Benchmarking Real-time Image Processing for Offloading at the Edge

Olivier Brochu[†], Dimitrios Spatharakis[*], Dimitrios Dechouniotis[*] Aris Leivadeas[†], Symeon Papavassiliou[†]

[†] *Department of Software and Information Technology Engineering, École de technologie supérieure, Montréal, Canada*
[*] *School of Electrical and Computer Engineering, National Technical University of Athens, Greece*
olivier.brochu.1@ens.etsmtl.ca, {dspatharakis, ddechou}@netmode.ntua.gr, aris.leivadeas@etsmtl.ca, papavass@mail.ntua.gr

Abstract—**Modern applications involve complex tasks such as machine learning and multimedia content. Resource-constrained devices are unable to guarantee real-time processing for time-critical applications. Therefore, Edge Computing provides the necessary resources to meet the stringent requirements.**

Index Terms—**Edge Computing, Task Offloading, Image Compression, Machine Learning, Yolov5**

I. Overview

This paper focuses on the use case of a robotic search and rescue in the context of Industrial Internet of Things (IIoT). In this use case, a robot or an Unmanned Aerial Vehicle, looks for subjects of interest (e.g., humans) in a designated area. The robot performs the task by periodically capturing high-resolution images of the landscape and running inference, i.e., applying a Machine Learning (ML) model, to identify a list of predicted objects and their location, and with some confidence regarding the prediction.

In a search and rescue mission, missing subjects may be in a life-threatening condition. Therefore, the inference must be as fast and as accurate as possible. Due to computing power improvements in IIoT, resource-constrained robots can run computationally-light ML models in real-time with low battery usage. However, these models lack the required accuracy to find subjects that can be only a few pixels wide. Nevertheless, computationally-heavy ML models are more accurate, however, executing them in the robot results in a significant increase in inference time and battery usage. Similarly to recent works in the literature [1], the robot may choose to offload such process to a more powerful local edge computing server deployed by the search and rescue team. In this work, we employ an Edge server instead of a Cloud infrastructure to reduce the transmission overhead [2]. The Edge server has the necessary resources to run a heavier ML model in real-time. Before offloading, the robot encodes the image to minimize the transmission time. Fig. 1 depicts the system's flowchart.

A. Featured Software

YOLOv5 [1] (You Only Look Once version 5) is a ML algorithm and a family of models that aims to permit real-time object detection on a GPU. It is based on the YOLOv4 [3] ML algorithm and improves its convenience. YOLOv5 offers five pre-trained models on the MS-COCO 2017 dataset [4] (nano, small, medium, large and x-large) provided in two input sizes (640 and 1280 pixels). Heavier models can offer better accuracy at the cost of more floating-point operations. We used the version 6.1 of the pre-trained models.

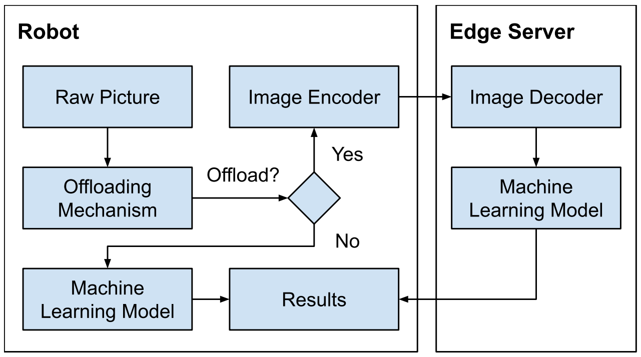

Fig. 1. Search and rescue system flowchart.

Moreover, libjxl[2] is the reference implementation of the JPEG XL image format specification. It generally offers a better compression ratio, image fidelity, and encoding and decoding speed than legacy JPEG formats (JPEG and JPEG 2000) and modern image formats such as WebP and HEIC [5]. JPEG XL aims to replace the original JPEG format by ensuring backward compatibility and lossless transcoding of JPEG files. New features such as support for the alpha channel, GIF-like animation, and 360 degree images are introduced. We used the version 0.6.1 of libjxl.

B. Featured Hardware

Table I details the hardware specifications of the robot (Intel NUC 11) and the edge server used to run the experiments.

