 DOI 10.1109/NANO58406.2023.10231293

Energy Efficient Object Detection for Automotive Applications with YOLOv3 and Approximate Hardware

Jordi Fornt, Pau Fontova-Musté, Martí Caro, Jaume Abella, Francesc Moll, Josep Altet, and Antonio Rubio

***Abstract*—Deep neural networks are the dominant models for perception tasks in the automotive domain, but their high computational complexity makes it difficult to execute them in real time with an acceptable power consumption on general-purpose devices. For this reason, the design of custom ASIC devices for real-time energy-efficient neural network deployment is a hot topic in academia and industry. Low-precision integer arithmetic and approximate computing are two popular optimizations often contemplated for saving hardware resources and power consumption. In this work, we evaluate two approximate computing circuits using different integer precisions in order to study the trade-offs that these techniques offer in terms of energy efficiency and degradation of the neural network outputs. In particular, we apply the approximations to the YOLOv3 object detection network, a popular model for critical applications in the automotive domain. By combining approximate arithmetic circuits with low precision we are able to reduce the power consumption of a MAD unit by over 50% compared to using only quantization.**

***Index Terms*—Approximate computing, object detection, low-precision, quantization, energy-efficient DNN acceleration.**

I. Introduction

Autonomous driving and advanced driver-assistance systems (ADAS) have made great leaps forward in recent years thanks to the widespread adoption of deep neural network (DNN) models to handle complex tasks, especially regarding camera-based perception. The outstanding accuracy achieved by these models comes at the expense of high complexity and a large number of learnable parameters, which makes them very computationally intensive. To perform a single forward pass, state-of-the-art DNNs have to perform arithmetic operations in the order of hundreds of billions. In an automotive setting, the required processing time of the perception pipeline is in the range of tens of milliseconds, so the underlying DNN models need to be executed at compute rates of the order of 10 TFLOP/s.

Typically, graphics processing units (GPUs) are used to run DNNs in real-time, but their high throughput comes at the cost of high power consumption, which is a liability in an automotive setting. For this reason, specialized hardware accelerators are becoming increasingly relevant in the industry, since custom application-specific integrated circuits (ASICs) can achieve higher energy efficiencies than general purpose GPUs. Dataflow accelerators are a popular example [1]–[4], based on the massive replication of simple arithmetic units that share and reuse data, exploiting the embarrassingly parallel nature of DNNs.

Nevertheless, even specialized hardware accelerators fall short of the required energy efficiency for executing state-of-the-art DNNs in real time with low power consumption. For this reason, optimizations are being applied at different levels to further push the energy efficiency of neural accelerators. Two popular examples of such techniques are low-precision quantization and approximate computing. Low-precision quantization consists on encoding the weights and feature maps as integer values with a limited number of bits, which decreases the hardware costs of multiply-add units as well as the memory footprint of the model. Approximate computing, on the other hand, seeks to optimize the hardware units by approximating the arithmetic operations of the network in ways that allow to decrease power consumption while retaining as much accuracy as possible [5].

In this work, we present a use-case of approximate computing and post-training quantization to improve the energy efficiency of accelerators targeting object detection for automotive applications using the YOLOv3 network [6]. We use this setting to study the trade-off between power consumption, low bit quantization, aggressiveness of the arithmetic approximations and object detection degradation.

II. Object Detection with YOLOv3

You Only Look Once (YOLO) is a real-time object detection DNN proposed in [7] and later updated to YOLOv3 in [6] which has seen a lot of success in both academia and industry [8]–[10]. It is also a popular model used in the automotive industry, with YOLOv3 being the backbone of the Apollo autonomous driving framework [11]. YOLO takes a color image as input and, after a single forward pass, outputs a set of bounding boxes defined by a class value plus 4 coordinates: two describing their position and two for their dimensions.

What differentiates YOLO from previous object detection networks is that, instead of generating many detection proposal regions and applying a DNN on each one, it performs a single forward pass to generate all bounding boxes. It does so by partitioning the image into a grid and posing the object detection task as a regression problem: for each

J. Fornt, M. Caro and F. Moll are with the Barcelona Supercomputing Center (BSC) and the Universitat Politècnica de Catalunya (UPC).

