Avaliação de Desempenho de Computadores Raspberry Pi com Algoritmos para o Reconhecimento Automático de Placas Veiculares

Felipe F. Soares
Programa de Pós-Graduação em Engenharia de Teleinformática (PPGETI)
Universidade Federal do Ceará (UFC)
Fortaleza, Brasil
felipefs@alu.ufc.br

Lucas de Sousa Fernandes
Departamento de Computação
Universidade Federal do Ceará (UFC)
Fortaleza, Brasil
lucasdesousafernandes@alu.ufc.br

Atslands R. da Rocha
Departamento de Engenharia de Teleinformática
Universidade Federal do Ceará (UFC)
Fortaleza, Brasil
atslands@ufc.br

Paulo A. L. Rego
Departamento de Computação
Universidade Federal do Ceará (UFC)
Fortaleza, Brasil
pauloalr@ufc.br

José G. R. Maia
Instituto UFC Virtual
Universidade Federal do Ceará (UFC)
Fortaleza, Brasil
gilvanm@ufc.br

José Neuman de Souza
Departamento de Computação
Universidade Federal do Ceará (UFC)
Fortaleza, Brasil
neuman@ufc.br

***Resumo*—Este trabalho apresenta uma avaliação de desempenho (acurácia, tempo de execução, uso de RAM e consumo de energia) de dois algoritmos para reconhecimento automático de placas (*Automatic License-Plate Recognition*), ALPR, em hardware de baixo custo comumente usado em ambientes de Internet das Coisas, computadores Raspberry Pi, modelos 3B, 3B + e 4B. Os objetivos dessa análise são verificar qual algoritmo entrega o melhor custo-benefício aos dispositivos sobre as métricas avaliadas, se os computadores podem executar este tipo de aplicação e como o desempenho melhorou nos diferentes modelos. Os resultados mostram que todos esses dispositivos podem lidar bem com a aplicação, embora o processamento de vídeo em tempo real não seja viável. Para o conjunto de dados testado, o algoritmo mais leve, que dispensa uma das etapas realizada pelo outro, quando executado no Raspberry Pi 4 superou os demais em todos os aspectos.**

***Index Terms*—algoritmos de reconhecimento automático de placas veiculares, análise de desempenho, raspberry pi, visão computacional, cidades inteligentes**

I. Introdução

Os avanços na microeletrônica e nas redes de comunicação possibilitaram o surgimento de uma miríade de dispositivos com tamanho reduzido, capazes de se comunicarem uns com os outros e de serem controlados remotamente por meio de conexão com a Internet. Tais evoluções tecnológicas impulsionaram o advento de um conceito denominado Internet das Coisas (*Internet of Things* - IoT), em que os dispositivos, as "coisas", estão espalhados por determinado ambiente, gerando informação útil sobre eles e atuando quando necessário ou quando ordenado [1].

É neste cenário que diversas aplicações disruptivas têm surgido e transformado o cotidiano das pessoas. Algumas das diferentes áreas que mais têm se beneficiado desta tecnologia são a da saúde, a rural, a da infraestrutura urbana, a militar e a da segurança. Além disso, as aplicações de IoT variam desde as mais simples, como coletar a temperatura de um ambiente, às mais complexas que demandam uso intensivo de recursos, como as aplicações de Visão Computacional e *Deep Learning* que exigem uma alta utilização de CPU ou de GPU para processar imagens e vídeos.

Embora possuam a capacidade de contornar diferentes problemas práticos do mundo contemporâneo, grande parte das aplicações de IoT necessita da cooperação de diversos dispositivos. Diante disso, a viabilidade econômica desse tipo de aplicação exige que o custo desses equipamentos seja baixo, o que impõe uma série de restrições aos seus recursos, como baixa capacidade de processamento, quantidade de memória RAM e de armazenamento. Os ambientes IoT também apresentam consideráveis limitações energéticas, o que torna a eficiência uma obrigação para esses dispositivos, sendo um tema que atrai a atenção de muitos pesquisadores [2].

Assim, devido a toda essa diversidade de aplicações de IoT, existem hoje várias classes de dispositivos que podem ser utilizados por elas [3], como os sensores, os atuadores e os computadores de placa única (*Single Board Computers* - SBCs). Em geral, estes últimos são capazes de embarcar sistemas operacionais e fornecem uma interface mais amigável para usuários que não são técnicos. Além de englobarem características presentes nos dispositivos dos dois primeiros tipos, SBCs destacam-se também pelo desempenho computacional equiparável ao de alguns *smartphones*, sendo comumente utilizados nas tarefas computacionais mais complexas das aplicações *IoT* [4]. Há diferentes fabricantes e modelos de SBCs no mercado, entre eles as placas Raspberry Pi, desenvolvidas pela Raspberry Pi Foundation, cujo custo-benefício tem tornado esses computadores populares no universo da IoT.

Este trabalho tem como objetivo principal avaliar o desempenho de três diferentes modelos de computadores *Raspberry Pi* na execução de uma aplicação que envolve Visão Computacional e *Deep Learning*, ou seja, complexa para os dispositivos IoT. Trata-se de um sistema de reconhecimento automático de placas veiculares (*Automatic License-Plate Recognition* - ALPR), que consiste em identificar placas de automóveis e fazer a leitura delas a partir de uma imagem submetida ao sistema. Neste trabalho são utilizados dois algoritmos de ALPR com abordagens e níveis de complexidade computacional diferentes. Por meio dos valores de *acurácia*, *tempo de execução*, *uso de RAM* e do *consumo energético*; espera-se determinar a viabilidade de se executar esses algoritmos nos dispositivos, bem como determinar a melhor combinação de algoritmo-dispositivo e verificar o desempenho da plataforma *Raspberry Pi* ao longo de diferentes modelos.

