Benchmarking modern video object detection methods on low TDP devices

Adam Watson
Pennsylvania State University
University Park, Pennsylvania
acw5549@psu.edu

Abstract

In the field of computer vision, object detection plays a vital role with a wide range of applications, including surveillance, robotics, and autonomous vehicles. As compact computing platforms gain prominence, it becomes increasingly important to evaluate the performance of various object detection techniques on devices such as the Raspberry Pi. This study conducts a comprehensive examination of more than 11 state-of-the-art object detectors on a Raspberry Pi, focusing on key performance indicators such as mAP values, FPS, latency, and energy consumption. The performance of each object detector is assessed on a test dataset after being trained on the COCO dataset. A detailed analysis of the results is provided, featuring graphical representations of mAP values plotted against other performance metrics. This research represents one of the first thorough investigations into object detection techniques on low TDP, general computing hardware, taking into account multiple performance criteria. The findings of this study will enable software developers and researchers to make well-informed decisions about the most suitable object detection methods for compact computing platforms, based on their specific needs. This master's thesis contributes significantly to the development of future object detection methodologies and the enhancement of their performance on compact IoT devices.

1. Introduction

Deep Learning's dominance in object detection algorithms has steadily increased over the past 20 years, and numerous techniques have been developed and improved upon. Over time, popular object detection algorithms such as YOLOv7, REPP, SELSA, Faster R-CNN, MEGA, and many others have made object detection more efficient and less-resource intensive at higher image resolutions and video frame rate. While there have been many advancements in this field, most complex solutions require expensive graphics processing units (GPUs) or tensor processing units (TPUs) to achieve sub-second processing. Some devices, such as the Nvidia Jetson embedded computer board, are designed around such machine learning tasks and are optimized for them. Since many modern object detection algorithms have been optimized for GPU, TPU, or specialized device workflows, this thesis will explore the following question:

> *How do recent advancements in object detection models, specifically mobile-focused Convolutional Neural Networks (CNNs), compare to standard CNNs in terms of performance and efficiency on selected general computing devices under synthetic and real-world conditions?*

The benchmark studies for popular object detection algorithms, such as YOLOv3, REPP, SELSA, Faster R-CNN, and MEGA [1, 2], compare each algorithm on mobile edge GPUs and Nvidia Jetson cards. These studies serve as the inspiration behind this research paper. However, they do not focus on two areas that our research aims to expand upon: the comparison of popular object detectors with CPU or mobile-optimized object detectors [3–11], and the evaluation of these algorithms based on their performance on mobile devices and small form factor computers without GPUs. Additionally, we will investigate the performance of these algorithms on small form factor computers with external TPU modules, such as the Google Edge TPU coprocessor.

With the results of our study, we can empirically quantify the differences between algorithms designed for more powerful systems and those intended for CPU or mobile applications when executed on small form factor or mobile devices. Researchers can use these findings to better understand the current limitations of object detection and the constraints of mobile or small form factor computing. Additionally, we can gain insights into which technique(s) proposed by papers [3–11] yield greater performance gains. Investigating these specific techniques in future research could advance the field and determine their viability.

Some questions we aim to answer include:

- *Which types of mobile neural networks deliver the best results on consumer hardware?*
- *Which model scales most effectively at different power consumption levels?*
- *Do older models still outperform newer versions of the YOLO model?*
- *Does the dataset on which a model is trained impact its performance in "real world" tests?*

The objective of this study is to compare algorithms designed for systems with GPUs to those intended for CPU or mobile applications by implementing each and training them on the standard datasets called COCO [12]. We will run the trained models on their respective test datasets using a Raspberry Pi. These tests will be performed multiple times at different device power levels, and the results will be recorded.

2. Related Works

To accomplish the objectives of this study, it is necessary to incorporate relevant supporting research. In section 2.1, we introduce the eight mobile object detectors we aim to benchmark, as well as YOLOv7. In section 2.2, we delve deeper into the methodologies presented in papers [1,2]. Lastly, In section 2.3, we provide a brief description of each dataset to be used for benchmarking purposes.

