Análise de Modelos de Aprendizado Profundo para Sistemas Embarcados aplicados à Classificação de Rachaduras em Construções

Adenilton Morais Arcanjo
RAI - Robotics and Artificial Intelligence
Centro de Ciências Exatas e Tecnológicas
Universidade Federal do Recôncavo da Bahia
Cruz das Almas, Brasil
adeniltonmorais@gmail.com

André Luiz Carvalho Ottoni
RAI - Robotics and Artificial Intelligence
Centro de Ciências Exatas e Tecnológicas
Universidade Federal do Recôncavo da Bahia
Cruz das Almas, Brasil
andre.ottoni@ufrb.edu.br

Abstract*—A Inteligência Artificial (IA) tem sido aplicada em diversas áreas para otimizar tarefas, agilizar atividades e reduzir custos. Na construção civil, a IA tem sido utilizada através do aprendizado de máquina e da visão computacional para automatizar a inspeção visual em obras e construções. Um dos principais usos é na detecção de patologias, como rachaduras, trincas e fissuras em estruturas como edifícios, pontes, tubulações, e afins. Outra área que vem aplicando técnicas de aprendizado de máquina é a de sistemas embarcados. É possível encontrar cada vez mais dispositivos com menor poder de processamento e menor capacidade de memória utilizando alguma técnica de aprendizado de máquina. Pode-se citar os *smartwatches*, assistentes virtuais, entre outros. O presente trabalho tem como objetivo selecionar modelos de aprendizado de máquina para a detecção de rachaduras em estruturas de concreto utilizando a plataforma Edge Impulse. Para isso, foram avaliadas diferentes versões da arquitetura de redes neurais *MobileNet*, levando em consideração critérios como acurácia, consumo de recursos (memória RAM e capacidade de armazenamento em Flash) e tempo de inferência. Ao término do processo, foram selecionados seis modelos candidatos que demonstraram potencial para serem implementados no dispositivo embarcado. Esses modelos apresentaram uma excelente acurácia, com valores variando de 97% a 99,6%. Além disso, o consumo de memória RAM necessário não ultrapassou 280 Kb, enquanto a utilização de memória FLASH ficou abaixo de 225 Kb. Em relação ao tempo de inferência, o modelo mais rápido registrou um tempo de 328 ms, enquanto o modelo mais lento atingiu 1240 ms. Todas as configurações da plataforma e a construção do modelo foram realizadas considerando o dispositivo embarcados de referência ESP32-CAM. Espera-se que este trabalho contribua para a seleção de modelos eficientes para a detecção de rachaduras em concretos, facilitando a implementação em sistemas embarcados.

***Index Terms*—crack detection, edge impulse,machine learning, mobilenet, tinyml**

I. Introdução

Nos últimos anos, a inteligência artificial (IA) vem sendo utilizada em diversas áreas para solucionar problemas complexos e também melhorar a precisão e execução de tarefas [1]. Seu poder de processamento e análise de dados permite que sejam identificados padrões e relações que, muitas vezes, passam despercebidos pelo ser humano. A utilização da IA tem mostrado resultados relevantes em uma variedade de aplicações, como na saúde [2], reconhecimento facial [3], e de voz [4], [5].

Na construção civil, a detecção precoce de patologias, como rachaduras, trincas e fissuras, é uma preocupação importante para garantir a segurança estrutural de edifícios e infraestruturas [6]. Nesse contexto, a utilização da inteligência artificial (IA), principalmente no campo de aprendizado de máquina, tem recebido cada vez mais destaque no setor da construção civil. Na literatura, é possível encontrar diversos trabalhos e estudos utilizando aprendizado de máquina no monitoramento e detecção de patologias na contrução civil [7]–[10].

Outra área que vem utilizando o aprendizado de máquina é a de sistemas embarcados. O TinyML refere-se à implementação de modelos de aprendizado de máquina em sistemas embarcados com recursos limitados [11]. O TinyML tem se mostrado uma abordagem promissora para a implementação desses modelos de aprendizado de máquina em dispositivos embarcados, especialmente na detecção de patologias na construção civil [12], [13].

O uso do TinyML nesse contexto pode apresentar vantagens relevantes. No entanto, a escolha de um modelo a ser implementado nem sempre é uma tarefa fácil de se alcançar. Para projetar uma arquitetura de aprendizado de máquina funcional e eficaz, é preciso levar em consideração características e recursos disponíveis do dispositivo de destino [14]. A escolha dos modelos adequados para esses sistemas muitas vezes requer atenção apropriada, levando em conta elementos como o consumo de memória RAM e ROM, a precisão do modelo, tempo de inferência, consumo de energia, entre outros [11].

Com base nesses critérios, o objetivo deste trabalho é identificar e selecionar os modelos de aprendizado de máquina de classificação de rachaduras em estruturas de concreto que sejam mais adequados para implementação em um sistema embarcado. Nesse aspecto, foram analisados critérios como baixo consumo de memória, alta precisão e tempo de inferência reduzido dos modelos. Além disso, a plataforma Edge Impulse foi utilizada como ferramenta de desenvolvimento e avaliação

dos modelos.

A estrutura deste trabalho compreende seis seções distintas. A Seção II apresenta a fundamentação teórica. A Seção III descreve as etapas realizadas para a construção, seleção e implantação dos modelos atarvés da plataforma Edge Impulse. A Seção IV expõe os resultados obtidos. A Seção V apresenta as considerações finais do trabalho, juntamente com sugestões para trabalhos futuros, e por fim os agradecimentos.

II. Fundamentação Teórica

A. TinyML

TinyML é um termo que se refere ao uso de modelos de aprendizado de máquina em dispositivos que possuem baixo consumo de energia, baixo poder de processamento e memória, como microcontroladores e alguns sistemas embarcados [15]. A ideia é permitir que esses dispositivos executem tarefas complexas sem depender de servidores externos para a execução.