II. Reconhecimento automático de placas veículares

Um sistema de Reconhecimento Automático de Placas Veiculares (ALPR) tem como propósito identificar e ler cada placa veicular presente em uma imagem, contendo um ou mais veículos. Estes sistemas estão presentes em um número importante de aplicações, como monitoramento de trânsito, coleta automática de pedágio, fiscalização e aplicação de leis de trânsito, controle de acesso de estacionamento, dentre outras [5]. O Governo do Estado do Ceará, por exemplo, tem implementado um sistema de videomonitoramento para auxiliar os trabalhos na área de segurança pública, reduzindo o número de roubos de veículos por meio do uso de inteligência artificial [6].

Esse tipo de sistema precisa lidar com condições adversas das imagens recebidas, tais como iluminação variável, diferentes ângulos de captura, presença ou não de sombras e de borrões, entre outras características inerentes ao ambiente onde a câmera está instalada. Para superar esses problemas, são utilizados sistemas mais robustos embasados em técnicas com custo computacional elevado que exigem equipamentos mais poderosos e que requerem grandes bases de imagens, como é o caso da *Deep Learning* [7], [8].

Reconhecendo a revolução que a *Deep Learning* tem trazido ao campo da Inteligência Artificial, em especial ao da Visão Computacional, decidiu-se utilizar neste trabalho algoritmos compostos por técnicas desse tipo, de modo que o desafio está em obter um equilíbrio entre a performance e a acurácia do sistema. Tipicamente, algoritmos de ALPR (*Algoritmo 1*) submetem as imagens recebidas em três etapas: detecção de veículo na imagem de entrada, detecção de placa nos recortes de veículos encontrados e leitura da placa. Como alternativa para reduzir o tempo de processamento, um algoritmo mais simples (*Algoritmo 2*), pode ser utilizado, em que a detecção da placa é realizada diretamente na imagem de entrada, seguindo para a leitura da mesma com uma etapa a menos. Dependendo da imagem recebida, essa última abordagem pode comprometer a acurácia do sistema, visto que a busca pela placa veicular será realizada na imagem inteira, não se restringindo ao recorte do veículo como no *Algoritmo 1*. O fluxograma das etapas do *Algoritmo 1*, abordagem completa, e do *Algoritmo 2*, abordagem simples, encontram-se na Figura 1.

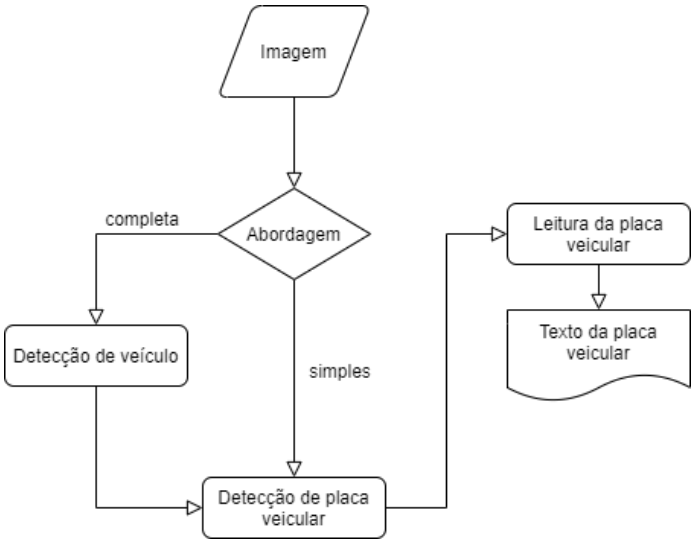

Figura 1. Fluxograma das etapas de um sistema ALPR com a possibilidade de dois algoritmos: *Algoritmo 1* contendo a etapa de detecção veicular e o *Algoritmo 2* sem ela.

Várias técnicas foram aplicadas na implementação dos algoritmos utilizados neste trabalho. Para a etapa de detecção veicular, presente apenas no *Algoritmo 1*, a técnica utilizada é chamada de *Single Shot Multibox Detector*, ou SSD [9], com a arquitetura de rede neural MobileNet-v2 [10] como *backbone* e entrada de tamanho 300×300. Como carros são objetos comuns no pré-treinamento de vários métodos de detecção de objetos com *Deep Learning* e já apresentam uma boa acurácia de identificação, foram aproveitados os pesos já treinados na base de imagens COCO [11]. Esse estágio recebe de entrada uma imagem e tem como saída as coordenadas de retângulos contendo os veículos detectados.

A seguir, um modelo baseado na rede neural Tiny YOLO-v3 [12] é utilizado para detectar a placa veicular na imagem do veículo resultante da etapa anterior ou na imagem original, a depender do algoritmo. Esse modelo possui entrada de tamanho 320×320 e pesos treinados a partir de imagens de uma base privada de carros em rodovias fornecida pela Secretaria da Segurança Pública e Defesa Social do Estado do Ceará. A rede foi modificada para sua saída ser apenas uma classe (placa), reduzindo sua complexidade em relação à original e mantendo robustez adequada. A saída dessa etapa é composta de coordenadas de placas veiculares contidas na imagem. Mais detalhes sobre essa rede podem ser vistos na Tabela I.

