Comparação de Algoritmos para Detecção de Nadadores Visando Automação de um Veículo Elétrico

Lucas Correa de Assis
Engenharia de Controle e Automação, Escola Politécnica, Pontifícia Universidade Católica do Paraná - PUCPR
Curitiba, Brasil
assis.correa@pucpr.edu.br

Roberto Zanetti Freire
Programa de Pós-Graduação em Engenharia de Produção e Sistemas - PPGEPS, Escola Politécnica, Pontifícia Universidade Católica do Paraná - PUCPR
Curitiba, Brasil
roberto.freire@pucpr.br

***Abstract*—Na análise biomecânica de nadadores, o uso de câmeras vem contribuindo para a melhora de rendimento esportivo. Porém grande parte dos vídeos são gravados de forma manual. Este trabalho tem como objetivo realizar uma análise comparativa entre os algoritmos de aprendizado profundo, *YOLO v4* e *YOLO v4 Tiny*, para detecção de nadadores, visando o projeto de um veículo capaz de seguir, de forma autônoma, os atletas ao longo da piscina. O treinamento dos algoritmos foi realizado com a base de dados *Video Diver Dataset* (VDD-C), que contém imagens anotadas de mergulhadores em piscinas e na costa de Barbados no Caribe. Para comparação entre os algoritmos, levou-se em consideração qualidade e tempo de detecção, baseando-se nas métricas de *Average Intersection Over Union* (IoU), *Recall*, *Average Precision (AP)* e tempo de latência. O algoritmo *YOLO v4*, apresentou superioridade na qualidade de detecção, atingindo 71,38% de *Average IoU*. Porém a *YOLO v4 Tiny*, apresentou um tempo de latência 6,93 vezes menor, mostrando ser uma opção para sistemas embarcados e aplicações em tempo real.**

***Index Terms*—Aprendizado Profundo, Análise Biomecânica de Nadadores, Detecção de objetos, YOLO.**

I. Introdução

A análise biomecânica, se tornou um instrumento de grande importância para ajudar a melhorar o desempenho de nadadores. Em [1], enfatiza-se a importância de registrá-los por meio de vídeo, principalmente no que se refere à alta demanda em termos de precisão associada a este tipo de análise [2]. Em sua maioria, os vídeos são gravados manualmente, acompanhando o nadador ao longo da piscina, ou por uma estrutura fixa de câmeras, conforme apresentado em [3] para estimar a pose humana em vídeos de nadadores.

Para a automação do processo de aquisição de imagens do nadador, é necessário desenvolver um sistema de visão computacional capaz de gerar referências de velocidade e posição para um veículo elétrico, que acompanhará o nadador ao longo da piscina.

Nesse contexto, este artigo considera o uso de técnicas de visão computacional para detecção de nadadores, como apresentado em [4], na abordagem *multi-related-targets*, e por [5] na análise de modelos de aprendizado profundo para detecção de mergulhadores. Assim, o presente trabalho tem como objetivo avaliar o algoritmo de aprendizado profundo *YOLO (You Only Look Once)* [6], a partir da análise comparativa entre as versões *YOLOV4* e *YOLOV4 Tiny* [7] para a detecção de nadadores.

A próxima seção deste artigo apresentam os trabalhos relacionados que serviram de inspiração para o desenvolvimento deste estudo. Na sequência, a seção III apresenta a base de dados, as técnicas adotadas e os critérios de avaliação de desempenho adotados neste trabalho. A seção IV apresenta os experimentos realizados e a seção V os resultados obtidos. Por fim, a seção VI traz as conclusões do presente estudo.

II. Trabalhos Relacionados

Os trabalhos apresentados nesta seção serviram de inspiração no desenvolvimento deste estudo.

No estudo apresentado em [4], foi desenvolvido o método *multi-related-targets* para *tracking* de nadadores. O método considera o nadador como um conjunto de *subtargets* que avançam na mesma velocidade, possibilitando a estimativa da localização de *subtargets* parcialmente ou totalmente oclusos. O método apresentou acurácia superior à abordagem clássica de *tracking* apenas da cabeça do nadador.

