Revista Brasileira
DE ESTUDOS CIENTÍFICOS

Comparative Analysis of Object Detection Algorithms for Pothole Recognition on Roads and Highways

Luiz Fernando Mello
luizf.mello@hotmail.com
Centro Universitário Avantis - UNIAVAN

Evandro L. Viapiana
evandro.luis.viapiana@gmail.com
0009-0008-4594-9988
Centro Universitário Avantis - UNIAVAN

Luiz Fernando M. Arruda
luiz.arruda@ieee.org
0000-0002-9159-2245
Centro Universitário Avantis - UNIAVAN

***Abstract* – This work presents a comparative analysis using the mAP metric of the YOLOv7 and SSD algorithms applied to the field of computer vision. The comparison is performed through the detection of potholes on highways and roads. In works defended by other authors, both techniques are considered for applications in computer vision. In this study, both models achieved promising results in pothole detection; however, it is observed that the YOLOv7 model reached an mAP metric of 80%, while the SSD model obtained an mAP of 73%. Thus, the results indicate that YOLOv7 is more efficient and accurate in pothole detection in this specific context.**

***Keywords* – Single Shot Multibox Detector, Pothole Detection, SSD, Computer Vision, You Only Look Once, YOLOv7**

Recebido em: 08/11/2025
Revisado em: 13/12/2025
Aprovado em: 15/12/2025

1 INTRODUCTION

Traffic accidents on roads and highways occur on a daily basis. Some of these accidents are caused by potholes or surface irregularities that arise from pavement deterioration or natural meteorological events, such as heavy rainfall. When vehicles encounter these potholes, they may cause significant inconvenience to drivers, including vehicle damage and/or serious accidents such as collisions and pedestrian run-overs, potentially resulting in fatal outcomes [1].

According to the open database of the Brazilian Federal Highway Police (PRF) [2], in 2021 a total of 3,974 accidents caused by potholes were recorded on highways (Fig. 1). Of these, 15.10% resulted in fatalities and 72.45% led to injured victims. In 2022, a 40% increase in pothole-related accidents was observed compared to 2021, of which 17.11% resulted in fatalities and 71.73% resulted in injured victims.

An analysis of accidents recorded up to the end of February 2023 shows that 1,312 incidents were caused by potholes on highways, of which 21.42% resulted in fatalities and 70.50% in injured victims. These data clearly indicate the urgent need for the adoption of mitigation measures to address this issue and enhance the safety of both drivers and pedestrians [2].

One effective strategy to reduce the number of accidents is

Fig. 1. Accident caused by potholes on public roads [3]

the implementation of intelligent autonomous systems for object detection. These systems are capable of addressing complex problems, such as pothole recognition, without relying on human intervention, by employing information input peripherals such as image and video acquisition systems. This approach is commonly referred to in the literature as computer vision or computer-assisted vision systems [4].

Fig. 2. Example of Computer Vision Application [5]

Computer vision is a subfield of artificial intelligence that aims to model and replicate human vision in computers through the use of *software* and *hardware*, as illustrated in Fig. 2. These systems extract meaningful information from images, videos, and other visual inputs in order to support decision-making processes or automate specific tasks [4].

Several object detection algorithms have been proposed in the literature, including the *Single Shot MultiBox Detector* (SSD) [6] and *You Only Look Once* (YOLO) [7], which belong to the category of single-stage detectors. These approaches are discussed in detail in previous studies [8]–

DOI: 10.5281/zenodo.18294543

[13].

In order to properly evaluate the performance of object detection algorithms, it is essential to select an appropriate evaluation metric. The *Mean Average Precision* (mAP) metric combines, in a weighted manner, three relevant metrics: *Precision*, *Recall*, and *Intersection over Union* (IoU). These metrics are widely recognized in the field of machine learning, making mAP a suitable choice for evaluating object detection algorithms [14].

2 OBJECT DETECTION ALGORITHMS

Object detection is considered a classical task in the field of computer vision, whose main objective is to classify and localize one or more objects in a static image (photograph) or a dynamic scene (video) [15]. Unlike image classification, which assigns a single class label to the entire image, object detection focuses on identifying specific objects by enclosing them within bounding boxes (Fig. 3).

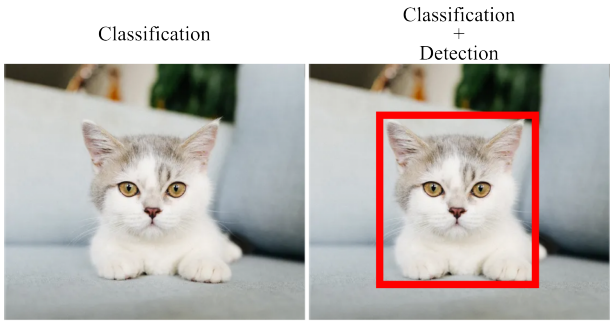


Fig. 3. Difference between image classification and object detection

Although techniques proposed more than 50 years ago exist to address computer vision problems—such as object detection based on edge detection (Fig. 4), as proposed by [16]—these approaches are not fully suitable for handling the challenges encountered in real-world scenarios. Consequently, over the years, new algorithms have been proposed with the aim of addressing this task in a more efficient and robust manner.

