Modelo de Inteligência Computacional e Processamento de Imagens para Interatividade Humano-Máquina Através de Gestos

Kéwen dos S. Silva[1], **Marlysson S. Dantas**[1], **Alcides X. Benicasa**[1]

[1]Departamento de Sistemas de Informação – Universidade Federal de Sergipe (UFS)
Caixa Postal – 49.506-036 – Itabaiana – SE – Brasil

{kewensilva,marlay}@academico.ufs.br,alcides@ufs.br

Abstract. *This paper presents a visual target recognition system using the ESP32-CAM microcontroller in conjunction with remote processing via neural networks. The embedded device captures and transmits images to a server for processing, where they are pre-processed, segmented, and classified using a multilayer perceptron (MLP) implemented in PyTorch. The solution explores the integration between embedded systems and artificial intelligence, optimizing computational resources and reducing latency. Experiments demonstrate the feasibility of the approach for low-cost intelligent monitoring applications. The proposed architecture is modular and adaptable, favoring deployment in various IoT contexts.*

Resumo. *Este artigo apresenta um sistema de reconhecimento de alvos visuais utilizando o microcontrolador ESP32-CAM em conjunto com processamento remoto via redes neurais. O dispositivo embarcado captura e envia imagens para um servidor, onde são pré-processadas, segmentadas e classificadas por uma rede MLP implementada em PyTorch. A solução integra sistemas embarcados com inteligência artificial, otimizando recursos computacionais e reduzindo a latência. Os experimentos demonstram a viabilidade da abordagem para aplicações de monitoramento inteligente de baixo custo. A arquitetura proposta é modular e adaptável a diferentes contextos da Internet das Coisas.*

1. Introdução

A crescente expansão da Internet das Coisas (IoT) tem impulsionado a criação de dispositivos inteligentes capazes de interagir com o ambiente e com os usuários de maneiras cada vez mais sofisticadas. Paralelamente, os avanços em Visão Computacional, um ramo da Inteligência Artificial (IA), tornaram as técnicas de análise e interpretação de imagens mais acessíveis e poderosas. A combinação dessas áreas abre novas fronteiras para o desenvolvimento de sistemas interativos, onde ações podem ser disparadas a partir da interpretação de dados visuais capturados em tempo real.

Entretanto, um desafio significativo reside na implementação de algoritmos complexos de processamento de imagens e IA em dispositivos com recursos computacionais limitados, como os microcontroladores. Esses dispositivos são ideais para aplicações de IoT devido ao seu baixo custo, tamanho reduzido e eficiência energética, mas não possuem o poder de processamento necessário para executar tarefas como detecção e reconhecimento de objetos em tempo real de forma autônoma.

Para superar essa limitação, este trabalho propõe o desenvolvimento de um modelo computacional distribuído, utilizando um microcontrolador com câmera integrada, o ESP32-CAM, para realizar a captura de cenas de um ambiente. Em vez de processar as imagens localmente, o dispositivo as envia via protocolo HTTP para um servidor remoto. Este servidor, com maior capacidade computacional, é responsável por aplicar técnicas de processamento de imagens e inteligência artificial para detectar, segmentar e reconhecer alvos previamente definidos.

Após o processamento, o servidor armazena as coordenadas dos alvos detectados para que no envio de novas imagens seja possível a detecção de interação com o ambiente físico. Essa abordagem busca oferecer uma solução viável e acessível para aplicações em sistemas distribuídos embarcados com reconhecimento visual, priorizando a eficiência na transmissão de dados e na inferência remota, mantendo a simplicidade na arquitetura local.

O objetivo geral deste projeto é, portanto, desenvolver e validar uma solução que permita a um microcontrolador capturar imagens, delegar o processamento inteligente a um serviço remoto e, com base na resposta recebida, executar ações de interação. Como parte dos objetivos específicos, o sistema deverá ser capaz de segmentar partes relevantes da imagem, identificar interações com base na análise computacional e acionar respostas adequadas, retornando, por exemplo, ações a serem executadas pelo microcontrolador.

Este artigo detalha a metodologia empregada, desde a escolha das tecnologias de hardware e software até a arquitetura de comunicação entre o dispositivo embarcado e o servidor. Serão apresentados os algoritmos de segmentação de imagem e a rede neural desenvolvida para o reconhecimento dos alvos, seguidos pelos resultados experimentais e discussões sobre o desempenho do modelo proposto.