Já no trabalho apresentado em [1], foi investigado o método *contrario* para detecção de nadadores, que consiste na modelagem do ruído aleatório da água e detecção do movimento estruturado do nadador. Para isso, apresentaram a técnica DLT (*Direct Linear Transform*) para calibrar o vídeo, onde foi extraído a raia do nadador. Na sequência, foi aplicada a subtração de *frames* para detecção do movimento, decompondo o *frame* em um *grid* e classificando cada célula do *grid* como nadador ou ruído. Os testes conduzidos mostraram bons resultados para localização e *tracking* de

nadadores.

Na mesma linha dos trabalhos anteriores, em [5] foram avaliados modelos de aprendizado profundo para detecção de mergulhadores, produzindo um *dataset* com mais de 105.000 imagens anotadas de mergulhadores, que foi utilizado para treinar redes neurais como *Single Shot MultiBox Detector* (SSD), *Mobilenet*, *Faster R-CNN* e *YOLO*. Tomando como base os resultados desses testes, os autores reportaram resultados relevantes com o uso da SSD e da *YOLO v4 Tiny* para aplicações em tempo real.

No estudo discutido em [8], foi introduzido o modelo *YOLO LITE* desenvolvido para ser executado em dispositivos sem a presença de uma *Graphical Processing Unit* (GPU). O modelo foi treinado com a base de dados PASCAL VOC [9] e posteriormente com a base de dados COCO [10], atingindo valores de *mean Average Precision* (mAP) de 33,81% e 12,26%, respectivamente. A *YOLO LITE* atingiu a marca de 21 *Frames per Second* (FPS) em computadores sem GPU e 10 FPS quando implementado em um *site* com apenas 7 camadas. Esta taxa de FPS é 3,8 vezes mais rápida, quando comparado ao modelo mais rápido *SSD Mobilenet v1*.

Seguindo a mesma proposta do trabalho anterior. O trabalho apresentado em [11], propõe um novo framework chamado *Fast YOLO*, que acelera a *YOLO v2* [12] para ser capaz de realizar detecção de objetos em sistemas embarcados para aplicações em tempo real. Neste caso, produziram uma arquitetura otimizada que possui 2,8 vezes menos parâmetros e uma queda de apenas 2% nos valores de *Intersection Over Union* (IoU). Resultados experimentais mostram que a *Fast YOLO* é capaz de aumentar a velocidade de detecção em média 3,3 vezes, se comparado a *YOLO v2* originalmente. Possibilitando a detecção de vídeos com taxa média de 18 FPS em um sistema embarcado na *Nvidia Jetson TX1*.

III. Materiais e Métodos

Esta seção descreve a coleta e tratamento dos dados, os algoritmos de aprendizado profundo e as métricas de avaliação dos modelos.

A. Base de Dados

A base de dados utilizada neste artigo é denominada *Video Diver Dataset (VDD-C)* [5]. que contêm 100.000 imagens anotadas de mergulhadores em piscinas e na costa de Barbados no Caribe, ambas embaixo d'água. Como o foco deste artigo é detectar e classificar nadadores, utilizou-se apenas as imagens coletadas em piscinas (Fig. 1), totalizando 5.620 imagens.

As imagens foram divididas em três conjuntos, o conjunto de treinamento que representa $71,91\%$, o conjunto de teste que representa $14,86\%$, e o conjunto de validação que representa $13,23\%$. Conforme apresenta a Tabela I.

Fig. 1: Representação das imagens em piscinas do *dataset VDD-C*.