II. Innovation

In this paper, we follow the current trends in Edge Robotics. Recent works involving real-time image recognition, data processing, and robot learning, e.g., [6] [7], rely heavily on a task offloading scheme to increase the accuracy of specific

[1]https://github.com/ultralytics/yolov5

[2]https://github.com/libjxl/libjxl

TABLE I
FEATURED HARDWARE SPECIFICATIONS

	Robot	**Edge Server**
OS	Ubuntu 20.4 LTS	
CPU	Intel Core i5-1135G7	AMD Ryzen 5 5600G
Memory Capacity	8 GiB	16 GiB
Memory Speed	3200 MHz DDR4	
GPU	–	Nvidia GeForce GTX 1660 SUPER
GPU Memory	–	6 GiB
GPU Driver	–	515.43.04
CUDA Version	–	11.7

applications, while reducing the response time and preserving resources on IIoT devices. Approximate computing is an innovative idea for selecting between different efficiency or quality configurations for a compute-intensive application that can be executed in different computing resources (i.e., GPU or CPU). However, deciding whether to offload such tasks is essential. Task-offloading decision schemes are crucial in maximizing resource utilization while preserving the Quality of Service (QoS) requirements. Moreover, utilizing the Edge Computing paradigm the overhead of communication is minimized. To this extent, we believe that this work is a significant preliminary benchmark using state-of-the-art software to support such a framework.

III. Relevance to Meditcom

The above-described challenges in Edge Robotics and real-time data processing are the cornerstones of current network and communication optimization. Investigating different settings for modern applications is crucial in finding a trade-off between accuracy, inference time, and resource utilization. It attracts many researchers from the academia and/or industry, who investigate relevant challenging research topics.

IV. Demo Summary

We conducted three sets of experiments to showcase three main aspects of the proposed scenario, namely; (i) YOLOv5 Accuracy Benchmark under different sets of image parameters, (ii) JPEG XL Compression Benchmark using different efforts and quality factors, and (iii) YOLOv5 Processing Benchmark using different number of requests. The code repositories[3] to reproduce the experiments are publicly available.

A. YOLOv5 Accuracy Benchmark

In this experiment, we selected 150 images that contained people from the People Overhead dataset[4]. The images were rotated to landscape and resized to different heights while keeping their aspect ratio and encoded in the JPEG XL format with different quality factors. The quality factor is a real number of at most 100 that controls the maximum butterraugli distance (i.e., psychovisual distance) of the encoded image

[3]https://github.com/NTUA-Edge-Robotics

[4]https://www.kaggle.com/datasets/hifrom/people-overhead

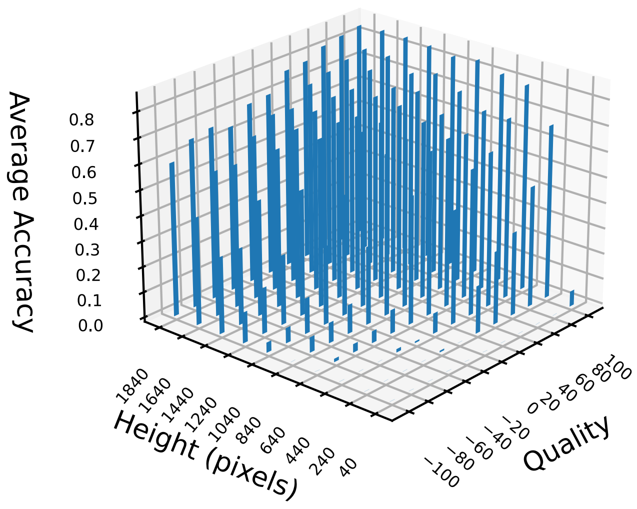

Fig. 2. Accuracy of the YOLOv5 predictions according to the height and the quality factor of the images.

with the input image. A quality factor of 100 produces a mathematically lossless image while a quality factor of 90 produces a visually lossless image. A lower quality factor reduces the bitrate of the encoded image at the cost of lower visual fidelity. We ran the inference on the edge server GPU using the YOLOv5 x-large model with an input size of 640 and recorded the accuracy of the predictions. In this experiment, we study the impact of the height and the quality factor of an image on the accuracy of the YOLOv5 predictions to identify the parameters that allow the fastest transmission time with minimal accuracy loss.