Com a capacidade de incorporar aprendizado de máquina em dispositivos com recursos limitados, o TinyML está se expandindo rapidamente. São várias as áreas que estão desenvolvendo soluções e pesquisas utilizando essa tecnologia. Pode-se encontrar o uso do TinyML em diversos setores, como na indústria, onde é aplicado na detecção de anomalias [16], na saúde [17]–[19], construção cívil [13], no reconhecimento de voz [20], [21], entre outros. Essas aplicações demonstram o potencial do TinyML em diversas áreas, impulsionando a adoção de sistemas embarcados mais inteligentes e eficientes.

B. Arquitetura MobileNet

As arquiteturas das redes neurais convolucionais (CNNs) têm se tornado cada vez mais profundas, o que implica na necessidade de mais memória e tempo de processamento para a execução desses modelos. Essa demanda também limita sua capacidade de serem implementados em dispositivos embarcados, principalmente aqueles que possuem recursos limitados [22].

Pesquisadores do Google desenvolveram as arquiteturas de redes neurais *MobileNet V1* e *MobileNet V2*, com o objetivo de equilibrar desempenho e eficiência [23]. Esses modelos são adequados para serem implementados em sistemas embarcados que possuem baixo desempenho de hardware e poder computacional limitado.

O *MobileNet V1* utiliza operações convolucionais separadas em profundidade, no qual é dividido uma operação de convolução tradicional em duas etaps diferentes. A etapa inicial, chamada de "depthwise convolution", aplica um filtro convolucional individual para cada canal da imagem de entrada. Em seguida, na etapa subsequente, conhecida como "pointwise convolution", ocorre a combinação linear dos resultados obtidos na primeira etapa, considerando todos os canais da imagem [24]. Essa estrutura pode ser vista na Figura 1.

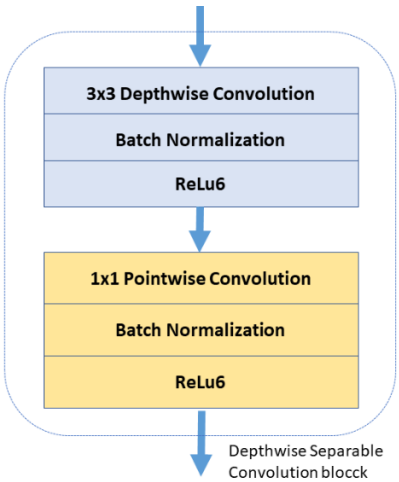

Fig. 1. Camada convolucional separavél em profundidade. Imagem adaptada de [25]

Na versão *MobileNet V2*, foram aplicadas melhorias com o objetivo de obter melhores resultados. Nessa versão, é introduzido o uso de blocos residuais, nos quais cada bloco é composto por três camadas distintas: (1) camada de expansão, (2) camada de convolução em profundidade e (3) camada de projeção [26]. Essa estrutura pode ser vista na Figura 2.

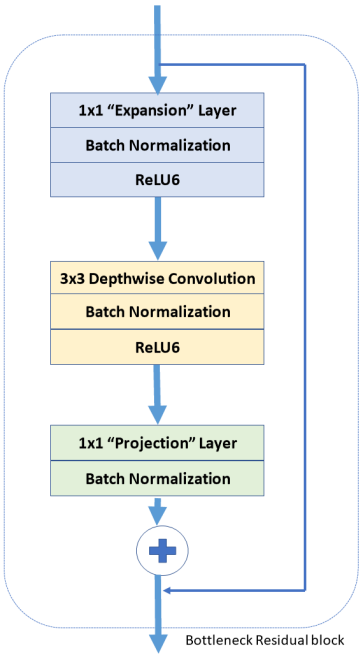

Fig. 2. Bloco residual bottleneck. Imagem adaptada de [25]

C. Plataforma Edge Impulse

O Edge Impulse é uma plataforma online criada para simplificar o fluxo de trabalho de aprendizado de máquina, permitindo aos usuários experimentar de forma completa todo o processo. Com o Edge Impulse, os usuários podem facil-

mente coletar dados, treinar modelos, avaliar o desempenho e implantar os modelos em dispositivos embarcados [27]. A Figura 3 apresenta um fluxo de trabalho de aprendizado de máquina que pode ser realizado na plataforma Edge Impulse.

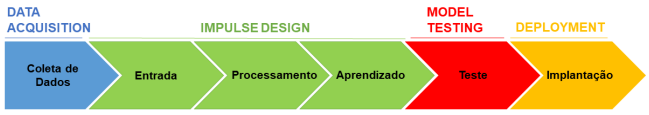

Fig. 3. Processo de classificação de imagens do Edge Impulse.

Na etapa de coleta de dados (*Data Acquisition*), a plataforma permite que o usuário envie dados ao projeto por meio de WebUSB, Edge Impulse API, Edge Impulse CLI ou por upload de arquivo. Nessa etapa, os dados são rotulados e divididos em conjuntos de treinamento e teste [28].

A etapa do projeto do Impulse Design [29] é composta por 3 blocos: o bloco de entrada, o bloco de processamento e o bloco de aprendizado. No bloco de entrada é determinado o tipo de dado de entrada usado para treinar o modelo. Nesse bloco, é possível redimensionar imagens, por exemplo. Em seguida, temos o bloco de processamento, responsável por definir e extrair recursos relevantes dos dados para a construção dos modelos de aprendizado. Em caso de classificação de imagens, é nesse bloco que podemos determinar a profundidade de cor a ser usada nas imagens. Por fim, temos o bloco de aprendizado, onde podemos escolher um modelo de rede neural e realizar o treinamento. A Figura 4 apresenta a visão do usuário dentro de um projeto Edge Impulse na seção de treinamento do modelo de aprendizado de máquina.

O Edge Impulse oferece uma variedade de arquiteturas para treinamento, incluindo as *MobileNets* V1 e V2, que foram treinadas em conjuntos de dados ImageNet. Essas redes pré-treinadas vêm com uma gama de blocos de entrada, variando de 96x96 a 320x320.

