EMERGING TECH CONFERENCE – Edge Intelligence

Volume 03, 2024, Pages 53 – 58

Proceedings of Emerging Tech Conference: Edge Intelligence 2024

Comprehensive comparison of YOLO-based object detection in edge applications: A use case on free parking spots

Vasileios Karvelas and Christoforos Kachris
University of West Attica Department of Electrical and Electronics Engineering Athens, Greece

Abstract

When it comes to edge applications that require object detection based on computer vision, there are several models that can be used. However there is not an easy way to identify what is the best model for each application. In this paper we present a com prehensive comparison of several YOLO-based models for object detection targeting edge applications using single board computers (Raspberry pi). We compare several versions of YOLO (YOLOv5, v8 and v10) in terms of accuracy, inference time and training time. For the specific comparison we evaluate the performance for an edge application where the system has to identify the empty parking spots in a parking lot. The comparison allows developers to select the best model based on the application requirements.

1 Introduction

Object detection is a fundamental task in computer vision that involves identifying and locating objects within an image or video. Unlike image classification, which simply categorizes an image as a whole, object detection not only determines the presence of objects but also provides their precise positions through bounding boxes. This capability is crucial for a wide range of applications, from autonomous driving to real-time video surveillance.

Among various approaches to object detection, single shot models have gained significant attention due to their efficiency and speed. Unlike multi-stage detectors that rely on multiple passes over the image, single shot models such as SSD (Single Shot MultiBox Detector) [3] and YOLO (You Only Look Once) [6] predict bounding boxes and class probabilities in a single forward pass through the network. This allows for real-time performance, making them ideal for time-sensitive tasks.

YOLO-based models, in particular, is a widely-used model in the domain of real-time object detection. Starting from the original YOLO model in 2015 [6], which introduced the concept of treating detection as a regression problem, these models have evolved through several iterations, each improving in accuracy, speed, and adaptability.

The idea is to segment the image into a grid and predict the existence of a bounding box for each of the classes we are considering. When it comes to labeling the data, a grid cell is labeled to contain an object only if the center of the box is in it. If the grid cell contains a center, the ”objectness” is labeled 1 and 0 otherwise. The model will try to predict the probability that a grid cell contains a center. If it contains a center or multiple ones, each class related to those centers will be labeled with 1s.

The success of YOLO-based models lies in their ability to balance computational efficiency with detection precision, pushing the boundaries of what is achievable in real-time object detection.

In this paper we present a comparison of several YOLO versions in terms of accuracy, inference througput and training time for a simple application ported to a Raspberry Pi board for edge applications. Specifically we compare several models for an application that need to identify the empty spots in a parking lot. First we train the model using a large dataset of parking areas and then we evaluate the performance of the models on inference in terms of accuracy, latency and training time.

2 Related work

There are several papers that have been presented looking to compare object detection models. For example, in [4] there is a comparison on the detection accuracy and speed measurements of several state-of-the-art models such as RetinaNet, GHM Faster R-CNN, Grid R-CNN, Double Head R-CNN , and Cascade R-CNN-for the task of object detection in commercial EO satellite imagery.

In [1] a comparison of YOLO-based models has been performed. However, the specific paper only compares the different versions of the YOLOv5 model and does not examine later YOLO models.

In [2], they compare several models for computer vision but the main focus is on privacy exploring the impact of data anonymization on deep learning models.

Naftali et al. [5] have also presented a comparison for object detection algorithms for street-level objects. This paper compares various one-stage detector algorithms such as SSD MobileNetv2 FPN-lite, YOLOv3, YOLOv4, YOLOv5l, and YOLOv5s for street-level object detection within real-time images. However it does not compare the latest versions of YOLO model.

As YOLO-based object detection has emerged as one of the most promising model for edge applications, we compare the performance of several newer versions in terms of accuracy, inference latency and training time for edge applications and specifically for a widely-used typical Raspberry pi 4 board.