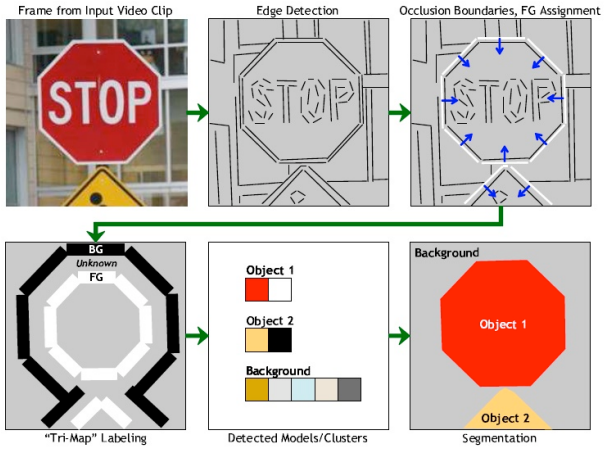


Fig. 4. Using boundaries for segmentation and recognition [17]

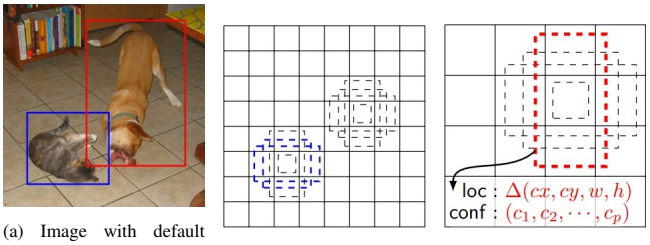
With the continuous evolution of *hardware* platforms and the rapid growth in the volume, velocity, and variety of generated data, the use of artificial neural network algorithms has become feasible. These algorithms play a crucial role in modern artificial intelligence-based object detection techniques [18], among which SSD and YOLO stand out.

2.1 *SINGLE SHOT MULTIBOX DETECTOR (SSD)*

The SSD algorithm, introduced by [6], is widely recognized for its ability to simultaneously perform object localization and classification tasks. The term *"Single Shot"* refers to the model's capability to execute both tasks in a single forward pass through the network. In addition, the term *"MultiBox"* alludes to the bounding box regression technique incorporated into SSD, which rapidly proposes bounding box coordinates regardless of the object class [19]. This model achieves a mAP score of 74.3% on the Pascal VOC2007 dataset [6].

Object detection in the SSD model is divided into two main stages: feature map extraction and the application of convolutional filters for object detection [19].

In the first stage of the SSD algorithm, fixed-size bounding boxes—also referred to as *default boxes*—are generated. The dimensions of these bounding boxes are precomputed by considering the sizes and locations of reference bounding boxes corresponding to each class in the dataset [20].



(a) Image with default boxes (b) 8×8 feature map (c) 4×4 feature map

Fig. 5. Default boxes of the input image [21]

Fig. 5 illustrates the operation of default bounding boxes on an input image. Fig. 5a shows the reference bounding boxes, while Figs. 5b and 5c present the feature maps that describe the original image partitioned into cells.

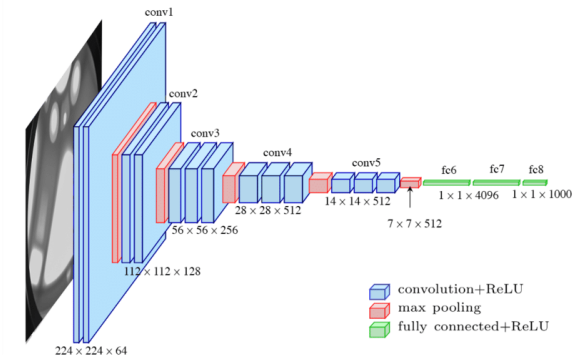


Fig. 6. Standard VGG16 network architecture [22]

DOI: 10.5281/zenodo.18294543

The initial layer of the neural network in the SSD algorithm employs a pretrained backbone network, such as VGG (Fig. 6), ResNet, or MobileNet. These networks are commonly used in classification tasks and have already been trained to extract representative feature maps [20].

The VGG architecture includes fully connected layers at the end of its structure. However, when addressing object detection problems, these fully connected layers are removed, and VGG16 is used exclusively for feature extraction. This feature extraction network is commonly referred to as the base network [23].

The *MultiBoxLoss* class (Fig. 7) is responsible for implementing the SSD loss function in detail. To ensure fidelity and accuracy, the loss function described in the original SSD paper was adopted as a reference.

