International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 7
DOI: https://doi.org/10.17762/ijritcc.v11i7.7904
***Article Received**: 04 May 2023 **Revised**: 20 June 2023 **Accepted**: 08 July 2023*

Comparative Review of Object Detection Algorithms in Small Single-Board Computers

Tuan Muhammad Naeem Bin Tuan Rashid[1], Lokman Mohd Fadzil[2]
[1]National Advanced IPv6 Centre (NAv6)
University Sains Malaysia (USM)
Penang, Malaysia
e-mail: tnaeemtrashid@student.usm.my
[2]National Advanced IPv6 Centre (NAv6)
University Sains Malaysia (USM)
Penang, Malaysia
e-mail: lokman.mohd.fadzil@usm.my

Abstract— Object detection is a crucial task in computer vision with a wide range of applications. However, deploying object detection algorithms on small single-board computers (SBCs) poses unique challenges. In this review article, we present an in-depth comparative analysis of object detection algorithms tailored for small SBCs. We have conducted an extensive literature review on existing research in object detection algorithms and evaluated the performance of different approaches on benchmark datasets. Our review encompasses cutting-edge deep learning methods, which are YOLO, SSD, and Faster R-CNN. We delve into the challenges and limitations of implementing these algorithms on small SBCs and offer recommendations for optimizing their performance in such environments. Our analysis aims to shed light on the strengths and weaknesses of various object detection algorithms for small SBCs, ultimately guiding practitioners in making informed decisions and identifying potential avenues for future research in this domain.

Keywords- computer vision, machine learning, embedded system, IoT applications, performance benchmarking.

I. INTRODUCTION

Object detection, a key task in computer vision, is centered around identifying the presence and location of objects within images or video sequences. It is increasingly gaining prominence with wide-ranging applications in robotics, surveillance systems, and autonomous vehicles [1]. In the contemporary digital era, deep learning-based object detection algorithms have been extensively explored and have achieved state-of-the-art performance. The algorithms leverage convolutional neural networks (CNNs) to extract representative features from input images and generate bounding boxes around objects of interest. Despite the significant improvements in object detection accuracy facilitated by deep learning, these algorithms are computationally intensive and require high-performance hardware for efficient operation [2].

In previous research, a variety of object detection algorithms have been evaluated and compared on different fronts, including accuracy, speed, and resource usage [7][8][9][10][11][12]. These studies have utilized a broad range of datasets, such as VOC2007, VOC2017, COCO, ImageNet, or a custom dataset, applying metrics like mean average precision (mAP), frame-per-second (FPS), receiver operating characteristic (ROC), training time, and precision for a comprehensive evaluation. Despite their considerable contributions, most investigations have been constrained to industrial-grade systems or high-performance computing platforms. Consequently, there remains a notable research gap in understanding and quantifying the performance of these algorithms when executed on small single-board computers (SBCs). This gap signifies the necessity for further study in this area, particularly considering the increasing relevance of low-cost, compact computing solutions in today's technologically driven world.

SBCs, though limited in computational resources and memory, are low-cost, compact devices extensively used in embedded systems and IoT applications [3]. Their constrained processing power, memory, and storage capabilities present unique challenges when running complex algorithms such as deep learning-based object detection [3]. Nevertheless, recent advancements in SBC technology, like the Jetson Nano, have started to make possible the execution of these algorithms on small form-factor devices, thereby expanding the horizons for object detection applications [3].

This review article casts a wide net to provide a comprehensive overview of the contemporary deep learning-based object detection algorithms applied to SBCs. We focus particularly on the Jetson Nano [4], given its standing as a representative example of current SBC technology. By analyzing the performance of these algorithms, we evaluate their suitability for deployment on SBCs. Our comparison primarily involves deep learning-based object detection algorithms, which

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 7
DOI: https://doi.org/10.17762/ijritcc.v11i7.7904
***Article Received**: 04 May 2023 **Revised**: 20 June 2023 **Accepted**: 08 July 2023*

are YOLOv7 [5], SSD [6] and Faster-RCNN [7], with emphasis on their smaller models. We also delve into the computational and memory limitations of this device.

The objective of this review is twofold: to offer valuable insights into the strengths and weaknesses of different object detection algorithms when deployed on small SBCs, and to identify potential areas for future research in this burgeoning field. We hope our findings will be useful for researchers and practitioners interested in developing object detection systems for small single-board computers. In this way, our work contributes to the ongoing effort to make deep learning-based object detection more accessible on low-cost devices.

