Câmera de Filmagem Inteligente para Contagem e Detecção de Veículos[1].

Vinícius Westphal de Paula[2]; Ênio dos Santos Silva[3].

Resumo Expandido

(1) Pesquisa apresentada para a Unidade Curricular Projeto Integrador 3 do Curso de Engenharia Elétrica

(2) Estudante da 9ª fase do Curso de Engenharia Elétrica

(3) Professor orientador da Unidade Curricular Projeto Integrador 3

RESUMO: Como parte da disciplina de Projeto Integrador 3, foi proposto um projeto envolvendo a implementação de um sistema embarcado contendo um equipamento de filmagem e com capacidade para detecção de veículos em tempo real, utilizando um microcontrolador e técnicas de aprendizado de máquina. O objetivo principal foi avaliar a viabilidade de implementar um modelo de detecção de objetos em um microcontrolador, considerando os obstáculos associados às restrições de recursos computacionais nesses dispositivos e aos requisitos para executar em tempo real tal modelo. Como método foram realizadas duas pesquisas principais: uma acerca de microcontroladores com capacidade de integrar câmera; outra focada no campo de aprendizado de máquina para aprender sobre modelos de detecção de objetos. Foram então realizados testes com um módulo microcontrolador ESP32-CAM, verificando a funcionalidade da sua câmera integrada. A pesquisa no campo do aprendizado de máquina foi voltada à abordagem tinyML, que foca em executar modelos diretamente em dispositivos com recursos limitados como microcontroladores, o que exige a otimização e criação de algoritmos leves para que possam executar de maneira eficiente dentro das restrições desses dispositivos. Foi então implementado no ESP32-CAM um modelo chamado FOMO (Faster Objects, Moving Objects), desenvolvido especificamente para executar uma versão simplificada de detecção de objetos em microcontroladores. Treinou-se o modelo, com auxílio da plataforma Edge Impulse, com imagens próprias de veículos. Após implementação, foi possível realizar a detecção dos veículos treinados, podendo diferenciá-los por quantidade, porém com algumas limitações, abrindo espaço para testes utilizando dispositivos com mais recursos, como as placas Raspberry Pi.

Palavras-chave: microcontroladores; detecção de objetos; tinyML.

INTRODUÇÃO

Câmeras são frequentemente utilizadas para capturar tráfego para a contagem de veículos, em situações como controle de tráfego, gestão de estacionamentos, monitoramento de estradas, entre outros. Algoritmos de visão computacional baseados em aprendizado de máquina podem ser empregados para identificar e contar automaticamente os veículos à medida que passam pela área monitorada.

A justificativa deste projeto se baseia no conceito de computação de borda, onde busca-se trazer poder computacional e processamento de dados mais próximos da “borda” da rede de informações através de sistemas embarcados, ao invés de depender de um computador central ou servidor baseado em nuvem. No geral, os modelos de visão computacional exigem recursos computacionais consideráveis, necessitando de hardware dedicado, como processadores, GPUs e memória extensa. Um campo do aprendizado de máquina que busca trazer a execução desses modelos

para dispositivos de baixa potência, como sistemas embarcados baseados em microcontroladores, é chamado de aprendizado de máquina minúsculo (tinyML). Ao executar algoritmos de aprendizado de máquina diretamente em sistemas embarcados, é possível realizar tarefas de visão computacional sem depender de serviços em nuvem.

Dessa forma, o intuito desse projeto foi de testar a viabilidade da execução em tempo real de um modelo de detecção de objetos utilizando uma abordagem tinyML, em um microcontrolador. O modelo deveria ser capaz de detectar veículos, através de um treinamento utilizando um banco de dados próprio, e então ser implementado em um microcontrolador com suporte a um módulo de câmera.

I. METODOLOGIA

A metodologia se baseou em pesquisas de cunho exploratório, sendo uma sobre microcontroladores com suporte a câmera, enquanto a outra buscou expandir o conhecimento sobre o campo de aprendizado de máquina, especificamente na área de detecção de objetos. Buscou-se então implementar um modelo de detecção de objetos em um microcontrolador com suporte à câmera, com base nos resultados das pesquisas.