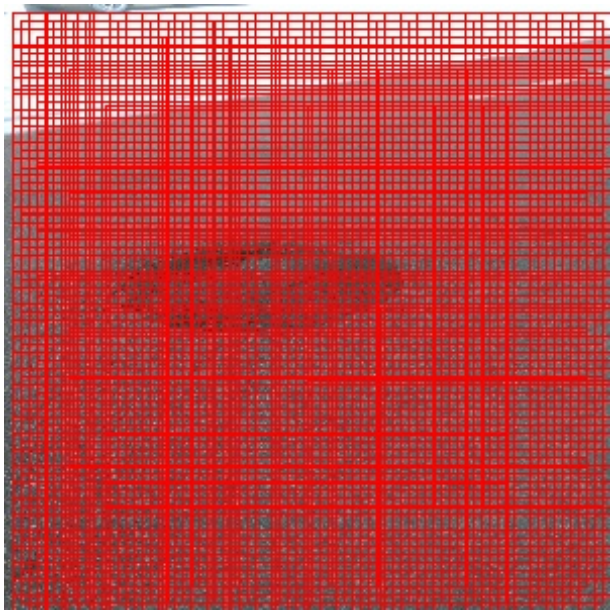


Fig. 7. Application of the MultiBox approach

The output of the base network is provided as input to six additional convolutional layers, where the feature maps progressively decrease in size, resulting in reduced spatial resolution. This process is performed to enable object detection at multiple scales [23], as illustrated in Fig. 8.

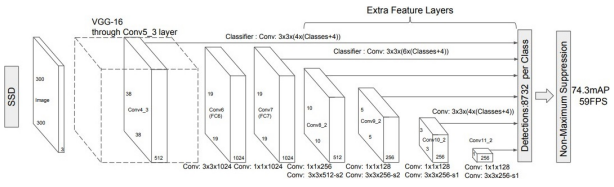


Fig. 8. SSD algorithm architecture

At each feature map, default detection boxes are defined, for which the network computes the probability of an object being present as well as variations in position, width, and height [6]. To perform class detection and bounding box regression, the algorithm applies convolutions on the feature maps using filters of size $3 \times 3 \times M \times N$, where M denotes the depth of the layer and N represents the number of applied filters. The convolutional layers use identical *stride* and *padding* values, ensuring that the output maintains the same width and height as the input feature map [23].

The number of filters applied to detect K classes on a feature map containing C default detection boxes is given by $(K+4) \cdot C$. This means that $K+4$ filters are used—one for each class and four additional filters to represent bounding box offsets—for each of the C boxes in the feature map [23].

All detections obtained across all cells of the convolutional layers are subsequently processed by a *Non-Maximum Suppression* (NMS) layer, whose objective is to eliminate overlapping detections and retain only a single bounding box per object present in the image. The NMS method employs the *Intersection over Union* (IoU) criterion, calculated as the ratio between the intersection area and the union area of two geometric regions. This criterion is used to determine whether a detection should be suppressed or retained during the filtering process [23].

The SSD loss function consists of two components: classification loss and localization loss. The classification loss measures the discrepancy between the predicted class probabilities and the ground truth object classes, computed using the cross-entropy loss function. In contrast, the localization loss evaluates the difference between the predicted bounding boxes and the ground truth bounding boxes. This loss component is computed using the *Smooth L1 Loss* function [6].

2.2 *YOU ONLY LOOK ONCE* (YOLO)

YOLO is a family of single-stage, real-time object detection algorithms that employ convolutional neural networks (CNNs) for object detection. The objective of the network is to perform object detection by formulating the task as a regression problem. In this way, a CNN is able to directly predict bounding boxes and class confidence scores [12].

The architecture selected for use in this work is YOLOv7. At the time this study was conducted, this architecture represents the state of the art in object detection for both static images and dynamic scenes [7].

According to [7], YOLOv7 is currently the fastest and most accurate real-time object detector. The algorithm achieved a significant performance improvement over previous YOLO-family models by substantially enhancing detection accuracy while maintaining high inference speed.

Improvements in speed and accuracy are achieved through the implementation of significant architectural refinements. Among these refinements are the adoption of the *Extended Efficient Layer Aggregation* (E-ELAN) technique, dimensional alignment for concatenation-based models, and reparameterization strategies. These modifications are designed to achieve an optimal balance between detection efficiency and accuracy [24].

The YOLOv7 algorithm consists of four main modules: the *input* module, the *backbone* network, the *head* network, and the *detection* (prediction) network, as illustrated in Fig. 9 [24].

DOI: 10.5281/zenodo.18294543

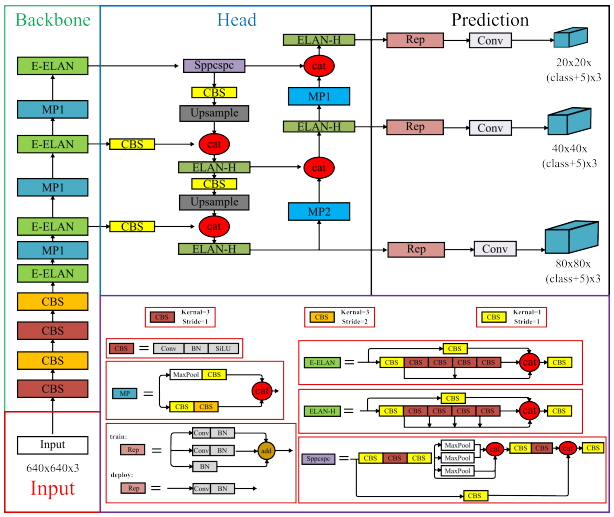


Fig. 9. YOLOv7 network architecture, adapted from [24]

The **input module** is responsible for ensuring that input images are uniformly resized to 640×640 pixels and comply with the input requirements of the main network. The preprocessing stage of YOLOv7 employs data augmentation techniques such as mosaic and hybrid augmentation and is based on the adaptive anchor box calculation method introduced in YOLOv5 [24].