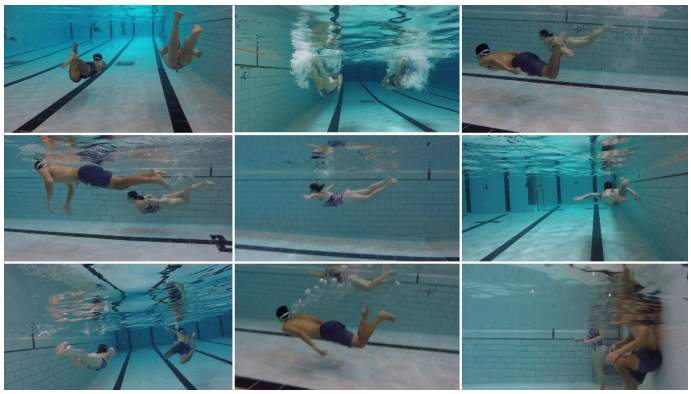


Tabela I: Distribuição dos dados

Categoria	Número de Imagens	Percentual
Treino	4.041	$71,91\%$
Teste	835	$14,86\%$
Validação	744	$13,23\%$

B. YOLO (You Only Look Once)

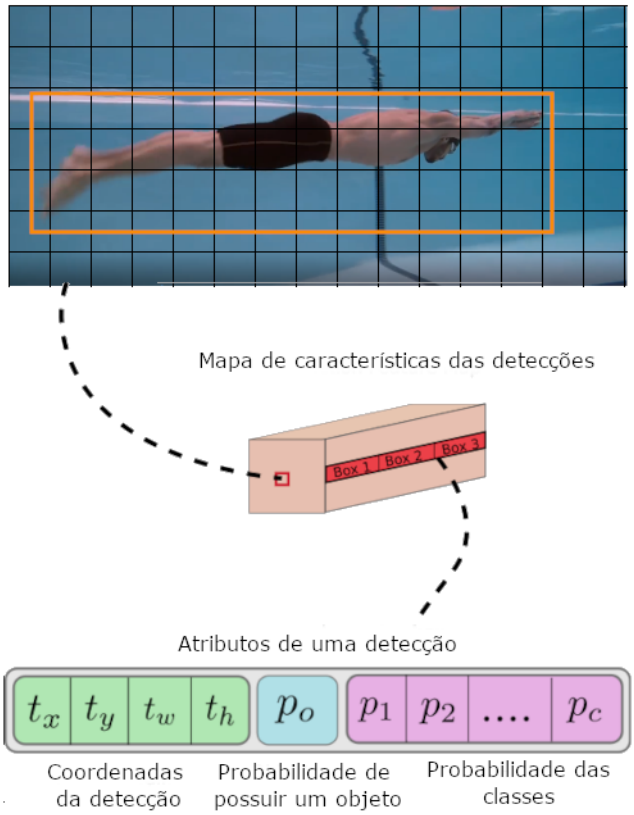
Redmon et al. (2016) introduziram uma nova abordagem para a detecção de objetos com o algoritmo *YOLO* [13], reformulando a solução para uma regressão única, diretamente dos pixels da imagem, para a localização e classificação dos objetos. Por apresentar apenas uma rede neural convolucional para toda imagem, a *YOLO* é considerada rápida e permite detecções em tempo real.

A *YOLOV4* é composta pela estrutura CSPDarknet53 [14], o módulo de *Spatial Pyramid Pooling* (SPP) [15], o ramo de *Path Aggregation Network* (PAN) [16], e a parte superior que é baseada na *YOLOV3* [6], que utiliza a arquitetura *Darknet-53* com 53 camadas convolucionais para classificação e outras 53 camadas de detecção, totalizando 106 camadas de profundidade.

Para a detecção, a imagem é dividida em um *grid*, onde cada bloco possui três regiões de detecção (*Boundig Boxes*). Cada uma delas possui suas coordenadas (*Box Co-ordinates*), a probabilidade de possuir um objeto (*Objectness Score*) e as probabilidades das classes (*Classes Score*), conforme Fig. 2.

Onde t_x, t_y, t_w e t_h representam a coordenada x do centro, a coordenada y do centro, a largura e altura relativa da região de detecção em relação a coordenada superior esquerda do bloco do *grid*, respectivamente. p_o representa a probabilidade da região possuir um objeto, seguido de p_1, p_2 ... p_c que representam as probabilidades de cada classe para aquela região, tendo c como número de classes.

