Aprendizado Profundo aplicado à Visão Robótica utilizando dispositivo embarcado Raspberry Pi

Myrella A. Bordado[1], Alisson A. Cardoso[1], Ricardo A. P. Franco[2]

[1]Escola de Engenharia Elétrica, Mecânica e de Computação – Universidade Federal de Goiás (UFG) – Goiânia – GO – Brasil

[2]Instituto de Informática – Universidade Federal de Goiás (UFG) – Goiânia – GO – Brasil

myrellaalves@discente.ufg.br,alsnac@ufg.br,ricardofranco@ufg.br

Abstract. *Due to the large amount of data that emerged, along with the evolution of Graphics Processing Units (GPU), it became possible to develop algorithms with greater performance and accuracy, these are the methods based on deep learning. This paper performs the training of the YOLOv5 model using a robotic vision dataset ARID in order to address the problem of object detection in a domestic environment considering resource limitations of the embedded system Raspberry Pi. The inference results show that the YOLOv5 model presented high accuracy for the object detection task in robotic vision, in addition to having versions that allow execution on low-cost embedded devices when comparing embedded devices to GPU.*

Resumo. *Devido a grande quantidade de dados que surgiram, juntamente com a evolução das Unidades de Processamento Gráfico (GPU), tornou-se possível desenvolver algoritmos com maior desempenho e acurácia, baseados em aprendizado profundo. Este trabalho realiza o treinamento do modelo YOLOv5 utilizando um conjunto de dados de visão robótica ARID, com o intuito de abordar o problema de detecção de objetos em um ambiente doméstico em conjunto com a limitação de recursos do sistema embarcado Raspberry Pi. Os resultados mostram que o modelo YOLOv5 possui alta precisão para a detecção de objetos utilizando visão robótica e possui versões que possibilitam a execução em dispositivos embarcados de baixo custo, quando comparados à dispositivos embarcados com GPU.*

1. Introdução

A Robótica é uma grande área do conhecimento dentro da ciência que pode ser dividida em várias subáreas, visto que se trata de um ramo vasto e multidisciplinar com grande aplicabilidade em problemas do mundo real. A Visão Computacional é uma dessas subáreas que busca simular a visão humana em sistemas computacionais[1]. Desde a década de 60 a Visão Computacional possui alta relevância visto que robôs podem executar tarefas com alta precisão e rapidez e até mesmo tarefas que são consideradas de grande risco aos seres humanos [1-2].

Devido a evolução das Unidades de Processamento Gráfico (*Graphics Processing Unit* - GPU) juntamente com a massiva quantidade de conjuntos de dados que surgiram a partir da década de 2010 como, por exemplo, o ImageNet [3], as técnicas de Visão Computacional basearam-se nesses avanços juntamente com a aprendizagem profunda para aprimorar e resolver muitos problemas que apenas as técnicas de visão tradicional não conseguiam resolver como, por exemplo, mudança de iluminação, escala, rotação, translação, etc.

Dentro da visão computacional, é possível encontrar vários métodos e técnicas que resolvem um mesmo problema de formas diferentes, variando em avanço, custo computacional, arquiteturas, robustez e complexidade de algoritmos [4]. Algumas dessas técnicas são: classificação de objetos, reconhecimento de pessoas, estimação de pose, segmentação semântica e detecção de objetos.

A detecção de objetos pode ser entendida como ensinar uma máquina a identificar e localizar objetos no plano [5]. Técnicas baseadas em aprendizagem profunda vêm sendo muito utilizadas atualmente na área de detecção de objetos devido à versatilidade, alto desempenho e acurácia em seus resultados, juntamente com a grande quantidade de dados disponibilizados [3]. Uma das aplicações é a detecção de objetos encontrados em ambientes domésticos através de visão robótica [6].

Este trabalho tem como objetivo mostrar uma análise acerca da detecção de objetos em ambiente doméstico em dispositivos com recursos de *hardware* limitados, isto é, sistemas embarcados que não possuem GPU para realizar a inferência de dados. O dispositivo *Raspberry Pi* 4 foi utilizado neste trabalho. Para implementar um modelo de aprendizagem profunda, utiliza-se o algoritmo YOLOv5 e suas versões (*small*, *medium* e *large*) para realizar o treinamento em um conjunto de dados de visão robótica e, posteriormente, a inferência no dispositivo embarcado.

2. Referencial teórico

A detecção de objetos divide-se em duas partes: o período dos métodos tradicionais de detecção de objetos e o período de detecção baseado em aprendizagem profunda, conforme figura abaixo [5].