2.1. Mobile Object Detection

In recent years, with modern object detectors, resource contention has become a significant issue on general-purpose, low TDP (thermal design power) devices when running large models. To address this challenge, researchers have explored various innovative techniques, including:

1. Encode both local processing and global interaction by combining both models in parallel rather than in series or intertwining them [3].
2. Using a field programmable gate array to offset processing from the mobile CPU [4].
3. Using a cloud server to reduce offloading detection latency and implemented variable region of interest (RoI) encoding. For future research, it would be interesting to implement crowd sourcing/ multiple low TDP processors combined in parallel to compute multiple frames of video and compare this to other object detection techniques. They also embedded motion vectors into video streams to track objects once detected [5].
4. Using feature collection and redistribution model (RFCR). This RFCR looked to strengthen raw features by the backbone [6].
5. Using inverted bottlenecks (IBN) and full convolutional sequences [7].
6. Generate compiler-based code for making pruning schemes (removing weights in the neural networks) for various neural network (NN) layers [8].
7. Compressed R-CNN detection heads (compressed further than what R-CNN has already done) and using 2-stage detectors [9].
8. Modifying existing object detectors such as Fast R-CNN (to trade accuracy with speed) and YOLO [10, 11].
9. And implement multistage pipelines [10].

As a comparison against more traditional object detectors, YOLOv7 [13] has been selected due to its predecessors being used in papers [3–11]. YOLO can be reconfigured into lightweight models (such as YOLOv7-tiny). A version called YOLOv7-tiny-SiLU achieves 38.7% AP, a reduction from the 51.2% AP of YOLOv7, but increases its frames per second (FPS) by 1.7 times (161 FPS compared to 286 FPS). YOLOv7-tiny outperforms YOLOv4-tiny by 10.3 AP while reducing FLOPS (floating point operations per second) by 49%. We aim to validate these numbers using the COCO dataset [12] and compare its results against the chosen mobile object detectors.

2.2. Benchmarking Techniques

To gather information on modern object detectors, research in this field [1,2] has proposed specific guidelines for benchmarking multiple object detectors. Instead of using pre-trained models from the authors, we should create our own for several reasons. One reason is the specific benchmarking tool we plan to test each model on: [12, 14, 15], as each model is tailored to the labels in the training dataset. Another reason is for verifying the original paper's findings. Standard datasets are used for evaluation in most papers and can provide a comparison point to validate researchers' claims on a given model. We can further extrapolate these claims, if consistent with our findings, to compare them against other novel object detectors in the study.

According to papers [1, 2], when evaluating object detectors, standard metrics of comparison include mAP values (mean average precision), % average precision, per frame latency, FLOPS, and TDP over time. We would also like to test FPS on a video source containing labeled objects. This evaluation metric was used in a subset [4–6, 10, 11] of papers, and we believe it is a crucial metric for evaluating consumer hardware and its typical applications.

For hardware, NVIDIA Jetson cards are used for benchmarking at four different GPU utilization levels: [0,25,50,90]% [1]. However, we will not follow this hardware recommendation. While Jetson cards have a considerably low TDP of 15 watts, they are specialized hardware platforms for machine learning. In the context of this study, Jetson cards would not produce useful information due to the focus on mobile consumer hardware. We will maintain the GPU utilization levels and implement the limits for the CPU for the selected mobile devices. In another study, mobile devices are used in conjunction with Jetson cards for benchmarking [8]. We decided to not use mobile phones in our benchmarking due to the hardware diversity and software constraints that could make it challenging to establish a consistent testing environment and draw generalizable conclusions.

Lastly, the study [1] simulates power constraints by artificially limiting the power draw of the device at different power profiles. Power constraints are relevant in many remote situations where TDP needs to be considered, such as when a device needs to conserve battery life, be powered from renewable sources, or for other reasons. In these situations, a low TDP device could be more cost-efficient per processed frame in terms of electricity used, which is something we aim to investigate.

2.3. Benchmarking Datasets

Three of the primary benchmarking datasets widely used in object detection research are "The Pascal Visual Object Classes (VOC) Challenge" [14], "Microsoft COCO: Common Objects in Context" [12], and "ImageNet" [15]. Most research papers in this domain utilize the latter two datasets for their evaluations.

The Pascal VOC Challenge [14] is a popular dataset that has been used in several object detection competitions. It contains a diverse range of images with annotated objects belonging to 20 different classes, including vehicles, animals, and household items. The dataset has been influential in shaping the field of object detection and fostering the development of new algorithms and techniques.

Microsoft COCO [12] is another widely-used dataset that focuses on object detection, segmentation, and captioning tasks. It contains over 200,000 images with more than 1.5 million object instances across 80 categories. COCO is designed to capture a wide variety of object scales, poses, and occlusions, making it particularly useful for evaluating the robustness and generalization capabilities of object detection algorithms.