Na etapa de teste (*Model Test*) é avaliado o desempenho do modelo através de testes com os dados não vistos. A plataforma Edge Impulse oferece diversas opções para realizar a implementação do modelo no dispositivo, entre elas, a utilização da biblioteca do Arduino e a criação de um código binário para implementação direta. o Edge Impulse fornece uma biblioteca C/C++ flexível e portátil que engloba o código de pré-processamento e o modelo de aprendizado de máquina treinado. Isso simplifica o processo de implementação, permitindo que os desenvolvedores integrem os modelos facilmente em diversos dispositivos. Além disso, o Edge Impulse oferece uma série de otimizações que ajudam a reduzir o tempo de inferência e o consumo de memória nos dispositivos, tornando-o ideal para dispositivos com recursos limitados [30].

III. Metodologia

Nesta seção, serão apresentadas as etapas realizadas para a construção e escolha dos modelos na plataforma Edge Impulse, as etapas são: (1) escolha da placa de desenvolvimento para a implantação do modelo de aprendizado de máquina, (2) seleção do conjunto de dados, (3) experimentos na plataforma Edge Impulse, (4) comparação entre os modelos, e por fim a (5) implantação dos modelos no dispositivo embarcado.

A. Dispositivo Embarcado

O dispositivo escolhido para a implantação do modelo de aprendizado de máquina foi o ESP32-CAM. Sua escolha foi baseada em seu poder de processamento, armazenamento e baixo custo. O dispositivo possui um processador dual-core de 32 bits, com frequência de trabalho de até 240MHz. Além disso, conta com uma memória RAM de 520 Kb e uma memória Flash de 4 Mb [31]. O dispostivo pode ser visto na Figura 5.

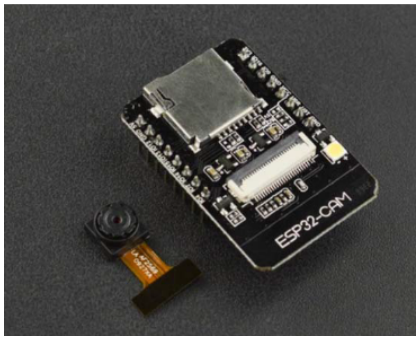

Fig. 5. Plataforma de Desenvolvimento ESP32- Cam. Fonte: [31]

O ESP32-CAM oferece suporte para as câmeras OV2640, que captura imagens com resolução de até 1600 x 1200 pixels, e OV7670, com resolução de 640 x 480 pixels. No presente projeto, foi utilizado o modelo OV2640. Além de suas especificações técnicas, o ESP32-CAM possui compatibilidade com a plataforma Edge Impulse e a existência de projetos prévios [13], [32] demonstra a viabilidade e eficácia da implementação de modelos de aprendizado de máquina nesse dispositivo.

B. Base de Dados

A escolha de uma base de dados é uma etapa importante na implementação de um modelo de aprendizado de máquina, pois é a partir dos dados que o modelo será treinado e validado. Dessa forma, o conjunto de imagens utilizado para a realização do projeto, foi obtido através da base de dados "*Concrete Crack Images for Classification*" [33], que já foi usada em diversos trabalhos e pesquisas na área [34]–[36]. Esse dataset possui 40 mil imagens de 227x227 pixels, sendo 20 mil referentes a concretos sem rachaduras (Negative) e as outras 20 mil, concretos com rachaduras (Positive). Esse conjunto de dados etá disponível para *download* no repositório *Mendeley*[1]. As Figuras 6 e 7 mostram exemplos de imagens do dataset utilizado.