Fig. 2: Representação do *grid* na imagem e dos dados contidos em uma região de detecção (adaptado de [17]).



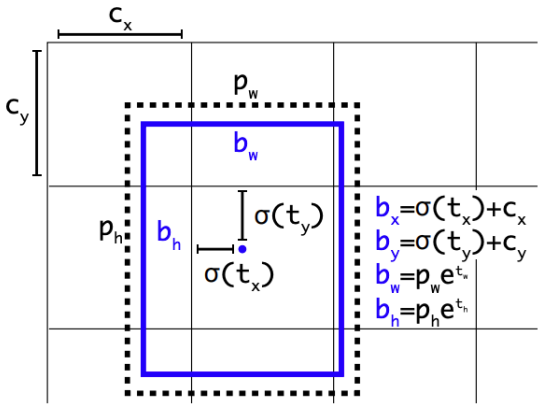
Como mencionado anteriormente, as coordenadas da região de detecção são relativas ao bloco do *grid*, logo as coordenadas absolutas podem ser obtidas conforme a Fig. 3. Onde C_x e C_y representam a coordenada absoluta do ponto superior esquerdo do bloco do *grid*. Os pontos b_x, b_y, b_w e b_h representam a coordenada x do centro, a coordenada y do centro, a largura e altura absoluta da região de detecção, respectivamente. Os valores de p_w e p_h representam a largura e altura da ancora da região de detecção. E o σ representa a função de ativação *sigmoid*.

A *YOLOV4 Tiny* é a versão comprimida da *YOLOV4*, possibilitando uma estrutura mais simples, para aplicações *mobile* e sistemas embarcados. A *YOLOV4 Tiny* tem como característica treinos e detecções mais rápidas devido a simplicidade da arquitetura. Ao contrário da *YOLOV4*, que possui 137 camadas convolucionais, a versão comprimida conta com 29 camadas convolucionais.

C. Métricas para Avaliação de Desempenho

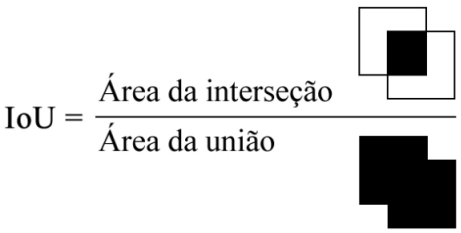
Para avaliar o desempenho das técnicas utilizadas neste estudo, é necessário utilizar métricas de avaliação já estabelecidas na literatura. Esta seção apresenta as métricas utilizadas.

Fig. 3: Representação das coordenadas da região de detecção relativas e absolutas [6].



A *Intersection Over Union* (IoU), é uma métrica capaz de representar a similaridade entre a localização detectada e a localização real do objeto [18]. A Fig. 4 mostra que para obter o valor percentual de IoU, é necessário dividir a área de interseção pela área de união entre a região detectada e a região real do objeto.

Fig. 4: Cálculo da *Intersection Over Union* (IoU) [19].



Porém para análise de uma época, utiliza-se o *Average IoU*, que pode ser calculado conforme a Eq. (1).

$$Average\ \ IoU = \frac{\sum IoU > T}{TP + FP} \tag{1}$$

Onde TP e FP representam os verdadeiros positivos e os falsos positivos, respectivamente. Para que uma classificação seja considerada um verdadeiro positivo (TP), a *Intersection Over Union* (IoU) deve ser maior que um limite pré-definido, $IoU > T$, onde T é o limite. Neste estudo, utilizou-se o valor de $0,5$ para T.

O *recall* é uma métrica utilizada para representar o percentual de detecções corretas e pode ser calculado conforme Eq. (2) [20].

$$recall = \frac{TP}{TP + FN} \tag{2}$$

A *precision* é uma métrica utilizada para avaliar a qualidade do modelo, para o cálculo pode ser visualizado na Eq. (3) [20].

$$precision = \frac{TP}{TP + FP} \tag{3}$$

A *Average Precision* (AP) é uma métrica utilizada para expressar a precisão média para todos os valores de *recall*, ou seja, é a área sob a curva *precision-recall* da seguinte forma:

$$AP = \int_0^1 p(r)dr \tag{4}$$

onde p e r, representam a *precision* e *recall* respectivamente.