De maneira geral, microcontroladores são circuitos integrados compactos que funcionam como um pequeno computador em um único chip semicondutor. São projetados para uso em aplicações embarcadas, em contraste com os microprocessadores usados em computadores pessoais ou em outra aplicação de propósito geral composta por vários chips discretos. Operam de maneira eficiente com um consumo de potência mínimo, o que os torna ideal para dispositivos alimentados por bateria ou sistemas onde a eficiência de energia é crucial. Existem diversos microcontroladores disponíveis no mercado que suportam interface à câmera para aplicações diversas. ESP32-CAM é um microcontrolador que possui uma câmera de vídeo integrada e um leitor para cartão microSD para armazenamento, além de possuir Wi-Fi e Bluetooth integrados, unindo todas essas funcionalidades a um preço acessível.

Jamshedji (2020) descreve uma desvantagem do ESP32-CAM o fato da placa não possuir um conector USB para programação da placa, sendo necessário utilizar um adaptador FTDI, o que exige uma série de conexões externas. De maneira alternativa, é possível adquirir uma placa adaptadora USB, com uma implementação bastante simples, já que a placa ESP32-CAM simplesmente se encaixa nela. Conjuntos como o da figura 1, contendo a placa ESP32-CAM, a placa adaptadora USB e um módulo de câmera baseado no sensor OV2640, são facilmente encontrados para compra on-line. O módulo OV2640 possui um sensor de 2MP (megapixels) e capacidade de transferência de imagem numa taxa de 15 a 60 quadros por segundo.

Figura 1 – Módulo ESP32-CAM, juntamente de um módulo de câmera OV2640 de 2MP e uma placa ESP32-CAM-MB para programação.

Jamshedji (2020) descreve a possibilidade de trabalhar com placas ESP32, incluindo o ESP32-CAM, com o Arduino IDE, sendo necessário adicionar o gerenciador de placas (Board Manager) "esp32 by Espressif Systems". Dessa forma, o módulo ESP32-CAM se mostrou bastante atrativo para execução do projeto, atendendo os requisitos básicos esperados de um microcontrolador contendo suporte à

câmera, e com programação facilitada por meio do Arduino IDE.

Entrando na área de aprendizado de máquina, a detecção de objetos é uma técnica de visão computacional usada para identificar e localizar objetos dentro de imagens ou vídeos. Envolve a detecção de instâncias de objetos dentro de uma imagem, determinando sua classe ou categoria específica, assim como sua localização específica dentro da imagem. Existem diversas abordagens para detecção de objetos, alguns dos métodos mais populares incluem o uso de abordagens baseadas em Aprendizado Profundo que utilizam redes neurais profundas, como Redes Neurais Convolucionais (CNNs).

A desvantagem das CNNs, apesar de serem poderosas para tarefas de visão computacional, é sua exigência de uma quantidade substancial de poder computacional, o que torna difícil sua implementação em dispositivos com recursos restritos. Visando contornar esse tipo de limitação, foi desenvolvida uma área chamada de aprendizado de máquina minúsculo (tinyML), que se concentra na otimização de algoritmos e modelos para que funcionem de maneira eficiente em dispositivos com recursos limitados. Técnicas como quantização de modelo, poda e arquiteturas especializadas são usadas para reduzir o tamanho dos modelos ainda mantendo uma precisão aceitável.

Um modelo de detecção de objetos que adota uma aplicação tinyML é FOMO (Faster Objects, Moving Objects), que segundo Moreau e Kelcey (2022), implementa uma versão simplificada de detecção de objetos, chamada também de detecção constrita de objetos, para casos onde o tamanho do objeto não é relevante, apenas sua posição na imagem. É feita uma detecção baseada nos centróides dos objetos, sendo uma abordagem que foi pensada especificamente para executar detecção de objetos em microcontroladores. FOMO divide uma imagem de entrada em uma grade e executa o equivalente a classificação de imagem em todas as células da grade de forma independente e em paralelo. Por padrão, o tamanho da grade é de 8x8 pixels, o que significa que para uma imagem de 96x96, a saída será de 12x12. Em outras palavras, FOMO executa a detecção de objetos em cada uma das células e determina a probabilidade de um objeto estar presente naquela célula.

Para um resolução de entrada de 96x96, observa-se na figura 2 (contendo parafusos) que o tamanho das células das grades é de 1/8 da resolução de entrada e em um formato quadrado. Para determinar o centróide de um objeto, é então escolhida a célula com maior probabilidade de conter determinado objeto, enquanto as células adjacentes com probabilidade menor de conter o mesmo objeto são ignoradas.