ImageNet [15] is a large-scale dataset primarily used for image classification tasks, containing over 14 million images organized into more than 20,000 categories. Although it is not specifically designed for object detection, it has been employed in some object detection research due to its extensive collection of annotated images.

As previously mentioned, utilizing standard datasets such as these is crucial for comparing the performance of different object detection algorithms. These datasets provide a common ground for evaluating and contrasting the claims made by researchers for their proposed models. By testing the mobile object detectors and popular object detectors from section 2.1 [3–11, 16] on these benchmark datasets, it becomes possible to obtain a comprehensive understanding of their relative performance and effectiveness in various scenarios.

3. Methods

This section outlines the methodology for selecting mobile object detection models, evaluation criteria, devices for testing, and the benchmarking technique. The research question aims to compare recent advancements in object detection models, specifically mobile-focused Convolutional Neural Networks (CNNs), to standard CNNs in terms of performance and efficiency on selected general computing devices under synthetic and real-world conditions.

3.1. Mobile Object Detection Models

We will benchmark 9 different mobile object detector models, including YOLOv7 as a control model [3–11]. These models were chosen for several reasons. First, they are designed for edge computing devices and have been tested on such devices. Additionally, each model claims improvement over traditional object detectors, including the control model YOLOv7, using various techniques. By analyzing these models, we hope to quantify the differences between techniques and their improvements on traditional object detectors, as well as compare them to the control model. Furthermore, these models represent a diverse range of development dates, which allows us to visualize and quantify improvements over time.

3.2. Evaluation Criteria

Data set: We will use the COCO dataset [12], a large-scale object detection, segmentation, and captioning dataset, for model validation. Each model will be trained and tested on a 70%-30% split of training and testing data.

Inference time (latency): Measured as the average time taken by the object detector model to process and return frames from a streamed video. The returned stream will have bounding boxes and confidence scores. This metric will provide insights into the speed of the implemented models on each hardware platform.

Frames per second (FPS): This is measured by the average amount of frames of a streamed video, to the object detector model that can be processed and returned per second. The returned stream of data will have bounding boxes

and confidences. This provides data on the speed of the implemented models on each hardware platform.

Mean Average Precision (mAP): A standard measure of accuracy for a model's predictions over the entire test on a specific dataset. It combines both precision (the proportion of detected objects that are correct) and recall (the proportion of actual objects detected). Higher mAP values indicate better performance in detecting objects accurately and consistently. This metric is standard for the computer vision community concerning object detection CNNs.

Energy Consumption: In the paper [1], tegrastats was used for monitoring power consumption on the Jetson devices. This software can not be used on the hardware we are going to test so an alternate, but similar, software will be used to measure this.

3.3. Benchmarking Device

We will test the models on the Raspberry Pi 4 (8 GB model) with non-overclocked speeds as our chosen general computing device. The Raspberry Pi has been selected for a few reasons:

1. **Affordability:** Raspberry Pi devices are cost-effective and easily accessible, making them suitable for a broad range of users who may not have access to specialized hardware.

2. **Versatility:** Unlike specialized devices like the Nvidia Jetson series, Raspberry Pi devices are designed for general computing purposes, making them more representative of typical computing devices that a wide range of users might employ for object detection tasks.

3. **Popularity:** The Raspberry Pi has gained significant popularity among hobbyists, researchers, and developers due to its flexibility and ease of use. Its widespread adoption makes it a relevant choice for testing mobile object detectors.

4. **Compatibility:** Raspberry Pi devices have extensive support for various software libraries, frameworks, and programming languages, making it easier to implement and test different mobile object detection models.

By focusing our tests on the Raspberry Pi, we can evaluate the performance and efficiency of mobile object detectors on a device that represents a general-purpose computing platform. This approach allows us to assess the potential benefits and limitations of mobile object detectors on widely-used and versatile computing hardware.

3.4. Benchmarking

Each model will be trained on a 70%-30% split of the COCO dataset. The models will be trained for as many epochs as needed until the loss flattens. After training, the models will be tested on the three devices outlined above. Pre-selected test videos and a live video feed will be tested for average FPS. Energy consumption, FPS, and mAP values will be collected for every test on every device. The testing framework will be TensorFlow Lite, and the trained models will be converted to the TensorFlow Lite format and run against the test dataset. By comparing energy consumption, accuracy (mAP values), and latency across different models and devices, we will assess the performance and efficiency of mobile object detectors on general computing hardware, such as Raspberry Pi devices, and gain insights into the impact of recent advancements in mobile-focused CNNs.