A última métrica utilizada neste artigo é o tempo de latência em milissegundos (ms), para a detecção do nadador. O tempo de latência, é tempo que o algoritmo leva para processar a imagem a partir do momento em que é requisitada, ou seja, quanto menor o tempo de latência mais rápido será a detecção dos objetos.

IV. Experimentos

Esta seção tem por objetivo mostrar o equipamento utilizado, a configuração dos parâmetros para cada algoritmo e os detalhes de treinamento.

A. Treinamento dos Modelos

A Tabela II apresenta as especificações técnicas do computador utilizado para treinamento dos dois algoritmos.

Tabela II: Especificações técnicas do computador utilizado

Sistema Operacional	Windows 10 Education 64 bits
Processador	i9-10900F @ 2.80 GHz
Memória RAM	128 GB DDR4
Placa de Vídeo	RTX 2080 Ti (11 GB)

Para definir o número de épocas necessário para o treinamento do modelo, recomenda-se utilizar a multiplicação do número de classes por 2.000, porém, recomenda-se o número mínimo de 6.000 épocas [7]. Como o foco é a detecção de nadador, ou seja, apenas uma classe, a quantidade de filtros nas camadas convolucionais que precedem as camadas *YOLO* e o número de épocas devem ser ajustados. O número de filtros é calculado de acordo com a Eq. (5), logo utilizou-se 18 filtros e o mínimo de 6.000 épocas para treinamento.

$$filtros = 3 \times (num.classes + 5) \tag{5}$$

A Tabela III apresenta os hiper-parâmetros utilizados para o treinamento dos algoritmos, segundo os padrões apresentados em [7].

Tabela III: Especificações de treinamento

Parâmetro	YOLO v4	YOLO v4 Tiny
Batch	64	64
Tamanho da imagem	416×416	416×416
Momentum	$0,949$	$0,9$
Decay	$0,0005$	$0,005$
Learning Rate	$0,001$	$0,00261$
Anchor Boxes	9	9
Épocas de treinamento	6.000	6.000
Tempo de treinamento	4 horas	$1,2$ horas

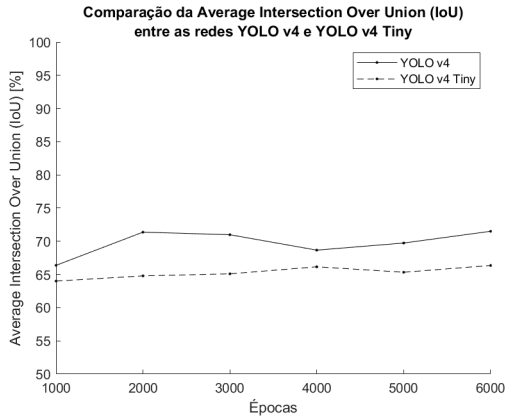
V. Resultados

Esta seção apresenta a discussão dos resultados obtidos, comparando os valores das métricas de avaliação para cada algoritmo.

Realizou-se a comparação dos resultados utilizando as métricas descritas na seção III-C, sendo elas a *Average IoU*, *Recall*, *Average Precision (AP)* e tempo de latência, utilizando o limite de IoU como 50%.

A Fig. 5 apresenta os valores obtidos de *Average IoU*, indicando uma semelhança entre os resultados. Contudo a *YOLO v4* atingiu seu valor máximo de $71,38\%$ com 2.000 épocas, já a *YOLO v4 Tiny* atingiu seu valor máximo de $66,16\%$ com 4.000 épocas. Portanto a *YOLO v4* apresenta vantagem de $5,22\%$ analisando os melhores resultados de ambos os algoritmos.