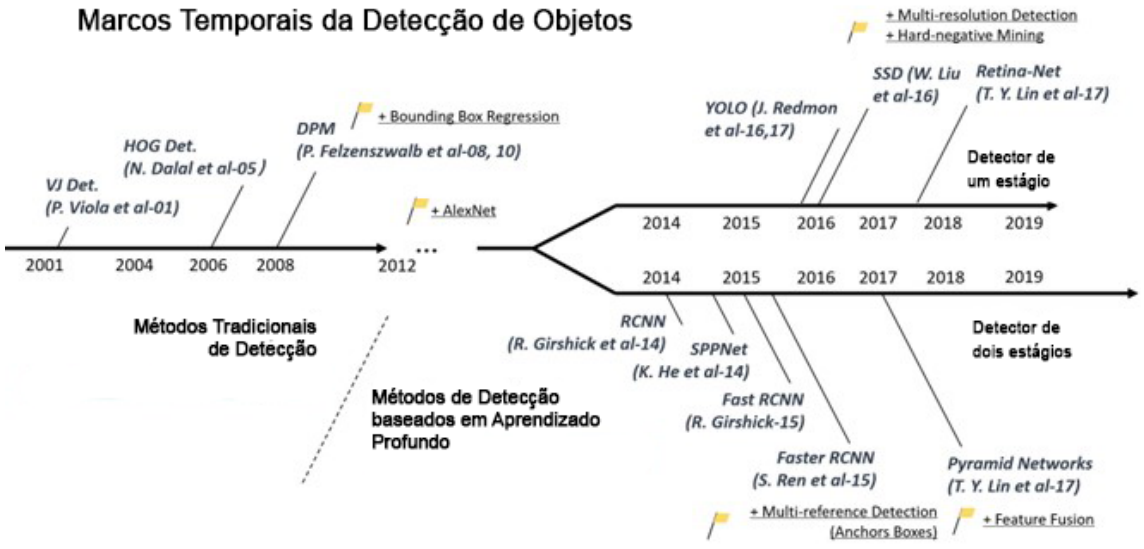


Figura 1: Marcos da detecção de objeto ao longo dos anos. Adaptado de [5].

2.1. Métodos tradicionais de visão computacional e detecção de objetos

A partir dos anos 90, surgiram os chamados descritores de características como a principal técnica de visão computacional. A primeira técnica utilizada por esses descritores e extratores de características era a detecção de bordas em uma imagem [7].

Com o tempo, além das bordas, as técnicas tradicionais de visão computacional também utilizavam da extração de características dos cantos, cores, iluminação, textura da imagem, etc, para descrever um objeto numa cena [8]. A partir disso, surgiram também os descritores de características invariantes a escalas, como o SIFT e o SURF.

O SIFT (*Scale Invariant Feature Transform*) [9-10] é um algoritmo invariável às características da imagem como escala, rotação e translação por completo e invariável parcialmente as informações de luz. Por conta disso, o SIFT é um dos descritores de características mais famosos e utilizados desde seu surgimento. Devido sua popularidade, diversos extratores basearam-se no SIFT para resolver limitações que o SIFT não conseguia com tanta eficiência como é o caso do SURF (*Speeded-Up Robust Features*) que, basicamente, difere do SIFT por utilizar regiões da imagem e não apenas um ponto de interesse melhorando os resultados e a performance do algoritmo [11].

Uma aplicação importante que o SIFT alavancou foi na robótica, pois suas soluções foram muito úteis para localização e mapeamento em ambientes reais para robôs móveis, onde possui variância de escala, de iluminação, oclusão de objetos em tempo real [12].

Apesar das técnicas tradicionais de detecção de objetos terem sido muito úteis para a visão computacional e alavancado bastante essa área ao longo dos anos, o surgimento das GPUs e da massiva quantidade de dados, juntamente com o desenvolvimento das aplicações utilizando aprendizado profundo, sobressaíram-se muito rapidamente. O motivo é que o aprendizado profundo resolveu grande parte dos problemas que as técnicas tradicionais não resolveram e, além disso, superaram as técnicas tradicionais em nível de assertividade e desempenho [13].

2.2. Métodos de detecção de objetos baseado em aprendizagem profunda

O aprendizado profundo começou a ser utilizado a partir de 2014 para realizar a detecção de objetos, conforme observa-se na figura 1. A ambição humana de simular o que se conhece sobre o comportamento do cérebro humano foi o que impulsionou o desenvolvimento das redes neurais artificiais e, posteriormente, das redes neurais convolucionais [14]. Dessa forma, os algoritmos de aprendizagem profunda conseguiram atingir o estado da arte em várias tarefas, inclusive na detecção de objetos [15-16].

Dentro da detecção de objetos, pode-se classificar dois tipos de detectores: os detectores de 1 estágio e os detectores de 2 estágios. Os detectores de 1 estágio destacam-se pela alta velocidade de inferência enquanto os detectores de 2 estágios possuem alta acurácia na localização e reconhecimento dos objetos[17].

O algoritmo YOLO (*You Only Look Once)*[18] é um detector de 1 estágio e foi projetado de tal forma que a imagem precisa passar só uma vez pela arquitetura da rede para realizar a detecção dos objetos. Essa rede realiza a detecção de objetos como um problema simples de regressão [19]. Outro diferencial do YOLO, é que esse algoritmo analisa a imagem como um todo (global) e não apenas para regiões locais como a maioria dos detectores e descritores fazem, como por exemplo, o SIFT[11] e o *Faster* R-CNN [20].