3 Comparison of the models

For the comparison, we selected a typical edge application that tries to identify the number of empty spots in a parking lot. The system takes as input the feed from a camera in a parking lot and tries to identify the empty parking spots. The training of the model has been performed using a dataset of 1600 images of parking lots. Out of the 1600 images, 1200 of them (75% of the images) were used for the training part, 200 images were used for the validation and 200 images for the testing.

Since the comparison is targeting edge applications that are using low-power processors, only the smaller versions of the YOLO models have been studied (nano and small versions) for the comparison. Also we evaluated the most widely used versions of YOLO models and specifically Yolov5, Yolov8 and Yolov10.

- **YOLOv5:** A highly popular version of YOLO (You Only Look Once) developed by Ultralytics, optimized for speed and accuracy in object detection, widely used in various computer vision applications.
- **YOLOv8:** The latest official release from Ultralytics, YOLOv8 builds upon previous versions with improved architecture, better performance, and a user-friendly interface for training, deployment, and inference.

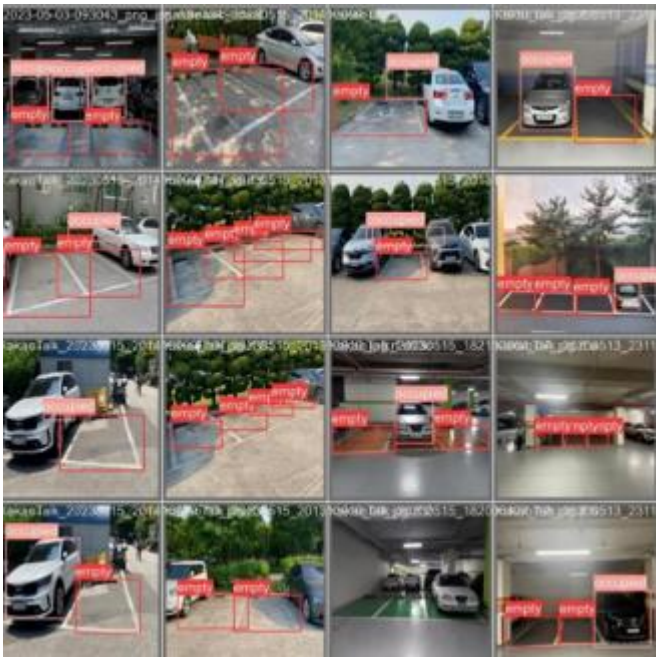

Figure 1: Identifying the empty spots in a parking lot using YOLOv5n

Model	mAP50	Precision	Recall
YOLOv5n	0.971	0.976	0.946
YOLOv5s	0.971	0.976	0.954
YOLOv8n	0.977	0.969	0.963
YOLOv8s	0.976	0.976	0.958
YOLOv10n	0.970	0.973	0.952
YOLOv10s	0.971	0.974	0.959

Table 1: Comparison of mAP, accuracy and recall for various models

- **YOLOv10:** YOLOv10 builds upon the strengths of its predecessors, addressing the limi tations in post-processing and model architecture. YOLOv10 allows for better adaptation in various real-world scenarios, from lightweight models for mobile applications to larger models for high-performance tasks.

Figure 1 shows a case where the YOLOv5n has been used to identify the empty parking spots after the training of the system.

The training of the models has been performed using Google Colab. An Intel CPU clocked at 2.2 GHz has been used coupled with a Tesla K80 GPU with 12 GB DDR5 RAM.

Table 1 shows the accuracy of the training that has been achieved for several versions of the YOLO models. As is it shown for the specific application on edge, the accuracy is quite high for each version. This may be due to the fact that the images form the camera are quite clear and there are not many objects that move very fast.