Tabela I
REDE DE DETECÇÃO DE PLACAS VEICULARES: UMA MODIFICAÇÃO DA TINY YOLOV3 PARA UMA ÚNICA CLASSE DE SAÍDA

	Camada	Filtros	Tamanho	Tamanho de Entada	Tamanho de Saída
0	conv	16	3 x 3 / 1	320 x 320 x 3	320 x 320 x 16
1	max		2 x 2 / 2	320 x 320 x 16	160 x 160 x 16
2	conv	32	3 x 3 / 1	160 x 160 x 16	160 x 160 x 32
3	max		2 x 2 / 2	160 x 160 x 32	80 x 80 x 32
4	conv	64	3 x 3 / 1	80 x 80 x 32	80 x 80 x 64
5	max		2 x 2 / 2	80 x 80 x 64	40 x 40 x 64
6	conv	128	3 x 3 / 1	40 x 40 x 64	40 x 40 x 128
7	max		2 x 2 / 2	40 x 40 x 128	20 x 20 x 128
8	conv	256	3 x 3 / 1	20 x 20 x 128	20 x 20 x 256
9	max		2 x 2 / 2	20 x 20 x 256	10 x 10 x 256
10	conv	512	3 x 3 / 1	10 x 10 x 256	10 x 10 x 512
11	max		2 x 2 / 1	10 x 10 x 512	10 x 10 x 512
12	conv	1024	3 x 3 / 1	10 x 10 x 512	10 x 10 x1024
13	conv	256	1 x 1 / 1	10 x 10 x1024	10 x 10 x 256
14	conv	512	3 x 3 / 1	10 x 10 x 256	10 x 10 x 512
15	conv	18	1 x 1 / 1	10 x 10 x 512	10 x 10 x 18
16	yolo				
17	route	13			10 x 10 x 256
18	conv	128	1 x 1 / 1	10 x 10 x 256	10 x 10 x 128
19	upsample		2x	10 x 10 x 128	20 x 20 x 128
20	route	19 8			20 x 20 x 384
21	conv	256	3 x 3 / 1	20 x 20 x 384	20 x 20 x 256
22	conv	18	1 x 1 / 1	20 x 20 x 256	20 x 20 x 18
23	yolo				

Por fim, a partir do recorte da placa, a rede neural OCR-NET [13] é responsável por reconhecer os caracteres contidos na imagem utilizando pesos pré-treinados com imagens artificiais e heurísticas de correção de erros [8].

III. TRABALHOS RELACIONADOS

Os trabalhos relacionados foram divididos em duas subseções: uma sobre estudos que envolvem ALPR e a outra sobre a avaliação de desempenho de computadores *Raspberry Pi* em diferentes cenários.

A. ALPR

Em [13], os autores propõem um sistema ALPR de placas veiculares brasileiras utilizando redes neurais convolucionais. O sistema é divido em três etapas: detecção da parte frontal do veículo, detecção da placa veicular e leitura da placa. Os autores reportam uma acurácia de $63,18\%$ em uma base de imagens própria, demorando 13 ms em média (o equivalente a 76 *frames por segundo*, ou FPS) para executar em uma placa de vídeo NVIDIA TITAN X.

Em [8], os autores propõem uma nova rede baseada na OCR-NET para detectar e consertar o alinhamento de placas visando um sistema ALPR para cenários sem restrições, utilizando imagens de países variados. O sistema consegue $91,23\%$ e $88,56\%$ de acurácia para a base de imagens de placas brasileiras da *OpenALPR*, e para a base de imagens de [13], respectivamente. O trabalho também propõe uma base com placas de vários países, na qual o sistema obtém 75% de acurácia. Em média, o sistema leva 200 ms por imagem (5 FPS) em uma NVIDIA TITAN X PASCAL. É importante destacar que a OCR-NET é treinada com imagens de placas veiculares geradas artificialmente.

No trabalho [7], análogo a [13] e [8], o sistema proposto utiliza uma rede neural baseada na FAST-YOLO para detecção veicular e de placa e a OCR-NET para leitura da placa. Além disso, utilizam-se redundância temporal para melhorar os resultados das leituras e a base aberta UFPR-ALPR, contendo 4.500 imagens de carros, motos, caminhões e ônibus. A redundância temporal resulta em $78,33\%$ de acurácia nessa base, uma melhora significativa dos $64,89\%$ atingidos quando essa técnica não é utilizada. É relevante observar que essa base contém imagens de motos, o que possibilita o treinamento e teste com placas de tamanho diferente. Em média, o sistema processa cada imagem em 28,57 ms (35 FPS) em uma NVIDIA TITAN X PASCAL. Na base de [13], os resultados são de 93.53% e 85.45% com e sem redundância temporal, respectivamente, operando em 21,27 ms (47 FPS) em média na mesma GPU.

Tendo como objetivo um sistema de ALPR para ser utilizado em um ambiente embarcado, em [14] é proposto um sistema mais simplificado, apenas com uma fase de detecção de placas através da rede neural Tiny YOLO-v3 e outra de leitura dos caracteres através de uma rede neural com arquitetura adaptada originalmente proposta para leitura de *captchas*, com o treinamento realizado por meio de geração de imagens artificiais de placas veiculares. Os modelos foram refinados com uma base de 240 imagens e o sistema foi validado em um computador *Raspberry Pi 3*, alcançando $98,36\%$ de acurácia e demorando 2,7 s por imagem.