P. Fontova-Musté and J. Abella are with the BSC.

J. Altet and A. Rubio are with the UPC.

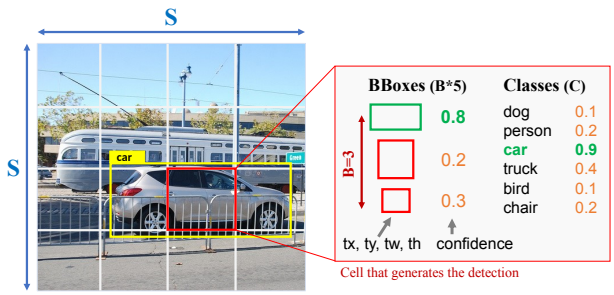

Fig. 1. Toy example showing the YOLO regression outputs of a grid cell.

grid location it generates a set of bounding boxes at different scales, consisting on 5 predictions: their 4 coordinates plus a confidence score that represents the probability of a bounding box being the correct one (see Fig. 1). Additionally, a class probability is generated for all classes on each grid cell. If the largest class probability exceeds a given threshold, an object of that class is detected on that specific cell and the bounding box prediction with highest confidence is chosen.

Since the publication of YOLOv3 in 2018, new versions of the network have appeared at a steady pace [12], [13], improving the detection accuracy by using more advanced CNN architectures and training techniques. Nevertheless, we use YOLOv3 as our evaluation network because of its aforementioned relevance in the automotive domain, as well as the fact that newer versions do not contain major changes that could radically alter the results of our study.

III. Methodology

We use the Multiply-Add (MAD) unit depicted in Fig. 2 as a baseline model. I_B determines the bit precision of the product inputs, and the parameter O_B describes the precision of the partial sum. For our study we set $O_B = 2I_B + 8$, since we found experimentally that 8 additional bits after the multiplication are enough to avoid overflows on YOLOv3.

For a given precision configuration (I_B value) of the MAD unit, we replace the multiplier by different approximate circuits, generating approximate MAD units with varying aggressiveness of approximation. In particular, in this case study we experiment with two approximate multiplier architectures:

1) A Broken Array Multiplier (BAM), as proposed in [14].
2) An Approximate Logarithmic Multiplier (ALM) with Set-One Adder (SOA) [15], in which we use log-encoded inputs in order to avoid the hardware overhead of the linear to log encoding modules, similar to [16].

These multipliers have parameters that determine how aggressive the approximation is, so each architecture can generate many design points of interest. The BAM has two parameters h_{BL} and v_{BL} (horizontal and vertical break lines), that determine the number of partial product cells skipped [14]. The ALM has a single approximation parameter m that determines the number of bits fixed to '1' by the SOA [15].

Both multiplier architectures are unsigned, so we deal with the product sign outside of the main multiplier logic, and

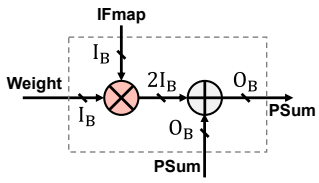

Fig. 2. Fixed-point MAD unit used for evaluation.

use the absolute value of the operands in the multiplication itself. For the case of the ALM, we expand the log-encoded absolute values by two bits: one for the sign and an additional bit that is set to '1' if the operand is zero. The latter allows our multiplier to deal with the case in which either of the original inputs is zero, which has a high error rate in logarithmic multipliers.

Even though in this study we do not consider applying approximations in the partial product sum, we still consider the adder as part of the MAD unit to be analyzed, for two reasons: first, the multiply-add operation is the core compute element of most dataflow accelerators, so considering both the multiplier and adder makes our power estimation results more meaningful. Second, some approximate multiplier architectures indirectly simplify the adder as well, which is an effect that can only be seen by considering the complete MAD unit. Nevertheless, the multiplier is typically the most energy-hungry block present in a multiply-add unit (over two thirds of the power in our exact int8 MAD), so it makes sense to focus on it when applying approximations.

Our experimental methodology is structured as follows. For each approximate MAD unit we test, we define two models: an HDL model of the circuit in SystemVerilog, which is used to synthesize the hardware; and a C model of the equivalent approximate operation, which is used to execute YOLOv3 with the approximate MAD as the main operation. Even though the HDL model would be enough to characterize the circuit and its behavior, running a DNN with RTL-level simulations is infeasible, so we define a C model that is much faster to execute and behaviorally equivalent.