2. Fundamentação Teórica

Esta seção apresenta os principais conceitos e tecnologias utilizados no desenvolvimento da solução proposta, organizados em quatro eixos: comunicação em rede, Internet das Coisas (IoT), processamento de imagens e inteligência artificial.

2.1. Protocolos de Comunicação

A comunicação entre os dispositivos no projeto é baseada no modelo TCP/IP. Destacam-se os protocolos HTTP e MQTT. O HTTP é utilizado para o envio das imagens capturadas pelo microcontrolador ao servidor, utilizando os métodos GET e POST. Já o MQTT, protocolo leve voltado para IoT, é empregado na comunicação assíncrona entre o servidor e o microcontrolador, com mensagens publicadas em tópicos gerenciados pelo broker HiveMQ. Essa abordagem permite uma comunicação eficiente mesmo em redes instáveis ou de baixa largura de banda.

2.2. Internet das Coisas (IoT)

Internet das Coisas (IoT), termo cunhado em 1999 por [Kramp et al. 2013], consolidou-se como uma das mais significativas revoluções tecnológicas contemporâneas. Evoluindo de conceito futurista para realidade presente.

Para [Magrani 2018], persistem importantes divergências conceituais sobre IoT, inexistindo definição universalmente aceita. Em linhas gerais, caracteriza-se como um

ecossistema de objetos físicos interconectados à internet mediante sensores compactos e embutidos, criando um ambiente de computação onipresente (ubíqua) orientado à facilitação do cotidiano e introdução de soluções funcionais nos processos diários.

No projeto, o ESP32-CAM representa o nó IoT responsável pela captura de imagens em tempo real, com envio para processamento remoto. Sua escolha se deve ao baixo custo, integração com câmera e conectividade Wi-Fi, tornando-o adequado para aplicações embarcadas de visão computacional.

2.3. API e Arquitetura de Servidor

A comunicação entre o microcontrolador e o servidor é intermediada por uma API REST desenvolvida com o framework Flask, em Python. Essa API é responsável pelo recebimento, armazenamento e encaminhamento das imagens para processamento. O servidor utiliza o Nginx como proxy reverso, otimizando o balanceamento de carga e o acesso à API. O sistema é containerizado com Docker, permitindo isolamento de ambientes, escalabilidade e fácil replicação em diferentes sistemas operacionais.

2.4. Processamento de Imagens

O processamento de imagens é realizado no servidor, utilizando a biblioteca OpenCV. Inicialmente, as imagens são convertidas do espaço de cores RGB para HSV, mais adequado à segmentação baseada em tonalidade. Técnicas como aplicação de máscaras, remoção de ruído com filtro mediano e detecção de contornos são utilizadas para isolar objetos com características cromáticas específicas (neste caso, objetos vermelhos). Após a segmentação, as regiões de interesse são extraídas para posterior análise e classificação.

2.5. Inteligência Artificial e Redes Neurais

Para [Luger 2004] a inteligência artificial (IA) pode ser definida como o ramo da ciência da computação que se ocupa da automação do comportamento inteligente, isto considerando a convicção de que a IA é uma parte da ciência da computação e, como tal, deve ser fundamentada em sólidos princípios teóricos e práticos deste campo.

Sobre redes neurais, para [Haykin 2001] O cérebro é um computador (sistema de processamento de informação) altamente complexo, não-linear e paralelo. Ele tem a capacidade de organizar seus constituintes, estruturais, conhecidos por neurônios, de forma a realizar diversos processamentos p.,.ex., reconhecimento de padrões., percepção e controle motor).

De acordo com [Haykin 2001] a rede Multilayer Perceptron (MLP) possui múltiplas camadas de neurônios. Basicamente, ela consiste de um conjunto de unidades sensoriais que constituem a camada de entrada, uma ou mais camadas ocultas de nós computacionais e uma camada de saída de nós computacionais.

3. Metodologia

Esta seção descreve a abordagem adotada para o desenvolvimento da solução proposta, abrangendo tanto os aspectos embarcados quanto os componentes do servidor. O sistema foi estruturado com base em uma arquitetura distribuída, na qual a responsabilidade pela captura das imagens recai sobre um microcontrolador, enquanto o processamento inteligente ocorre remotamente em um servidor dedicado.