Fig. 2 shows the results. Images bigger than the YOLOv5 input size yield more accurate results. We believe it is due to the bilinear scaling algorithm that resizes the images and favors speed over quality. Images with a quality factor of 100 generally yield the most accurate results, as long as they are at least as big as the YOLOv5 input size. However, as shown in the next experiment, these images have a bitrate that is at least 3.4 times higher than the other quality factors, which increases their transmission time. In the offloading mechanism, a reasonable compromise between the transmission time and accuracy of the predictions may be to encode at a quality factor of 90 while keeping their high resolution.

B. JPEG XL Benchmark

In this experiment, we selected 213 images from the DOTA dataset [8] that we cropped to 640 by 640 pixels. We selected this dataset because it offers the content diversity needed to properly benchmark an image codec. We encoded the images using libjxl with different quality factors and different efforts on the robot CPU. We ignored quality factors lower than 40 because, as shown in Fig. 2, the accuracy of the predictions is too low. Efforts are presets that enable or disable JPEG

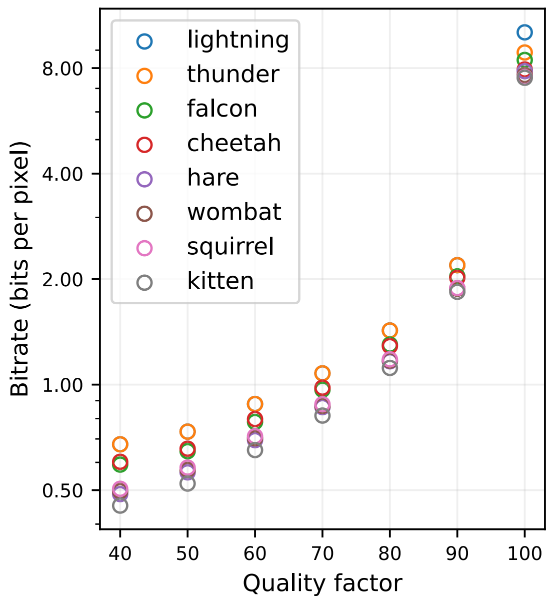

Fig. 3. Average bitrate, in bits per pixel, according to the quality factor for each effort. The y-axis is in logarithmic scale.

TABLE II
AVERAGE ENCODING SPEED, IN MEGAPIXELS PER SECOND, ACCORDING TO THE QUALITY FACTOR FOR EACH EFFORT

	Quality Factor						
Effort	**40**	**50**	**60**	**70**	**80**	**90**	**100**
lightning	33.04	32.92	50.65	48.2	45.4	40.52	26.21
thunder	32.08	31.60	49.95	48.08	44.96	40.12	22.83
falcon	31.61	31.44	48.99	46.81	43.89	38.78	12.38
cheetah	30.96	30.29	39.24	37.84	42.64	37.86	1.46
hare	18.77	18.39	21.15	20.79	21.17	20.1	1.28
wombat	16.31	16.25	18.51	18.03	18.56	17.25	1.20
squirrel	10.26	10.00	11.01	13.78	14.32	13.81	0.83
kitten	0.84	0.83	0.88	0.9	0.9	0.90	0.35

XL encoding tools. Slower animals (e.g., kitten) enable more tools to reduce the bitrate at the cost of a slower encoding speed. We ignored the effort "tortoise" as it is too slow for real-time processing. Each combination of quality factor and effort is executed 10 times and the metrics are averaged. In this experiment, we study the impact of the effort and the quality factor on the bitrate and the encoding speed to identify the parameters that allows the fastest transmission time with minimal accuracy loss.

Fig. 3 and Table II show the results. The behavior of JPEG XL is not deterministic. Images with different content and resolution yield different results. However, effort generally offers diminishing returns. For example, "kitten", compared to "cheetah", allows an average bitrate saving of 9.5% while being 97.5% slower to encode. For an image of 640 by 640 pixels, this is an increase of 490 ms in encoding time. Lossless encoding (i.e., a quality factor of 100) is slower to encode than a lossy encoding (i.e., a quality factor below 100). We believe that this is because a lossy image has more data to encode. Moreover, in lossless encoding, the efforts "lightning", "thunder" and "falcon" offer an encoding speed that is 8.5 to 74.9 times faster than the other efforts. We believe that this is because these three efforts disable most or all coding tools which greatly reduces the required processing. In Table III, we select the optimal JPEG XL parameters for our system. The accuracy refers to the results of the previous experiment.