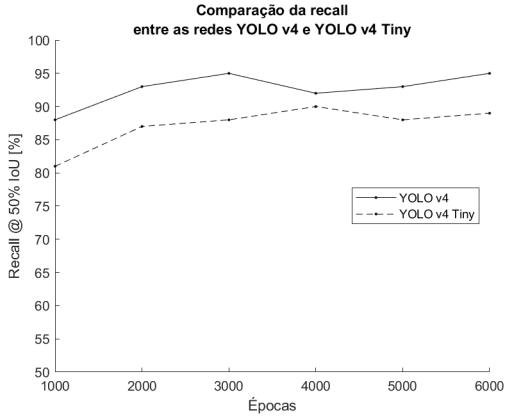
Fig. 5: Comparação da *Average Intersection Over Union (IoU)* entre as redes *YOLO v4* e *YOLO v4 Tiny*



A Fig. 6 mostra a comparação dos valores de *recall* entre os algoritmos. Novamente, observa-se a semelhança entre os valores e os limites máximos de ambos os casos, estes acima de 90%, sendo a *YOLO v4* com o melhor resultado, atingindo 95%.

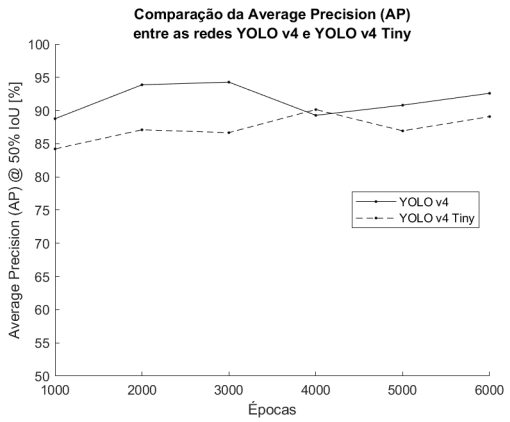
A comparação da *Average Precision* é apresentada na Fig. 7, os dados apresentam valores próximos, e os limites

Fig. 6: Comparação da *recall* entre as redes *YOLO v4* e *YOLO v4 Tiny*



máximos acima de 90% em ambos os casos. Novamente, a *YOLO v4* apresentou o melhor resultado, atingindo $94,28\%$.

Fig. 7: Comparação da *Average Precision (AP)* entre as redes *YOLO v4* e *YOLO v4 Tiny*

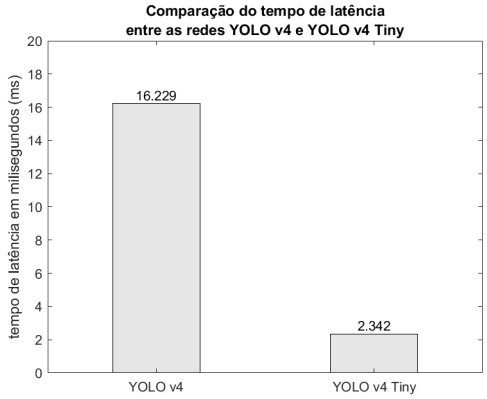


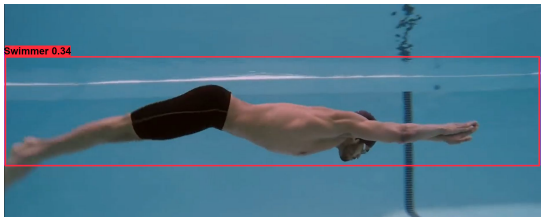
Já quando os resultados analisados levam em conta o tempo de latência (Fig. 8), observa-se a inversão do cenário, o tempo de latência da *YOLO v4 Tiny* chega a ser $6,93$ vezes menor que o tempo de latência da *YOLO v4*, mostrando vantagens em aplicações de tempo real.

Na Fig. 9, observa-se a detecção realizada pelos algoritmos. A *YOLO v4* apresentou uma região bem definida e uma probabilidade de identificar um nadador superior ao da versão comprimida. Contudo, a *YOLO v4 Tiny* também apresentou uma região de detecção aceitável.

