Benchmarking Video Object Detection Systems on Embedded Devices under Resource Contention

Jayoung Lee*
Pengcheng Wang*
lee3716@purdue.edu
wang4495@purdue.edu
Purdue University

Ran Xu
Purdue University
xu943@purdue.edu

Venkat Dasari
Army Research Lab
venkateswara.r.dasari.civ@mail.mil

Noah Weston
Army Research Lab
noah.d.weston.civ@mail.mil

Yin Li
University of Wisconsin at Madison
yin.li@wisc.edu

Saurabh Bagchi
Purdue University
sbagchi@purdue.edu

Somali Chaterji
Purdue University
schaterji@purdue.edu

ABSTRACT

Adaptive and efficient computer vision systems have been proposed to make computer vision tasks, e.g., object classification and object detection, optimized for embedded boards or mobile devices. These studies focus on optimizing the model (deep network) or system itself, by designing an efficient network architecture or adapting the network architecture at runtime using approximation knobs, such as image size, type of object tracker, head of the object detector (e.g., lighter-weight heads such as one-shot object detectors like YOLO over two-shot object detectors like FRCNN). In this work, we benchmark different video object detection protocols, including FastAdapt, with respect to accuracy, latency, and energy consumption on three different embedded boards that represent the leading edge mobile GPUs. Our set of protocols consists of Faster R-CNN, YOLOv3, SELSA, MEGA, and REPP. Further, we characterize their performance under different levels of resource contention, specifically GPU contention, as would arise due to co-located applications on these boards, contending with the video object detection task. Our key insights are that object detectors have to be coupled with trackers to keep up with the latency requirements (e.g., 30 fps). With this, FastAdapt achieves up to 76 fps on the most well-resourced NVIDIA Jetson-class board—the NVIDIA AGX Xavier. Second, adaptive protocols like FastAdapt, FRCNN, and YOLO (specifically our adaptive variants, FRCNN+ and YOLO+) work well under resource constraints. Among the latest video object detection heads, SELSA achieves the highest accuracy but at a latency of over 2 sec per frame. Our energy consumption experiments bring out that FastAdapt, adaptive FRCNN, and adaptive YOLO are best-in-class, relative to the non-adaptive protocols SELSA, MEGA, and REPP.

*Both authors contributed equally to this research.

EMDL '21, June 25, 2021, Virtual, WI, USA

ACM ISBN 978-1-4503-8597-8/21/06.
https://doi.org/10.1145/3469116.3470010

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; • **Computer systems organization** → **Embedded & cyber-physical systems**.

KEYWORDS

Adaptive inference, mobile devices, video object detection, energy consumption, approximate algorithms, context-aware analytics, mobile GPUs, embedded computing, configuration tuning.

ACM Reference Format:
Jayoung Lee, Pengcheng Wang, Ran Xu, Venkat Dasari, Noah Weston, Yin Li, Saurabh Bagchi, and Somali Chaterji. 2021. Benchmarking Video Object Detection Systems on Embedded Devices under Resource Contention. In *5th International Workshop on Embedded and Mobile Deep Learning (EMDL '21), June 25, 2021, Virtual, WI, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3469116.3470010

1 INTRODUCTION

Video analytic systems have seen widespread success in various domains, ranging from computationally heavy tasks such as recognizing faces for surveillance, to mobile applications such as detecting objects for mobile-based augmented reality (AR) to real-time systems, e.g., localizing pedestrians and cars for autonomous driving. A key component shared by these applications is the ability to detect objects in videos. There is ongoing research on pushing the accuracy further for video object detection tasks. Some of the recent breakthroughs involve frame aggregation, a technique that utilizes features from other frames during inference to enhance the detection results. SELSA [20] widens the window for selecting the frames for aggregation by expanding the semantic neighborhood of the frames from their immediate neighborhood. MEGA [1] extends SELSA further and adds global frame aggregation considering frames from other videos, sharing semantic similarity. While these techniques are performed during the runtime of the inference task, REPP [17] reuses the detection output from a baseline model to further post-process the detection output to enhance the detection results after the analysis of a video. Other works make use of optical flow [24], or techniques such as knowledge distillation [4] that have not been commonly deployed in video object detection.