[1]https://data.mendeley.com/datasets/5y9wdsg2zt/2

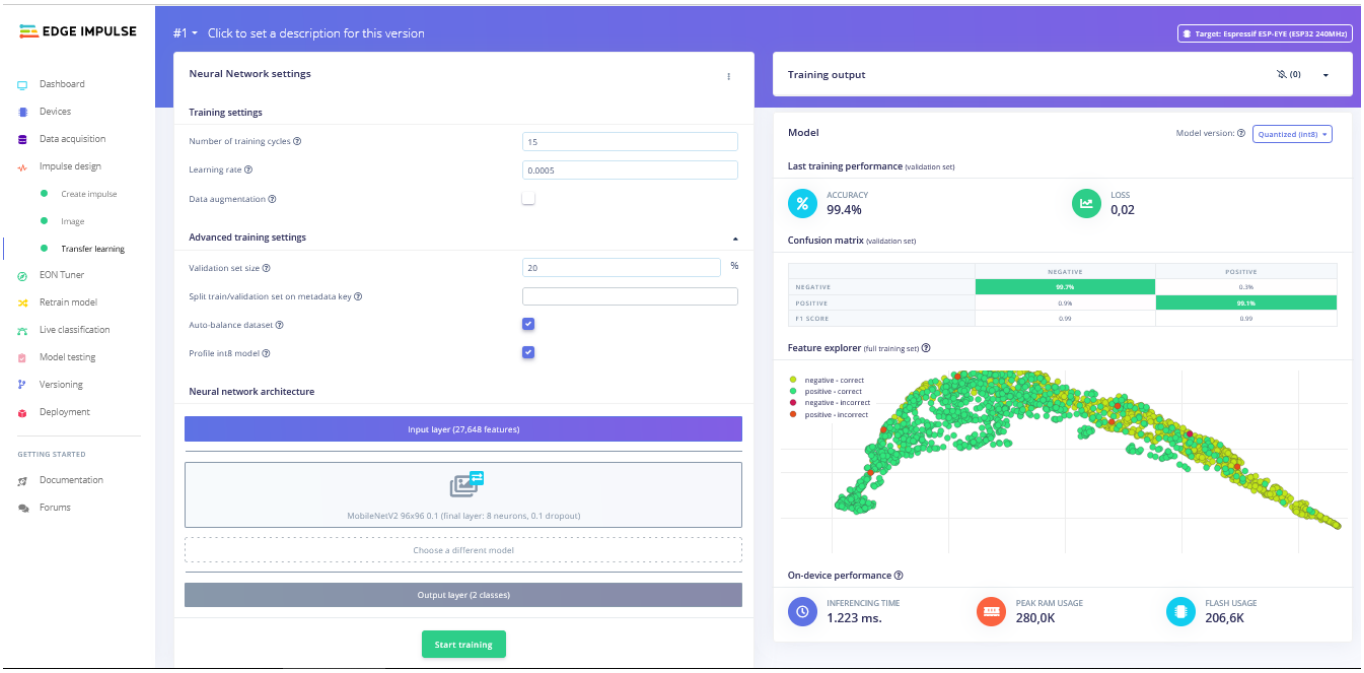

Fig. 4. Captura de tela mostrando a visão do usuário dentro de um projeto Edge Impulse na seção de treinamento do modelo.

Fig. 6. Concreto sem rachadura. Fonte: [33]

Fig. 7. Concreto com rachadura. Fonte: [33]

C. Experimentos na Plataforma Edge Impulse

O desenvolvimento e implantação dos modelos de aprendizado de máquina foram realizados no "*Edge Impulse Studio*" da plataforma Edge Impulse, a Figura 8 mostra um diagrama de blocos das etapas realizadas.

Fig. 8. Fluxo de desenvolvimento e implantação do modelo de classifiação .

1) Coleta de dados: A seleção da quantidade de dados utilizada para treinar o modelo de classificação no Edge Impulse foi determinada considerando as restrições da versão gratuita da plataforma. Devido aos limites de tempo de trabalho, estabelecidos em 20 minutos, e ao limite de quantidade de dados permitidos nessa versão, foi necessário fazer uma seleção dos dados disponíveis para garantir a criação de um modelo eficaz dentro dessas limitações.

A paitir disso, no processo de coleta de dados, foram selecionadas 4 mil imagens do conjunto de dados original. Essas imagens consistem nas primeiras 2.000 do grupo "*Positive*" e nas primeiras 2.000 do grupo "*Negative*". Os rótulos escolhidos para os dados foram mantidos os mesmos da base de dados, ou seja, "*Positive*" para imagens com rachaduras e "*Negative*" para imagens sem rachadura. Além disso, os dados foram divididos em 20% para teste e 80% para treinamento. Todo esse processo foi realizado na seção "*Data Acquisition*".

2) Pré-Processamento de dados: O pré-processamento de dados foi realizado na seção "*Impulse Design*", nos blocos de entrada e de processamento. Nessa etapa, as imagens de entrada foram redimensionadas com base na arquitetura escolhida para realizar os testes, no qual a dimensões passaram de 227x227 pixels para 96x96 pixels. Também foi selecionada a profundidade de cor para cada modelo. Para cada um deles, foram escolhidas tanto a opção de profundidade de cor em escala de cinza ("*grayscale*") quanto em RGB.

3) Treinamento: Na seção "*Impulse Design*", no bloco de aprendizado, foi selecionada a utilização da técnica de transferência de aprendizado (*Tranfer Learning*), esse paradgma possui a capacidade de reutilizar modelos já existentes como ponto de partida para a criação de novos modelos [37]. Essa abordagem oferece a vantagem de economia de tempo e de recursos necessarios para o treinamento.

Nessa etapa também foram realizados os treinamentos dos modelos de classificação utilizando tranferência de aprendizado com a arquitetura *MobileNet* nas versões 96x96 V1 e 96x96 V2, incluindo suas respectivas atualizações (V1: 0.1, 0.2, 0.25; V2: 0.05, 0.1, 0.35). O treinamento dos modelos contou com a configuração de alguns hiperparâmetros com valores fixos. Essas configurações podem ser visualizadas na Tabela I.

TABLE I
TABELA COM CONFIGURAÇÕES E VALORES FIXOS DO TREINAMENTO DO MODELO NO EDGE IMPULSE

Hiperparâmetros e Configurações	Ação/Quantidade
Número de ciclos de treinameto	15
Taxa de aprendizagem	0,0005
Aumento de dados	Não Selecionado
Tamanho do conjunto de validação	20%
Balanceamento automático do conjunto de dados	Selecionado
Perfil de modelo int8	Selecionado
Camada de saída (2 classes)l	(Positive ou Negative)

4) Testes: Para validar o desepenho dos modelos, os testes foram realizados na seção "*Model Testing*". Para isso, foram utilizados 20% dos dados, um total de 800 imagens que não eram conhecidas pelo modelo.

5) Implantação do modelo: A etapa de implantação do do modelo no dispositivo embarcado, foi realizado na seção "*Deployment*", optou-se pela implementação dos modelos por meio da plataforma Arduino e da biblioteca gerada pelo Edge Impulse.

D. Análise dos Modelos de Aprendizado Profundo

Após o treinamento de cada modelo, foi realizada a comparação entre eles. Para isso, foi proposta uma análise rankings dos modelos, inspirada em outros estudos da literatura que realizam ranqueamento de hiperparâmetros, como descrito em [38], [39].

Os modelos foram ranqueados de acordo alguns requesitos, como maior acurácia dos dados de teste, menor consumo de memórias RAM e Flash, e menor tempo de inferência. Para realizar o ranqueamento, foi feita a soma das posições dos modelos em cada requisito. Por exemplo, o Modelo 1 apresenta a menor acurácia, ocupando a posição 12 no ranking de acurácia, sendo a última posição nesse critério. No quesito de tempo de inferência, o modelo obteve a segunda menor pontuação, ocupando a posição 2. Em relação à memória FLASH e RAM, o modelo obteve o menor valor, ocupando a posição 1. Dessa forma, a soma das posições do Modelo 1 é igual a 16. Os modelos que obtiverem os menores valores são considerados os mais desejáveis. Vale ressaltar que todos os requisitos possuem o mesmo grau de relevância. A Tabela II mostra um exemplo do ranqueamento de dois modelos.