II. LITERATURE REVIEWS

A. Region-Based Convolutional Neural Network

1) R-CNN

R-CNN was introduced by Girshick et al. in 2014. The primary goal of R-CNN was to improve detection accuracy by using deep learning techniques to generate powerful feature representations for objects within an image. R-CNN consists of three main steps:

- Region proposal generation using selective search [27], which identifies a set of candidate regions (bounding boxes) in the image that may contain objects,
- Feature extraction uses a pre-trained CNN, which processes each region proposal and generates a high-dimensional feature vector.
- Classification using a set of support vector machines (SVM) [28] classifiers, which assign a class label to each region proposal based on the extracted features.

While R-CNN achieved significantly better accuracy compared to previous object detection methods, it had several limitations:

- It was computationally expensive due to the need to process each region's proposal separately, resulting in slow inference times.
- The selective search algorithm for region proposal generation was slow and not optimized for object detection tasks.
- The multi-stage pipeline, involving selective search, CNN, and SVM, was complex and could not be trained end-to-end.

2) Fast R-CNN

Fast R-CNN, introduced by Girshick in 2015, further improved the efficiency of R-CNN by introducing several key advancements. The Region of Interest (ROI) pooling layer was added, allowing for faster processing of region proposals. Fast R-CNN also simplified the training process by replacing the SVM classifiers with a single multi-task loss function that optimized classification and bounding box regression. Furthermore, Fast R-CNN processed the entire image through CNN to create a feature map, unlike R-CNN, which processed each region proposal individually. Fast R-CNN consisted of three main steps:

- Feature extraction using CNN is to process the entire image and generates a feature map, capturing high-level features from different parts of the image.
- Region proposal generation using selective search which shared with R-CNN, is to identify a set of candidate regions (bounding boxes) in the image that may contain objects. However, it is important to note that the selective search algorithm can be slow, which limits both Fast R-CNN and R-CNN.
- ROI pooling and classification using a fully connected network is to assign a class label and refine the bounding box coordinates for each region proposal based on the pooled features extracted from the feature map.

Fast R-CNN significantly improved the inference speed and efficiency compared to R-CNN and Spatial Pyramid Pooling (SPP-net). However, it relied on the slow selective search algorithm for region proposal generation.

3) Faster R-CNN

Faster R-CNN, introduced by Ren et al. in 2015, aimed to overcome the remaining bottleneck in Fast R-CNN by replacing the selective search algorithm with a Region Proposal Network (RPN), a neural network designed specifically for generating region proposals. This change resulted in a fully end-to-end trainable object detection pipeline that was both fast and accurate. Faster R-CNN consists of two main components:

- The RPN processes the feature map a CNN generates, producing a set of candidate region proposals (bounding boxes) and their corresponding objectness scores. The RPN uses anchor boxes, predefined bounding box shapes and sizes, to generate proposals that are more likely to contain objects.
- The ROI pooling and classification module takes the region proposals generated by the RPN and pools features from the CNN feature map for each proposal. A fully connected network then assigns a class label and refines the bounding box coordinates for each region proposal based on the pooled features.

By incorporating the RPN into the pipeline, Faster R-CNN significantly improved the inference speed compared to Fast R-CNN while maintaining high accuracy. Faster R-CNN has become one of the most widely used and well-regarded object detection algorithms in the computer vision community due to its combination of speed and accuracy.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 7
DOI: https://doi.org/10.17762/ijritcc.v11i7.7904
Article Received: 04 May 2023 Revised: 20 June 2023 Accepted: 08 July 2023

TABLE I. RCNN-FAMILY PERFORMANCE

Reference	Models	Test Set	Input Size	mAP (%)	FPS
[5]	SSD	VOC 2007	300x300	77.2	46
[5]	SSD	VOC 2012	300x300	75.8	46
[5]	SSD	COCO	300x300	43.1	46
[5]	SSD	VOC 2007	500x500	79.8	19
[5]	SSD	VOC 2012	500x500	78.5	19
[5]	SSD	COCO	500x500	48.5	19

B. Single-Shot Detection

SSD, introduced by Liu et al. in 2016, is a fast and accurate object detection algorithm that aims to address some of the limitations of the R-CNN family, such as the need for a separate region proposal generation step. Instead, SSD directly predicts the class labels and bounding box coordinates in a single forward pass through the network, hence the name "Single Shot." SSD consists of the following main components:

- A base CNN architecture processes the input image and generates a feature map. This base network is typically a pre-trained deep CNN, such as a 16-layer-depth Visual Geometry Group Very Deep Convolutional Networks (VGG-16) or Residual Network (ResNet), with the fully connected layers removed to allow for variable input sizes.
- Convolutional layers with different aspect ratios and scales are added to the base network. These additional layers enable the detection of objects at various scales and aspect ratios, improving the ability of the network to manage objects with different shapes and sizes.
- Multi-scale feature maps are generated by applying convolutional layers to the outputs of the base network and the additional layers. These feature maps are used to make predictions at different scales, which helps improve the detection performance for objects of various sizes.
- Default bounding boxes (anchor boxes or priors) are distributed across the feature maps. The network predicts the class probabilities and bounding box offsets for each default box.
- A non-maximum suppression (NMS) step removes overlapping bounding boxes and retains only the most confident predictions for each object class.