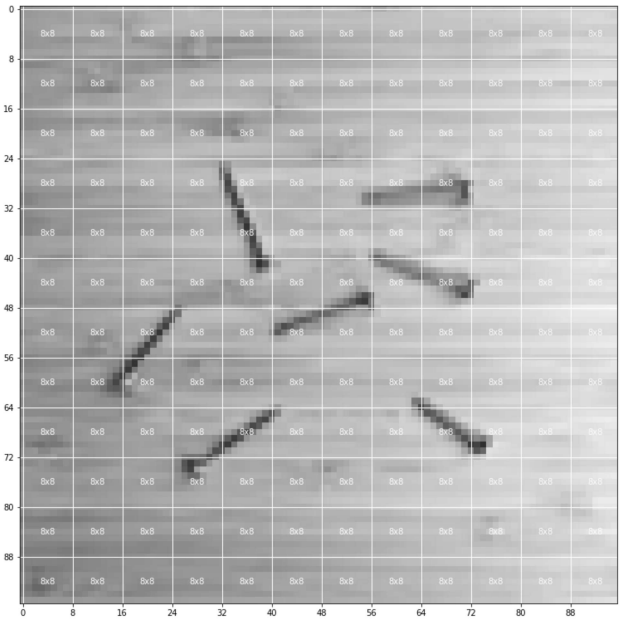

Figura 2 – Imagem de 96x96 pixels dividida em uma grade 12x12.

Moreau e Kelcey (2022) argumentam que o cenário ideal para utilização do modelo FOMO envolve objetos que sejam todos de tamanho similar, e ocupem pelo menos uma célula por completo. Também é importante que objetos não estejam tão próximos uns dos outros, pois se objetos diferentes ocuparem a mesma célula da grade, apenas o objeto de maior probabilidade será considerado. Uma maneira de contornar essa limitação seria aumentando a resolução da imagem, aumentando assim o número de células,

porém exigindo mais poder computacional do dispositivo embarcado.

Para implementar um modelo de detecção de objetos, seja ele FOMO ou qualquer outro, é necessário treiná-lo, o que geralmente segue um fluxo de trabalho bem definido. O processo de treinamento começa com a coleta e preparação dos dados, onde são coletadas imagens e rotulados os objetos de interesse para formar um conjunto de dados, seguido pela escolha da arquitetura do modelo, como FOMO, por exemplo. Posteriormente, realiza-se o pré-processamento dos dados, definindo uma resolução padrão, no caso do modelo FOMO geralmente utiliza-se uma resolução de 96x96 pixels com imagens em escala de cinza, e é feita a divisão dos dados em conjuntos de treinamento e teste. O treinamento ocorre utilizando algoritmos de otimização e técnicas como transferência de aprendizado, fazendo previsões e ajustando os pesos com base nas diferenças entre as previsões e os rótulos reais para minimizar a função de perda. Esse processo é repetido por várias épocas até que o modelo atinja um desempenho satisfatório, medido por métricas como o F1 score, medindo precisão e recall. Após o treinamento, o modelo é avaliado no conjunto de testes para verificar sua eficácia na detecção de objetos antes de ser implantado para uso prático. Uma das vantagens do modelo FOMO é que após treinado pode ser exportado como uma biblioteca do Arduino, facilitando sua implementação no ESP32-CAM.

III. RESULTADOS E DISCUSSÃO

Com base nas pesquisas optou-se por utilizar como microcontrolador um ESP32-CAM, adquirido junto de uma placa adaptadora USB e módulo de câmera OV2640, visando implementar um modelo FOMO treinado com imagens próprias, com o intuito de realizar a detecção de veículos.

O treinamento do modelo FOMO foi realizado com auxílio da plataforma Edge Impulse. Seguindo o fluxo de trabalho, foram utilizadas 48 imagens próprias contendo 3 veículos distintos, divididas numa razão treinamento/teste de 73/27. A figura 3 demonstra um exemplo de imagem usada no conjunto de dados. Inicialmente o pré-processamento de imagens foi definido utilizando uma resolução de 96x96 pixels definindo as imagens em escala de cinza.

Figura 3 – Exemplo de imagem utilizada no conjunto de treinamento.

O treinamento foi realizado utilizando FOMO baseado em MobileNetV2 com um fator de escala 0.1, utilizando 30 ciclos de treinamento, com uma taxa de aprendizado de 0.001. Nota-se pela figura 4, que o primeiro treinamento obteve um F1 score 88.4%, indicando uma boa precisão de previsões para os objetos treinados, entretanto sua performance prevista em um ESP32-CAM indicou um tempo de inferência elevado acima de 1 segundo. Na prática ao exportar o modelo e executá-lo diretamente no ESP32-CAM, onde não foi possível detectar nenhum objeto de maneira concisa.