B. Avaliação de Desempenho de Computadores Raspberry Pi

Entre os trabalhos que focam em avaliação de desempenho de computadores *Raspberry Pi*, há uma grande diversidade de aplicações e análises realizadas nos mais diferentes cenários. No trabalho [15], são destacadas as dificuldades em portar aplicações de Visão Computacional para ambientes com restrições de recursos, como os de IoT. Os autores utilizam um *cluster* de 16 *Raspberry Pi modelo 2* para executar aplicações médicas de análise de imagens microscópicas e comparam com dois microcomputadores, com processadores *Core2Duo* e *i7*, comumente utilizados nesse tipo de aplicação. O *cluster* de *Raspberry* provou ser uma plataforma excelente e promissora para a aplicação proposta, sendo mais rápido e economizando mais energia que os micro-computadores.

Em [16], os autores propõem um detector, utilizando *Deep Learning*, idealizado para rodar em dispositivos com limitação de recursos. Ele consiste em identificar perigos em ambientes de supermercados, como objetos no piso, a partir de uma imagem. O trabalho comparou diversas arquiteturas de redes neurais e computadores, sendo eles um *desktop*, um *Coral Dev Board* e um *Raspberry Pi*. Embora este último tenha sido o mais lento e o que consumiu mais memória, ainda assim conseguiu executar o detector de maneira excelente, demorando cerca de 0,506 segundos para classificar a ausência ou não de perigos em uma imagem e utilizando pouco mais de 5 MB de RAM.

Em [17], os autores utilizam um *Intel Movidius*™ acoplado a um *Raspberry Pi modelo 3* para identificar objetos em imagens e vídeos. É destacado também o dilema entre a acurácia e a latência de aplicações *Deep Learning*, uma vez que o acréscimo de camadas pode melhorar a classificação feita pelo sistema, mas o tornará mais lento. Os resultados

do trabalho mostram que a quantidade de *frames por segundo* do computador, quando utilizando o *Movidius*, chega a 70, enquanto sem o dispositivo alcança 11 FPS.

Embora muitos trabalhos avaliem o desempenho dos dispositivos *Raspberry Pi* sob diversos aspectos e, até mesmo, comparam eles com outros computadores, há uma carência de avaliação da performance dos diferentes modelos da plataforma, uma vez que tal estudo pode fornecer informações importantes para o planejamento de aplicações IoT e para a redução de custos de projeto. Este trabalho faz essa avaliação, de modo a analisar a viabilidade de se executar sistemas de ALPR na plataforma *Raspberry Pi* e como ela se comporta com esse nicho de aplicação.

IV. Experimentos

Nesta seção, são descritos o design experimental e os materiais utilizados, cujo objetivo é avaliar o desempenho de diferentes modelos de computadores *Raspberry Pi* na execução dos dois algoritmos de ALPR descritos na Seção II. Tais algoritmos possuem níveis de complexidade distintos, tendo em vista que uma das três etapas do *Algoritmo 1* não é aplicada no *Algoritmo 2*.

O sistema de ALPR foi escolhido para os experimentos devido à demanda que aplicações de Visão Computacional e de *Deep Learning* utilizadas em ambientes IoT têm tido nos últimos anos, especialmente por governos e entidades de segurança, que podem se beneficiar do monitoramento e do rastreamento que esse sistema oferece. Além disso, a complexidade dos algoritmos é um fator importante, pois possibilita que este trabalho sirva como *benchmark* para esse nicho de aplicações em computadores *Raspberry Pi*.

Para avaliar o desempenho dos computadores *Raspberry Pi* na execução dos dois algoritmos de ALPR, realizaram-se experimentos que consistiram na seguinte sequência de passos:

1) Inicializar as redes neurais com seus pesos de treinamento nos dispositivos;
2) Carregar o dataset OpenALPR que possui as imagens a serem processadas;
3) Submeter, em sequência, cada uma das imagens ao sistema ALPR, cujas etapas variam conforme o algoritmo utilizado;
4) Computar e apresentar as métricas de resposta da avaliação.

As especificações dos dispositivos *Raspberry Pi* utilizados nos experimentos são apresentadas na Tabela II e o detalhamento desses experimentos encontra-se na Tabela III.

Ressalta-se a utilização da mesma fonte de alimentação e do mesmo cartão de memória em todos os computadores. Dessa forma, os experimentos ficam isolados de ruídos ocasionados por diferenças nas versões do sistema operacional ou nos processos em execução. Sendo assim, as variações nos resultados de cada métrica estão inteiramente relacionadas com componentes do *hardware*.

Os algoritmos foram implementados utilizando a linguagem *Python*, versão 3.6, e com o auxílio da biblioteca *OpenCV* na versão 4.2.0. Em relação às redes neurais, o detector de placas foi treinado com *framework* de código aberto *Darknet* [18]. O detector de veículos foi o disponibilizado pela *OpenCV*. A OCR-NET foi inicializada com os mesmos pesos disponibilizados pelos seus autores [8].

Durante a execução dos experimentos, os tempos de processamento total e de cada etapa dos algoritmos são capturados e o programa *top*, disponível nativamente em muitas distribuições *Linux* (incluindo o *Raspbian*), é utilizado para obter os valores de uso de RAM. Para a obtenção do consumo de energia, é utilizado um *Wattimetro* de tomada acoplado à fonte da *Raspberry*, externo aos dispositivos, de forma que essa medição não influencie no desempenho dos algoritmos. Por fim, ao terminar a execução, a acurácia dos algoritmos é calculada.