A metodologia adotada contempla a integração de diferentes tecnologias de hardware e software, a definição dos protocolos de comunicação utilizados e a aplicação de técnicas de visão computacional e inteligência artificial. Nos tópicos seguintes, são detalhadas a arquitetura do modelo, os processos de inicialização do microcontrolador, a segmentação das imagens, os métodos de reconhecimento e os mecanismos de interação com o ambiente.

3.1. Arquitetura do Modelo

A arquitetura proposta é composta por um microcontrolador ESP32-CAM, responsável pela captura de imagens, e por uma estrutura de microsserviços em um servidor remoto, encarregada do processamento dessas imagens e do gerenciamento dos dados. Além disso, o servidor realiza o reconhecimento dos alvos presentes nas imagens e identifica possíveis interações com eles. Quando uma interação é detectada, o servidor responde enviando comandos específicos ao microcontrolador, que os executa em tempo real.

As imagens capturadas pelo microcontrolador são enviadas ao servidor por meio do protocolo HTTP, utilizando requisições do tipo POST. Após o recebimento, o servidor realiza o processamento da imagem e responde com o status da operação. Já a comunicação assíncrona é realizada via protocolo MQTT, permitindo que o servidor envie comandos ao microcontrolador por meio de tópicos aos quais o dispositivo está previamente inscrito. Essa abordagem garante maior eficiência na troca de mensagens simples e em tempo real.

Esse modelo híbrido de protocolos foi adotado visando maior flexibilidade na comunicação com o microcontrolador. O protocolo HTTP, embora altamente eficiente para o envio de imagens, devido à sua otimização e velocidade nesse tipo de transferência, impõe um tempo máximo de resposta, o que pode ser problemático diante da variabilidade no tempo de processamento do servidor. Por outro lado, o protocolo MQTT é mais indicado para o envio de mensagens simples, como comandos ou textos curtos, por ser leve e eficiente em comunicações assíncronas. Diante disso, a combinação de ambos os protocolos atende melhor aos requisitos do sistema, especialmente em contextos de tempo real, nos quais a latência e a confiabilidade da comunicação são fatores críticos.

A Figura 1 ilustra a interação entre os componentes do sistema.

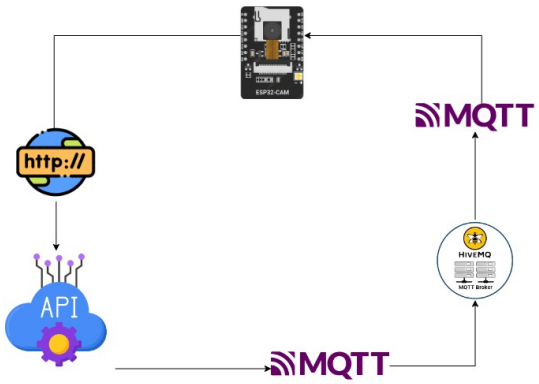


Figura 1. Arquitetura do modelo proposto

3.1.1. Inicialização do Microcontrolador

O microcontrolador ESP32-CAM é responsável por capturar imagens e enviá-las ao servidor via protocolo HTTP. Sua inicialização segue a estrutura padrão de sistemas embarcados, com uma fase de configuração inicial e uma de execução contínua. Na fase inicial, são realizadas as conexões Wi-Fi, o ajuste da câmera, a autenticação com o broker MQTT e a sincronização do horário com um servidor externo.

Durante a execução contínua, o dispositivo verifica a conectividade com o broker e realiza a captura e envio das imagens ao servidor.

A câmera OV2640, integrada ao ESP32-CAM, foi configurada com resolução Full HD, proporcionando boa qualidade nas capturas sem comprometer o desempenho. Ajustes como o número de quadros em buffer e o nível de compressão JPEG foram definidos para otimizar a transmissão via HTTP. Essa configuração garantiu estabilidade e qualidade mesmo em comparação a resoluções menores.

A captura e o envio de imagens, que ocorrem de forma cíclica, com o microcontrolador realizando periodicamente a aquisição de uma nova imagem através do módulo de câmera. Após a captura, os dados são preparados e transmitidos via protocolo HTTP para uma API web previamente configurada.