SSD offers several advantages over the R-CNN family of algorithms:

- It is computationally efficient due to the single-shot detection mechanism, eliminating the need for a separate region proposal generation step.
- It can manage objects of varying scales and aspect ratios using multiple feature maps and convolutional layers with different aspect ratios and scales.
- It has a more straightforward end-to-end training process compared to the multi-stage pipelines of the R-CNN family.

Despite these advantages, SSD may sometimes underperform in detection accuracy compared to Faster R-CNN, particularly for small objects. However, the trade-off between speed and accuracy has made SSD popular for real-time object detection applications.

TABLE II. SSD PERFORMANCE

Reference	Models	Test Set	Input Size	mAP (%)	FPS
[28]	RCNN	ILSVRC2013	Variable	31.4	0.077
[28]	RCNN	VOC 2010	Variable	53.7	0.077
[6]	Fast R-CNN	VOC 2007	Variable	70	3.33
[6]	Fast R-CNN	VOC 2010	Variable	68.8	3.33
[6]	Fast R-CNN	VOC 2012	Variable	68.4	3.33
[27]	Faster RCNN	VOC 2007	Variable	78.8	5
[27]	Faster RCNN	VOC 2012	Variable	75.9	5
[27]	Faster RCNN	COCO	Variable	42.1	5

C. You-Only-Look-Once

1) YOLOv1

YOLOv1 (You Only Look Once) is the first version of the YOLO object detection algorithm, introduced by Redmon et al. in 2016. The primary goal of YOLOv1 was to address the limitations of the existing object detection methods at that time, which often involved complex pipelines and were computationally expensive. YOLOv1 aims to simplify the object detection process by framing it as a single regression problem, resulting in a faster and more efficient algorithm.

YOLOv1 divides the input image into a fixed grid (usually 7x7 or 13x13), and each grid cell predicts a certain number of bounding boxes and class probabilities. These predictions are then combined to produce the final detection results. This approach allows YOLOv1 to process images in real-time, achieving high FPS rates. However, YOLOv1 also has some limitations:

- Due to the coarse grid structure, it struggles to detect small objects or objects that are close together.
- The fixed number of bounding box predictions per grid cell can lead to suboptimal performance for images with varying objects.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 7
DOI: https://doi.org/10.17762/ijritcc.v11i7.7904
***Article Received**: 04 May 2023 **Revised**: 20 June 2023 **Accepted**: 08 July 2023*

- The localization accuracy could be improved, as YOLOv1 tends to produce imprecise bounding boxes.

2) YOLOv2

YOLOv2, also known as YOLO9000, was introduced by Redmon and Farhadi in 2017 as an improvement over the original YOLOv1. YOLOv2 addressed some of the limitations of YOLOv1 by introducing several new techniques and modifications:

- Using anchor boxes improves localization accuracy and better handles objects of different shapes and sizes.
- It uses a finer-grained feature by removing one pooling layer to obtain an output feature map. Alternatively, it uses a grid of 13x13 for input images of 416x416 to better detect small objects and objects that are close together.
- Batch normalization in the network to improve the training stability and reduce overfitting.
- The introduction of multi-scale training features enables object detection at various scales.
- It uses a high-resolution classifier by pre-training the model with ImageNet at 224x224, similar to YOLOv1. This time, however, they fine-tuned the model for ten epochs on ImageNet with a resolution of 448x448. It improves network performance on higher-resolution input and the mAP.
- Use Darknet 19 as the backbone classifier.

These improvements led to higher accuracy and mAP scores while maintaining the real-time processing speed of YOLOv1.

3) YOLOv3

YOLOv3, introduced by Redmon and Farhadi in 2018, further improved upon YOLOv2 by introducing several key changes:

- Using multi-scale predictions by employing feature pyramids allows for more accurate detection of objects at different scales and aspect ratios.
- Predicts four coordinates for each bounding box and an objectness score using logistic regression. It assigns one anchor box to each object. Only classification loss is affected if no anchor box is assigned, not localization or confidence loss.
- It uses binary cross-entropy for class prediction, allowing multiple labels for the same bounding box. This feature helps handle complex cases, like an object being both a "Person" and a "Man".