TABLE II
EXEMPLO DE RANQUEAMENTO

Modelo	Acurácia	Tempo Inferência	RAM	Flash	Ranking Final
1	12	2	1	1	16
2	1	3	4	10	18

IV. RESULTADOS E DISCUSSÕES

Nesta seção, apresentaremos e discutiremos os resultados e a seleção dos modelos para a implementação no sistema embarcado.

A. Acurácia dos modelos

O gráfico da Figura 9 apresenta a acurácia dos modelos com os dados de validação.

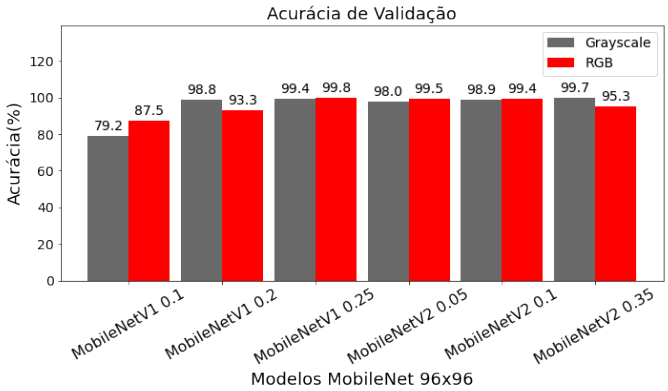

Fig. 9. Resultados da acurácia com os dados de validação.

Nota-se que o modelo *MobileNet V1 0.25*(RGB) obteve o melhor desempenho nos dados de validação, com uma acurácia de 99.8%. Por outro lado o modelo treinado com a arquitetura *MobileNet V1 0.1*(Grayscale) apresentou o pior desempenho nessa etapa, alcançando 79.2%.

O gráfico da Figura 10 exibe os resutados da acurácia com os dados de teste.

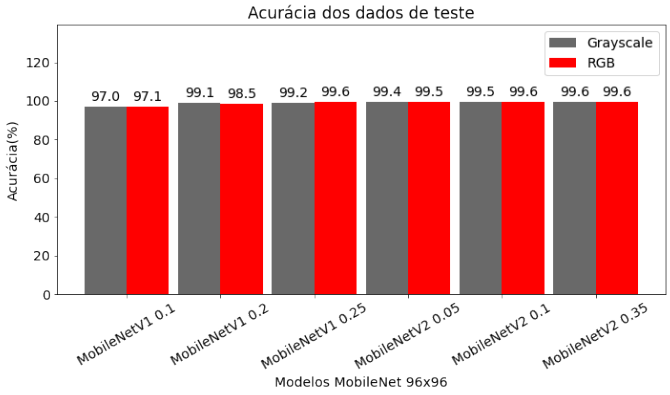

Fig. 10. Resultados da curácia com os dados de teste.

Ao analisar os resultados, nota-se que todos os modelos obtiveram um bom desempenho na tarefa de classificação com os dados de teste, alcançando valores de acurácia superiores a 97%. Destacam-se os modelos *MobileNet V1* 0.25 (RGB), *MobileNet V2* 0.1 (RGB), *MobileNet V2* 0.35 (GrayScale) e *MobileNet V2* 0.35 (RGB), que alcançaram o melhor desempenho em acurácia, com um valor de 99.6%.

B. Uso da memória RAM

Com relação ao uso da memóra RAM necessária para a execução do modelo, o gráfico da Figura 11 apresenta os resultados obtidos.

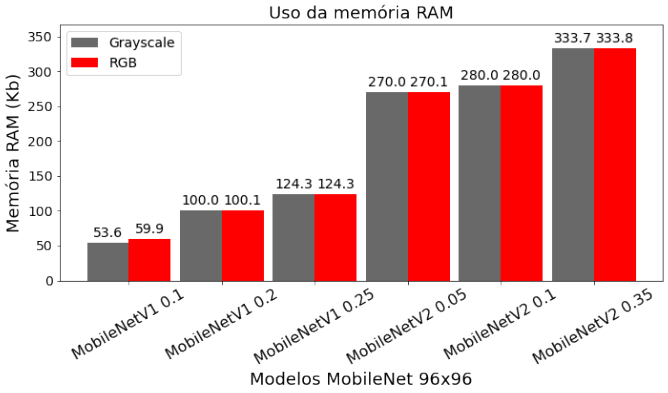

Fig. 11. Resultados da quantidade de memória RAM necessária para a utilização dos modelos.

Ao examinar os dados, observa-se que os modelos do *MobileNet V1* requerem menos de 125 Kb de memória RAM, enquanto as versões do *MobileNet V2* exigem mais de 270 Kb. O modelo que necessita de menos memória RAM para a execução, é o *MobileNet V1* 0.1 (Grayscale), precisando de 53.6 Kb. Por outro lado, o modelo que exige uma quantidade maior de memória é o *MobileNet V2* 0.35, com 333 Kb.

Devido à arquitetura mais complexa e com um maior número de camadas do modelo *MobileNet V2* em comparação com o *MobileNet V1*, é certo que isso teve um impactado na quantidade de memória RAM exigida para executar o modelo.

C. Uso da memória Flash

O gráfico da Figura12 mostra os resultados da memória flash necessaria para armazenar cada modelo no dispositivo embarcado.