A primeira versão do YOLO surgiu em 2015 e, com o passar dos anos, foram desenvolvidas outras versões como, o YOLOv2, YOLOv3 e YOLOv4 [21]. Atualmente, a mais utilizada por apresentar um melhor desempenho é a YOLOv5, que difere das anteriores visto que essa é implementada utilizando o *framework* PyTorch, sendo que as demais foram implementadas utilizando o *framework* Darknet, mais denso que o PyTorch.

3. Metodologia

3.1. Definição do modelo

Neste trabalho foi utilizado o algoritmo YOLO [22]. A arquitetura do YOLO é simples, de forma que esse algoritmo obtenha bom desempenho e seja mais rápido, em tempo de treinamento que os outros modelos, como por exemplo, a *Faster* R-CNN [23]. Sua rapidez de processamento se dá pelo fato de que o algoritmo realiza a leitura da imagem uma única vez, no fim da detecção para classificar e detectar o objeto [24]. Além disso, o YOLO é atualmente um dos modelos considerado como estado da arte na detecção de objetos.

Dentro de suas variações, foi escolhida a YOLOv5, visto que é uma das últimas versões disponibilizadas até o momento com melhor relevância. Foram utilizadas as versões *small*, *medium* e *large* do YOLOv5 para realizar o treinamento e a inferência do conjunto de dados.

3.2. Treinamento e Validação do modelo

A metodologia desenvolvida realiza o treinamento e validação do modelo em GPU e, posteriormente, o modelo é inserido no dispositivo embarcado para a realização das inferências. A escolha para tal abordagem utilizada foi devido ao fato de que é necessário a utilização de GPU para o treinamento do modelo, tornando-se inviável seu treinamento no *Raspberry Pi*, visto que, o mesmo não possui GPU em sua arquitetura de *hardware* limitada. Por isso, utilizou-se um computador externo para tal procedimento.

O treinamento do *YOLOv5* foi realizado via *Colab*, disponibilizado pela *Google*, em um computador com as seguintes configurações: processador Intel(R) Xeon(R) CPU @ 2.20GHz, com 25GB de memória RAM e com uma GPU Tesla P100-PCIE-16GB.

3.3. Inferência no Dispositivo Embarcado

A inferência foi realizada em um dispositivo com as seguintes configurações: *Raspberry Pi* 4 modelo B, com processador 64-*bits quad-core* Cortex-A72 e 4 GB de memória RAM.

O ato de primeiro treinar e validar um modelo e posteriormente transferi-lo para o dispositivo embarcado é uma solução para a robustez e complexidade dos algoritmos utilizados.

Devido às restrições de recursos de *hardware* do dispositivo embarcado, utilizou-se o modelo YOLOv5 com pesos pré-treinados no conjunto de dados COCO128 [25], e foi realizada a transferência de conhecimento (*transfer learning*) no conjunto de dados ARID.

3.4. A escolha do conjunto de dados

O objetivo deste trabalho é analisar a viabilidade de um sistema de detecção de objetos em ambientes domésticos em um dispositivo embarcado com recursos limitados. Dessa forma, foi escolhido o conjunto de dados ARID [26] para essa proposta, pois a maioria dos conjuntos de dados existentes atualmente são muito restritos a um ambiente controlado. Quando se trata de um robô móvel em um ambiente doméstico, o treinamento em um ambiente controlado, ou seja, nas condições ideais, não é muito interessante. Considerando que o modelo seja treinado em um conjunto de dados onde a iluminação é sempre adequada, os objetos estejam sempre visíveis e dispostos nas imagens, dentre outros itens, o robô provavelmente não generalizará de forma satisfatória quando essas condições não forem supridas. Por isso, foi escolhido um conjunto de dados com imagens de visão robótica, ou seja, imagens e cenas captadas a partir de um robô que considera os desafios de um ambiente aberto como variação de iluminação, desordem, oclusão, dentre outros.

4. Resultados

O modelo YOLOv5 utilizado possui pesos pré-treinados do conjunto de dados COCO128 [25]. Todavia, como o objetivo deste trabalho é o treinamento e aplicação do modelo na tarefa de detecção de objetos em ambiente doméstico em visão robótica, foi realizado o treinamento do modelo com o conjunto de dados ARID [26].

O treinamento foi realizado utilizando 3.420 imagens capturadas a partir de um robô em ambientes distintos, totalizando 2.659 imagens de treinamento e 639 imagens utilizadas para validação. Além disso, 122 imagens foram utilizadas para teste no dispositivo embarcado, isto é, imagens que não foram utilizadas no treinamento e na validação.