YOLOv3 maintained the real-time processing capabilities of its predecessors while achieving better mAP scores and improved localization accuracy.

4) YOLOv4

YOLOv4, introduced by Bochkovskiy et al. in 2020, aimed to improve the performance of YOLOv3 by incorporating several state-of-the-art techniques and modifications:

- The introduction of CSPNet, a novel network architecture, improves the model's information flow and gradient propagation.
- The incorporation of modern techniques, such as Bag of Freebies (BoF) and Bag of Specials (BoS), enhances the model's overall performance and efficiency.

YOLOv4 achieved even better mAP scores than YOLOv3 while maintaining real-time processing speed.

5) YOLOv5

YOLOv5, developed by Glenn Jocher et al. in 2020, is an unofficial continuation of the YOLO series that introduces several enhancements and modifications to improve the performance and efficiency of the object detection algorithm:

- Switching to the PyTorch framework from the original Darknet framework provides greater flexibility and compatibility with other machine-learning tools and libraries.
- Introducing new network architecture variants (YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x) with different sizes and computational requirements allows users to choose a model tailored to their specific needs and hardware constraints.
- Advanced data augmentation techniques, such as Mosaic and MixUp, improve the model's generalization capabilities and robustness against various image transformations.

No scientific paper was published on YOLOv5 at the time of this writing. YOLOv5 builds upon the foundations of the previous YOLO versions, offering improved mAP scores and maintaining real-time processing capabilities while providing a more flexible and user-friendly framework for object detection tasks. However, it should be noted that YOLOv5 is not an official release from the original YOLO authors, as it has been developed independently.

6) YOLOv6

YOLOv6, designed for industrial applications [18], was developed by Li et al. in 2022, focusing on hardware-efficient design and better performance. The improvements include:

- A 51% increase in speed using the Anchor-free paradigm.
- Dynamic allocation of positive samples using the SimOTA label assignment strategy further enhances detection accuracy.
- Adoption of the SIoU [20] bounding box regression loss function to supervise the network during the learning

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 7
DOI: https://doi.org/10.17762/ijritcc.v11i7.7904
***Article Received**: 04 May 2023 **Revised**: 20 June 2023 **Accepted**: 08 July 2023*

phase, reducing the degree of freedom of regression, improving network convergence, and increasing regression accuracy.

As a result, YOLOv6 significantly improves mAP and inference speed compared to its predecessor.

7) YOLOv7

YOLOv7, developed in 2022 by Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao, introduces several key improvements as outlined in the comprehensive review of YOLO architectures in [21] by Terven and Cordova-Esparza (2023):

a) Architecture changes

- They are accomplished by extending an efficient layer aggregation network (E-ELAN). ELAN [22] is a strategy that allows a deep model to learn and converge more efficiently by controlling the shortest longest gradient path. YOLOv7 proposed E-ELAN that works for models with unlimited stacked computational blocks. E-ELAN combines the features of different groups by shuffling and merging cardinality to enhance the network's learning without destroying the original gradient path.
- They are scaling the model for a concatenation-based model. Scaling generates models of different sizes by adjusting some model attributes. The architecture of YOLOv7 is a concatenation-based architecture in which standard scaling techniques, such as depth scaling, cause a ratio change between the input channel and the output channel of a transition layer which, in turn, leads to a decrease in the hardware usage of the model. YOLOv7 proposed a new strategy for scaling concatenation-based models in which the depth and width of the block are scaled with the same factor to maintain the optimal structure of the model.

b) Bag-of-freebies updates

- They planned a re-parameterized convolution. Like YOLOv6, the architecture of YOLOv7 is also inspired by re-parameterized convolutions (RepConv) [23]. However, they found that the identity connection in RepConv destroys the residual in ResNet [24] and the concatenation in DenseNet [25]. For this reason, they removed the identity connection and called it RepConvN.
- The coarse label assignment is targeted at the auxiliary head, and the fine label assignment is for the lead head. The lead head is responsible for the final output, while the auxiliary head assists with the training.
- Batch normalization in conv-bn-activation integrates the mean and variance of batch normalization into the bias and weight of the convolutional layer at the inference stage.
- The implicit knowledge is inspired by YOLOR [26].
- Exponential moving average as the final inference model.