These tasks are mostly geared toward accuracy, leveraging server-class machines or cloud enterprises. In contrast, mobile applications typically encounter limited energy or constrained computational resources while still requiring strict latency guarantees. To improve the user's immersive experience (e.g., in AR/VR games) or to give high-confidence outputs from streaming videos (e.g., for pedestrian recognition in autonomous driving), such tasks must be performed in near real-time under stringent latency budgets of 33 msec or lower. Even with advancements in mobile GPUs, these devices lack the computation power and resource isolation that a server-class machine/cloud enterprise provides. In response, following are some proposed innovations: applying reinforcement learning based neural architecture search (NAS) for a lightweight network (e.g., [19]), more efficient computation of the network [6], database's parameter tuning/cloud VM selection/serverless optimization [10–12], or using a dynamic pipeline adapting to content at runtime [5, 21, 22].

However, to our knowledge, no prior work has benchmarked state-of-the-art (SOTA) video object detection works, under varying resource contention on heterogeneous devices. Further, no prior work has benchmarked these solutions on embedded devices for energy measurement, using different power budgets. Thus, in this paper, we claim the following contributions: We engineered FastAdapt—a faster variant of ApproxDet [22]—by optimizing the frame decoding and resizing modules, plus a better choice of library APIs. We also implemented FRCNN+ and YOLO+, which are engineered versions of FRCNN and YOLO, with image resizing as the adaptive approximation knob and with the addition of an object tracker (MedianFlow) for faster video object detection with lower jitter, as in our framework FastAdapt[1]. We benchmark the SOTA models in video object detection on different mobile devices and benchmark under different GPU resource contention levels, or different power modes to investigate the impact of contention or power on video object detection solutions.

2 RELATED WORK

As a key problem in computer vision, video object detection seeks to locate object instances in video frames using bounding boxes and simultaneously classifying the instance into target categories with their class probabilities. The most widely used detection models adopt convolutional neural networks (CNNs), broken down into: a backbone network that extracts features from images (e.g., ResNet) and a detection network/head that classifies object regions based on the extracted features (e.g., Faster R-CNN, YOLO). The detection network can be further categorized into two-stage [3, 15, 19] or single-stage [9, 14, 23]. A general trend in object detection has been to make deeper and more complex object detection networks in order to achieve higher accuracy such as in recent video object detection algorithms [1, 17, 20]. However, these advancements in accuracy do not make these algorithms more efficient in terms of the network size and the latency of the detection task. Further, in many real-world object detection tasks, the task has to be carried out in real-time on a computationally constrained platform such as a mobile or embedded device[2]. In such cases, object detection, and more specifically in FastAdapt, video object detection becomes challenging because of the resource constraints and the often stringent latency budget (30–50 msec/frame) for acceptable video quality.

[1]We experimentally determine that it is crucial to couple a tracking algorithm with the detection algorithm for the streaming analytics to keep up with the streaming video rate. In this setup, for a Group of Frames (GoF), object detection runs on one frame and the tracker on the remaining frames. A typical range for the size of GoF for FastAdapt is 1–100 frames, where 1 means detection only.

3 FASTADAPT'S DESIGN VIS-À-VIS BENCHMARK DESIGN

FastAdapt comes with a total of five different tuning knobs that could be adaptively selected at runtime to match the latency Service Level Objectives (SLOs). The tuning knobs are as follows, 1) image shape of the object detector, 2) number of region proposals, 3) interval of the object detector being triggered (sampling interval), 4) tracker type, and 5) image downsampling ratio in the object tracker. FastAdapt is an engineered version of ApproxNet [21] and ApproxDet [22], which focus on video object classification and video object detection, respectively.

FRCNN+ comes with tuning knobs of the image shape and number of region proposals and YOLO+ comes with the tuning knob of image shape. Both of them are combined with a tracker with a sampling interval of 8, which was empirically determined as optimal, meaning the object detector will be triggered every 8 video frames, and the other frames will be handled with an object tracker.

Our concept of using tuning knobs comes with the trade-off between accuracy and latency, which results in accuracy degradation for faster inference under low resource budget. The biggest difference between FastAdapt and the simple tunable models is that FastAdapt is able to schedule the tuning knob policy with more options, especially the sampling interval that controls how often the tracker will be triggered. The tracker primarily leverages the CPU rather than the GPU, unlike the detector, and suffers relatively low accuracy loss for most cases where the tracking of the object is not lost. Plus, use of tracking in video object detection is a video stabilization technique, rendering the overall video viewing experience more jitter-free. This gives FastAdapt the ability to match the latency requirement more flexibly while maintaining similar accuracy with simple tunable models.

Next, we introduce object detection methods benchmarked, our evaluation metric, dataset, and embedded devices, and our contention generator to create controlled levels of GPU contention.