Os parâmetros considerados para análise foram: precisão, sensibilidade, mAP *(mean Average Precision)* e *f1-score.* A precisão representa a taxa de quantos rótulos foram classificados corretamente em relação ao total, enquanto a sensibilidade representa a taxa de rótulos positivos que foram classificados corretamente [27]. Em detecção de objetos, pode-se dizer que a sensibilidade é uma métrica representando a quantidade de objetos que foram identificados e a precisão uma métrica representando que os objetos identificados foram classificados corretamente.

Para obter as métricas de precisão e sensibilidade na tarefa de identificação de objetos, é necessário analisar o quão similar/sobreposto a posição do *box* predito pela rede está em relação ao *box* verdadeiro previamente anotado (*ground truth box)*. Para isto, define-se o valor IoU (*Intersection over Union*) como um limiar para poder dizer se a classe identificada será considerada verdadeira ou falsa [27].

Tabela 1. Divisão do dataset ARID nos conjuntos de treinamento e teste.

	Treinamento	Validação	Teste	Total
Divisão	78%	19%	3%	100%
Total de imagens	2659	639	122	3420
Quantidade de Objetos	32718	8006	1459	40724

Destaca-se que os parâmetros de desempenho (precisão e sensibilidade) são sensíveis à escolha do limiar de IoU. Assim, define-se a métrica de desempenho mAP representando a média da curva precisão-sensibilidade com variações sobre o limiar IoU. Neste trabalho, considera-se mAP.5 a média da curva precisão-sensibilidade com limiar de IoU em 0,5 e mAP a média da curva precisão-sensibilidade com limiar de IoU variando de 0,5 até 0,95. Por fim, o parâmetro *f1-score* é utilizado para maior robustez na análise dos resultados obtidos, pois retorna a média harmônica entre a precisão e a sensibilidade.

Na Tabela 2, apresenta-se os resultados do treinamento das três versões do YOLOv5 (*small*, *medium* e *large*) para o conjunto de teste do *dataset* ARID.

Tabela 2. Comparação entre as métricas de treinamento (precisão, sensibilidade, map e *f1-score*) para as três versões do YOLOv5.

Modelo	Precisão	Sensibilidade	*F1 Score*	mAP.5	mAP	Quantidade de Parâmetros (Milhões)	Tempo médio de Execução (*Raspberry*)	Tempo em relação a *small*
***small* (Yolov5s6)**	98,2 %	98,1 %	98,1%	96,8 %	85,0 %	12	942,9 ms	1x
***medium* (Yolov5m6)**	98,2 %	98,1 %	98,1%	99,0 %	87,9 %	35	2350,4 ms	2,49x
***large* (Yolov5l6)**	98,4 %	98,3 %	98,3%	99,0 %	88,1 %	76	4351,1 ms	4,61x

De acordo com os resultados apresentados na Tabela 2, nota-se que o modelo *large* (Yolov5l6) obteve os maiores valores para as métricas de precisão, sensibilidade, mAP.5 e mAP em relação aos modelos *small* e *medium*. Esses resultados indicam que o modelo *large* consegue detectar e predizer corretamente uma maior quantidade de rótulos positivos no conjunto de teste em relação aos demais.

Ainda na Tabela 2, observa-se o tempo computacional para realizar a inferência de uma imagem utilizando os modelos *small* e *medium*. Nota-se que o tempo computacional do modelo *small* foi 2,49 vezes menor que o tempo para o modelo

medium e 4,61 vezes menor do que o modelo *large*. Destaca-se que a diferença na performance nas métricas precisão, sensibilidade e mAPs dos modelos maiores não tem um impacto suficiente maior em detrimento do modelo *small* para ser utilizado em dispositivos com processamento limitado, uma vez que a quantidade de parâmetros (memória) e tempo computacional (processamento) do modelo *small* é bem inferior aos demais.

Na Figura 2, são apresentadas as imagens capturadas pelo robô juntamente com as inferências realizadas pelas três versões do modelo YOLOv5.