From our HDL models we perform a synthesis of each MAD version using Cadence Genus, with 22 nm physical libraries (GF22FDX). The synthesis runs are performed at a target frequency of 500 MHz, considering operating conditions of 0.8 V and 25ºC. From the post-synthesis netlist we perform a power estimation using Cadence Joules, considering the switching activities obtained from a gate-level simulation of the same netlist with inputs obtained from YOLOv3.

From our C models we estimate the accuracy of the object detection by simulating a forward pass of YOLOv3 for a set of test images from the COCO dataset [17], replacing the multiplications and additions in the matrix-matrix multiplications by our approximate circuit models. We use the Mean Average Precision (mAP) metric [18] to quantify the accuracy of the output detections. Given that we are interested in object detection for automotive applications, we consider only classes corresponding to vehicles and people for our

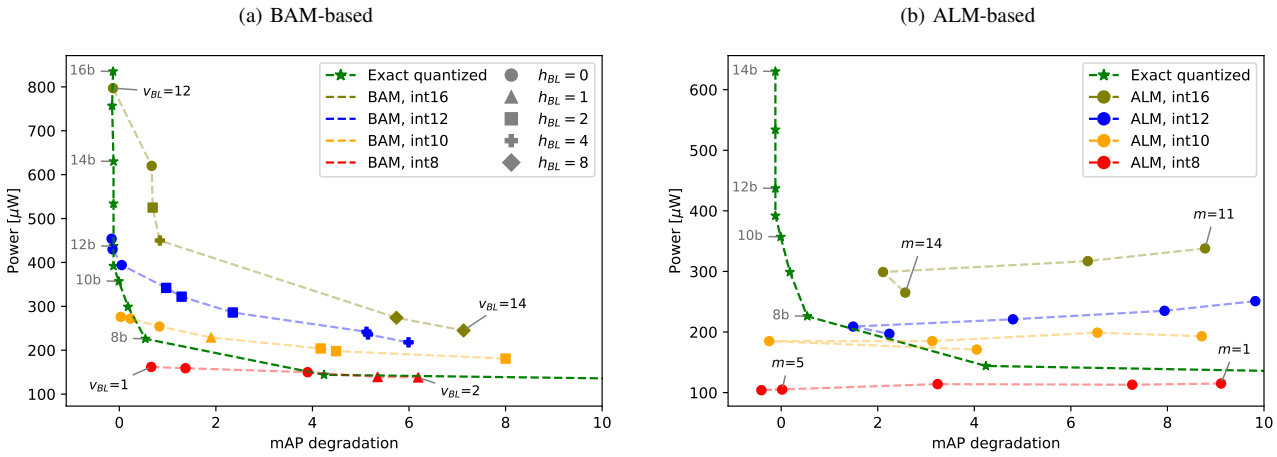

Fig. 3. Power versus mAP degradation ($mAP_{FP32} - mAP_{unit}$) for the approximate multipliers under study, at different baseline precisions. (a) shows only the pareto-optimal points resulting from a design space exploration of the BAM parameters, while (b) shows all the design points in the plot range.

calculation of the mAP, which are the most interesting objects for many ADAS applications. In particular, we consider the following 5 classes:

(a) Person
(b) Car
(c) Motorcycle
(d) Bus
(e) Truck

As we are only interested in this subset of classes, we filter the COCO test dataset and only consider the images in which at least one of these objects appear, resulting in a test set of 869 images, including all the instances of the aforementioned classes in the dataset.

Regarding the integer quantization of YOLOv3, we use post-training quantization with linear scales, using per-layer scaling values for the feature maps and per-channel scalings for the weights, similar to [19] with the zero-points fixed to zero. We calibrate the values of the feature map scales by analyzing the execution of the original network in full precision and keeping track of the maximum absolute values on each layer. Before computing the quantization scales, we apply a tolerance factor of 0.3 to the max values in order to increase the effective dynamic range during execution, and let the higher values saturate in the range $[-2^{I_B-1}, 2^{I_B-1}-1]$.