3.1 Selected Video Object Detection Models

We benchmark 6 SOTA adaptive models for video object detection, a total of 10 protocols (including variants of the same model), selected using the following criteria: 1) The code and model could be deployed on an Jetson TX2 board. 2) The model is opensourced with pretrained weights on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2015 VID dataset [16] available. These models can be divided into two groups.

Adaptive video object detection models: We evaluate 3 adaptive video object detection models. FastAdapt, a sped-up variant of ApproxDet [22], sped up with optimization in frame decoding and image resizing and improved library APIs to represent a state-of-the-art adaptive protocol. FRCNN+ and YOLO+ in our improvement over FRCNN [15] and YOLO [13, 14]. **FastAdapt**: FastAdapt has

[2]We use the terms "mobile device" and "embedded device" synonymously.

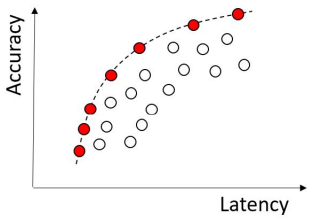


Figure 1: Pareto frontier for the accuracy-latency tradeoff for a particular model, tuning its configuration knobs. This has to be considered separately for each contention level.

five tuning knobs: Image shape; Number of region proposals; Interval of the object detector being triggered; Tracker type; & Image downsampling ratio. **FRCNN+**: We added two tuning knobs for FRCNN: 1) Image shape, and 2) Number of proposals. Medianflow tracker [7] was employed for acceleration. With different combinations of the image shape and number of proposals, there are 28 branches with the shape in the following range [224, 320, 448, 576] and number of proposals in the following range [1, 3, 5, 10, 20, 50, 100]. We used the MedianFlow tracker to speed up while keeping a sampling interval of 8 frames for the object detector. **YOLO+**: We included one tuning knob for YOLO, which is the shape of the input image, and also used it in tandem with the MedianFlow object tracker [7] for acceleration. There are a total of 12 branches with the shape in the following range [224, 256, 288, 320, 352, 384, 416, 448, 480, 512, 544, 576]. Similarly, object detection was triggered every 8 frames—the sampling interval knob in FastAdapt.

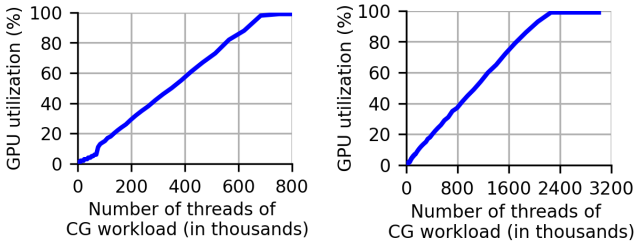
SOTA video object detection models: We also consider several latest video detection models. REPP comes with a total of three model (baseline) variants - YOLOv3, SELSA, FGFA - and only their implementation of the YOLOv3 baseline was able to run on a TX2 board. We address the baseline model as YOLOv3 to avoid confusion with our own YOLO implementations (PyTorch YOLO and YOLO+), and the full model with the post-processing technique as REPP+YOLOv3. MEGA is provided with two different backbone models with different levels of features usage. Here, we use the BASE model provided by the authors with ResNet-50 as the feature extractor, as this was the only configuration that could be run on the TX2 board. We call this model MEGA base. For SELSA, we utilize ResNet-50 and ResNet-101, with the corresponding variants referred to as SELSA 50 and SELSA 101.

3.2 Evaluation Metrics and Dataset

Accuracy of all models is measured with mean average precision (mAP), following the widely adapted protocol [8] on the dataset.

Latency of all models are measured as per-frame latency during inference, further averaged to acquire the mean latency per frame. Each model can achieve a variety of accuracy-vs-latency points corresponding to different settings of the configuration parameters, specific to each protocol. Generally, for a usable range for each protocol, a setting that improves the accuracy leads to increased latency. This tradeoff space defines a set of Pareto optimal frontiers, as shown in Fig. 1, representing the accuracy and latency achieved by all the configuration knobs for a given protocol.

Energy consumption is measured by using the API from NVIDIA - *tegrastats*. *Tegrastats* provides the information of current power



(a) Xavier NX (b) AGX Xavier

Figure 2: Calibration of the GPU contention generator. The CG workload is increased by a fixed factor 1.1, until the GPU utilization reaches 100%. The CG workload at desired GPU utilization levels are used to create the contention.

usage for each module — whether CPU or GPU. User can also specify the interval of the output from *tegrastats*. We use 1-sec interval in our experiments. This choice balances the need to collect accurate measurements at a fine time granularity while bounding the overhead of the measurement and corresponding impact on the performance of the main application.