3.5. Statistical Methods

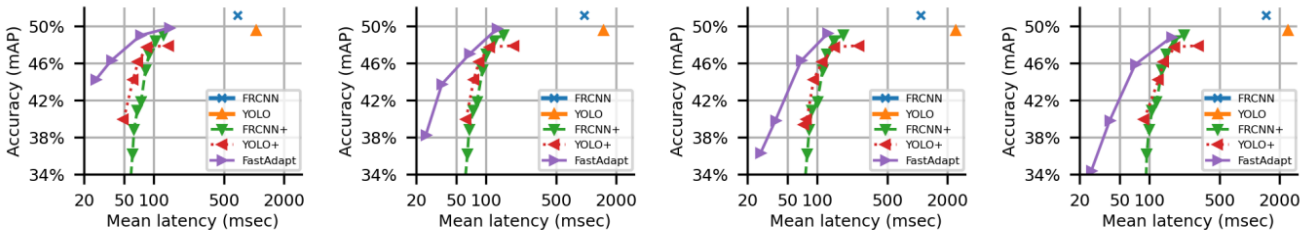
Several statistical methods can be employed to compare the performance and efficiency of mobile object detectors on general computing devices, such as Mean Average Precision (mAP), Analysis of Variance (ANOVA), and Regression analysis. After reviewing papers in this area [3–11], we chose to use mAP values as the primary method for comparing the models and graphing them against other evaluation criteria.

The rationale behind this decision is that mAP is a widely recognized and standard metric for evaluating object detection models, as it takes into account both precision and recall, making it a comprehensive measure of model performance. Additionally, using mAP values allows for straightforward comparisons between models, without the need for complex statistical methods or assumptions. This makes the analysis more accessible and easier to interpret. Finally, graphing mAP values against other evaluation criteria (such as latency and energy consumption) provides a clear way to assess the trade-offs between accuracy, speed, and efficiency for each model.

While ANOVA and regression analysis can provide additional insights into the relationships between variables and the performance of models, they were deemed unnecessary for this study, given the focus on evaluating the performance and efficiency of mobile object detectors on general computing devices using mAP values.

3.6. Data Analysis

These metrics will be used to draw conclusions about the performance and efficiency of the mobile object detectors on the Raspberry Pi. By comparing the results, we will identify the strengths and weaknesses of each model and assess the impact of recent advancements in mobile-focused CNNs on general computing devices. The data will be displayed using graphs (1,2) and tables (3) to provide a clear visualization of the performance differences between the models and how they relate to the evaluation criteria.



(a) No contention (b) 20% GPU contention (g20) (c) 50% GPU contention (g50) (d) 90% GPU contention (g90)

Figure 1. An example of graphs for GPU contention [1].

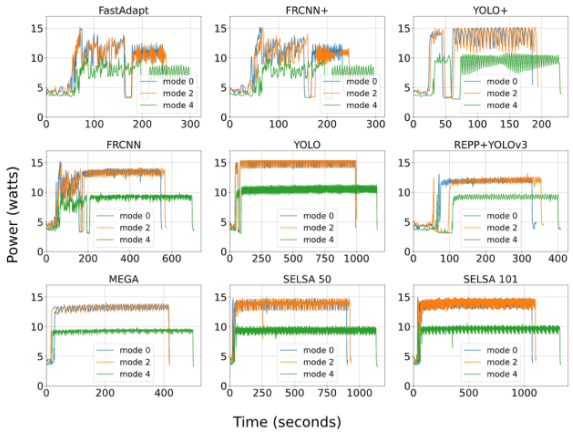


Figure 2. An example of a graph for comparing Watts vs the run time [1]