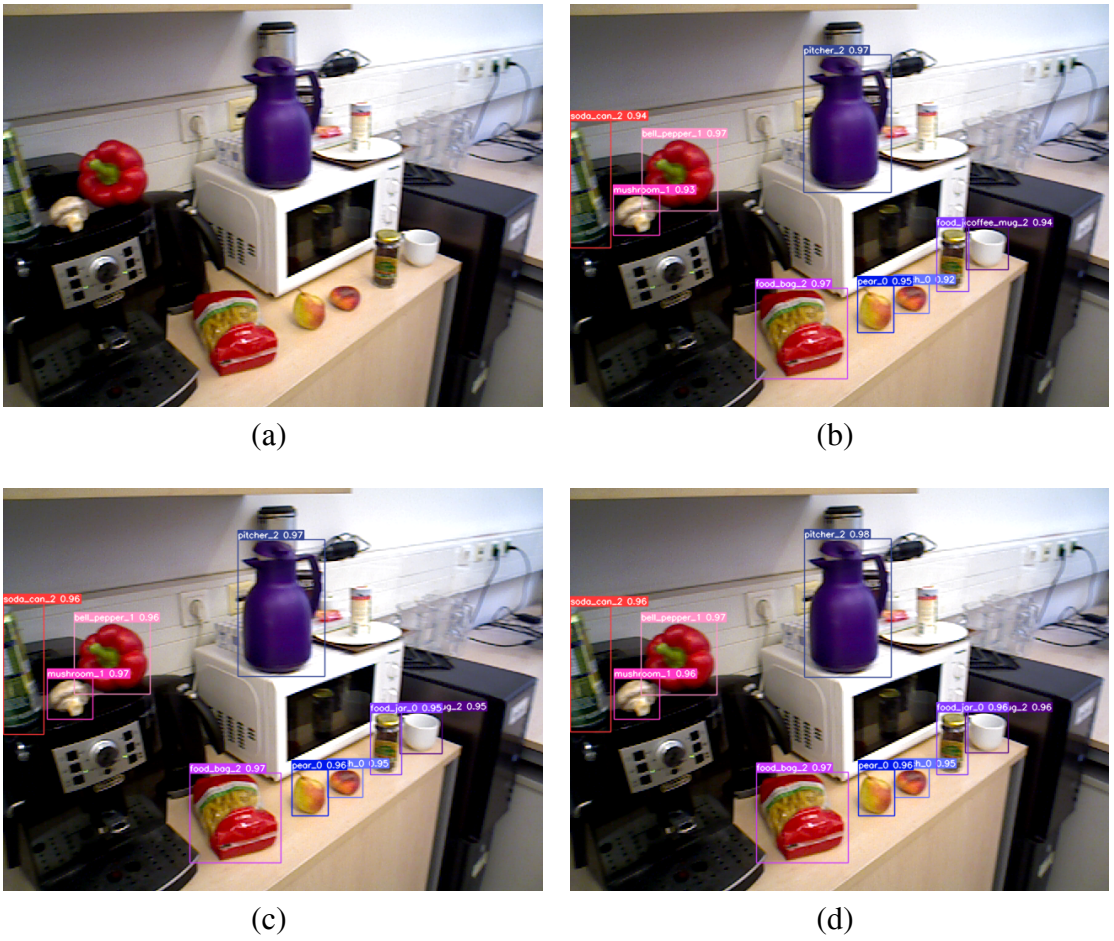
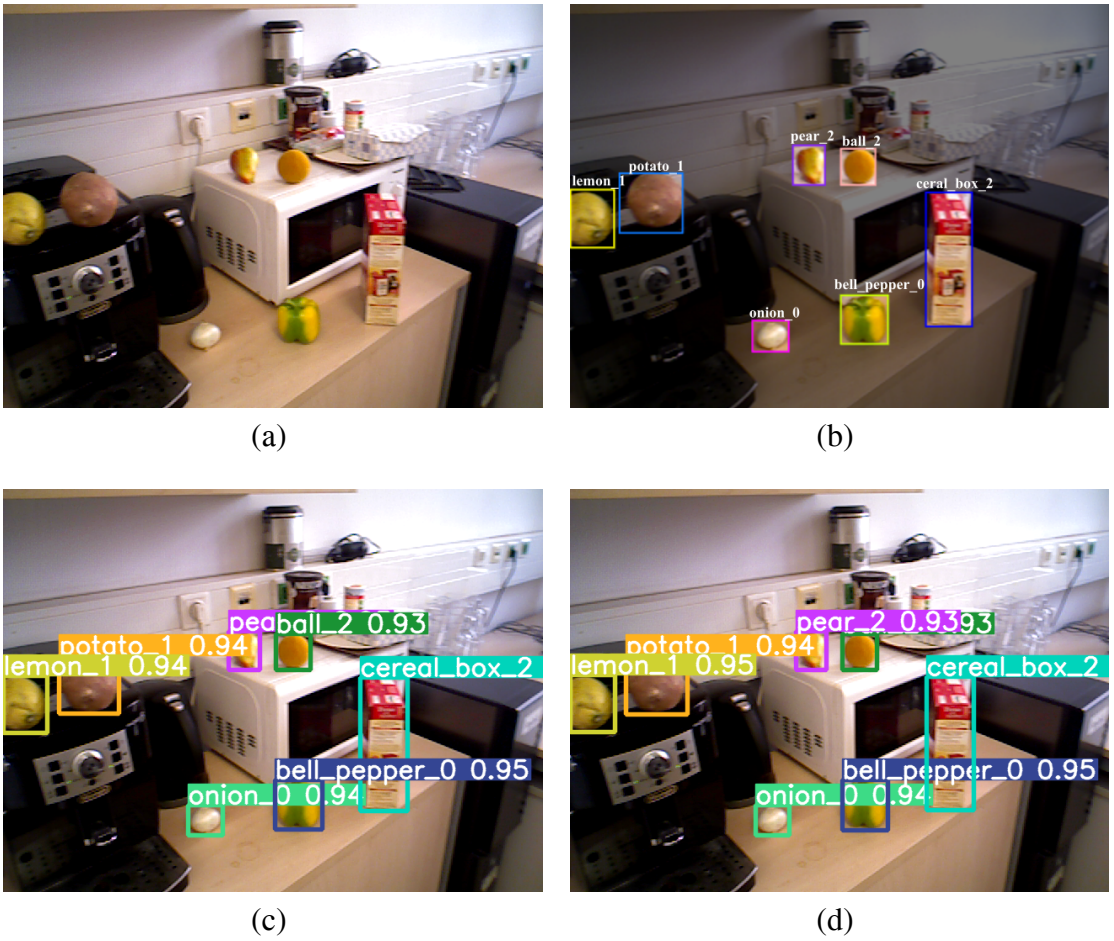