The ***backbone*** **network** is composed of three main components: CBS, E-ELAN, and MP1. The CBS block consists of convolution, batch normalization, and SiLU activation functions. E-ELAN preserves the original ELAN design architecture and enhances the network's learning capability by guiding different computational blocks of feature groups to learn more diverse representations while maintaining the original gradient flow path [24].

The MP1 block is composed of CBS and MaxPool operations and is divided into two paths: an upper path and a lower path. In the upper path, MaxPool is used to reduce the spatial dimensions (height and width) of the feature maps by half, while a CBS block with 128 output channels reduces the number of channels by half [24].

In the lower path, a CBS block with a 1×1 kernel and stride is used to reduce the number of channels by half, followed by another CBS block with a 3×3 kernel and a 2×2 stride to reduce the spatial resolution by half. The features extracted from both paths are then combined using a concatenation (Cat) operation [24].

MaxPool aims to extract the maximum pixel value information within small local regions, whereas CBS extracts complete value information from local regions, thereby improving the network's feature extraction capability [24].

The ***head*** **network** is structured using a *Feature Pyramid Network* (FPN) architecture combined with the PANet design. This network is composed of multiple blocks of convolution, batch normalization, and SiLU activation (CBS blocks), along with the incorporation of a *Spatial Pyramid Pooling and Convolutional Spatial Pyramid* (SPPESPE) structure, the E-ELAN network, and the MaxPool-2 (MP2) block [24].

The SPPESPE structure enhances the receptive field of the network by embedding a *Convolutional Spatial Pyramid* (CSP) within the *Spatial Pyramid Pooling* (SPP) framework, along with a wide residual connection to facilitate optimization and feature extraction [24].

The ELAN-H layer fuses multiple feature layers based on E-ELAN, further improving feature extraction performance. The MP2 block has a structure similar to MP1, with a slight modification in the number of output channels [24].

The ***prediction*** **network** employs a structure known as REP to adjust the number of feature channels produced by the *head* network. A 1×1 convolution is applied to generate confidence scores, class predictions, and bounding box coordinates. The REP structure is inspired by RepVGG [25] and adopts a residual design to facilitate the training process. During the inference stage, this residual structure can be simplified into a single convolutional layer, thereby reducing network complexity without compromising detection performance [24]. After bounding box prediction, the YOLO algorithm applies the same post-processing technique used in SSD, namely *non-maximum suppression* (NMS).

In the post-processing stage, YOLO employs the NMS technique to eliminate overlapping and duplicate detections, ensuring that only the most confident bounding box is retained for each detected object in the image.

2.3 EVALUATION METRICS

To evaluate object detection models, evaluation metrics are employed by considering the ground-truth object locations in an image and the predictions generated by the model, enabling the assessment of model accuracy. One of the most widely used metrics for this purpose is the *Mean Average Precision* (mAP). The mAP metric combines three fundamental measures: *precision*, *recall*, and *Intersection over Union* (IoU) [26].

2.3.1 Precision

The precision metric is defined as the ratio between true positives (T_P) and the total number of predictions classified as positive by the model, which include true positives and false positives (F_P). Therefore, precision indicates the proportion of relevant results among all positive predictions [12].

$$Precision = \frac{T_P}{T_P + F_P} \tag{1}$$

2.3.2 Recall

The *recall* metric is defined as the ratio between T_P and the sum of T_P and false negatives (F_N). Thus, recall represents the ability of a model to identify all relevant objects in a dataset, i.e., the proportion of total relevant instances that are correctly detected by the algorithm [12].

DOI: 10.5281/zenodo.18294543

$$Recall = \frac{T_P}{T_P + F_N} \quad (2)$$

2.3.3 Intersection over Union

The IoU metric is widely used to evaluate the quality of object detection models. Images in the dataset are previously annotated to specify the ground-truth object locations. Model performance is then assessed by computing the intersection between the predicted bounding box and the ground-truth bounding box divided by the union of both bounding box areas, as illustrated in Fig. 10 [12].

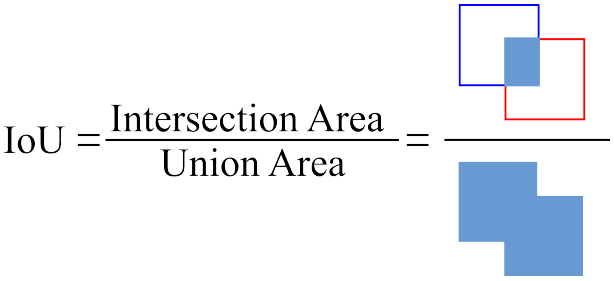


Fig. 10. IoU metric formulation [12]

The IoU metric produces values ranging from 0 to 1, where higher values indicate better model performance [12].

2.3.4 *Mean Average Precision*

The mAP metric corresponds to the mean of the *Average Precision* (AP) values computed for each object class in an object detection problem. It is a widely adopted metric for evaluating the overall performance of object detection systems. The mAP value ranges from 0 to 1 (or 0–100%), with values closer to 1 indicating superior detection performance [27].