V. Resultados

Nesta seção, são apresentados os resultados experimentais, os quais estão divididos em três subseções: resultados de acurácia, de tempo de execução e de uso de recursos pelos algoritmos.

A. Acurácia

O *Algoritmo 1* possui uma acurácia de 89,57%, enquanto o *Algoritmo 2* alcança 91,30%. Tais valores permanecem constantes independentemente do dispositivo utilizado, uma vez que o mesmo *dataset* e pesos das redes neurais são utilizados nos experimentos. A diferença de 1,73% na acurácia dos algoritmos decorre da forma que cada um lida com a imagem de entrada e as etapas às quais ela é submetida.

Por conter a etapa de detecção de veículos, o *Algoritmo 1* é mais robusto que o *Algoritmo 2*, uma vez que a identificação do veículo reduz o espaço de busca da placa e torna esse processo menos suscetível a erros. Contudo, a acurácia do primeiro é menor do que a do segundo, que é mais leve em termos de processamento. Essa diferença é explicada pela natureza do próprio *dataset*, que é composto majoritariamente por imagens nítidas, com boa iluminação e centralizadas em uma única placa veicular, fazendo com que a etapa de detecção de veículos torne-se desnecessária em muitas imagens.

Como o objetivo do *dataset* é apresentar imagens de placas veiculares, há, entre elas, casos com extrema oclusão de veículos, em que apenas partes bem específicas do carro como o para-choque estão visíveis. Isso atrapalha a detecção de veículos, pois o detector de carros não teve seu treinamento realizado para lidar com esse tipo de imagem. Consequentemente, uma vez que o veículo não seja detectado, o algoritmo não é capaz de realizar a leitura da placa, sendo contabilizado como um erro que degrada a acurácia. Dois exemplos dessas imagens estão ilustradas na Figura 2.

Tabela II
ESPECIFICAÇÕES DOS COMPUTADORES UTILIZADOS NO EXPERIMENTO

Recursos / Dispositivos	Raspberry Pi 3 B (RPi3B)	Raspberry Pi 3 B+ (RPi3B+)	Raspberry Pi 4 (RPi4B)
Processador	Cortex-A53 64-bit, quad-core, 1.2 GHz	Cortex-A53 64-bit, quad-core, 1.4 GHz	Cortex-A72 64-bit, quad-core, 1.5 GHz
Armazenamento	SanDisk micro SD Ultra	SanDisk micro SD Ultra	SanDisk micro SD Ultra
RAM	1GB LPDDR2 (900 MHz)	1GB LPDDR2 (900 MHz)	1GB LPDDR4 (2400 MHz)
S.O.	Raspbian 10 (Buster)	Raspbian 10 (Buster)	Raspbian 10 (Buster)
Alimentação	Micro USB 5V/3.0A	Micro USB 5V/3.0A	USB-C 5V/3.0A
Data de Lançamento	Fevereiro 2016	Março 2018	Junho 2019

Tabela III
DETALHAMENTO DA AVALIAÇÃO DE DESEMPENHO

Técnica de avaliação	Aferição
Delineamento	Análise comparativa de modelos de computadores *Raspberry Pi* na execução de sistemas de ALPR com variação no nível de complexidade dos algoritmos
Dispositivos usados	RPi3B, RPi3B+ e RPi4B
Carga de trabalho	115 imagens de carros brasileiros do *dataset OpenALPR* submetidas aos algoritmos sequencialmente
Fatores	Dispositivos (RPi3B, RPi3B+ e RPi4B) e algoritmos de ALPR (algoritmo 1 e algoritmo 2)
Iterações	Para cada combinação de níveis dos fatores foram realizadas 50 execuções para cada imagem do *dataset*
Métricas de resposta	Acurácia dos algoritmos, tempo de processamento total e de cada etapa dos algoritmos, uso de RAM e consumo energético dos algoritmos
Análise dos dados	Interpretação dos resultados do tempo de execução de cada etapa tos algoritmos e do uso de recursos por cada computador

(a) Imagem com um veículo completo

(b) Imagem apenas com parte de um veículo

Figura 2. Exemplo de imagens do dataset OpenALPR

Contudo, tal resultado não desqualifica a aplicação do *Algoritmo 1* em um cenário real, mas apenas mostra a necessidade de conhecer o ambiente em que ele será utilizado e treinar a etapa de *detecção de veículos* para o cenário de atuação e as características do ambiente.

B. Tempo de Execução dos Algoritmos

Os resultados envolvendo o tempo de processamento total e de cada etapa dos *Algoritmos 1 e 2* encontram-se nas Tabelas IV e V, respectivamente. Analisando a variação de dispositivos, verifica-se que, conforme o ano de lançamento, mais rápidos para essa tarefa os computadores *Raspberry* são. No *Algoritmo 1*, da *Raspberry Pi 3B* para a *3B+* há um decréscimo de tempo de processamento total igual a 16,379%. Da *3B+* para a *4B* é de 46,771%. Comparando os modelos *3B* para o *4B*, a redução de tempo de processamento é de 55,681%. Para o *Algoritmo 2*, há um decréscimo de tempo de processamento total de 14,210% da *Raspberry Pi 3B* para a *3B+* e 49,889% da *3B+* para a *4B*. Entre a *3B* e *4B*, o decréscimo é de 57,010%. Essa vantagem do *Raspberry Pi 4B* em relação aos demais modelos se dá, principalmente, por utilizar uma RAM LPDDR4 de 2400 MHz enquanto os outros, LPDDR2 de 900 MHz. Entre o *3B* e a *3B+*, tem-se a diferença dada pela frequência de operação dos seus processadores, conforme mostrado na Tabela II.