Dataset: We use the ILSVRC 2015 VID validation dataset [16] as the benchmark dataset for the video object detection task to classify and localize the objects in 30 classes, and over 555 videos.

3.3 Embedded Devices

We selected embedded devices in the Jetson class with mobile GPUs leveraging different CPU, GPU, and memory capacities, specifically—Jetson TX2, Jetson Xavier NX, and Jetson AGX Xavier—for our benchmarking. The relation between their computational capacities is Jetson TX2 < Jetson Xavier NX < Jetson AGX Xavier. Each GPU and CPU on these devices come with a native Dynamic Voltage and Frequency Scaling (DVFS) functionality, enabled by default, which we disable in our experiments. DVFS provides a way to reduce (static and dynamic) power consumption of the embedded chips on the fly by scaling down the voltage and frequency based on the targeted performance of the application. We disable DVFS by fixing the frequency of the modules at their max frequencies. We realized this through our detailed empirical investigations and found our step to be essential for reproducibility.

3.4 Contention Generator

We benchmark object detection models for the contention experiments using a synthetic contention generator (CG) that we created. The CG simulates the processes that consume GPU resources, which could co-exist with the object detection tasks. The CG occupies designated levels of resource on the GPU module by a percentage of the maximum capacity. Each embedded device has a different GPU architecture, and thus, different amounts of resources and computational ability. Our GPU CG performs add operation on a certain size of arrays by employing a CUDA kernel function. By changing the workload size (number of threads) of the CG, we control the number of GPU cores that are kept busy in unit interval. For the observation of our calibration experiment, we use *tegrastats* to record the GPU utilization. We calibrate the CG for each device so that 99% GPU utilization at max frequency was achieved and all contention values less than that use a proportionately lesser

percentage of workloads. We show the offline profiling experiment results of the GPU CG on Xavier NX and AGX Xavier in Fig. 2. As can be seen, with the increasing number of threads in the CG, the GPU utilization keeps increasing and finally saturates at 99% when the workload is large enough. After the offline profiling experiment, we choose 12 discrete levels [0%, 1%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, 99%] to serve as the CG's calibrated output, and accordingly, map to 12 internal parameters that correspond to the size of the workloads in the CG. Further, CPU and I/O contention could be a relevant criterion with efficient models such as MobileNets, e.g., [18] or GhostNet [6] that utilize depthwise separable convolution operations that are more CPU friendly. Our current models are GPU-heavy ones since they have heavier backbones.

4 IMPLEMENTATION AND EVALUATION

4.1 Accuracy and Latency Benchmark

Experiment setup: We evaluate each object detection model on our 3 embedded devices under [0, 20, 50, 90]% GPU contention on the ILSVRC VID 2015 validation set, as is standard practice. For adaptive video object detection models, we consider a streaming setting for inference with video frames fed one-by-one, and report the mean latency per video frame. For other methods, we use batch processing for inference. Fig. 3, 4, and 5 summarize our main results. Power modes for our latency-accuracy experiments: TX2: mode 0; Xavier NX: mode 2; AGX Xavier: mode 0.