Durante esse processo, são definidos cabeçalhos apropriados para o envio, como o tipo do conteúdo (por exemplo, `image/jpeg`), e é feito o tratamento das respostas HTTP para assegurar que as imagens foram corretamente recebidas e processadas pelo servidor. Em caso de falhas, o sistema é capaz de registrar e identificar o erro, garantindo maior robustez na transmissão dos dados.

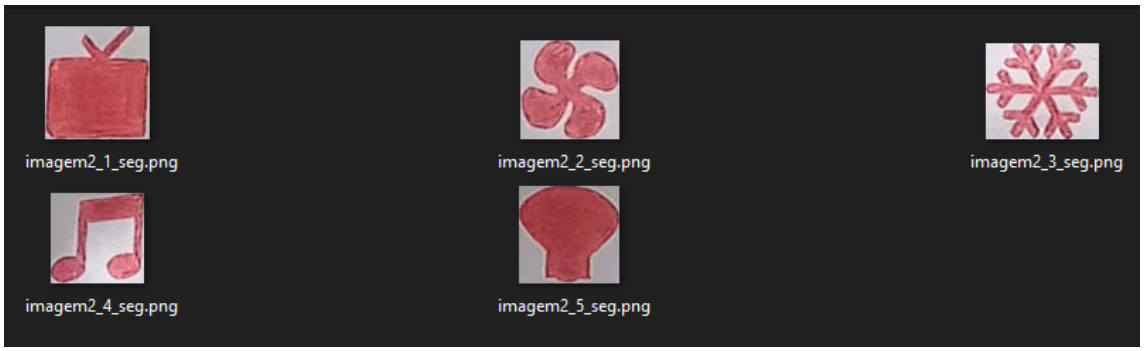
3.1.2. Segmentação de imagens

A etapa de segmentação de imagens nesta pesquisa exigiu o uso de técnicas de pré-processamento, como ajustes dimensionais, tratamento de cores e controle de pixels, para preparar adequadamente os dados para análise. As imagens são capturadas e enviadas pelo ESP32-CAM, sendo transmitidas via protocolo HTTP para uma API, que realiza todo o tratamento prévio.

O algoritmo de segmentação implementado é baseado em cores e utiliza processamento digital de imagens para identificar e isolar objetos com características cromáticas específicas. Para isso, as imagens são convertidas do espaço de cores RGB para HSV, o que permite maior robustez na detecção de tonalidades, especialmente em condições de iluminação variáveis. A segmentação foca na cor vermelha, com aplicação de máscaras que isolam regiões da imagem com essa tonalidade.

Funções auxiliares são responsáveis por carregar imagens, converter espaços de cores e remover ruídos. A conversão RGB para HSV é feita utilizando a biblioteca OpenCV, essencial para separar tonalidade (matiz) de brilho e saturação. A remoção de ruídos é realizada com filtro mediano, eficaz contra ruídos do tipo ”sal e pimenta”, preservando as bordas dos objetos. Já a criação das máscaras baseia-se em uma cor de referência convertida para HSV e aplicada com margens de variação, isolando os objetos desejados.

Figura 2. Alvos segmentadas



Fonte: Autoria própria.

A função principal do algoritmo segue uma sequência lógica: carrega a imagem, converte para HSV, aplica múltiplas máscaras com variações da cor vermelha, combina essas máscaras, aplica o filtro de ruído e realiza a detecção e filtragem dos contornos encontrados. Cada objeto detectado é verificado quanto ao seu tamanho, sendo descartados os que estão abaixo de um limiar mínimo, considerados ruídos.

O sistema também permite o processamento em lote de várias imagens, com parâmetros ajustáveis como variação de tonalidade, intensidade do filtro e tamanho mínimo do objeto. Essa abordagem possibilita testar a robustez da segmentação em diferentes condições. Além disso, a estratégia de usar múltiplas definições da cor vermelha combinadas aumenta a precisão na identificação dos objetos-alvo.

Por fim, o algoritmo implementa duas camadas de filtragem, remoção de ruído e verificação do tamanho dos segmentos, garantindo que apenas os objetos relevantes sejam considerados na análise. A estrutura modular do código, com funções bem definidas, facilita sua manutenção e expansão para aplicações mais complexas.