Figura 2. Imagem pura (a) e imagem contendo as inferências realizadas pelas três versões *small* (b), *medium* (c) e *large* (d) do modelo YOLOv5 no *Raspberry Pi*.

Ainda na Figura 2, observa-se que os modelos conseguem identificar os objetos de forma satisfatória, apresentando valores de confiança maiores que 90% para todos os objetos na imagem. Nota-se que os valores de confiança de ser a classe no modelo *small* é menor em comparação aos demais modelos, ou seja, o modelo *small* tem menos certeza sobre o objeto. Destaca-se que essa diferença é mínima (em torno de 1%), onde o modelo *small* conseguiu identificar todos objetos, assim como os demais, embora com um tempo de processamento bem inferior.

Na Figura 3, visualiza-se a imagem de teste capturada pelo robô, a imagem com as caixas delimitadoras anotadas (*ground truth*) e a imagem com a inferência realizada no dispositivo embarcado da versão *small* e *medium* do modelo YOLOv5.



(a) (b)

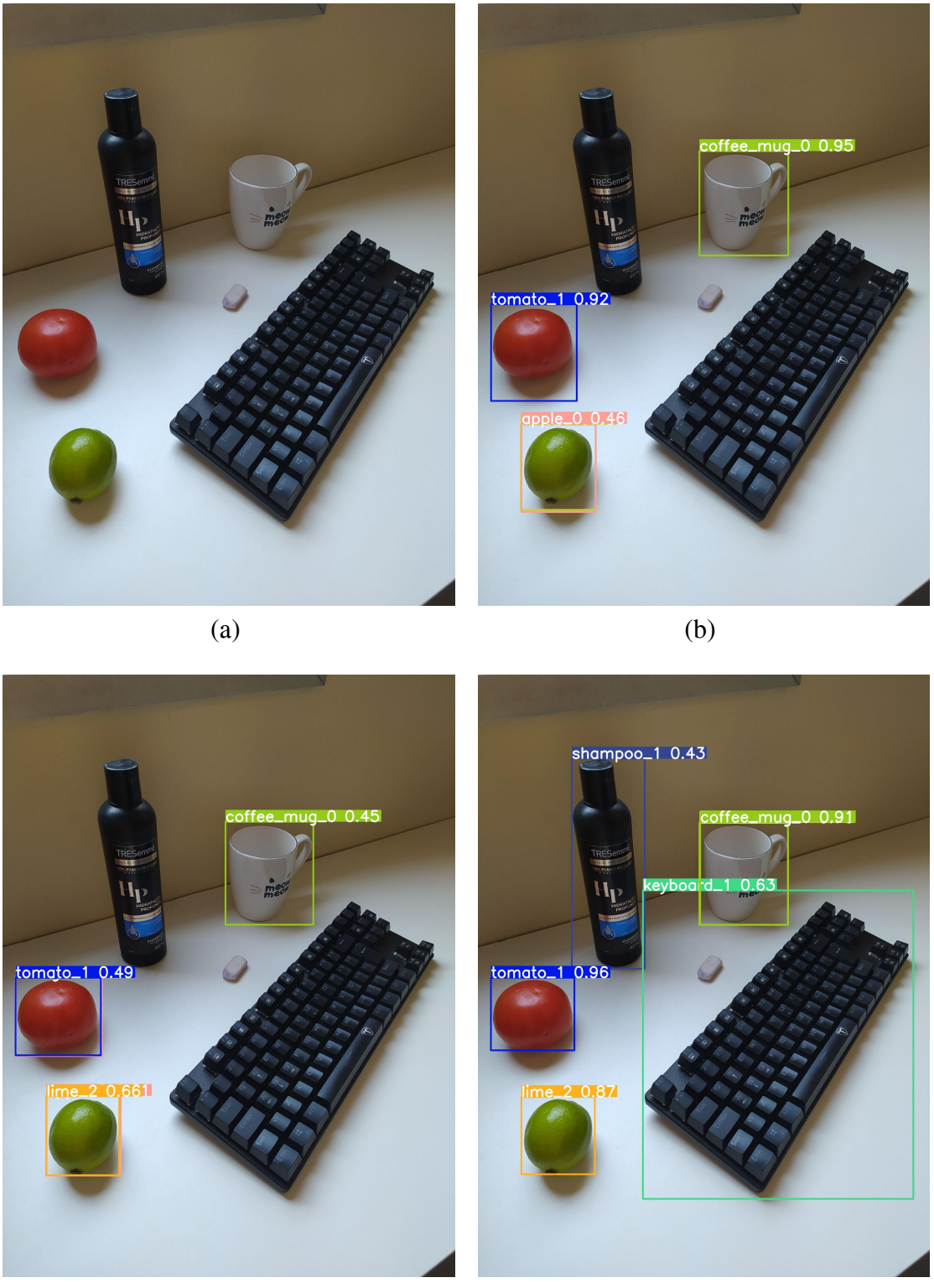
(c) (d)