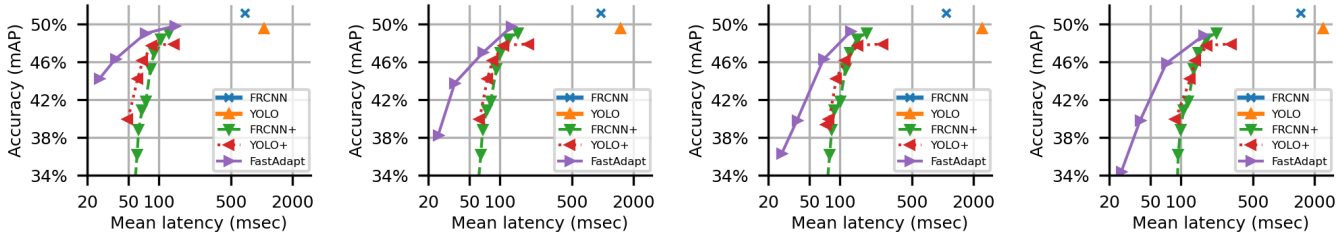
4.1.1 Adaptive Video Object Detection Models. Fig. 3 reports the accuracy and latency of all models with varying contention levels on the NVIDIA Jetson TX2 board. In subfigure (a), we observe that FastAdapt has the lead in the accuracy-latency frontier over FRCNN+ and YOLO+ over a wide range of latency budgets, varying from 33.3 msec to 200 msec. For FastAdapt, the accuracy goes up to 49.8% mAP, running at 142.7 msec per frame (roughly 7.0 fps) and the accuracy drops by 5.6%, running at 37.9 fps. For FRCNN+, it is able to achieve a similar accuracy to FastAdapt but the accuracy drops significantly (>15%) as we dial up the speed to 20 fps. On the other hand, YOLO+, though with lower accuracy given high latency budget (100 to 200 msec), wins at speed over FRCNN+ since the accuracy drop is 8% from 150 msec speed to 50 msec speed. Finally, we also show that latency and accuracy of FRCNN and YOLO without our optimization in the figure, where the latency axis is in log-scale. For FRCNN and YOLO, the latency is 693 and 1051 msec per frame while the accuracy is 51.1% and 49.5%, respectively. Based on our empirical experiments, we have used the contention generator on different devices, while varying the GPU contention level from 0% to 90% by increasing the contention level in steps of 10%. However this workload is calibrated when there is no other workload on the GPU. When the contention generator is run with the object detection model, the contention generator cannot take up more than 50% GPU, i.e., the OS scheduler throttles it. This suggests 50% GPU contention is already large enough to impact the object detection model, saturating it at that level.

In subfigures (b), (c), (d), we further measure the performance of these protocols under GPU contention. Compared to the subfigure (a), we observe that FastAdapt maintains the latency with reduced accuracy and superior to FRCNN+ and YOLO+, validating its claim of approximation (at runtime) to resource availability. Given a 50% GPU contention, FastAdapt's accuracy drop is 7.8% given a 30 fps speed and 1.0% given a 5 fps speed. In contrast, FRCNN+ and YOLO+ are 1.5X-2X slower with 50% to 90% GPU contention.

For the latency-accuracy experiments on Xavier NX and AGX Xavier, we set the latency budget as [20ms, 33ms, 50 ms, 100ms], in most cases FastAdapt is superior at both accuracy and latency than FRCNN+ and YOLO+. As we can see the results on Xavier NX from Fig. 4, in subfigure (a), FastAdapt achieves 49.3% of accuracy with 59.3 msec per frame (16.9 fps), and accuracy drops to 43.9% running at 35.2 fps. FRCNN+ is comparable to or even better than FastAdapt when the latency budget is sufficient. However, it drops quickly to 40.9% when the latency budget decrease to 27.8 msec (36.0 fps). YOLO+ has a accuracy at 44.2% when the latency budget is 52.2 msec (19.2 fps). We further add GPU contention for the latency-accuracy experiment, in subfigure (b), the overall latency of FastAdapt, FRCNN+, and YOLO+ increased as the contention increase from 0% to 50%. However, FastAdapt can still keep the fps about 10 fps when there is 50%. Fig. 4 (c) and (d) show that the models achieve roughly 2 times higher fps on AGX Xavier than on TX2. FastAdapt can achieve 47.5% of accuracy at 49.8 fps. FRCNN+ can achieve 38.8% accuracy at 57.3 fps and YOLO+ can achieve 40.0% at 43.1 fps. All 3 groups of experiments suggest that as contention increased, the accuracy is influenced and will drop accordingly.

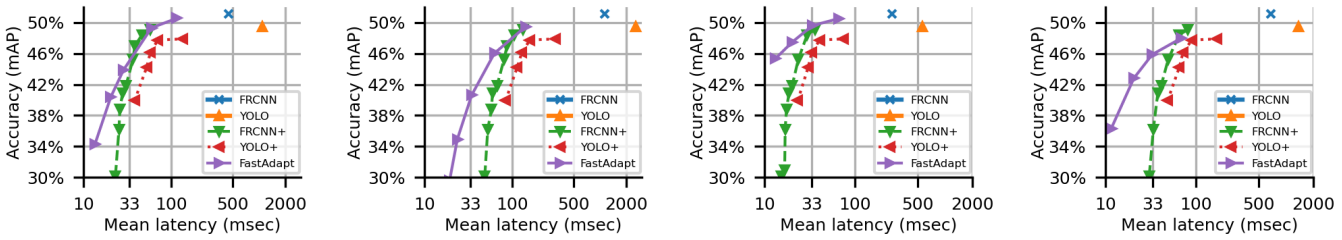
4.1.2 SOTA Video Object Detection Models. SOTA models (YOLOv3, REPP+YOLOv3, MEGA base, SELSA 50, 101) focus on the accuracy of video object detection. As we can see from the caption of Fig. 5, without latency budgets, they achieve higher mAP than adaptive models, ranging from 51.25% to 81.5%. The accuracy values obtained in our benchmark differ from those of the original authors, most significantly for REPP. This we believe is due to two reasons—first, we use an IoU threshold of 0.6 consistently for all protocols (original authors use different thresholds for different protocols—0.5 for REPP) and second, for the REPP evaluation, we use the code's "demo" version, rather than the "full" version, which has most parameter values hard-coded and thus not easy to use. We find from our evaluation that these models all suffer from poor latency, further exacerbated under contention, showing up to 2X increased latency with contention over 50%. This is because of their model sizes and inability to adapt to contention. Among all models: YOLOv3 has the lowest latency along with the lowest mAP on all boards in line with prior works. SELSA 101 achieves the highest mAP but with the highest latency on all boards. In contrast, REPP+YOLOv3 has a superior mAP at 74.81% while still maintaining a low latency.