The *Average Precision* (AP) is computed as the area under the *Precision–Recall* curve. To compute AP, a condition must be established to determine whether a prediction is considered correct. This condition is defined by an IoU threshold (*threshold*), whereby a detection is regarded as correct if its IoU value is equal to or greater than the specified threshold [27].

For example, consider a predefined IoU threshold of 0.5 for classifying a prediction. If the IoU value is greater than or equal to 0.5, the object detection is classified as a true positive (TP), which occurs when the detector correctly predicts both the object class and its location. Conversely, if the IoU value is lower than 0.5, the detection is considered incorrect and is classified as a false positive (FP). A false positive represents a detection error and occurs when the classifier incorrectly predicts an object.

A true negative (TN) occurs when the detector correctly predicts that no object of interest is present in a given region, and indeed no object exists. On the other hand, a false negative (FN), also considered a detection error, occurs when the detector predicts that no object is present in a region where an object actually exists [28].

3 APPLICATION

This section presents the methodology adopted for the development of this work, which is divided into three main parts: Dataset Acquisition, Training of the SSD Algorithm, and Training of the YOLO Algorithm. The complete implementation of the comparative analysis is publicly available on GitHub[1].

3.1 DATASET ACQUISITION

To perform the comparative analysis between the SSD and YOLO models, a real-world dataset was collected from the Kaggle data science competition platform[2].

The use of a real dataset is essential to properly evaluate the performance and effectiveness of the SSD and YOLO models under practical application conditions. The selected Kaggle dataset presents relevant characteristics for comparative analysis, providing a diverse set of samples for both training and testing.

The dataset consists of a collection of 665 images containing potholes on roads and highways, with each image accompanied by its corresponding annotated bounding boxes. The annotations follow the widely adopted Pascal VOC format in the computer vision community, ensuring a standardized and easily interpretable structure. The dataset is already divided into training and test subsets.

In total, 80% of the images were allocated to the training set, allowing the models used in this work to be trained effectively. The remaining 20% were reserved for the test set, which was used to evaluate model performance and assess their generalization capability on previously unseen data.

This dataset partitioning strategy is a common practice in machine learning and computer vision, ensuring an objective evaluation of model performance. From these evaluations, reliable metrics such as mAP and IoU can be obtained, which are essential for assessing the effectiveness and reliability of the models in pothole detection on roads and highways.

3.2 TRAINING OF THE SSD ALGORITHM

To implement the SSD algorithm, the original SSD paper was used as the primary reference, along with the official repository provided by Wei Liu and hosted on GitHub[3].

The algorithm implementation followed the object-oriented programming paradigm, using classes as the fundamental structural elements. All implemented files are available in the repository referenced in [1], allowing full access to and inspection of the source code.

After implementing the SSD algorithm, model training was initiated using the pothole dataset described in Table 1. During training, data augmentation techniques were not applied to generate synthetic samples. However, transfer learning was adopted by leveraging the pretrained weights

[1] https://github.com/luizfmello01/tcc_ssd_yolo

[2] https://www.kaggle.com/datasets/chitholian/annotated-potholes-dataset

[3] https://github.com/weiliu89/caffe/tree/ssd

DOI: 10.5281/zenodo.18294543

Tab. 1 SSD algorithm training

Starting training				
Updating learning rate to 0.0001				
Epoch [0]	batch_idx [0]	loc_loss [2.65]	cls_loss [7.44]	total_loss [10.09]
Epoch [0]	batch_idx [1]	loc_loss [2.86]	cls_loss [7.09]	total_loss [9.96]
Epoch [0]	batch_idx [2]	loc_loss [2.92]	cls_loss [6.83]	total_loss [9.75]
Epoch [0]	batch_idx [3]	loc_loss [2.93]	cls_loss [6.53]	total_loss [9.47]
Epoch [0]	batch_idx [4]	loc_loss [2.92]	cls_loss [6.29]	total_loss [9.21]
Epoch [0]	batch_idx [5]	loc_loss [2.88]	cls_loss [6.04]	total_loss [8.92]
Epoch [0]	batch_idx [6]	loc_loss [2.90]	cls_loss [5.81]	total_loss [8.71]
Epoch [0]	batch_idx [7]	loc_loss [2.87]	cls_loss [5.61]	total_loss [8.48]
Epoch [0]	batch_idx [8]	loc_loss [2.87]	cls_loss [5.41]	total_loss [8.28]
Epoch [0]	batch_idx [9]	loc_loss [2.87]	cls_loss [5.23]	total_loss [8.10]
Epoch [0]	batch_idx [10]	loc_loss [2.87]	cls_loss [5.07]	total_loss [7.94]

of the VGG-16 network, which was originally trained on the ImageNet dataset.

The trained model was subsequently evaluated through inference on new images that were not previously seen during the training phase.

3.3 TRAINING OF THE YOLO ALGORITHM

The first step in initiating the YOLOv7 training process consisted of cloning the official YOLOv7 repository available on GitHub[4]. This process creates an exact copy of all files and directories contained in the repository, enabling access to the resources required for training.