Figura 3. Imagem pura (a), *ground truth* (b) e imagem contendo as inferências realizadas no *Raspberry Pi* pela versões *small* e *medium* do modelo YOLOv5 (c) e (d), respectivamente.

Observa-se na Figura 3 que os modelos *small* e *medium* conseguiram identificar todos os rótulos da imagem *ground truth.* Destaca-se que os dois modelos obtiveram valores de confiança iguais, exceto para o objeto *lemon_1*, onde o modelo *small* obteve 0,94, enquanto, o modelo *medium* obteve 0,95.

Por fim, foi conectado uma câmera ao dispositivo embarcado para realizar a captura de uma imagem de um ambiente doméstico, de forma a simular a conexão do robô ao dispositivo embarcado, de forma a obter a visão robótica. Pode-se observar o resultado da inferência dos modelos *YOLOs* no sistema embarcado na Figura 4.

Nota-se, na Figura 4, que o modelo *large* apresentou a maior quantidade de objetos sendo detectados. Enquanto, os modelos *small* e *medium* realizaram a detecção de três objetos (caneca, tomate e limão). Pode-se observar que, para este exemplo, houve confusão do objeto limão, sendo identificado como maçã pelo modelo *small* e com baixos valores de confiança obtidos pelo modelo *medium*.



(a) (b)

(c) (d)

Figura 4. Imagem pura (a), imagem contendo as inferências realizadas pelas versões *small*, *medium* e *large* (b), (c) e (d), respectivamente do modelo YOLOv5 no *Raspberry Pi*.

Dessa forma, de acordo com os resultados se observa que o modelo treinado (YOLOv5 *small*) é adequado para uso em dispositivos embarcados com recursos de *hardware* limitado e que possui boa assertividade na detecção de objetos domésticos. Nota-se que, embora os modelos *medium* e, em destaque, *large* possuem boa assertividade na detecção de objetos, os mesmos apresentam um tempo de processamento no dispositivo *Raspberry Pi* que impossibilitam seu uso em sistemas robóticos de tempo real.

5. Conclusões

A partir dos resultados obtidos das implementações dos treinamentos das versões do modelo YOLOv5 em ambiente computacional e das inferências do modelo YOLOv5 *small* no dispositivo *Raspberry Pi*, conclui-se que os objetivos deste trabalho foram atingidos de forma satisfatória, dentro das limitações de *hardware* definidas. Portanto, é possível constatar a viabilidade da construção de um sistema de detecção de objetos em dispositivos embarcados limitados e de custo reduzido, em comparação com dispositivos embarcados com GPU. Sendo possível, para trabalhos futuros analisar mais a fundo esse tópico no intuito de implementar novos algoritmos no dispositivo, comparando seus resultados com aqueles fornecidos neste trabalho. Além disso, é possível utilizar visão robótica, em robôs com características de assistente doméstico, com capacidade de identificar objetos, evitando utilizar dispositivos com GPU.

Referências

[1] SZELISKI, R., Computer Vision: Algorithms and Applications. Springer, set. 2010.

[2] MILANO, D.; HONORATO, L. B. Visão Computacional. Faculdade de Tecnologia, Universidade Estadual de Campinas, Limeira. Acesso em: 20 ago. 2022.

[3] RUSSAKOVSKY, O. et al. ImageNet Large Scale Visual Recognition Challenge. In: arXiv:1409.0575. Acesso em: 20 ago. 2022.

[4] YADAV, N; BINAY, U. Comparative Study of Object Detection Algorithms. 2017, IRJET | Impact Factor value: 6.171 | ISO 9001:2008 Certified Journal | Página 591.

[5] ZOU, Z. et al. Object Detection in 20 Years: A Survey. 2019. Disponível em: arXiv:1905.05055v2. Acesso em: 20 ago. 2022.

[6] MARTINEZ-MARTIN, E.; DEL POBIL, A. P. Object Detection and Recognition for Assistive Robots: Experimentation and Implementation. In: IEEE Robotics & Automation Magazine, v. 24, n. 3, p. 123 - 138, set. 2017. Acesso em: 20 ago. 2022.