Overall, FastAdapt outperforms other object detection algorithms since it is designed to adapt to different (video) content characteristics and varying resource contention in the embedded device. However, other models like FRCNN and YOLO cannot instantiate such dynamic adaptations and thus lead to inferior accuracy-latency performance. FastAdapt is lighterweight, especially engineered for mobile GPUs with co-located applications (contention-aware), and this comes at the expense of accuracy. The other models, e.g., REPP and SELSA, specifically perform averaging of computations across frames of the same video or global averaging across many videos, with further post-processing, with accuracy being the primary criterion for the optimization (of performance).



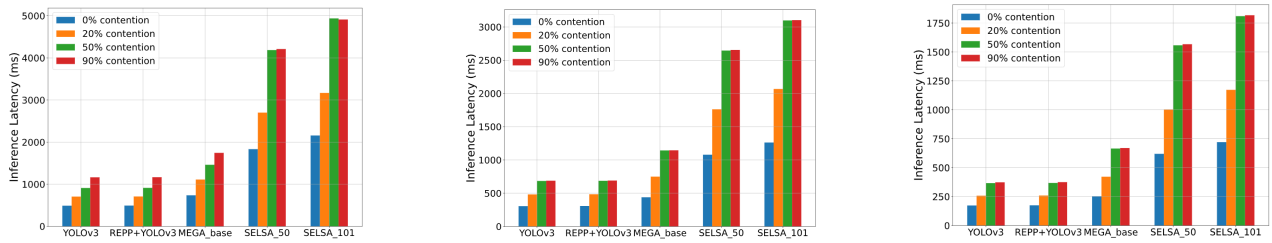
(a) No contention (b) 20% GPU contention (g20) (c) 50% GPU contention (g50) (d) 90% GPU contention (g90)

Figure 3: Benchmarking the object detection algorithms on NVIDIA Jetson TX2 under varying contention. Datapoints for FastAdapt are acquired using the adaptive scheduler; FRCNN+, YOLO+ are pareto-style datapoints from experimenting with multiple fixed configurations; & FRCNN and YOLO has a single datapoint, lacking adaptive features.



(a) Xavier NX, g0 (b) Xavier NX, g50 (c) AGX Xavier, g0 (d) AGX Xavier, g50

Figure 4: Benchmarking the object detection algorithms on the NVIDIA Jetson Xavier NX and AGX Xavier under different contention levels. The datapoints are acquired with the same procedure as from the NVIDIA Jetson TX2.



(a) Jetson TX2 (b) Jetson Xavier NX (c) Jetson AGX Xavier

Figure 5: Benchmarking SOTA models on NVIDIA Jetsons under varying contention. The accuracy remains invariant across boards, as follows (in mAP). YOLOv3: 51.25%, REPP+YOLOv3: 74.81%, MEGA base: 68.11%, SELSA 50: 77.31%, SELSA 101: 81.5%.

Table 1: Energy consumption (J) for all protocols under different power modes: We run all models on a randomly selected video dataset (828 frames), then measured the energy consumption for the whole inference time (Xavier NX).