Comparando os dois algoritmos, os resultados mostram que o tempo de processamento necessário pelo *Algoritmo 2* é bem inferior ao do *Algoritmo 1*. Analisando detalhadamente os valores, verifica-se que isso se dá, principalmente, por conta da etapa de *detecção veicular*, que é a mais demorada, estar presente apenas no *Algoritmo 1*. Nas demais etapas, o *tempo de detecção de placa* é menor do que o *tempo de processamento do OCR*.

Verifica-se também que o *tempo de processamento total* do *Algoritmo 1* para o *Algoritmo 2* tem uma redução de $44,892\%$ para a *Raspberry Pi 3B*, $43,219\%$ para a *3B+* e $46,544\%$ para a *4B*.

Por meio da equação 1, é possível converter os valores de tempo de processamento para a taxa de *frames por segundo* (FPS), ou seja, quantas imagens, ou *frames*, são processadas

Tabela IV
RESULTADOS DE TEMPO DE EXECUÇÃO DO *Algoritmo 1*

Métrica	Dispositivo	Média	Desvio Padrão	Interv. Conf. 95%
Tempo Detecção Veículo (ms)	RPi3B	1418,275	51,713	[1403,578 - 1432,972]
	RPi3B+	1187,510	9,028	[1184,945 - 1190,076]
	RPi4B	615,058	23,095	[608,494 - 621,621]
Tempo Detecção Placa (ms)	RPi3B	1002,526	36,697	[992,097 - 1012,955]
	RPi3B+	833,859	4,665	[832,533 - 835,184]
	RPi4B	448,868	15,890	[444,352 - 453,383]
Tempo Processamento OCR (ms)	RPi3B	882,743	33,397	[873,252 - 892,234]
	RPi3B+	729,206	4,5140	[727,923 - 730,489]
	RPi4B	400,180	13,921	[396,224 - 404,136]
Tempo Processamento Total (ms)	RPi3B	3303,544	121,674	[3268,964 - 3338,123]
	RPi3B+	2750,575	18,049	[2745,446 - 2755,704]
	RPi4B	1464,105	52,413	[1449,210 - 1479,001]

Tabela V
RESULTADOS DE TEMPO DE EXECUÇÃO DO *Algoritmo 2*

Métrica	Dispositivo	Média	Desvio Padrão	Interv. Conf. 95%
Tempo Detecção Placa (ms)	RPi3B	966,680	70,179	[946,736 - 986,625]
	RPi3B+	834,774	4,999	[833,353 - 836,194]
	RPi4B	409,823	21,897	[403,600 - 416,046]
Tempo Processamento OCR (ms)	RPi3B	853,834	66,156	[835,033 - 872,636]
	RPi3B+	727,039	4,3158	[725,812 - 728,265]
	RPi4B	372,825	2,391	[368,420 - 377,229]
Tempo Processamento Total (ms)	RPi3B	1820,515	136,301	[1781,779 - 1859,251]
	RPi3B+	1561,813	9,232	[1559,189 - 1564,436]
	RPi4B	782,647	37,140	[772,092 - 793,202]

por segundo, algo fundamental para o processamento de vídeos em tempo-real. Os valores convertidos para cada dispositivo encontram-se na Tabela VI.

$$FPS = \frac{1}{TPTA} \tag{1}$$

Em que, TPTA é o tempo total de processamento do algoritmo em segundos.

Tabela VI
RESULTADOS DE FPS DOS ALGORITMOS

Métricas	RPi3B	RPi3B+	RPi4B
Taxa Média de Processamento do *Algoritmo 1* (FPS)	0,303	0,364	0,683
Taxa Média de Processamento do *Algoritmo 2* (FPS)	0,549	0,640	1,28

Como pode ser visto na Tabela VI, o valor de FPS mais alto ocorre com o *Algoritmo 2* na *Raspberry Pi 4B* e, mesmo assim, é inferior a 2 FPS. Tendo em vista que a grande maioria das câmeras atuais filmam com uma taxa de 30 ou 60 FPS, pode-se verificar pelos resultados a inviabilidade de embarcar esses algoritmos em computadores *Raspberry Pi* para o processamento de vídeos em tempo real, uma vez que haveria bastante perda de informação. Contudo, para análise de imagens e vídeos que não necessitem resposta imediata, os computadores *Raspberry Pi* demonstraram ser capazes de executá-la sem problemas. Com destaque para o modelo 4B.

C. Uso de RAM

Assim como nos resultados de acurácia, os valores para uso de RAM, permanecem praticamente constantes para a mudança de dispositivo, sofrendo variações relevantes apenas com a variação do algoritmo usado. Novamente, isso é algo esperado, uma vez que os *scripts* em execução nas diferentes máquinas são os mesmos, a diferença está apenas no algoritmo.

Conforme pode ser visto na Figura 3, o uso médio de RAM do *Algoritmo 1* é de, aproximadamente, 273 MB, enquanto o uso pelo *Algoritmo 2* fica em torno de 243 MB. Ou seja, há uma diferença de cerca de 40 MB entre os algoritmos, algo esperado, uma vez que o *Algoritmo 1* possui uma etapa a mais. De todo modo, tal diferença, considerando que o *Raspberry Pi* executará exclusivamente o sistema de ALPR e demais processos do sistema operacional, é irrelevante, já que corresponde a aproximadamente 4% de toda a memória disponível, 1 GB, desses computadores.