TABLE III
CHOSEN JPEG XL PARAMETERS FOR THE SYSTEM ACCORDING TO NETWORK CONDITIONS AND THE RESULTING IMAGE SIZE, ENCODING TIME AND ACCURACY

	Good Network Conditions	**Poor Network Conditions**
Quality Factor	100	90
Effort	thunder, falcon	cheetah or faster
Image Size (Kb)	443	108
Encoding Time (ms)	26	10
Accuracy	0.82	0.71

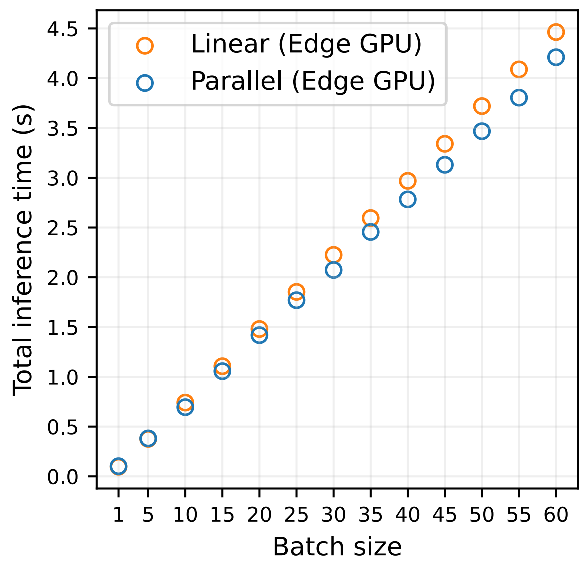

Fig. 4. Total inference time, in seconds, according to batch size in linear and parallel processing with YOLOv5 on the edge server GPU.

C. YOLOv5 Processing Benchmark

In this experiment, we used the cropped images from the previous experiment. The images were divided in batches of 1 to 60 images with a step of 5. An image can appear in multiple batches. The upper bound of 60 corresponds to the maximum number of images that the GPU can hold in memory at the same time. For each batch, we ran the inference with the x-large model on the edge server GPU and the robot CPU, linearly (i.e., one image after the other) and in parallel (i.e., all the images at the same time). This experiment studies the

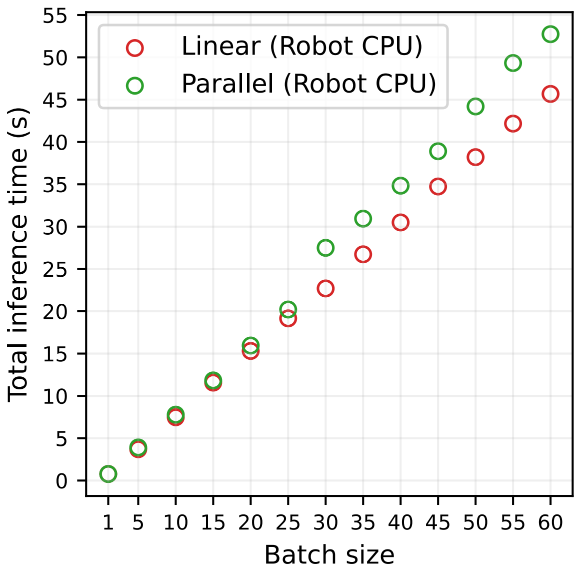

Fig. 5. Total inference time, in seconds, according to batch size in linear and parallel processing with YOLOv5 on the robot CPU.

impact of the batch size on the total inference time and the average inference time on the robot and the edge server.

Figs. 4 and 5 and Table IV show the results. In the linear average inference time, the edge server GPU is about 13.3 times faster than the robot's CPU. On the edge server's GPU, the total inference time is faster in parallel than linearly. On the robot CPU, it is the opposite. In larger batch sizes, a greater gap in the total inference time can be observed between the two processing modes. We believe that this is because YOLOv5 is a fully parallelized application optimized for GPUs. In Table IV, in the linear average inference time, the batch sizes 1 and 5 yield different results. This is because the first image of each batch is always slower to infer. This behavior raises the average in smaller batches. The results show an interesting trade-off for the system. While processing more images in parallel reduces the total inference time, the response time increases.