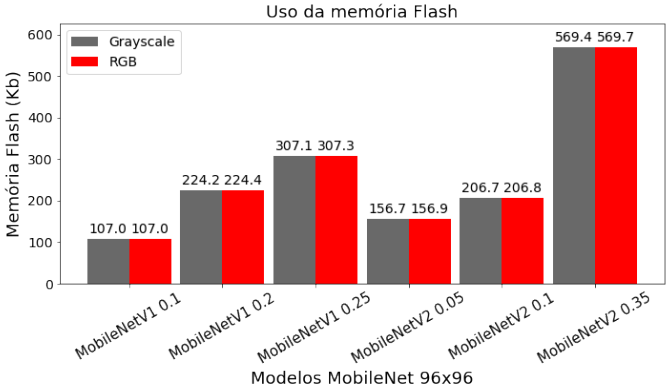

Fig. 12. Resultados da quantidade de memória Flash necessária para a implementação dos modelos.

Através dos resultados, foi possível observar que o modelo *MobileNet V1* 0.1 necessita de menos espaço em memória para ser armazenado, precisando de 107 Kb de memória Flash. Por outro lado, o modelo que exigiu um maior armazenamento em memória foi o *MobileNet V2* 0.35, com um valor de 569 Kb. Além disso, foi possível observar que os modelos *MobileNet V2* 0.1 e *MobileNet V2* 0.05 requerem menos memória Flash do que os modelos *MobileNet V1* 0.2 e *MobileNet V1* 0.25.

D. Tempo de inferência do modelo

Com relação ao tempo de inferência dos modelos, o gráfico da Figura 13 mostra os resultados de cada modelo.

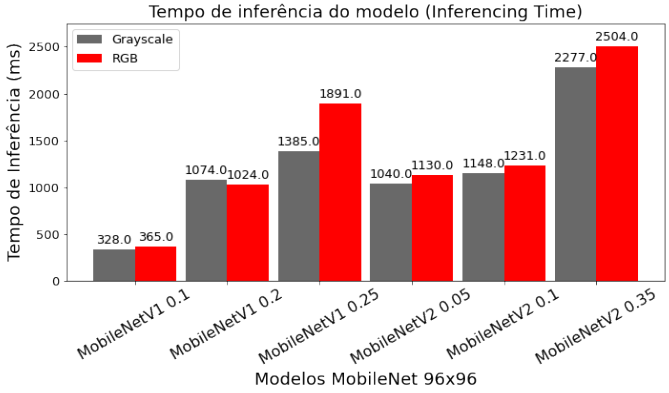

Fig. 13. Tempo de Inferência do modelo

Ao avaliar os dados, percebe-se que o tempo necessário para realizar a classificação do modelo no dispositivo varia de acordo com a escala de cor e os modelos. O modelo que obteve o menor tempo de execução da classificação foi o *MobileNet V1* 0.1 (Grayscale), com um valor de 328 ms. Por outro lado, o maior tempo de execução foi de 2504 ms, correspondente ao modelo *MobileNet V2* 0.35 (RGB).

E. Ranqueamento e escolha dos modelos

O resultado do ranqueamento dos modelos está apresentado na Tabela III.

TABLE III
TABELA DE RANQUEAMENTO DOS MODELOS

MobileNet	Escala cor	Acurácia	Tempo Inferência	RAM	Flash	Total
V1 0.1	Grayscale	12	1	1	1	15
V1 0.1	RGB	11	2	2	1	16
V2 0.05	Grayscale	7	4	7	3	21
V2 0.05	RGB	5	6	8	4	23
V2 0.1	RGB	1	8	9	6	24
V1 0.2	Grayscale	9	5	3	7	24
V1 0.2	RGB	10	3	4	8	25
V1 0.25	RGB	1	10	5	10	26
V2 0.1	Grayscale	5	7	9	5	26
V1 0.25	Grayscale	8	9	5	9	31
V2 0.35	Grayscale	1	11	11	11	34
V2 0.35	RGB	1	12	12	12	37

Atavés do ranqueamento, os 6 modelos candidatos para implantação no dipositivo embarcado foram os modelos:

1) MobileNetV1 96x96 0.1 (Grayscale)
2) MobileNetV1 96x96 0.1 (RGB)
3) MobileNetV2 96x96 0.05 (Grayscale)
4) MobileNetV2 96x96 0.05 (RGB)
5) MobileNetV1 96x96 0.2 (Grayscale)
6) MobileNetV2 96x96 0.1 (RGB)

Esse resultado evidência que a escolha de um modelo de aprendizado de máquina para implementação em um dispositivo embarcado requer que seja considerado diversos fatores, além da acurácia do modelo .Pode-se observar que o modelo *MobileNet V1* 0.1 ele não possuiu a maior acurácia, com o valor de 97%, mas ele é um candidato a ser implementado no dispositivo. Embora a acurácia seja uma métrica importante, dependendo das necessidades do projeto, outros aspectos críticos podem ter um impacto significativo no desempenho do modelo no sistema embarcado. Entre eles, destacam-se a utilização das memórias RAM e Flash pelo modelo e o tempo de inferência necessário para a execução do modelo.

Esses parâmetros devem ser cuidadosamente avaliados, especialmente quando o dispositivo embarcado possui recursos bastante limitados e requer uma execução rápida do processo de classificação. Portanto, é fundamental considerar não apenas a acurácia, mas também as restrições de memória, a capacidade de armazenamento em Flash e o tempo de inferência ao selecionar um modelo de aprendizado de máquina.

F. Implantação e execução do modelo no dispositivo embarcado

Com relação a implantação dos modelos selecionados no dispositivo ESP-32 CAM, a Figura 14 apresenta saída do monitor serial do Arduino com o resultado do teste de classificação de um modelo.

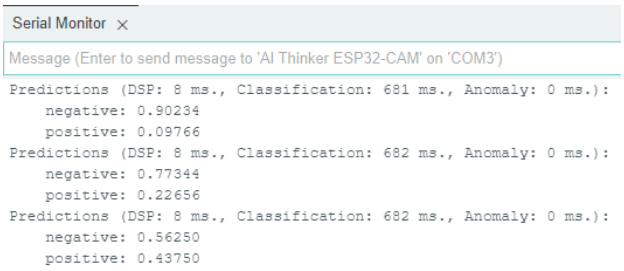

Fig. 14. Saída do resultado de inferência do modelo na plataforma Arduino.