Figura 4 – Métricas de precisão e desempenhos previstos para o primeiro treinamento.

Foi então realizado um novo treinamento, ajustando a etapa de pré-processamento das imagens ao utilizar uma menor resolução de 48x48 pixels,

ainda definindo as imagens em escala de cinza. Observa-se pela figura 5 que houve uma diminuição do F1 score para 83.3%, indicando uma precisão menor das previsões. Entretanto, a performance prevista em um ESP32-CAM acabou amplamente melhorada, com um tempo de inferência pelo menos 5 vezes menor que do treinamento anterior.

Figura 5 – Métricas de precisão e desempenhos previstos para o segundo treinamento.

Na prática, exportando o modelo com o segundo treinamento para o ESP32-CAM, foi possível realizar a detecção de objetos. O programa utilizado no Arduino IDE envolvia fotografar um fundo branco utilizando a câmera do ESP32-CAM, executar a biblioteca do modelo de detecção FOMO treinado, e imprimir através do monitor serial o rótulo de cada objeto detectado.

Conforme o exemplo da figura 6, foi possível diferenciar os veículos do fundo branco e contá-los por quantidade. Retirando os veículos de cena também refletia na perda de sua detecção. Isso mostrou que houve sucesso na implementação do modelo FOMO no ESP32-CAM.

Entretanto houveram limitações na detecção, principalmente em momentos onde os veículos se encontravam muito próximos uns dos outros. Esse era um problema já esperado, devido ao modo de como o modelo FOMO funciona, onde devido a resolução de 48x48 pixels no pré-processamento das imagens, o número de grades que compõem cada imagem é reduzido e os objetos podem acabar ocupando a mesma célula, o que acarreta na perda detecção de objetos próximos, considerando apenas um deles. Para contornar esse problema seria necessário uma resolução maior no pré-processamento das imagens, mas como observado pelo primeiro treinamento, essa não seria uma solução viável de ser implementada em um ESP32-CAM.

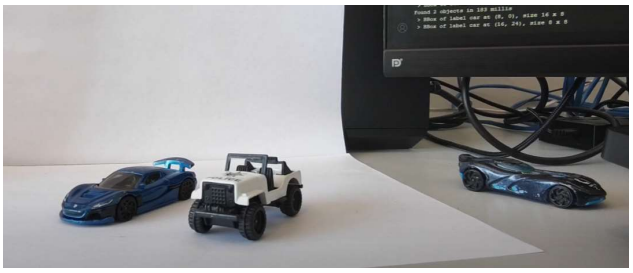

Figura 6 – Execução em tempo real do modelo FOMO em um ESP32-CAM.

IV. CONSIDERAÇÕES FINAIS

Nota-se que o ESP32-CAM é capaz de executar o modelo FOMO para detecção de objetos com sucesso.

Devido às limitações do funcionamento do modelo FOMO, as aplicações para uma implementação no ESP32-CAM também se restringem.

Uma possível aplicação real para detecção de veículos que se ajusta às limitações do modelo FOMO envolve contabilizar o número de veículos em um estacionamento durante o dia, onde a resolução da imagem pode ser ajustada para que cada veículo ocupe exatamente uma grade de maneira ideal.

No entanto, espera-se também a necessidade de mais poder computacional envolvido, e é provável que um ESP32-CAM não seja totalmente capaz de realizar essa tarefa, onde uma placa Raspberry Pi pode ser considerada como um substituto viável para essa aplicação.

REFERÊNCIAS BIBLIOGRÁFICAS

BitMaker. **ESP32 CAM com Câmera + Programador ESP32-CAM-MB**. Disponível em < https://www.bitmaker.com.br/produto/esp32-cam-com-camera-programador-esp32-cam-mb.html >. Acesso em 10 ago. 2023.

JAMSHEDJI, W. **Getting started with the ESP32-CAM**. Disponível em: < https://dronebotworkshop.com/esp32-cam-intro/ >. Acesso em 01 ago. 2023.

MOREAU, L; KELCEY M. **Announcing FOMO (Faster Objects, More Objects).** Disponível em : < https://edgeimpulse.com/blog/announcing-fomo-faster-objects-more-objects >. Acesso em 15 nov. 2023