V. Demo Presentation and Conclusion

A demo of the offloading mechanism will be presented using a prerecorded video. The robot offloads batches of tiles to the edge server for inference. The inference result will be shown by superimposing the identified objects of interest, their estimated location and the confidence to the tiles. In this article, we conducted benchmarks of the technologies that support an edge offloading mechanism in an Edge IIoT search and rescue use case. The results allow us to identify the parameters that will enable the system to operate in real-time.

Acknowledgment

This work was supported in part by Mitacs and École de technologie supérieure through the Mitacs Globalink program and in part by Les Offices jeunesse internationaux du Québec.

TABLE IV
Average inference time per image, in seconds, according to batch size in linear and parallel processing with YOLOv5

Batch Size	**Average Inference Time (Linear)**		**Average Inference Time (Parallel)**	
	GPU	**CPU**	**GPU**	**CPU**
1	0.096	0.777	0.101	0.774
5	0.075	0.740	0.377	3.899
10	0.074	0.747	0.683	7.615
15	0.074	0.771	1.026	11.559
20	0.074	0.765	1.386	15.531
25	0.074	0.766	1.727	19.658
30	0.074	0.756	2.028	26.882
35	0.074	0.763	2.363	30.002
40	0.074	0.762	2.707	33.397
45	0.074	0.772	3.057	36.894
50	0.074	0.764	3.372	41.755
55	0.074	0.767	3.691	47.236
60	0.074	0.761	4.08	49.773

This work was supported by the CHIST-ERA grant CHIST-ERA-18-SDCDN-003 (DRUID-NET), and is co-financed by Greece and European Union under the Operational Programme ”Competitiveness, Entrepreneurship and Innovation” (EPAnEK) through the Greek General Secretariat for Research and Innovation (GSRI), grant number T11EPA4-00022.

References

[1] D. Spatharakis, M. Avgeris, N. Athanasopoulos, D. Dechouniotis, and S. Papavassiliou, “Resource-aware estimation and control for edge robotics: a set-based approach,” *IEEE Internet of Things Journal*, pp. 1–1, 2022.

[2] M. Afrin, J. Jin, A. Rahman, A. Rahman, J. Wan, and E. Hossain, “Resource allocation and service provisioning in multi-agent cloud robotics: A comprehensive survey,” *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 842–870, 2021.

[3] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “Yolov4: Optimal speed and accuracy of object detection,” 2020. [Online]. Available: https://arxiv.org/abs/2004.10934

[4] 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,” 2014. [Online]. Available: https://arxiv.org/abs/1405.0312

[5] E. Kliuchnikov, E. Upenik, J. Wassenberg, J. Sneyers, J. Alakuijala, L. Vandevenne, L. Versari, S. Boukortt, and T. Ebrahimi, “Benchmarking jpeg xl lossy/lossless image compression,” in *Optics, Photonics and Digital Technologies for Imaging Applications VI*, 2020. [Online]. Available: http://infoscience.epfl.ch/record/277420/files/Submitted%20manuscript.pdf

[6] S. Chinchali, A. Sharma, J. Harrison, A. Elhafsi, D. Kang, E. Pergament, E. Cidon, S. Katti, and M. Pavone, “Network offloading policies for cloud robotics: a learning-based approach,” *Autonomous Robots*, vol. 45, no. 7, pp. 997–1012, 2021.

[7] Z. Cai, Y. Qu, and C. Dong, “Edge intelligence-based uav human target recognition with improved yolov5 algorithm,” in *2021 IEEE 23rd Int Conf on High Performance Computing & Communications; 7th Int Conf on Data Science & Systems; 19th Int Conf on Smart City; 7th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*, 2021, pp. 861–868.

[8] G.-S. Xia, X. Bai, J. Ding, Z. Zhu, S. Belongie, J. Luo, M. Datcu, M. Pelillo, and L. Zhang, “Dota: A large-scale dataset for object detection in aerial images,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.