Todos os 6 modelos selecionados foram submetidos a testes de implantação, e foram executados com sucesso no dispositivo embarcado.

V. Conclusão

O objetivo principal do estudo foi selecionar modelos de aprendizado de máquina para a detecção de rachaduras em estruturas de concreto, visando avaliar sua viabilidade de implementação no dispositivo embarcado ESP32-CAM. Durante a seleção dos modelos, foram considerados critérios como menor consumo de memórias RAM e Flash, maior acurácia e menor tempo de inferência. Além disso, os testes foram realizados para cada modelo e sua versão, tanto utilizando imagens em RGB quanto em Grayscale.

Os resultados obtidos mostram que a mudança de escala de cor não alterou significativamente os resultados nos testes com o mesmo modelo, em relação à acurácia, uso de memória Flash e RAM. No entanto, no que se refere ao tempo de inferência, o uso de imagens em RGB aumentou o tempo de inferência do modelo. Além disso, os modelos *MobileNet V2* requerem um maior consumo de memória RAM em comparação com os modelos *MobileNet V1*.O menor valor memória RAM necessária para a execução do *MobileNet V2* é de 270 Kb e o maior valor do *MobileNet V1* é de 124.3 Kb. Com relação ao ranqueamento o modelo que obteve o melhor desempenho levando em consideração aos critérios desejados, foi o modelo *MobileNet V2* 0.1, com acurácia do modelo de 97%, memória Flash necessária de 107 Kb, memória RAM de 53.6 Kb, e tempo de inferência de 328 ms.

Para trabalhos futuros, recomenda-se a exploração de otimizações adicionais nos modelos selecionados, por exemplo, a investigação de diferentes combinações de hiperparâmetros. Além disso, é importante conduzir testes em ambientes reais, com o intuito de avaliar a eficácia dos modelos em cenários práticos. Através da experimentação em ambientes reais, será possível obter informações valiosas sobre a capacidade dos modelos em lidar com situações práticas, contribuindo para o aprimoramento de sua eficiência e confiabilidade, além de impulsionar o avanço do conhecimento na área específica da construção civil.

VI. Agradecimentos

Os autores agradecem ao apoio da Universidade Federal do Recôncavo da Bahia.

References

[1] C. M. B. da Cruz and A. R. da Silva, "Produção científica relacionada à inteligência artificial no brasil," *Revista Expressão Científica (REC)*, vol. 5, no. 3, pp. 81–88, 2020.

[2] D. C. S. G. Bruno Oliveira, João Moreira and F. G. aes., "Uma abordagem de classificação de ovos de parasitos intestinais humanos em imagens microscópicas usando redes neurais convolucionais," in *Anais do 15 Congresso Brasileiro de Inteligência Computacional*, C. J. A. B. Filho, H. V. Siqueira, D. D. Ferreira, D. W. Bertol, and R. C. L. ao de Oliveira, Eds. Joinville, SC: SBIC, 2021, pp. 1–8.

[3] R. R. Ely, C. A. Künas, L. P. Heck, and E. L. Padoin, "Ia aplicada no reconhecimento facial," *Salão do Conhecimento*, vol. 6, no. 6, 2020.

[4] J. C. F. Igor R. Sousa and G. A. Barreto., "Revisitando o uso da transformada de fourier no reconhecimento de voz para robótica móvel," in *Anais do 15 Congresso Brasileiro de Inteligência Computacional*, C. J. A. B. Filho, H. V. Siqueira, D. D. Ferreira, D. W. Bertol, and R. C. L. ao de Oliveira, Eds. Joinville, SC: SBIC, 2021, pp. 1–8.

[5] W. R. da Silva, J. C. de Almeida, E. F. de Oliveira, E. R. Neto, C. A. Ynoguti, and J. P. Henriques, "Manufatura avançada para produçao agrıcola usando inteligência artificial e iot," 2019.

[6] G. B. Heerdt, V. M. Pio, and N. C. T. Bleichvel, "Principais patologias na construção civil," *Trabalho de Graduação, Bacharelado em Engenharia Civil-Faculdade Metropolitana de Rio do Sul–UNIASSELVI/FAMESUL, Rio do Sul*, 2016.

[7] Y. Gao and K. M. Mosalam, "Deep transfer learning for image-based structural damage recognition," *Computer-Aided Civil and Infrastructure Engineering*, vol. 33, no. 9, pp. 748–768, 2018.

[8] L. Ali, F. Alnajjar, H. A. Jassmi, M. Gocho, W. Khan, and M. A. Serhani, "Performance evaluation of deep cnn-based crack detection and localization techniques for concrete structures," *Sensors*, vol. 21, no. 5, p. 1688, 2021.

[9] X. Yang, H. Li, Y. Yu, X. Luo, T. Huang, and X. Yang, "Automatic pixel-level crack detection and measurement using fully convolutional network," *Computer-Aided Civil and Infrastructure Engineering*, vol. 33, no. 12, pp. 1090–1109, 2018.

[10] A. L. C. Ottoni, "Métodos para recomendação de hiperparâmetros de aprendizado de máquina na classificação de imagens da construção civil," Ph.D. dissertation, Universidade Federal da Bahia, 2022.

[11] P. P. Ray, "A review on tinyml: State-of-the-art and prospects," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 4, pp. 1595–1623, 2022.

[12] L. Falaschetti, M. Beccerica, G. Biagetti, P. Crippa, M. Alessandrini, and C. Turchetti, "A lightweight cnn-based vision system for concrete crack detection on a low-power embedded microcontroller platform," *Procedia Computer Science*, vol. 207, pp. 3948–3956, 2022.

[13] X. Zhou, Z. Kang, R. Canady, S. Bao, D. A. Balasubramanian, and A. Gokhale, "Exploring cloud assisted tiny machine learning application patterns for phm scenarios," in *Annual Conference of the PHM Society*, vol. 13, no. 1, 2021.