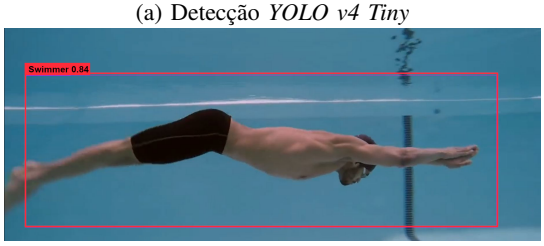
Também se realizou a análise para quando os nadadores estão parcialmente oclusos (Fig. 10). A versão comprimida conseguiu identificar apenas um dos nadadores oclusos, já a *YOLO v4* identificou ambos os nadadores parcialmente oclusos.

Fig. 8: Comparação do tempo de latência entre as redes *YOLO v4* e *YOLO v4 Tiny*





(a) Detecção *YOLO v4 Tiny*



(b) Detecção *YOLO v4*

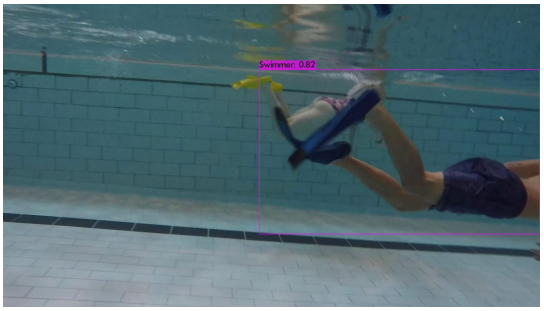
Fig. 9: Comparação da detecção realizadas pelos algoritmos

VI. Conclusão

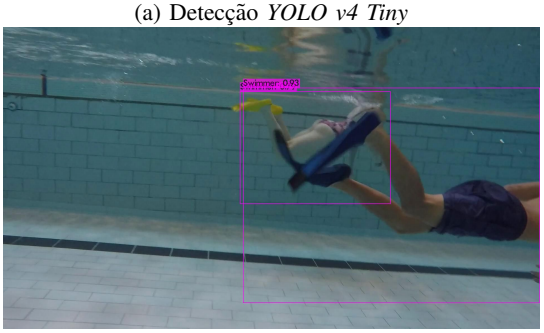
Este trabalho apresentou a coleta de dados de um *dataset* com imagens anotadas de nadadores em baixo d'água, treinamento e análise comparativa entre dois algoritmos de aprendizado profundo, *YOLO v4* e *YOLO v4 Tiny*. Para comparação, utilizou-se as seguintes métricas de desempenho: *Average IoU*, *Recall*, *Average Precision (AP)* e tempo de latência, utilizando o limite de IoU como 50%.

Ao analisar o desempenho das métricas, observa-se uma qualidade de detecção superior com a *YOLO v4*, o que era esperado por possuir uma estrutura mais complexa. Porém no tempo de latência, a versão comprimida usa a simplicidade de sua estrutura, para atingir valores até 6,93 vezes menor quando comparado com a *YOLO v4*, com o tempo de latência e tamanho reduzidos, a *YOLO v4 Tiny* se mostra apta para ser implementada em sistemas embarcados com foco em atividades de tempo real.

Observou-se falsos positivos para objetos como a divisória



(a) Detecção *YOLO v4 Tiny*



(b) Detecção *YOLO v4*

Fig. 10: Comparação da detecção de nadadores parcialmente oclusos realizadas pelos algoritmos

da raia e também dificuldade para detectar nadadores a partir de ângulos inferiores ou superiores, devido à falta de exemplares nas imagens da base de dados. Portanto enfatiza-se a importância dos trabalhos futuros, para expansão e melhoria das amostras da base de dados, tendo como objetivo aumentar a robustez e generalização do modelo.

Para os trabalhos futuros, propõe-se o aumento da base de dados utilizada para o treinamento, por meio da produção e anotação de mais vídeos de nadadores, buscando aumentar a robustez e generalização do modelo. Incorporação do algoritmo a um sistema embarcado, para que testes de qualidade e tempo de detecção possam ser conduzidos. E modelagem do controlador capaz de gerar referência de posição e velocidade para um veículo elétrico.