IV. Results

The results of our trade-off analysis of the multipliers are summarized in Fig. 3, where we show the evolution of power consumption versus mAP degradation for different versions of the MAD unit. The green curve present in both Fig. 3a and Fig. 3b locates the power and mAP values corresponding to exact MAD units with increasingly low bit precision. We use this curve to evaluate the trade-off of the different approximate MAD versions: only design points below the exact quantized curve are worth considering, since they provide a clear benefit over simply applying further quantization to the network. In particular, the exact int8 version has the lowest bit precision that retains a negligible mAP degradation of less than 1 unit, so it is the benchmark to beat by approximating the MAD hardware.

We test the approximate architectures with different precision baselines in order to study the interaction between the post-training quantization and the approximations in terms of mAP-Power trade-off. Lowering the precision by itself is clearly beneficial in terms of power savings. However, as Figures 4 and 5 indicate, it limits the aggressiveness we can apply to the approximation without major performance degradation, hence also limiting potential power reductions. Nevertheless, the results from Fig. 3 suggest that the best design points are always obtained by reducing the baseline precision as much as possible and then applying the approximation to further improve the energy efficiency.

Focusing on the results of the BAM, which are shown in Fig. 3a, we see that the architecture shows the typical trade-off of approximate compute units: higher approximation parameters result in lower power but also a degradation in network accuracy. Fig. 4 depicts the evolution of the YOLOv3 mAP as we increase the aggressiveness of the multiplier via its parameters v_{BL} and h_{BL}. Using a higher number of bits allows for more approximation, as there is less quantization noise, but the difference is not enough to surpass the mAP-Power trade-off of the lower-bit approximate versions. At its best design point, the 8-bit version of the BAM achieves a 30% power reduction with respect to the exact int8 MAD, while maintaining almost the same mAP. Comparatively, an exact int7 unit reduces power by 36%

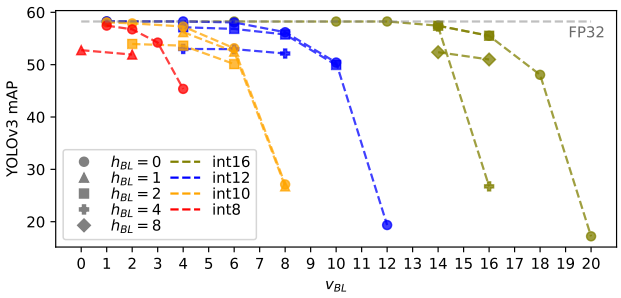

Fig. 4. YOLOv3 mAP using the BAM, for different values of $[v_{BL}, h_{BL}]$.

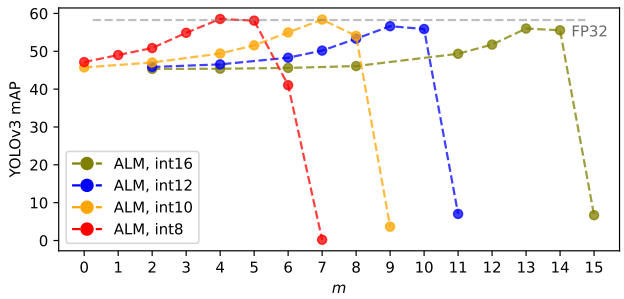

Fig. 5. YOLOv3 mAP using the ALM, versus its parameter m.

but has a mAP degradation of 4.24 units. Higher precision versions of the MAD unit are not able to beat the low-bit quantization benchmark, laying consistenly above the exact quantized curve.

Regarding the ALM, we see in Fig. 5 that the evolution of the mAP with its approximation parameter m is quite different due to a particularity of this architecture: the logarithmic multiplier itself, without further approximation, is inexact (at $m = 0$ some error is expected) and when using the ALM-SOA architecture, increasing the aggressiveness of the approximation improves the mAP up to a certain point. This effect has been studied in previous works [15] and comes from the fact that the internal SOA adder that applies the approximation compensates the error introduced by Mitchell's algorithm [20]. Thanks to this fact and the relatively low cost of logarithmic multipliers without encoders (in which the inputs are already log-encoded, as explained in Section III), ALMs present a very competitive mAP-Power trade-off as evident in Fig. 3b. Both the 10-bit and the 8-bit versions of the architecture show better trade-offs than further quantizing the exact baseline, and the best performing design point, based on int8, achieves a power reduction of 54% with respect to the exact 8-bit unit, while slightly improving the mAP. This power reduction is similar to what could be obtained using an exact unit with 6-bit precision, which would degrade the mAP by 28.92 units.