[7] ALAKE, Richmond. A Beginner's Guide To Computer Vision. Towards Data Science, 22 set. 2020. Disponível em: https://towardsdatascience.com/a-beginners-guide-to-computer-

vision-dca81b0e94b4. Acesso em: 20 ago. 2022.

[8] THILAKARATHNE, Haritha. Deep Learning Vs. Traditional Computer Vision. NaadiSpeaks, 12 ago. 2018. Disponível em: https://naadispeaks.wordpress.com/2018/08/12/deep-

learning-vs-traditional-computer-vision/. Acesso em: 20 ago. 2022

[9] LOWE, David G. Object recognition from local scale-invariant features. In: Seventh IEEE International Conference on Computer Vision, 7., 1999. Anais eletrônicos [...]: IEEE, 1999. p. 1150 - 1157 vol. 2.

[10] LINDEBERG, Tony. Scale Invariant Feature Transform. Scholarpedia 7(2012): 10491. Disponível em: http://www.scholarpedia.org/article/Scale_Invariant_Feature _Transform. Acesso em: 20 ago. 2022.

[11] CHELLURI, H. B.; MANJUNATHACHARI, K. SIFT and it's Variants: An Overview In: INTERNATIONAL CONFERENCE ON SUSTAINABLE COMPUTING IN SCIENCE, TECHNOLOGY AND MANAGEMENT (SUSCOM-2019). Anais [...] Jaipur - India: Amity University Rajasthan, 26-28 fev. 2019. Disponível em: https://ssrn.com/abstract=3358743. Acesso em: 20 ago. 2022.

[12] HOLTKAMP, Michiel; JONG, Sjoerd de. Robot Localisation Using SIFT and Active Monocular Vision. Orientador: Dr. Gert Kootstra. 2006. 20 f. TCC (Graduação) - Department of Artificial Intelligence, University of Groningen, 2006. Disponível em: https://www.research gate.net/publication/229048592_Robot_Localisation_Using_SIFT_and_Active_Monocu lar_Vision. Acesso em: 20 ago. 2022.

[13] MAHONY, N. O'. et al. Deep Learning vs. Traditional Computer Vision. 2019. In: arXiv:1910.13796. Acesso em: 20 ago. 2022.

[14] VOULODIMOS A. et al. Computational Intelligence and Neuroscience, 2018. Acesso em: 20 ago. 2022.

[15] PATHAK, A. R.; PANDEY, M.; RAUTARAY, S. Application of Deep Learning for Object Detection. Procedia Computer Science, v. 132, p. 1706 - 1717, 2018.

[16] OUYANG, W. et al. Deep Learning for Generic Object Detection: A Survey. 2018. In: arXiv:1809.02165. Acesso em: 20 ago. 2022.

[17] Lohia, Aditya; Kadam, Kalyani Dhananjay; Joshi, Rahul Raghvendra; and Bongale, Dr. Anupkumar M., "Bibliometric Analysis of One-stage and Two-stage Object Detection" (2021). Library Philosophy and Practice (e-journal). 4910

[18] REDMON, J. et al. You Only Look Once: Unified, Real-Time Object Detection. 08 jun. 2015.In: arXiv:1506.02640. Acesso em: 20 ago. 2022.

[19]CHABLANI, M. YOLO - You only look once, real time object detection explained. Towards Data Science, 21 ago. 2017. Disponível em: https://towardsdatascience.com/yolo-you-only-look-once-real-time-object-detection-exp lained-492dc9230006. Acesso em: 31 ago. 2022.

[20] REN, S. et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. 04 jun. 2015. in: arXiv:1506.01497.

[21] NELSON, J. Your Comprehensive Guide to the YOLO Family of Models. Disponível em: https://blog.roboflow.com/guide-to-yolo-models/ . Acesso em: 03 set. 2022.

[22] YOLOv5, Disponível em: https://github.com/ultralytics/yolov5. Acesso em: 31 ago. 2022.

[23] JIANG, P. et al. A Review of Yolo Algorithm Developments. Procedia Computer Science, v. 199, p. 1066 - 1073, 2022.

[24] KARIMI, G. Introduction to YOLO Algorithm for Object Detection, 2021. Disponível em: https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/. Acesso em: 03 set. 2022.

[25] COCO 128 Computer Vision Project. Disponível em: https://universe.roboflow.com/team-roboflow/coco-128. Acesso em 03 set. 2022

[26] LOGHMANI, M. R.; CAPUTO, B.; VINCZE, M. Recognizing Objects In-the-wild: Where Do We Stand? IEEE International Conference on Robotics and Automation (ICRA), 2018. Disponível em: https://www.acin.tuwien.ac.at/en/vision-for-robotics/software-tools/autonomous-robot-indoor-dataset/. Acesso em: 31 ago. 2022.

[27] FACELI, K. et al. Inteligência Artificial - Uma abordagem de aprendizado de máquina. Editora LTC, 2021.