TABLE III. YOLO-FAMILY PERFORMANCE

Reference	Models	Test Set	Input Size	mAP (%)	FPS
[11]	YOLOv1	VOC 2007	448x448	63.4	45
[12]	YOLOv2	VOC 2007	288x288	69	91
[12]	YOLOv2	VOC 2007	352x352	73.7	81
[12]	YOLOv2	VOC 2007	416x416	76.8	67
[12]	YOLOv2	VOC 2007	480x480	77.8	59
[12]	YOLOv2	VOC 2007	544x544	78.6	40
[13]	YOLOv3	COCO	320x320	51.5	38
[13]	YOLOv3	COCO	416x416	55.3	31
[13]	YOLOv3	COCO	608x608	57.9	23
[14]	YOLOv4	COCO	416x416	62.8	38
[14]	YOLOv4	COCO	512x512	64.9	31
[14]	YOLOv4	COCO	608x608	65.7	23
[15]	YOLOv5	COCO	640x640	67.3	99
[16]	YOLOv6	COCO	640x640	70	98
[17]	YOLOv7	COCO	640x640	69.7	161

III. METHODOLOGIES

A. Algorithms

In this study, we compare the performance of three state-of-the-art object detection algorithms:

- **Faster-RCNN:** A region-based convolutional neural network that combines region proposal networks (RPNs) with a Fast R-CNN model for accurate object detection [7].
- **Single-Shot Detector (SSD):** A single-shot multi-box detector streamlines the object detection process by simultaneously predicting object categories and bounding box coordinates [6].
- **YOLOV7:** At the time of writing, it is one of the latest versions of the YOLO family, a real-time object detection algorithm that utilizes a single convolutional network to predict object classes and bounding boxes in one pass [5].

B. Evaluation Metrics

The performance of each object detection algorithm will be accessed using the following metrics.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 7
DOI: https://doi.org/10.17762/ijritcc.v11i7.7904
Article Received: 04 May 2023 Revised: 20 June 2023 Accepted: 08 July 2023

- **Frame Per Second:** The algorithm's processing speed is calculated as the number of processed frames per second.
- **Inference Time:** Time taken by the algorithm to generate object detection predictions for a single input image.
- **Power Usage:** Amount of electrical power consumed by the small single-board computer during the execution of each object detection algorithm.
- **Mean Average Precision:** We will use each algorithm's reported mAP on the COCO dataset, as this study will not retrain the models.

C. Experimental Setup

We will experiment on a small single-board computer platform: NVIDIA Jetson Nano. The object detection algorithms will be implemented using one of the popular deep learning frameworks called PyTorch. Pre-trained models for each algorithm, trained on the COCO dataset, will be used for inference. The evaluation metrics will be computed for each algorithm, and the results will be compared and analyzed to identify the best-performing algorithm for small single-board computer platforms.

D. Data Analysis

The performance results obtained for each algorithm on the small single-board computer platforms will be analyzed using a table to present the comparative results for FPS, inference time, power usage, and mAP. The findings will be discussed in the context of the research question and objectives. Recommendations will be made for the most suitable object detection algorithm for deployment in resource-constrained environments.

IV. COMPARATIVE ANALYSIS

TABLE IV. ALGORITHMS COMPARISON

Algorithms	Average Inference Time (ms)	Average FPS	Average Power Usage (W)	mAP (%)
Faster R-CNN Mobilenet v3 Large FPN	0.52	1.91	8.70	32.8 [41]
Faster R-CNN Mobilenet v3 Large 320 FPN	0.22	4.45	7.46	22.8 [41]
SSD 300 VGG16	0.54	1.88	8.23	43.1 [5]
SSDLite 320 Mobilenet	0.31	3.26	5.69	22.0 [40]
YOLOv7	0.89	1.12	7.51	69.7 [17]
YOLOv7x	1.45	0.668	3.041	71.2 [17]
YOLOv7-w6	0.85	1.18	7.516	72.6 [17]
YOLOv7-tiny	0.21	4.7	7.2	56.7 [17]

Table IV shows a detailed comparison of object detection models, each with its unique trade-offs concerning inference time, frame per second (FPS), GPU power usage, and mAP (mean Average Precision).

Starting with the Faster R-CNN models, which employ the MobileNet v3 with a large FPN, two configurations are evident based on input image dimensions. The standard Faster R-CNN recorded an inference time of 0.52 ms, achieving an FPS of 1.91, consuming 8.70W of power, and an mAP of 32.8% [41]. Its counterpart, processing a smaller input image size of 320x320 pixels, boasts a speedier inference time of 0.22 ms and a superior FPS of 4.45, but sacrifices some accuracy, with an mAP of 22.8% [41]. It is also marginally more energy-efficient, drawing only 7.46W.