The official YOLOv7 repository was originally trained using the MS COCO dataset, whose annotation format differs from that of the dataset used in this work. To address this incompatibility, two dedicated *notebooks* were developed to convert the dataset from the Pascal VOC format to the format supported by YOLOv7.

During the adaptation process to the YOLO format, it was necessary to copy the *ImageSets*, *Annotations*, and *JPEGImages* directories from the standard Pascal VOC structure into a new directory. This new directory follows a simplified structure containing only the folders `train/images`, `train/annotations`, `test/images`, and `test/annotations`. This simplification facilitates the transformation of the data into the YOLO-compatible format.

After executing the aforementioned *notebook*, the resulting images were organized into the `train/images` and `test/images` directories, while the object annotations were grouped into the `train/annotations` and `test/annotations` directories. This directory structure properly separates images and annotations according to the training and testing partitions.

In the subsequent step, the object annotations were converted from the Pascal VOC standard format into the YOLO format. To perform this transformation on each XML file individually, a separate *notebook* was developed. This *notebook* is comprehensive and includes all the steps required for correct conversion, such as reading XML files, extracting relevant information (e.g., annotated object coordinates), and generating the corresponding YOLO-format text files.

During the training of the YOLO algorithm, data augmentation techniques were employed to generate synthetic samples, and transfer learning was adopted by leveraging pretrained weights obtained from the MS COCO dataset. Table 2 presents the initial training process of the YOLOv7 algorithm using the pothole dataset.

Tab. 2 YOLOv7 algorithm training

Epoch	Gpu_Mem	Box	Obj	Cls	Total	Labels	Img Size	mAP@.5
0/299	9.93G	0.089	0.008264	0	0.097	24	320	0.002
1/299	9.97G	0.082	0.008746	0	0.091	34	320	0.008
2/299	9.97G	0.079	0.008431	0	0.087	9	320	0.009
3/299	9.97G	0.074	0.009733	0	0.084	32	320	0.007
4/299	9.93G	0.071	0.009912	0	0.081	45	320	0.001

4 RESULTS

To evaluate the models, a training process consisting of a total of 300 epochs was conducted. In addition, 133 images were considered for evaluation in both models. It is worth noting that the dataset used in this study includes exclusively a single class, namely potholes.

Tab. 3 **Model evaluation results**

Algorithm	Epochs	Total Images	mAP
SSD	300	133	0.736
YOLOv7	300	133	0.806

Table 3 presents the evaluation results of the models, including the number of training epochs, the total number of images used during the evaluation phase, and the evaluation metric known as mAP, which was obtained during the analysis. After performing the evaluation, the results indicate that the YOLOv7 algorithm outperformed SSD in terms of the mAP metric.

4.1 POTHOLE DETECTION IN TEST SET IMAGES

In Figs. 11a and 11b, the YOLO algorithm identified a total of six potholes on the roadway. However, in Fig. 11b, the model incorrectly classified an object as a pothole when, in fact, it corresponds to a minor surface irregularity. Despite this misclassification, it is important to emphasize that, overall, the algorithm demonstrated satisfactory performance in pothole detection.

On the other hand, Figs. 12a and 12b show that the SSD algorithm exhibited inferior performance in pothole detection when compared to YOLOv7. Specifically, SSD identified a total of three potholes in the analyzed images, failing to detect two potholes present on the roadways.

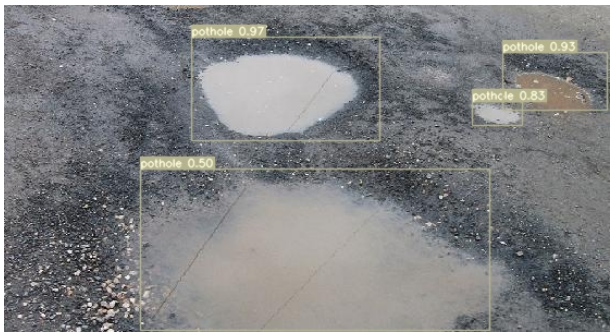
Therefore, it is evident that both models were successful in detecting potholes in images not previously seen during the training process, highlighting the models' ability to effectively learn the underlying pothole detection patterns.

4.2 POTHOLE DETECTION IN NEW IMAGES CAPTURED BY THE AUTHOR USING A MOBILE PHONE

To assess model generalization capability, four new images (Figs. 13a, 13b, 14a, and 14b) were captured in the region of

[4] `https://github.com/WongKinYiu/yolov7`

DOI: 10.5281/zenodo.18294543

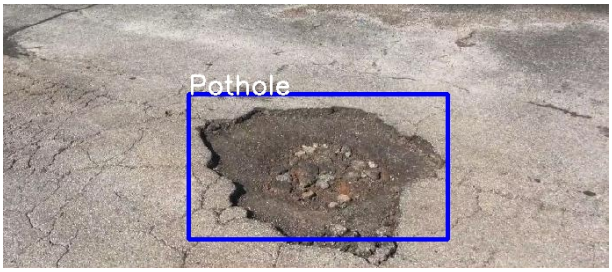


(a) Pothole detection in img-449 using YOLOv7

(b) Pothole detection in img-464 using YOLOv7

Fig. 11. Analysis of test set images using YOLOv7

(a) Pothole detection in img-449 using *Single Shot Detector*



(b) Pothole detection in img-464 using *Single Shot Detector*

Fig. 12. Analysis of test set images using the *Single Shot Detector*

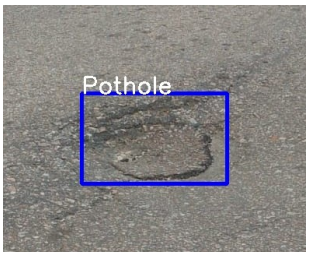
Balneário Camboriú. The results show that both algorithms successfully detected potholes on the roads, as illustrated in Figs. 13 and 14.