AGRADECIMENTOS

Os autores agradecem a Pontifícia Universidade Católica do Paraná (PUCPR) e a Fundação Araucária de apoio ao desenvolvimento científico e tecnológico do Paraná (PRONEX 042/2018) pelo financiamento desta pesquisa.

REFERÊNCIAS

[1] D. Benarab, T. Napoléon, A. Alfalou, A. Verney, and P. Hellard, "Swimmer's head detection based on a contrario and scaled composite jtc approaches," *International Journal of Optics*, vol. 2020, pp. 1–12, 04 2020.

[2] N. M. N. Adnan, M. N. A. A. Patar, H. Lee, S.-I. Yamamoto, L. Jong-Young, and J. Mahmud, "Biomechanical analysis using kinovea for sports application," *IOP Conference Series: Materials Science and Engineering*, vol. 342, p. 012097, apr 2018. [Online]. Available: https://doi.org/10.1088/1757-899x/342/1/012097

[3] M. Einfalt, D. Zecha, and R. Lienhart, "Activity-conditioned continuous human pose estimation for performance analysis of athletes using the example of swimming," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018, pp. 446–455.

[4] D. Benarab, T. Napoléon, A. Alfalou, A. Verney, and P. Hellard, "Optimized swimmer tracking system based on a novel multi-related-targets approach," *Optics and Lasers in Engineering*, vol. 89, pp. 195–202, 2017, 3DIM-DS 2015: Optical Image Processing in the context of 3D Imaging, Metrology, and Data Security. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0143816616300902

[5] K. de Langis, M. Fulton, and J. Sattar, "An analysis of deep object detectors for diver detection," 2020.

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

[7] A. Bochkovskiy, C. Wang, and H. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *CoRR*, vol. abs/2004.10934, 2020. [Online]. Available: https://arxiv.org/abs/2004.10934

[8] R. Huang, J. Pedoeem, and C. Chen, "Yolo-lite: A real-time object detection algorithm optimized for non-gpu computers," in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 2503–2510.

[9] M. Everingham, L. Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *Int. J. Comput. Vision*, vol. 88, no. 2, p. 303–338, Jun. 2010. [Online]. Available: https://doi.org/10.1007/s11263-009-0275-4

[10] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312

[11] M. J. Shafiee, B. Chywl, F. Li, and A. Wong, "Fast YOLO: A fast you only look once system for real-time embedded object detection in video," *CoRR*, vol. abs/1709.05943, 2017. [Online]. Available: http://arxiv.org/abs/1709.05943

[12] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," *CoRR*, vol. abs/1612.08242, 2016. [Online]. Available: http://arxiv.org/abs/1612.08242

[13] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2016.

[14] C. Wang, H. M. Liao, I. Yeh, Y. Wu, P. Chen, and J. Hsieh, "Cspnet: A new backbone that can enhance learning capability of CNN," *CoRR*, vol. abs/1911.11929, 2019. [Online]. Available: http://arxiv.org/abs/1911.11929

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *CoRR*, vol. abs/1406.4729, 2014. [Online]. Available: http://arxiv.org/abs/1406.4729

[16] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," *CoRR*, vol. abs/1803.01534, 2018. [Online]. Available: http://arxiv.org/abs/1803.01534

[17] A. Kathuria, "How to implement a yolo (v3) object detector from scratch in pytorch," https://blog.paperspace.com/how-to-implement-a-yolo-object-detector-in-pytorch/, 2018, accessed: 2021-06-19.

[18] M. Rahman and Y. Wang, "Optimizing intersection-over-union in deep neural networks for image segmentation," vol. 10072, 12 2016, pp. 234–244.

[19] A. Rosebrock, "Intersection over union (iou) for object detection," https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/, 2016, accessed: 2021-06-18.

[20] B. S. dos Santos and M. T. A. Steiner, "Metodologia baseada em kdd para a classificação de mulheres fumantes quanto ao consumo de cigarros industrializados," Ph.D. dissertation, Pontifícia Universidade Católica do Paraná, Curitiba, 2020, accessed: 2021-06-16.