Regarding the SSD family, the SSD model uses the VGG16 backbone and an input image of 300x300 pixels. It has an inference time of 0.54 ms, similar in speed to the standard Faster R-CNN with MobileNet v3, generating 1.88 FPS with a power demand of 8.23W. Notably, it leads to accuracy with a mAP of 43.1% [5]. In contrast, the SSDLite, leveraging the MobileNet backbone and processing images of 320x320 pixels, finds a balance with an inference time of 0.31 ms, FPS of 3.26, and a power consumption of 5.69W. Its mAP stands at 22.0%, illustrating the trade-off for its speed and efficiency.

The YOLOv7 family, apt for SBD, highlights a diverse range of performance metrics. The base YOLOv7 model has an inference time of 0.89 ms, 1.12 FPS, consumes 7.51W, and achieves an impressive mAP of 69.7% [17]. The YOLOv7x variant, possibly fine-tuned for power efficiency, exhibits an inference time of 1.45 ms, 0.668 FPS, and a lower power footprint at 3.041W, but slightly outperforms the base model with a mAP of 71.2% [17]. The YOLOv7-w6 aligns with an inference time of 0.85 ms, 1.18 FPS, power usage of 7.516W, and a mAP of 72.6% [17]. Lastly, the YOLOv7-tiny, optimized for speed, clocks an inference time of 0.21 ms, the highest FPS at 4.7, with a power consumption of 7.2W. Its mAP at 56.7%

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 7
DOI: https://doi.org/10.17762/ijritcc.v11i7.7904
***Article Received**: 04 May 2023 **Revised**: 20 June 2023 **Accepted**: 08 July 2023*

[17] suggests a slight compromise in accuracy compared to its larger counterparts.

In summary, Faster R-CNN models with MobileNet v3 and large FPN offer variations based on image size, weighing between speed and accuracy. SSD models present a choice between the robust VGG16 backbone and the streamlined MobileNet for efficiency. The YOLOv7 variants cater to a wide array of applications, from precision to real-time detection, matching diverse project needs.

It is vital to recognize that these results might shift when models are fine-tuned with TensorRT or executed in different hardware or software settings. Hence, a comprehensive understanding of a project's requirements should drive the final model selection, weighing parameters like speed, accuracy, and power efficiency.

V. APPLICATIONS AND CHALLENGES

A. Applications

Object detection algorithms, especially when deployed on small single-board computers, have numerous practical applications across various domains. Some of these applications include:

a) Surveillance and security: Object detection algorithms can monitor public spaces, detect suspicious activities, identify objects left unattended, and recognize unauthorized entries [29].

b) Autonomous vehicles: Object detection plays a crucial role in autonomous vehicles, where accurate and real-time detection of pedestrians, vehicles, and other obstacles is essential for safe navigation [30].

c) Smart agriculture: Farmers can utilize object detection algorithms to identify crop diseases, monitor livestock, and track the growth of plants, enabling more efficient and sustainable farming practices [31].

d) Retail and inventory management: Object detection can be employed to track products on shelves, monitor stock levels, and detect misplaced or missing items in retail environments [32].

e) Healthcare: In medical imaging, object detection algorithms can identify and localize abnormalities, such as tumors or lesions, in medical scans, assisting doctors in diagnosing and treating various conditions [33].

f) Robotics: Object detection is a critical component in robotic systems, enabling robots to navigate their environments, recognize and manipulate objects, and perform complex tasks [34].

B. Challenges

Despite the potential benefits of object detection algorithms in small single-board computers, there are several challenges associated with their deployment:

a) Computational limitations: Small single-board computers often have limited processing power, memory, and storage, which may hinder the performance of object detection algorithms, particularly deep learning-based methods that require substantial computational resources [35][36].

b) Power consumption: Object detection algorithms can be power-intensive, and optimizing their power usage is crucial for deployment in battery-powered devices or energy-constrained environments [36].

c) Model complexity and size: Deep learning models for object detection can be large and computationally expensive, making deploying them on resource-constrained platforms challenging. Model compression and optimization techniques may be required to reduce the size and complexity of these models without sacrificing accuracy [35][36].

d) Real-time performance: Some applications, such as autonomous vehicles or robotics, demand real-time object detection. Ensuring that object detection algorithms can process and analyze data at high speeds is critical for these use cases [11].

e) Adaptability and generalization: Object detection algorithms should be capable of adapting to different environments and conditions, such as varying lighting, occlusions, or object orientations. Ensuring that the algorithms generalize well to new situations is an ongoing challenge [38].

f) Privacy and ethical considerations: As object detection algorithms are increasingly used in surveillance and monitoring applications, concerns about privacy, data protection, and potential biases in the algorithms must be addressed [39].