Model	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	MAdds (G)	#Params (M)
Shuffle-V2 [26]	25.9	41.9	26.9	12.4	28.0	36.4	**2.6** (161)	0.8 (10.4)
MF-151M	**34.2**	53.4	36.0	19.9	36.8	45.3	**2.6** (161)	4.9 (14.4)
Mobile-V3 [16]	27.2	43.9	28.3	13.5	30.2	37.2	4.7 (162)	2.8 (12.3)
MF-214M	**35.8**	55.4	38.0	21.8	38.5	46.8	**3.9** (162)	5.7 (15.2)
ResNet18 [14]	31.8	49.6	33.6	16.3	34.3	43.2	29 (181)	11.2 (21.3)
MF-294M	**36.6**	56.6	38.6	21.9	39.5	47.9	**5.5** (164)	6.5 (16.1)
ResNet50 [14]	36.5	55.4	39.1	20.4	40.3	48.1	84 (239)	23.3 (37.7)
PVT-Tiny [39]	36.7	56.9	38.9	22.6	38.8	50.0	70 (221)	12.3 (23.0)
ConT-M [43]	37.9	58.1	40.2	23.0	40.6	50.4	65 (217)	16.8 (27.0)
MF-508M	**38.0**	58.3	40.3	22.9	41.2	49.7	**9.8** (168)	8.4 (17.9)

Figure 3. An example table showing mAP values [3]

References

[1] J. Lee, P. Wang, R. Xu, V. Dasari, N. Weston, Y. Li, S. Bagchi, and S. Chaterji. Benchmarking video object detection systems on embedded devices under resource contention. In *Proceedings of the 5th International Workshop on embedded and Mobile Deep Learning*. ACM Conferences, 2021. 1, 2, 3, 4, 5

[2] K. Li, G. Wan, G. Cheng, L. Meng, and J. Han. Object detection in optical remote sensing images: A survey and a new benchmark. *ISPRS Journal of Photogrammetry and Remote Sensing*, 2019. 1, 2

[3] Y. Chen, X. Dai, D. Chen, M. Liu, X. Dong, L. Yuan, and Z. Liu. Mobile-former: Bridging mobilenet and transformer. In *CVF Open Access*, 2022. 1, 2, 3, 4, 5

[4] D. Honegger, H. Oleynikova, and M. Pollefeys. Real-time and low latency embedded computer vision hardware based on a combination of fpga and mobile cpu. *IEEE Xplore*, 2014. 1, 2, 3, 4

[5] L. Liu, H. Li, and M. Gruteser. Edge assisted real-time object detection for mobile augmented reality. In *The 25th Annual International Conference on Mobile Computing and Networking*. ACM Conferences, 2019. 1, 2, 3, 4

[6] P. Ganesh, Y. Chen, Y. Yang, D. Chen, and M. Winslett. Yolo-ret: Towards high accuracy real-time object detection on edge gpus. In *CVF Open Access*, 2022. 1, 2, 3, 4

[7] Y. Xiong, H. Liu, S. Gupta, B. Akin, G. Bender, Y. Wang, P.-J. Kindermans, M. Tan, V. Singh, and B. Chen. Mobiledets: Searching for object detection architectures for mobile... In *CVF Open Access*, 2021. 1, 2, 3, 4

[8] Z. Li, G. Yuan, W. Niu, P. Zhao, Y. Li, Y. Cai, X. Shen, Z. Zhan, Z. Kong, Q. Jin, Z. Chen, S. Liu, K. Yang, B. Ren, Y. Wang, and X. Lin. Npas: A compiler-aware framework of unified network pruning and architecture search for beyond real-time mobile acceleration. In *CVF Open Access*, 2021. 1, 2, 3, 4

[9] Z. Qin, Z. Li, Z. Zhang, Y. Bao, G. Yu, Y. Peng, and J. Sun. Thundernet: Towards real-time generic object detection on mobile devices. In *CVF Open Access*, 2019. 1, 2, 3, 4

[10] H. Mao, Y. Wang, J. Yao, B. Li, T. Tang, and S. Yao. Towards real-time object detection on embedded systems. *IEEE Xplore*, 2016. 1, 2, 3, 4

[11] B. Ullah. Cpu based yolo: A real-time object detection algorithm. *IEEE Xplore*, 1:1–7, Nov. 2020. 1, 2, 3, 4

[12] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár.

Microsoft coco: Common objects in context. arXiv.org, 2015. 2, 3

[13] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. arXiv.org, 2022. 2

[14] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge 2012 (voc2012) development kit. Visual Object Classes Challenge 2012 (VOC2012), 2012. 2, 3

[15] J. Deng, W. Dong, R. Socher, L.-J. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. https://image-net.org/static_files/papers/imagenet_cvpr09.pdf, 2009. [Accessed: 14-Nov-2022]. 2, 3

[16] October 2022. Steam Hardware Software Survey, 2022. 3