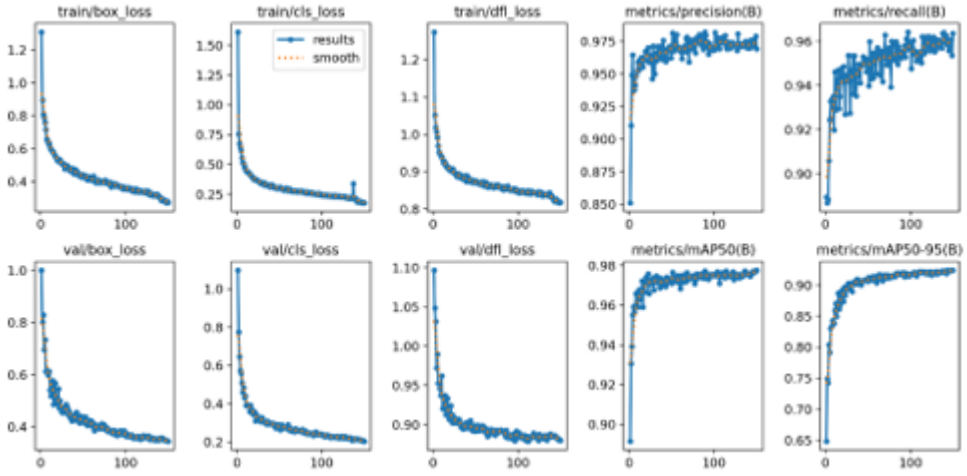

Figure 2: Results of the training using YOLOv8n

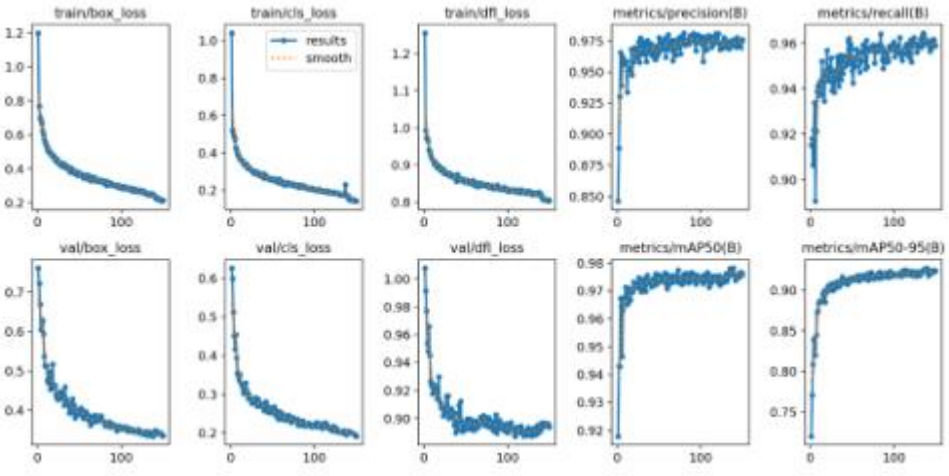

Figure 3: Results of the training using YOLOv8s

Figure 2 and Figure 3 show the results of the training for each model.

Figure 4 shows the training time for each model. As it is shown the latest version of the model require almost 40% higher training time compared to the early versions of the models.

Figure 5 shows the model size for each model. As it is shown the latest versions of the model usually are much larger than the early versions of the model. The most efficient one in terms of model size is the YOLOv5n that uses only 3.73MB and it makes it easier to fit into embedded systems with limited memory and storage.

Table 2 shows the comparison of the YOLO version in terms of the inference time, the pre processing and the post-processing. As it is shown the earlier versions of the model offer much lower inference latency compared to the latest versions. YOLOv5n takes only 584 ms for the inference on a raspberry pi 4 board while YOLOv10s takes around 2200ms for the inference. For the specific application of identification of empty spots the requirements in terms of inference is around 1 fps.