V. Conclusions

We present a case study of two approximate multiplier circuits under different bit precisions for the execution of YOLOv3 targeting object detection for automotive applications. By comparing the mAP degradation of the network with the hardware gains in terms of power, we analyze the trade-offs of each approximate MAD version under different baseline precisions and approximation parameter values.

Thanks to the redundancy and error-tolerance of YOLOv3, we are able to apply aggressive approximations and low-bit quantization to the network while retaining accuracy values very close to the full precision baseline. We find that, by combining both techniques, we are able to generate MAD units with better energy-accuracy trade-off than by just further reducing the bit precision. Even though using approximate circuits with low bit-precision limits the range of aggressiveness that can be applied to the approximation, the energy-accuracy trade-off is better than using more aggressive approximate units with more bits. Hence, the recommendation is to first quantize as much as possible and then fine-tune by approximating the computation.

From the approximate units that we have studied, we find that an ALM-SOA multiplier based on int8 precision yields the best energy-accuracy profile, with the best design point achieving a power reduction of 54% with respect to the exact int8 implementation, while retaining the same mAP as the full precision baseline.

References

[1] A. Yazdanbakhsh *et al.*, "An Evaluation of Edge TPU Accelerators for Convolutional Neural Networks," *CoRR*, vol. abs/2102.10423, 2021.

[2] Y.-H. Chen *et al.*, "Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 2, pp. 292–308, 2019.

[3] B. Moons *et al.*, "14.5 envision: A 0.26-to-10tops/w subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28nm fdsoi," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 246–247, 2017.

[4] S. Venkataramani *et al.*, "RaPiD : AI Accelerator for Ultra-low Precision Training and Inference," in *Proceedings - 2021 48th International Symposium on Computer Architecture, ISCA 2021*, pp. 153–166, 2021.

[5] H. Jiang *et al.*, "Approximate Arithmetic Circuits: A Survey, Characterization, and Recent Applications," *Proceedings of the IEEE*, vol. 108, no. 12, pp. 2108–2135, 2020.

[6] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *CoRR*, vol. abs/1804.02767, 2018.

[7] J. Redmon *et al.*, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015.

[8] "Apollo Open Platform." https://developer.apollo.auto/.

[9] E. N. Ukhwah *et al.*, "Asphalt Pavement Pothole Detection using Deep learning method based on YOLO Neural Network," in *2019 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, pp. 35–40, 2019.

[10] J. Hu *et al.*, "Detection of Workers Without the Helments in Videos Based on YOLO V3," in *2019 12th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pp. 1–4, 2019.

[11] "Apollo 3.0. Perception (Github)." https://github.com/ApolloAuto/apollo/blob/master/docs/specs/perception_apollo_3.0.md.

[12] A. Bochkovskiy *et al.*, "Yolov4: Optimal speed and accuracy of object detection," *CoRR*, vol. abs/2004.10934, 2020.

[13] X. Zhu *et al.*, "Tph-yolov5: Improved yolov5 based on transformer prediction head for object detection on drone-captured scenarios," *CoRR*, vol. abs/2108.11539, 2021.

[14] H. R. Mahdiani *et al.*, "Bio-Inspired Imprecise Computational Blocks for Efficient VLSI Implementation of Soft-Computing Applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 4, pp. 850–862, 2010.

[15] W. Liu *et al.*, "Design and Evaluation of Approximate Logarithmic Multipliers for Low Power Error-Tolerant Applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 9, pp. 2856–2868, 2018.

[16] L. Oliveira *et al.*, "Design of power-efficient fpga convolutional cores with approximate log multiplier," in *2019 European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 04 2019.

[17] T. Lin *et al.*, "Microsoft COCO: Common Objects in Context," *CoRR*, vol. abs/1405.0312, 2014.

[18] M. Everingham *et al.*, ""The Pascal Visual Object Classes (VOC) Challenge"," *International Journal of Computer Vision*, vol. 88, pp. 303–338, June 2010.

[19] "Tensorflow lite 8-bit quantization specification." https://www.tensorflow.org/lite/performance/quantization_spec.

[20] J. N. Mitchell, "Computer Multiplication and Division Using Binary Logarithms," *IRE Transactions on Electronic Computers*, vol. EC-11, no. 4, pp. 512–517, 1962.