3.1.3. Reconhecimento de alvos

A tarefa de reconhecimento de alvos foi implementada por meio de uma rede neural do tipo Multilayer Perceptron (MLP), desenvolvida com o framework PyTorch. A MLP utilizada é composta exclusivamente por camadas lineares densamente conectadas (fully connected layers), sendo alimentada com imagens previamente segmentadas e submetidas a um processo de transformação, o qual inclui conversão para escala de cinza, redimensionamento para 64×64 pixels e conversão para tensores.

O conjunto de dados foi estruturado em diretórios separados por classe, correspondentes às categorias: lâmpada, colcheias, floco, hélice e televisão. Essa organização foi utilizada tanto para o treinamento quanto para a validação do modelo por meio do K-fold.

A arquitetura final adotada no experimento consiste em uma entrada com 4096 unidades (relativas aos 64×64 pixels da imagem em tons de cinza), uma camada oculta com 256 neurônios, função de ativação ReLU, e uma camada de saída com 5 neurônios — um para cada classe.

Essa abordagem permitiu realizar o reconhecimento de objetos simples com bom desempenho, validando a viabilidade de aplicar modelos de IA leves em arquiteturas distribuídas com microcontroladores e servidores.

A etapa de reconhecimento dos alvos foi conduzida por meio de redes neurais artificiais, dada sua capacidade de identificar padrões complexos. Utilizou-se a biblioteca PyTorch, com a classe `torch.nn.Linear` para representar camadas densas. Essa camada aplica uma transformação linear da forma:

$$y = \sum_{i=1}^{n} x_i \cdot w_i + b$$

em que x_i são as entradas, w_i os pesos, b o viés e y a saída.

Como entrada, o modelo recebe imagens convertidas em vetores unidimensionais utilizando `torch.nn.Flatten`. A função de ativação escolhida foi a ReLU (*Rectified Linear Unit*), que introduz não linearidade à rede:

$$ReLU(x) = max(0, x)$$

As imagens foram organizadas em cinco classes e processadas com as transformações: `Grayscale`, `Resize` (64×64) e `ToTensor`. Para o treinamento, utilizou-se a função de perda `CrossEntropyLoss`, comum em tarefas de classificação. Essa função mede a divergência entre a predição e o rótulo correto, assumindo a ativação Softmax:

$$\mathcal{L} = -\log(p_y)$$

onde p_y é a probabilidade prevista para a classe correta.

O processo de treinamento foi realizado com o otimizador Adam, por meio de uma função que percorre várias épocas. Em cada época, o modelo realiza predições, calcula a perda, aplica retropropagação e atualiza os pesos.

Para avaliação, adotou-se como referência o valor de perda inicial esperado de $\log(5) \approx 1{,}61$, correspondente a uma rede sem aprendizado (acurácia de 20%). Valores de perda abaixo desse indicam aprendizado, classificados em intervalos: 1,5–1,7 (baixo desempenho), 1,0–1,5 (aprendizado inicial), 0,7–1,0 (aprendizado moderado), 0,3–0,7 (boa performance) e abaixo de 0,3 (alto desempenho).

3.1.4. Interação com os alvos

A interação com os alvos é o principal objetivo funcional do sistema proposto e constitui a etapa atual de desenvolvimento. Neste estágio, dois métodos distintos estão sendo testados para verificar a tentativa de interação do usuário com os elementos identificados na imagem.

O primeiro método baseia-se na comparação entre histogramas de matiz (Hue) no espaço de cor HSV. O procedimento consiste na comparação entre uma imagem base

do objeto segmentado e outra obtida durante a tentativa de interação. Ambas são convertidas para o espaço HSV, e seus histogramas de matiz são extraídos, normalizados e comparados por meio da distância do cosseno. Caso a dissimilaridade ultrapasse um limiar predefinido (por exemplo, 0,6), considera-se que houve uma tentativa de interação. Gráficos comparativos são gerados para apoiar a análise visual dos resultados.

No segundo método, a tentativa de interação é tratada como uma nova classe no processo de classificação. Para isso, foi criada a classe ”interação”, composta por imagens em que o alvo identificado está parcialmente encoberto, como, por exemplo, pela presença de uma mão. O modelo é então treinado para distinguir as classes originais dos alvos e a nova classe de interação, permitindo que o sistema reconheça diretamente, por meio da rede neural, quando ocorre uma tentativa de interação por alteração visual no alvo.