VI. CONCLUSIONS

This study aimed to compare the performance of three state-of-the-art object detection algorithms—Faster R-CNN, SSD, and YOLOv7—on small single-board computers, specifically Raspberry Pi and NVIDIA Jetson Nano. We evaluated the algorithms based on various performance metrics, including frames per second (FPS), inference time, power usage, and accuracy (mAP). Our analysis highlighted the trade-offs between these metrics, which are crucial in determining the most suitable algorithm for deployment in resource-constrained environments.

Based on our comparative analysis, YOLOv7 emerged as the most promising algorithm for small single-board computers, given its high processing speed, low inference time, and competitive accuracy. However, it is essential to consider the specific requirements and constraints of the target application

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 7
DOI: https://doi.org/10.17762/ijritcc.v11i7.7904
***Article Received**: 04 May 2023 **Revised**: 20 June 2023 **Accepted**: 08 July 2023*

when selecting an object detection algorithm, as different scenarios may prioritize different aspects of performance.

Object detection algorithms have numerous potential applications across various domains, including surveillance, autonomous vehicles, smart agriculture, retail, healthcare, and robotics. However, challenges are associated with deploying these algorithms on small single-board computers, such as computational limitations, power consumption, model complexity, real-time performance, adaptability, and privacy concerns. Future research should focus on developing algorithms and techniques that address these challenges and are tailored to the specific constraints of small single-board computer platforms.

In conclusion, this study offers valuable insights into the performance of different object detection algorithms on small single-board computers. It provides a foundation for further research and development in this area. By continuing to explore and optimize these algorithms, we can unlock the full potential of small single-board computers and enable their widespread adoption across various applications and industries.

ACKNOWLEDGMENT

This paper is the outcome of the Intelligent Connected Streetlights research project work supported by the Renesas-USM industry matching grant as per MoA#A2021098 agreement with grant account no 7304.PNAV.6501256.R128.

REFERENCES

[1] Pathak, A. R., Pandey, M., & Rautaray, S. (2018). Application of deep learning for object detection. Procedia computer science, 132, 1706-1717.

[2] O'Mahony, N., Campbell, S., Carvalho, A., Harapanahalli, S., Hernandez, G. V., Krpalkova, L., ... & Walsh, J. (2020). Deep learning vs traditional computer vision. In Advances in Computer Vision: Proceedings of the 2019 Computer Vision Conference (CVC), Volume 1 1 (pp. 128-144). Springer International Publishing.

[3] Ildar, R. (2021). Increasing FPS for single board computers and embedded computers in 2021 (Jetson nano and YOVOv4-tiny). Practice and review. arXiv preprint arXiv:2107.12148.

[4] NVIDIA. (n.d.). Jetson Nano Developer Kit. NVIDIA Developer. Retrieved May 2, 2023, from https://developer.nvidia.com/embedded/jetson-nano-developer-kit.

[5] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. In Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14 (pp. 21-37). Springer International Publishing.

[6] Girshick, R. (2015). Fast r-CNN. In Proceedings of the IEEE international conference on computer vision (pp. 1440-1448).

[7] Zhao, Z. Q., Zheng, P., Xu, S. T., & Wu, X. (2019). Object detection with deep learning: A review. IEEE Transactions on neural networks and learning systems, 30(11), 3212-3232. [8] Lee, Y. H., & Kim, Y. (2020). Comparison of CNN and YOLO for Object Detection. Journal of the semiconductor & display technology, 19(1), 85-92.

[8] Malhotra, P., & Garg, E. (2020, July). Object detection techniques: a comparison. In 2020 7th International Conference on Smart Structures and Systems (ICSSS) (pp. 1-4). IEEE.

[9] Lee, C., Kim, H. J., & Oh, K. W. (2016, October). Comparison of faster R-CNN models for object detection. In 2016 16th international conference on Control, automation and systems (iccas) (pp. 107-110). IEEE.

[10] Jabir, B., Falih, N., & Rahmani, K. (2021). Accuracy and Efficiency Comparison of Object Detection Open-Source Models. International Journal of Online & Biomedical Engineering, 17(5).

[11] [12] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).

[12] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).

[13] Redmon, J., & Farhadi, A. (2017). YOLO9000: better, faster, stronger. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7263-7271).

[14] Farhadi, A., & Redmon, J. (2018, June). Yolov3: An incremental improvement. In Computer vision and pattern recognition (Vol. 1804, pp. 1-6). Berlin/Heidelberg, Germany: Springer.

[15] Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934.