D. Consumo Energético

As medições de consumo energético de cada *Raspberry* baseiam-se na potência elétrica média desses computadores. Em estado ocioso, ou seja, sem a execução dos algoritmos avaliados nesse trabalho, os valores da potência são de 1,7 W para a *3B*; 2,0 W para a *3B+* e 2,8 W para a *4B*.

A potência elétrica média para o processamento de cada imagem do *dataset* de placas por cada computador e algoritmo encontra-se na Tabela VII. Pelos valores dessa tabela, verifica-se que o aumento do poder computacional ao longo das versões dos computadores resultou no aumento da potência elétrica. Possivelmente uma consequência da atualização e inserção de novos componentes de *hardware*, como GPU no caso da *Raspberry Pi 4*.

De acordo com os resultados obtidos na Tabela VII, a *Raspberry Pi 4B* é a que possui uma maior potência elétrica.

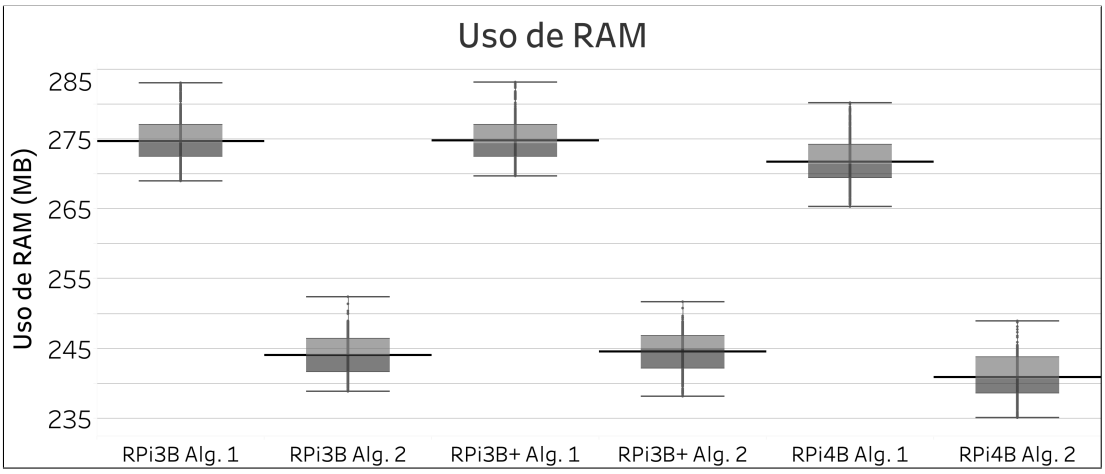

Figura 3. Utilização média de RAM por algoritmo e por dispositivo

Tabela VII
POTÊNCIA ELÉTRICA DOS COMPUTADORES DURANTE A EXECUÇÃO DOS ALGORITMOS

Métrica	Dispositivo	Média	Desvio Padrão	Interv. Conf. 95%
Potência Elétrica do *Algoritmo 1* (W)	RPi3B	4,090	2,187	[4,041 - 4,140]
	RPi3B+	5,610	0,365	[5,568 - 5,651]
	RPi4B	6,570	0,532	[6,479 - 6,661]
Potência Elétrica do *Algoritmo 2* (W)	RPi3B	4,033	0,426	[3,876 - 4,191]
	RPi3B+	5,548	0,330	[5,519 - 5,578]
	RPi4B	6,520	0,470	[6,455 - 6,582]

Contudo, por meio dos valores das Tabelas IV e V, é possível verificar que o tempo de processamento médio total é inversamente proporcional à potência elétrica. Dessa forma, utilizando os valores dessas tabelas, é possível calcular o consumo de cada computador para cada algoritmo, em kWh, mostrados na Tabela VIII.

A partir dos valores da Tabela VIII, verifica-se que a *Raspberry Pi 3B+* é a mais ineficiente em termos de gasto de energia por tempo de processamento. Já a *Raspberry Pi 4B* mostra-se como a melhor opção em termos de consumo energético e tempo de processamento.

VI. CONCLUSÃO E TRABALHOS FUTUROS

Foi realizada uma avaliação de desempenho por meio da execução de dois algoritmos de uma aplicação de Visão Computacional e *Deep Learning*, um sistema de ALPR, em *hardwares* de baixo custo e amplamente adotados em aplicações de *Internet das Coisas*. Na ocasião, foram utilizados os modelos *Raspberry Pi 3B*, *3B+* e *4B*.

O *Raspberry Pi 4B*, o modelo mais recente dos computadores, mostra-se mais preparado para esse tipo de aplicação, sendo o mais rápido dentre os três, além de ser o mais econômico em termos de consumo energético por tempo de processamento. Os demais dispositivos também conseguem executar ambos os algoritmos sem problemas, com o modelo 3B+ sendo um pouco mais rápido que o 3B, mas sendo menos eficiente quando analisado sob ponto de vista do consumo de energia. Contudo, os resultados obtidos mostram a inviabilidade de se utilizar um sistema de ALPR nesses computadores para o processamento de vídeo em tempo real.