Como mencionado na Seção 3.1.3, temos definidos cinco alvos, e o significado de cada um deles dependerá da estrutura externa e do comportamento programado no microcontrolador. Por exemplo, o reconhecimento do alvo ”lâmpada”pode ser interpretado como um comando para acionar uma única lâmpada ou um conjunto delas, a depender do contexto e da lógica embarcada no sistema. Dessa forma, o sistema é flexível e permite que cada alvo represente diferentes ações, conforme as necessidades da aplicação.

4. Resultados Parciais

No estágio atual do projeto, o microcontrolador ESP32-CAM envia as imagens diretamente para o servidor/API em qualidade HD+ (1600x900). Essa alta resolução foi mantida graças à arquitetura híbrida de protocolos adotada, na qual o protocolo HTTP é utilizado para o envio das imagens. Essa escolha proporcionou uma transmissão rápida e mais robusta para arquivos de maior tamanho, o que não seria viável utilizando apenas o protocolo MQTT, que é mais adequado para dados leves e de baixa latência. Para uma realizar uma decisão eficiente, foi realizado testes com o envio das imagens utilizando ambos os protocolos, MQTT e HTTP, como na figura 1 abaixo .

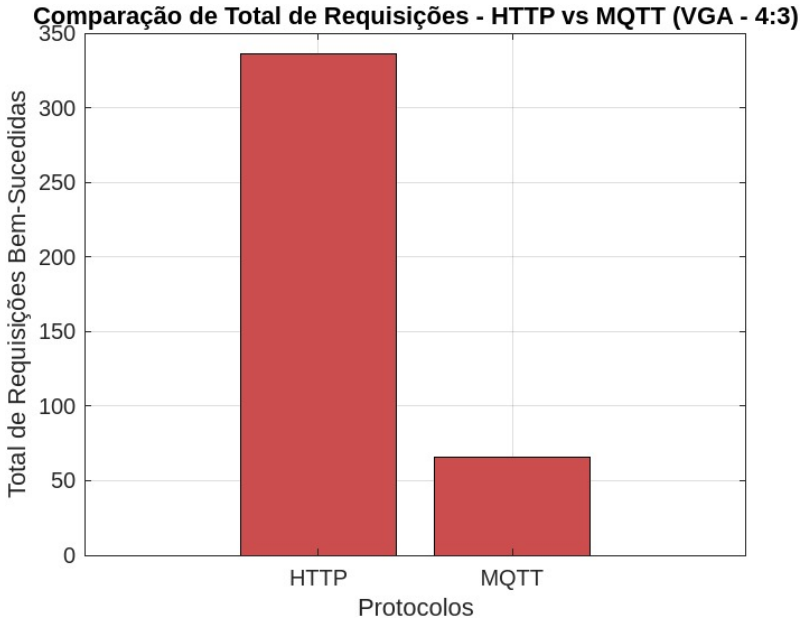


Figura 3. Comparação do Total de Requisições HTTP X MQTT

Houve uma diferença nítida entre os protocolos. Embora os testes tenham sido realizados com a resolução mais baixa, o protocolo MQTT é projetado para mensagens pequenas e leves, enquanto o HTTP lida melhor com grande volumes de dados, como imagens.

Além do envio das imagens, o servidor/API também é responsável por todo o processamento posterior, iniciando pela etapa de segmentação dos alvos. Nesse processo, as imagens recebidas passam por uma série de etapas de pré-processamento, que incluem redimensionamento, correção e conversão de cores, além do controle do tamanho dos pixels. Essas operações são essenciais para padronizar e preparar adequadamente os dados visuais, garantindo que estejam em condições ideais para as fases seguintes de segmentação e análise. Com os ajustes corretos aplicados, o algoritmo de segmentação consegue isolar com precisão apenas os elementos de interesse (alvos) eliminando ruídos e informações irrelevantes à tarefa proposta.