[16] Jocher, G., et al. (2022). YOLOv5: State-of-the-art object detection model (v7.0) [Computer software]. Zenodo. https://doi.org/10.5281/zenodo.7347926

[17] Li, C., Li, L., Jiang, H., Weng, K., Geng, Y., Li, L., ... & Wei, X. (2022). YOLOv6: A single-stage object detection framework for industrial applications. arXiv preprint arXiv:2209.02976.

[18] Wang, C. Y., Bochkovskiy, A., & Liao, H. Y. M. (2022). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. arXiv preprint arXiv:2207.02696.'

[19] Gevorgyan, Z. (2022). SIoU loss: More powerful learning for bounding box regression. arXiv preprint arXiv:2205.12740.

[20] Terven, J., & Cordova-Esparza, D. (2023). A Comprehensive Review of YOLO: From YOLOv1 to YOLOv8 and Beyond. arXiv preprint arXiv:2304.00501.

[21] C.-Y. Wang, H.-Y. M. Liao, and I.-H. Yeh, "Designing network design strategies through gradient path analysis," arXiv preprint arXiv:2211.04800, 2022.

[22] X. Ding, X. Zhang, N. Ma, J. Han, G. Ding, and J. Sun, "Repvgg: Making vgg-style convnets great again," in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 13733–13742, 2021.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 7
DOI: https://doi.org/10.17762/ijritcc.v11i7.7904
***Article Received**: 04 May 2023 **Revised**: 20 June 2023 **Accepted**: 08 July 2023*

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE Conference on computer vision and pattern recognition, pp. 770–778, 2016.

[24] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in Proceedings of the IEEE Conference on computer vision and pattern recognition, pp. 4700–4708, 2017.

[25] C.-Y. Wang, I.-H. Yeh, & H.-Y. M. Liao, "You only learn one representation: Unified network for multiple tasks," arXiv preprint arXiv:2105.04206, 2021.

[26] Uijlings, J. R., Van De Sande, K. E., Gevers, T., & Smeulders, A. W. (2013). Selective search for object recognition. International journal of computer vision, 104, 154-171.

[27] Noble, W. S. (2006). What is a support vector machine?. Nature Biotechnology, 24(12), 1565-1567.

[28] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems, 28.

[29] Zou, Z., Chen, K., Shi, Z., Guo, Y., & Ye, J. (2023). Object detection in 20 years: A survey. Proceedings of the IEEE.

[30] Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? The Kitti Vision Benchmark Suite. 2012 IEEE Conference on Computer Vision and Pattern Recognition. https://doi.org/10.1109/cvpr.2012.6248074

[31] Kamilaris, A., & Prenafeta-Boldú, F. X. (2018). Deep learning in agriculture: A survey. Computers and Electronics in Agriculture, 147, 70–90. https://doi.org/10.1016/j.compag.2018.02.016

[32] Deng, C., & Liu, Y. (2021). A deep learning-based inventory management and demand prediction optimization method for ANOMALY DETECTION. Wireless Communications and Mobile Computing, 2021, 1–14. https://doi.org/10.1155/2021/9969357

[33] Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A., Ciompi, F., Ghafoorian, M., van der Laak, J. A. W. M., van Ginneken, B., & Sánchez, C. I. (2017). A survey on Deep Learning in medical image analysis. Medical Image Analysis, 42, 60–88. https://doi.org/10.1016/j.media.2017.07.005

[34] Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J., & Quillen, D. (2017). Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. The International Journal of Robotics Research, 37(4–5), 421–436. https://doi.org/10.1177/0278364917710318

[35] Zhang, X., Zhou, X., Lin, M., & Sun, J. (2018). Shufflenet: An extremely efficient convolutional neural network for mobile devices. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 6848-6856).

[36] Han, S., Mao, H., & Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149.

[37] Cheng, Y., Wang, D., Zhou, P., & Zhang, T. (2017). A survey of model compression and acceleration for deep neural networks. arXiv preprint arXiv:1710.09282.

[38] Geirhos, R., Temme, C. R., Rauber, J., Schütt, H. H., Bethge, M., & Wichmann, F. A. (2018). Generalisation in humans and deep neural networks. Advances in neural information processing systems, 31.

[39] Cath, C., Wachter, S., Mittelstadt, B., Taddeo, M., & Floridi, L. (2018). Artificial intelligence and the 'good society': the US, EU, and UK approach. Science and engineering ethics, 24, 505-528.

[40] Howard, A., Sandler, M., Chu, G., Chen, L. C., Chen, B., Tan, M., ... & Adam, H. (2019). Searching for mobilenetv3. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 1314-1324).

[41] Models and pre-trained weights. Models and pre-trained weights - Torchvision 0.15 documentation. (n.d.). https://pytorch.org/vision/stable/models.html