Power Modes	*Fast Adapt*	*FRCNN+*	*YOLO+*	*FRCNN*	*YOLO*	*REPP w YOLOv3*	*MEGA*	*SELSA 50*	*SELSA 101*
0	2244	2205	2196	6567	14270	3427	5269	12233	14565
2	2293	2296	2246	6745	14407	3490	5368	12519	14967
4	2069	2064	1914	5906	11836	3122	4528	10493	12451
Avg	2202	2189	2119	6406	13504	3346	5055	11748	13995

4.2 Energy Consumption Benchmark

Experiment setup: Embedded devices need to be energy efficient as in most cases they may need to run on limited power or batteries. Thus, we measure the energy consumption of our video object detection models—FastAdapt and baselines on our mid-tier embedded device, Xavier NX using 3 different power modes [0, 2, 4], out of 5 available ones [2]. Power mode 0 consumes the highest power (15W), with the highest frequency setting for each of GPU, CPU, and the DL accelerator. Power mode 4 provides the lowest power level. Under each power mode, we first measure the idle status power using *tegrastats*. Mode [0, 2, 4] have idle status power of [3.25, 3.38, 3.08]W. Then, we run each video object detection model on a randomly selected video dataset, which has 828 frames. At the same time, we record the energy consumption using *tegrastats*. The results are shown in Fig. 6 and Table. 1. Since the major part of power consumption comes from the GPU module, and FastAdapt, FRCNN+, YOLO+ leverage an object tracker in our adaptive versions, we notice significant oscillations in their energy plots [Fig. 6]. This is because the object tracker mainly uses the CPU corresponding to the dips in the curve. We see that Mode 4 helps reduce the instantaneous power by around 33% and total energy consumption by around 10%. Among all models, FastAdapt, FRCNN+, and YOLO+ have the lowest average total energy consumption at around 2,200 J. REPP with YOLOv3 (baseline) consumed 3,346 J on average while SELSA has the highest average energy consumption of 13,995 J, 6X larger than adaptive models. Thus, our energy benchmarking experiments validate that adaptive

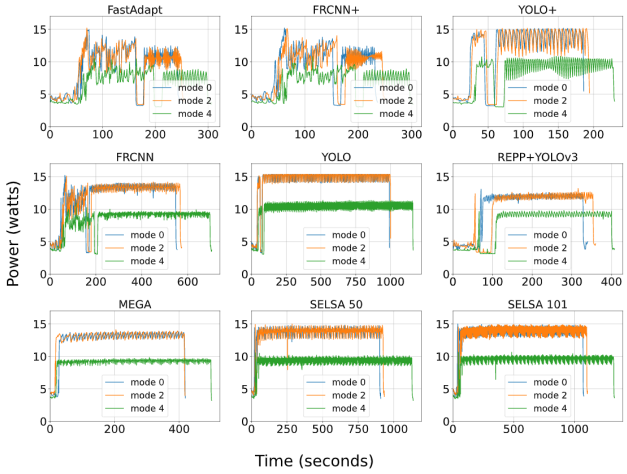


Figure 6: During the inference stage, on the randomly selected video dataset, the instantaneous power is calculated every 1-sec interval, using the tegrastats API (Xavier NX).

variants of YOLO and FRCNN, as well as FastAdapt demonstrate the highest energy efficiency relative to their rigid variants.

The results from the YOLOv3 backbone of REPP and YOLO suggest that different implementations and frameworks result in different energy consumption. Also, from the observation of the SELSA model with different backbones, a deeper backbone (ResNet 101) has a larger energy consumption, validating our intuition that more layers of computation translate to higher energy consumption.

5 CONCLUSION

In this paper, we benchmark the state-of-the-art video object detection frameworks on the NVIDIA Jetson family of embedded devices under different GPU contention and power-mode scenarios. We show the latency, accuracy, and power consumption of FastAdapt compared with other video object detection models under each scenario. According to our observation, FastAdapt provides the best Pareto frontier of accuracy-vs-latency for all three boards and under all contention levels. However, if there are loose latency bounds, among adaptive models, FRCNN+ (i.e., FRCNN coupled with a tracker) achieves the highest accuracy while among non-adaptive models, SELSA outperforms all others. For the most well-resourced board, the AGX Xavier, and under the most stringent latency requirement, FastAdapt is able to sustain 76 fps in the no-contention scenario, while on Xavier NX, it is 74 fps, and on Jetson TX2, it is 38 fps. Among the latest video object detection models, SELSA achieves the highest mAP (81.5%, with ResNet-101) but also suffers from a high latency (> 2 seconds per frame). Also these models, SELSA, MEGA, and REPP, are not adaptive, and therefore, suffer in latency when faced with any level of contention. We also perform energy consumption experiments and find that FastAdapt, FRCNN+, and YOLO+ achieve the best, and comparable, energy efficiency, while SELSA has 6X the energy consumption of these best-in-class protocols. Thus, if one runs the boards at the lowest power level, it does not always translate to equivalent energy savings. We hope that this work points to much further work in understanding the suitability of various object detection heads on embedded boards. The understanding must encompass varying levels of resource availability on these devices as well as varying power levels of operation.