5 CONCLUSIONS

Based on the obtained results, it can be concluded that both models performed effectively in the detection of potholes on roadways. The YOLOv7 model, in particular, demonstrated superior performance compared to the SSD model, as evidenced by the mAP metric, achieving an mAP of 80% in contrast to the 73% obtained by SSD. This performance gap can be attributed to YOLOv7's greater effectiveness in handling variations such as scale, illumination, occlusion, and other challenging factors. These aspects are commonly encountered in object detection tasks within dynamic, complex, and uncontrolled environments, such as road and highway scenarios.

Nevertheless, it is important to note that both models present potential for further improvement in terms of accuracy. The use of a larger and more diverse training dataset could enable the models to learn from a wider variety of pothole examples, including those with less common characteristics. In addition, the application of advanced data augmentation techniques—such as geometric transformations and illumination variations, which were not explored in this study—could enrich the training dataset without the need for manual image collection.

Another relevant aspect concerns the optimization of model parameters. Fine-tuning parameters such as learning rate, batch size, and number of training epochs may have a significant impact on model performance. Furthermore, experiments involving different network architectures and attention mechanisms may lead to additional improvements in pothole detection accuracy.



(a) Image analysis using the *Single Shot Detector*

(b) Image analysis using the *Single Shot Detector*

Fig. 13. Pothole detection in new images captured by the author using a mobile phone

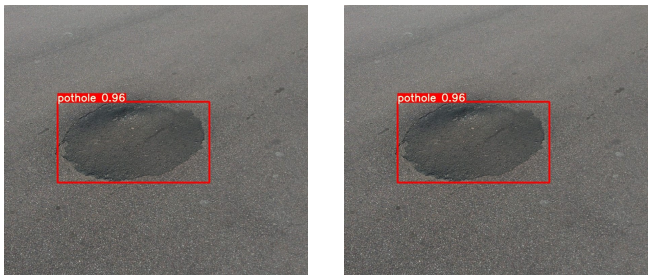
In summary, the current results are promising, yet there remains room for enhancement. Future research focused on expanding the dataset, evaluating data augmentation strategies, and optimizing model parameters has the potential to further improve the performance of pothole detection models applied to road and highway environments.

REFERENCES

[1] R. S. Rodrigues, *Detecção de buracos em estradas: uma abordagem automatizada baseada na transformada Wavelet de Haar*, Master's thesis, UNIVERSIDADE FEDERAL DE SANTA MARIA, Santa Maria, RS, Jun. 2019.

[2] PRF, "Documento CSV de Acidentes 2023 (Agrupados por pessoa - Todas as causas e tipos de acidentes)", `https://www.gov.br/prf/pt-br/acesso-a-informacao/dados-abertos/dados-abertos-da-prf`, 2023.

[3] G. Almeida, "Após acidente por causa de buraco, moradores reclamam de abandono no Caiobá", , 2017, URL: `https://www.campograndenews.com.br/direto-das-ruas/apos-acidente-por-causa-de-buraco-moradores-reclamam-de-abandono-no-caioba`.



(a) Image analysis using YOLO (b) Image analysis using YOLO

Fig. 14. Pothole detection in new images captured by the author using a mobile phone

[4] D. de Milano, L. B. Honorato, "Visao computacional", , 2010.

[5] R. Zrenner, "Explorando o potencial prático da visão computacional: o que é, aplicação e possibilidades", , 2023, URL: `https://blog.pti.org.br/visao-computacional/`.

[6] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A. C. Berg, "SSD: Single Shot MultiBox Detector", *in* B. Leibe, J. Matas, N. Sebe, M. Welling, eds., *Computer Vision – ECCV 2016*, pp. 21–37, Springer International Publishing, Cham, 2016.

[7] C.-Y. Wang, A. Bochkovskiy, H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors", *arXiv preprint arXiv:220702696*, 2022.

[8] P. F. N. Ferreira, D. L. G. Junior, A. M. Santos, "Detecçao de uso de máscaras em ambientes fechados com MobileNetV2 e Single Shot Multibox Detector", *in Anais da X Escola Regional de Computação do Ceará, Maranhão e Piauí*, pp. 61–68, SBC, 2022.

[9] M. A. de Sousa, K. de SOUZA, J. CAMARGO NETO, S. TERNES, I. YANO, "Usando a rede neural SSD para identificar frutos verdes em pomares de laranja.", *in "CONGRESSO INTERINSTITUCIONAL DE INICIAÇÃO CIENTÍFICA"*, pp. 1–10, Embrapa Agricultura Digital, "Campinas - SC", 2021.