[14] G. M. Iodice, *TinyML Cookbook*. Packt Publishing Pvt Ltd, 2022.

[15] D. S. Pete Warden, *TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers*, 1st ed. O'Reilly Media, 2019.

[16] M. Antonini, M. Pincheira, M. Vecchio, and F. Antonelli, "An adaptable and unsupervised tinyml anomaly detection system for extreme industrial environments," *Sensors*, vol. 23, no. 4, p. 2344, 2023.

[17] M. O.-E. Aoueileyine, "Tiny machine learning for iot and ehealth applications: Epileptic seizure prediction use case," in *Digital Technologies and Applications: Proceedings of ICDTA'23, Fez, Morocco, Volume 2*. Springer, 2023, pp. 242–251.

[18] A. Ukil, I. Sahu, A. Majumdar, S. C. Racha, G. Kulkarni, A. D. Choudhury, S. Khandelwal, A. Ghose, and A. Pal, "Resource constrained cvd classification using single lead ecg on wearable and implantable devices," in *2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. IEEE, 2021, pp. 886–889.

[19] M. Z. M. Shamim, "Hardware deployable edge-ai solution for prescreening of oral tongue lesions using tinyml on embedded devices," *IEEE Embedded Systems Letters*, vol. 14, no. 4, pp. 183–186, 2022.

[20] "Limitaccess: reconhecimento de fala robusto baseado em tinyml no dispositivo e classificação de idade," vol. 3.

[21] M. e. P. M. e. S. S. Wong, Alexander e Famouri, "Tinyspeech: Condensadores de atenção para redes neurais de reconhecimento de fala profunda em dispositivos de ponta," *arXiv preprint arXiv:2008.04245*.

[22] G. Edel and V. Kapustin, "Exploring of the mobilenet v1 and mobilenet v2 models on nvidia jetson nano microcomputer," in *Journal of Physics: Conference Series*, vol. 2291, no. 1. IOP Publishing, 2022, p. 012008.

[23] G. Yu, L. Wang, M. Hou, Y. Liang, and T. He, "An adaptive dead fish detection approach using ssd-mobilenet," in *2020 Chinese Automation Congress (CAC)*. IEEE, 2020, pp. 1973–1979.

[24] A. S. d. Macena, "Rastreamento de múltiplos objetos utilizando modelos de aprendizado profundo em hardware limitado," 2021, trabalho de Conclusão de Curso (Bacharel em Ciência da Computação).

[25] J. R. N. Sousa, "Deteção de ações a partir da pose corporal dentro de veículos autónomos partilhados," Ph.D. dissertation, 2021.

[26] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[27] I. N. Mihigo, M. Zennaro, A. Uwitonze, J. Rwigema, and M. Rovai, "On-device iot-based predictive maintenance analytics model: Comparing tinylstm and tinymodel from edge impulse," *Sensors*, vol. 22, no. 14, p. 5174, 2022.

[28] E. IMPULSE, "Data acquisition," Disponível em: https://docs.edgeimpulse.com/docs/edge-impulse-studio/data-acquisition, acesso em: 20/05/2023.

[29] ——, "Impulse design," Disponível em: https://docs.edgeimpulse.com/docs/edge-impulse-studio/impulse-design, acesso em: 20/05/2023.

[30] S. Hymel, C. Banbury, D. Situnayake, A. Elium, C. Ward, M. Kelcey, M. Baaijens, M. Majchrzycki, J. Plunkett, D. Tischler *et al.*, "Edge impulse: An mlops platform for tiny machine learning," *arXiv preprint arXiv:2212.03332*, 2022.

[31] DFROBOT, "ESP32-CAM Development Board Datasheet," PDF, dfrobot, 2019. [Online]. Available: https://tinyurl.com/27ck44ca

[32] L. S. Gid, J. P. Ganhor, and O. M. de Souza, "Lowcost siv-sistema inteligente de vigilância de baixo custo desenvolvido com esp32," in *Anais do XVIII Congresso Latino-Americano de Software Livre e Tecnologias Abertas*. SBC, 2021, pp. 142–145.

[33] F. Özgenel and A. Gönenç Sorguç, "Performance comparison of pretrained convolutional neural networks on crack detection in buildings," in *ISARC 2018*, Berlin, 2018.

[34] M. M. M. Islam and J.-M. Kim, "Vision-based autonomous crack detection of concrete structures using a fully convolutional encoder–decoder network," *Sensors*, vol. 19, no. 19, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/19/4251

[35] M. M. Islam, M. B. Hossain, M. N. Akhtar, M. A. Moni, and K. F. Hasan, "Cnn based on transfer learning models using data augmentation and transformation for detection of concrete crack," *Algorithms*, vol. 15, no. 8, 2022. [Online]. Available: https://www.mdpi.com/1999-4893/15/8/287

[36] H. Tang, J. Shi, X. Lu, Z. Yin, L. Huang, D. Jia, and N. Wang, "Comparison of convolutional sparse coding network and convolutional neural network for pavement crack classification: A validation study," in *Journal of Physics: Conference Series*, vol. 1682, no. 1. IOP Publishing, 2020, p. 012016.

[37] M. B. Azevedo, T. d. A. d. Medeiros, M. Medeiros, I. Silva, and D. G. Costa, "Detecting face masks through embedded machine learning algorithms: A transfer learning approach for affordable microcontrollers," *Available at SSRN 4404567*.

[38] F. Hutter, L. Kotthoff, and J. Vanschoren, *Automated machine learning: methods, systems, challenges*. Springer Nature, 2019.

[39] A. L. C. Ottoni, M. S. Novo, and D. B. Costa, "Hyperparameter tuning of convolutional neural networks for building construction image classification," *The Visual Computer*, vol. 39, no. 3, pp. 847–861, 2023.