REFERENCES

[1] Yihong Chen, Yue Cao, Han Hu, and Liwei Wang. 2020. Memory enhanced global-local aggregation for video object detection. In *CVPR*. 10337–10346.

[2] NVIDIA Corporation. 2021. NVIDIA Jetson Linux Developer Guide. https://docs.nvidia.com/jetson/l4t/index.html#page/Tegra%20Linux%20Driver%20Package%20Development%20Guide/power_management_jetson_xavier.html#wwpID0E0VO0HA.

[3] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. 2016. R-FCN: Object detection via region-based fully convolutional networks. In *NeurIPS*. 379–387.

[4] Jiajun Deng, Yingwei Pan, Ting Yao, Wengang Zhou, Houqiang Li, and Tao Mei. 2019. Relation distillation networks for video object detection. In *ICCV*. 7023–7032.

[5] Biyi Fang, Xiao Zeng, and Mi Zhang. 2018. Nestdnn: Resource-aware multi-tenant on-device deep learning for continuous mobile vision. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*. 115–127.

[6] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. 2020. Ghostnet: More features from cheap operations. In *CVPR*. 1580–1589.

[7] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. 2010. Forward-backward error: Automatic detection of tracking failures. In *2010 20th international conference on pattern recognition*. IEEE, 2756–2759.

[8] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *ECCV*. Springer, 740–755.

[9] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. 2016. SSD: Single shot multibox detector. In *ECCV*, Vol. 9907. 21–37.

[10] Ashraf Mahgoub, Alexander Michaelson Medoff, Rakesh Kumar, Subrata Mitra, Ana Klimovic, Somali Chaterji, and Saurabh Bagchi. 2020. OPTIMUSCLOUD: Heterogeneous Configuration Optimization for Distributed Databases in the Cloud. In *2020 USENIX Annual Technical Conference (USENIX ATC 20)*. 189–203.

[11] Ashraf Mahgoub, Karthick Shankar, Subrata Mitra, Ana Klimovic, Somali Chaterji, and Saurabh Bagchi. 2021. SONIC: Application-Aware Data Passing for Chained Serverless Applications. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*. USENIX Association, Virtual. forthcoming

[12] Ashraf Mahgoub, Paul Wood, Alexander Medoff, Subrata Mitra, Folker Meyer, Somali Chaterji, and Saurabh Bagchi. 2019. SOPHIA: Online Reconfiguration of Clustered NoSQL Databases for Time-Varying Workloads. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*. USENIX Association, Renton, WA. https://www.usenix.org/conference/atc19/presentation/mahgoub

[13] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *CVPR*. 779–788.

[14] Joseph Redmon and Ali Farhadi. 2018. YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767* (2018).

[15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NeurIPS*. 91–99.

[16] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* 115, 3 (2015), 211–252.

[17] Alberto Sabater, Luis Montesano, and Ana C Murillo. 2020. Robust and efficient post-processing for video object detection. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*.

[18] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*. 4510–4520.

[19] Mingxing Tan, Ruoming Pang, and Quoc V Le. 2020. EfficientDet: Scalable and efficient object detection. In *CVPR*. 10781–10790.

[20] Haiping Wu, Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. 2019. Sequence level semantics aggregation for video object detection. In *ICCV*. 9217–9225.

[21] Ran Xu, Rakesh Kumar, PengCheng Wang, Ganga Meghanath, Somali Chaterji, Subrata Mitra, and Saurabh Bagchi. 2021. ApproxNet: Content and Contention Aware Video Analytics System for the Edge. *ACM Trans. Sensor Netw.* (2021).

[22] Ran Xu, Chen-lin Zhang, Pengcheng Wang, Jayoung Lee, Subrata Mitra, Somali Chaterji, Yin Li, and Saurabh Bagchi. 2020. ApproxDet: content and contention-aware approximate object detection for mobiles. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems (SenSys)*. 449–462.

[23] Shifeng Zhang, Longyin Wen, Xiao Bian, Zhen Lei, and Stan Z Li. 2018. Single-shot refinement neural network for object detection. In *CVPR*. 4203–4212.

[24] Xizhou Zhu, Yujie Wang, Jifeng Dai, Lu Yuan, and Yichen Wei. 2017. Flow-guided feature aggregation for video object detection. In *ICCV*. 408–417.