Por meio de algoritmos de aprendizado de máquina, foi possível treinar o modelo responsável pelo reconhecimento dos alvos diretamente na API. Antes da definição final do modelo, utilizou-se a técnica de validação cruzada K-Folds com o objetivo de obter uma avaliação mais robusta e confiável do desempenho durante o treinamento. Essa abordagem permite dividir o conjunto de dados em múltiplas partições, alternando entre conjuntos de treino e teste, o que contribui para uma melhor generalização do modelo. A partir dessa validação, é possível observar a quantidade ideal de épocas necessárias para atingir uma perda mínima, a acurácia média obtida e a taxa de aprendizado por época. Além disso, durante o processo de treinamento, são geradas matrizes de confusão que permitem uma análise mais detalhada sobre os acertos e erros do modelo, fornecendo uma visão clara sobre seu comportamento e desempenho na classificação dos alvos.

Nos testes parciais realizados até o momento, o método que utiliza uma rede neural com uma classe adicional denominada ”interação”apresentou desempenho superior. Nesse método, imagens contendo uma mão cobrindo o alvo são rotuladas como interação e utilizadas no treinamento da rede. Enquanto o método baseado na dissimilaridade de histogramas apresentou valores médios entre 0,75 e 0,85 para casos de interação, a rede neural alcançou taxas de acerto entre 98% e 99% na classificação dessas imagens como interação. Esses resultados indicam, até o estágio atual do projeto, uma vantagem significativa do segundo método em termos de precisão.

Portanto, este projeto, desenvolvido no contexto de um sistema embarcado com microcontrolador ESP32-CAM e processamento remoto via API, configura-se como uma solução inovadora para o reconhecimento de alvos em tempo real.

O sistema permite uma avaliação precisa do desempenho do modelo de aprendizado, utilizando métricas de perda e acurácia que refletem sua evolução desde os estágios iniciais até níveis mais avançados, assegurando a confiabilidade da classificação dos alvos.

A arquitetura adotada, a qual integra um microcontrolador compacto, envio de imagens por HTTP, respostas e comandos via protocolo MQTT, e processamento realizado com Flask e PyTorch, demonstra-se eficiente, escalável e flexível, oferecendo uma base sólida para futuras expansões e adaptações a novos cenários.

Essa combinação tecnológica evidencia o potencial da integração entre hardware

embarcado e inteligência artificial, promovendo uma automação inteligente com uso otimizado de recursos. O projeto pode ser facilmente adaptado a diferentes dispositivos e aplicações, contribuindo significativamente para o avanço da visão computacional no contexto de sistemas IoT.

5. Conclusão

Com base nos resultados obtidos até o momento, o projeto tem demonstrado a viabilidade da integração entre tecnologias de IoT e APIs para realizar, de forma eficiente, a captura, o envio, o processamento de imagens e o reconhecimento de alvos. Através do sistema embarcado, foi possível capturar imagens em tempo real utilizando o microcontrolador ESP32-CAM, que as envia ao servidor por meio do protocolo HTTP. No servidor, que dispõe de maior capacidade computacional em comparação ao microcontrolador, as imagens são processadas com etapas de segmentação e detecção dos alvos. Após o processamento, comandos específicos são gerados e enviados de volta ao microcontrolador utilizando o protocolo MQTT, completando o ciclo de comunicação e controle entre os dispositivos. Essa integração evidencia o potencial da arquitetura proposta para aplicações em visão computacional embarcada e automação inteligente.

Por fim, vale ressaltar que o projeto ainda está em desenvolvimento e não se encontra totalmente finalizado. Como etapas futuras, está prevista a conclusão da implementação do reconhecimento de interação com os alvos detectados, permitindo que uma pessoa interaja com um determinado alvo e realize uma ação previamente definida. Além disso, será incorporado o reconhecimento facial, com o objetivo de autenticar usuários e adicionar uma camada extra de segurança e personalização ao sistema.

Referências

Haykin, S. (2001). *Redes Neurais: Princípios e Prática*. Bookman, Porto Alegre, 2 edition. Tradução do original: Neural Networks: A Comprehensive Foundation.

Kramp, T., Van Kranenburg, R., and Lange, S. (2013). Introduction to the internet of things. *Enabling things to talk: Designing IoT solutions with the IoT architectural reference model*, pages 1–10.

Luger, G. F. (2004). *Inteligência Artificial: estruturas e estratégias para a solução de problemas complexos*. Bookman, Porto Alegre, 4 edition. Tradução da obra original *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*.

Magrani, E. (2018). *A Internet das Coisas*. FGV Editora, Rio de Janeiro.