[10] P. F. C. Araujo, "Protótipo de braço robótico capaz de capturar objetos fazendo uso de processamento de imagens e redes neurais", *RE3C-Revista Eletrônica Científica de Ciência da Computação*, vol. 13, no. 1, 2018.

[11] I. Costa, E. Silva Jr, A. Rodrigues, L. Angeloni, E. Dias, "Avaliação do Processo para Embarcar uma Rede Neural Baseada em YOLO Utilizando um Acelerador de Hardware Dedicado", *in Anais Estendidos do X Simpósio Brasileiro de Engenharia de Sistemas Computacionais*, pp. 174–179, SBC, 2020.

[12] A. M. Araújo, *Detecção e destaque em vídeo de objetos utilizando YOLO*, Master's thesis, Universidade Federal da Paraíba - UFPB, João Pessoa , PB, Oct. 2022.

[13] L. Barbado, L. M. R. dos Santos, M. D. Matrakas, J. Moreira, "Aplicação da rede convolucional Yolo para análise de fluxo de veículos", *in Anais do XIX Congresso Latino-Americano de Software Livre e Tecnologias Abertas*, pp. 43–49, SBC, 2022.

[14] A. C. Banderchuk, *Detecção de defeitos de fabricação em placas de circuito impresso utilizando técnicas de aprendizado profundo*, Instituto Federal de Santa Catarina - IFSC, Florianópolis, SC, May 2021.

[15] Z. Li, F. Liu, W. Yang, S. Peng, J. Zhou, "A survey of convolutional neural networks: analysis, applications, and prospects", *IEEE Transactions on neural networks and learning systems*, 2021.

[16] Y. Yakimovsky, "Boundary and object detection in real-world images", *Journal of the ACM (JACM)*, vol. 23, no. 4, pp. 599–618, 1976.

[17] M. Zakariah, K. A. AlShalfan, "Image Boundary, Corner, and Edge Detection: Past, Present, and Future", , 2020, URL: `https://api.semanticscholar.org/CorpusID:231771734`.

[18] J. V. E. Gomes, *Detecção de objetos com a arquitetura YOLO.*, Universidade Federal de Ouro Preto - UFOP, João Monlevade, MG, Outubro 2022.

[19] A. C. Rios, *Comparação dos algoritmos YOLOv3 e SSD para identificação de objetos em imagens, em um dataset para navegação de robôs em ambientes interiores*, Universidade Federal de Santa Maria - UFSM, Santa Maria, RS, Feb. 2022.

[20] J. O. G. CALIS, *APLICAÇÃO DE REDES NEURAIS CONVOLUCIONAIS PARA RECONHECIMENTO AUTOMÁTICO DE PLACAS DE VEÍCULOS*, Universidade Estadual Paulista "Júlio de Mesquita Filho, São José do Rio Preto - SP, Dec. 2018.

[21] A. Ivanov, N. Chivarov, "Methods for object recognition and classification for tele-controlled service robots", *IOP Conference Series: Materials Science and Engineering*, vol. 878, p. 012005, 07 2020, doi:10.1088/1757-899X/878/1/012005.

[22] M. Ferguson, R. ak, Y.-T. Lee, K. Law, "Automatic localization of casting defects with convolutional neural networks", pp. 1726–1735, 12 2017, doi:10.1109/BigData.2017.8258115.

[23] R. V. de Sousa Junior, *Estudo de métodos de inteligência computacional para detecção de humanos em imagens de webcam*, Universidade Federal do Ceará - UFC, Fortaleza, CE, 2018.

[24] K. Liu, Q. Sun, D. Sun, L. Peng, M. Yang, N. Wang, "Underwater target detection based on improved YOLOv7", *Journal of Marine Science and Engineering*, vol. 11, no. 3, p. 677, 2023.

[25] X. Ding, X. Zhang, N. Ma, J. Han, G. Ding, J. Sun, "RepVGG: Making VGG-style ConvNets Great Again", *CoRR*, vol. abs/2101.03697, 2021, URL: `https://arxiv.org/abs/2101.03697`, `2101.03697`.

[26] A. B. Nogueira, *et al.*, "Análise de viabilidade no uso de Deep Learning para contagem de pessoas com câmeras de segurança", , 2018.

[27] A. C. Serra, J. V. F. França, J. A. de Sousa, R. W. de Sousa Costa, I. F. S. da Silva, S. V. da Rocha, A. C. de Paiva, A. C. Silva, E. M. G. Monteiro, I. F. S. da Silva, *et al.*, "Segmentaçao semântica de medidores de energia elétrica e componentes de identificaçao", *Brazilian Applied Science Review*, vol. 4, no. 3, pp. 2002–2013, 2020.

[28] R. R. Leocádio, A. K. Segundo, J. R. Souza, J. Galaschi-Teixeira, P. de Souza, G. Pessin, "Detecçao de abelhas nativas em colmeias em campo utilizando visao computacional", *in Anais do XII Workshop de Computação Aplicada à Gestão do Meio Ambiente e Recursos Naturais*, pp. 59–68, SBC, 2021.