A acurácia dos algoritmos e o uso de recursos mostram-se independentes do dispositivo utilizado, como esperado. Considerando o *dataset* utilizado, o *Algoritmo 2* supera *Algoritmo 1* em todas as métricas, sendo mais econômico no uso de recursos e de energia, mais rápido e mais acurado.

Como trabalhos futuros, pretende-se analisar o desempenho dos algoritmos em um *cluster* de computadores *Raspberry Pi* e comparar com máquinas comumente utilizadas em aplicações de ALPR. Deseja-se também investigar o impacto no desempenho da plataforma *Raspberry* com o uso de diferentes componentes, como cartões micro SD de diferentes velocidades e fontes de alimentação não-oficiais, incluindo componentes falsificados. Tal estudo é pertinente, pois preços atrativos que esses componentes oferecem à primeira vista podem vir a comprometer o desempenho do *hardware* quando da sua implantação. Serão realizados experimentos com um *Intel Movidius*™ acoplado aos computadores devido aos ótimos resultados que têm sido reportados por outros trabalhos, como em [17], onde foi possível realizar o processamento de vídeo em tempo real em uma *Raspberry Pi 3B* com o uso desse dispositivo. Por fim, com o objetivo de analisar o custo-benefício de *hardwares* na aplicação apresentada, outros dispositivos serão investigados, tais como outras plataformas embarcadas e FPGAs (*Field Programmable Gate Array*).

AGRADECIMENTOS

Os autores gostariam de agradecer à Fundação Cearense de Apoio ao Desenvolvimento Científico e Tecnológico (FUNCAP) pelo suporte financeiro (6945087/2019).

Tabela VIII
ESTIMATIVA DO CONSUMO DE ENERGIA POR PERÍODO DE PROCESSAMENTO

Métrica	Dispositivo	Consumo (kWh)
Algoritmo 1	RPi3B	3.763×10^{-6}
	RPi3B+	4.256×10^{-6}
	RPi4B	2.694×10^{-6}
Algoritmo 2	RPi3B	2.057×10^{-6}
	RPi3B+	2.386×10^{-6}
	RPi4B	1.434×10^{-6}

REFERÊNCIAS

[1] B. Nour, K. Sharif, F. Li, S. Biswas, H. Moungla, M. Guizani, and Y. Wang, "A survey of internet of things communication using icn: A use case perspective," *Computer Communications*, vol. 142-143, p. 95–123, 2019.

[2] K. Wang, Y. Wang, Y. Sun, S. Guo, and J. Wu, "Green industrial internet of things architecture: An energy-efficient perspective," *IEEE Communications Magazine*, vol. 54, no. 12, p. 48–54, 2016.

[3] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke, "Middleware for internet of things: A survey," *IEEE Internet of Things Journal*, vol. 3, no. 1, p. 70–95, 2016.

[4] S. J. Johnston, M. Apetroaie-Cristea, M. Scott, and S. J. Cox, "Applicability of commodity, low cost, single board computers for internet of things devices," *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, 2016.

[5] S. Du, M. Ibrahim, M. Shehata, and W. Badawy, "Automatic license plate recognition (alpr): A state-of-the-art review," *IEEE Transactions on circuits and systems for video technology*, vol. 23, no. 2, pp. 311–325, 2012.

[6] SSPDS, "Roubos de veículos no ceará têm redução de 46% em 2019," https://www.ceara.gov.br/2019/12/11/com-30-meses-seguidos-de-diminuicao-no-cvp-roubos-de-veiculos-no-ceara-se-destacam-com-reducao-de-46-em-2019/, Governo do Estado do Ceará, 2019, acessado em 01 de abril de 2020. [Online]. Available: https://www.ceara.gov.br/2019/12/11/com-30-meses-seguidos-de-diminuicao-no-cvp-roubos-de-veiculos-no-ceara-se-destacam-com-reducao-de-46-em-2019/

[7] R. Laroca, E. Severo, L. A. Zanlorensi, L. S. Oliveira, G. R. Gonçalves, W. R. Schwartz, and D. Menotti, "A robust real-time automatic license plate recognition based on the yolo detector," in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–10.

[8] S. M. Silva and C. R. Jung, "License plate detection and recognition in unconstrained scenarios," in *European Conference on Computer Vision*. Springer, 2018, pp. 593–609.

[9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[10] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[11] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[12] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[13] S. M. Silva and C. R. Jung, "Real-time brazilian license plate detection and recognition using deep convolutional neural networks," in *2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. IEEE, 2017, pp. 55–62.

[14] D. M. Izidio, A. P. Ferreira, H. R. Medeiros, and E. N. d. S. Barros, "An embedded automatic license plate recognition system using deep learning," *Design Automation for Embedded Systems*, pp. 1–21, 2019.

[15] R. M. Ramos, C. Ralha, and G. Teodoro, "Avaliação de cluster raspberry pi para execução de aplicações de análise de imagens microscópicas médicas," in *Anais do XLIII Seminário Integrado de Software e Hardware*. SBC, 2020, pp. 175–186.

[16] M. Murshed, E. Verenich, J. J. Carroll, N. Khan, and F. Hussain, "Hazard detection in supermarkets using deep learning on the edge," *arXiv preprint arXiv:2003.04116*, 2020.

[17] J. Hochstetler, R. Padidela, Q. Chen, Q. Yang, and S. Fu, "Embedded deep learning for vehicular edge computing," in *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 2018, pp. 341–343.

[18] J. Redmon, "Darknet: Open source neural networks in c," http://pjreddie.com/darknet/, 2013–2016.