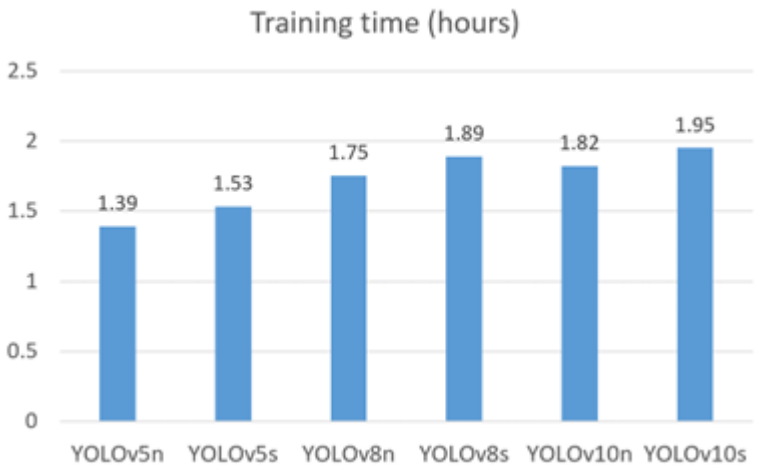

Figure 4: Comparison in terms of training time for the specific dataset

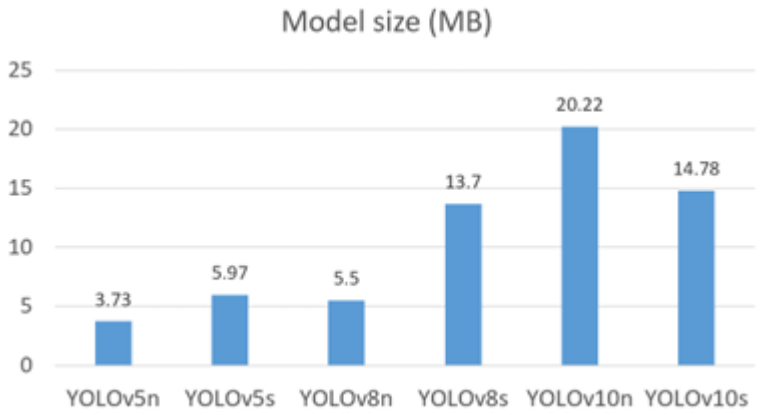

Figure 5: Comparison of the size model for the specific dataset

That means that a raspberry pi 4 could support up to 2 cameras using a YOLOv5n model.

Model	**Postprocess (ms)**	**Inference (ms)**	**Preprocess (ms)**
YOLOv5n	3.8	584	34
YOLOv5s	5.5	1405	34.3
YOLOv8n	3.2	932	15.3
YOLOv8s	3.3	1672	5.8
YOLOv10n	2.6	1075	9.6
YOLOv10s	8.1	2201	8.1

Table 2: Comparison of pre-process, inference time and post-process for various models

4 Conclusions

Based on these results it is shown that for edge applications targeting single board computers like Raspberry pi, earlier versions of the YOLO models can offer very good accuracy, low inference latency and small model size. For applications where better accuracy is required, latest version of the YOLOmodel such

as YOLO 10could offer better results but would require more powerful processors.

5 Acknowledgments

Based on these results it is shown that for edge applications targeting single board computers like Raspberry pi, earlier versions of the YOLO models can offer very good accuracy, low inference latency and small model size. For applications where better accuracy is required, latest version of the YOLOmodel such as YOLO 10could offer better results but would require more powerful processors.

References

[1] Youngkwang Kim, Woochan Kim, Jungwoo Yoon, Sangkug Chung, and Daegeun Kim. Deep learning-based multiple droplet contamination detector for vision systems using a you only look once deep learning. 02 2024.

[2] Jun Lee and Su You. Balancing privacy and accuracy: Exploring the impact of data anonymization on deep learning models in computer vision. IEEE Access, PP:1–1, 01 2024.

[3] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single Shot MultiBox Detector, page 21–37. Springer International Publishing, 2016.

[4] D. Mishra, K. Rout, S. Mishra, and S. Salkuti. Various object detection algorithms and their comparison. Indonesian Journal of Electrical Engineering and Computer Science, 29:330, 2022.

[5] Martinus Grady Naftali, Jason Sebastian Sulistyawan, and Kelvin Julian. Comparison of object detection algorithms for street-level objects, 2022.

[6] Joseph Redmon, S. Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 779–788, 2015.