An Automotive Case Study on the Limits of Approximation for Object Detection

 ${\rm Mart}{\rm i} \ {\rm Caro}^{^{\dagger}, \, ^{\ddagger}}, \, {\rm Hamid} \ {\rm Tabani}^{^{\dagger}}, \, {\rm Jaume} \ {\rm Abella}^{^{\dagger}}, \, {\rm Francesc} \ {\rm Moll}^{^{\dagger}, \, ^{\ddagger}}, \, {\rm Enric}$

Morancho[‡], Ramon Canal^{†, ‡}, Josep Altet[‡], Antonio Calomarde[‡], Francisco J.

Cazorla^{\dagger}, Antonio Rubio^{\ddagger}, Pau Fontova^{\dagger , \ddagger}, and Jordi Fornt^{\dagger , \ddagger}

[†]Barcelona Supercomputing Center (BSC) [‡]Universitat Politècnica de Catalunya (UPC)

Abstract

The accuracy of camera-based object detection (CBOD) built upon deep learning is often evaluated against the real objects in frames only. However, such simplistic evaluation ignores the fact that many unimportant objects are small, distant, or background, and hence, their misdetections have less impact than those for closer, larger, and foreground objects in domains such as autonomous driving. Moreover, sporadic misdetections are irrelevant since confidence on detections is typically averaged across consecutive frames, and detection devices (e.g. cameras, LiDARs) are often redundant, thus providing fault tolerance.

This paper exploits such intrinsic fault tolerance of the CBOD process, and assesses in an automotive case study to what extent CBOD can tolerate approximation coming from multiple sources such as lower precision arithmetic, approximate arithmetic units, and even random faults due to, for instance, low voltage operation. We show that the accuracy impact of those sources of approximation is within 1% of the baseline even when considering the three approximate domains simultaneously, and hence, multiple sources of approximation can be exploited to build highly efficient accelerators for CBOD in cars.

1 Introduction

Systems based on Artificial Intelligence are becoming ubiquitous these days for a variety of applications across domains. Those based on deep learning are particularly popular for object detection, and hardware and software designs to achieve increasingly high accuracy and confidence on the predictions within limited time bounds (e.g. to process images from cameras at a high rate) are continuously improved.

Camera-based object detection (CBOD) building upon deep learning has been deployed in a variety of applications across multiple domains, some of them with safety requirements, and thus with limitations in the level of errors that can be tolerated. This is, for instance, the case of CBOD in Autonomous Driving (AD), where accurate object detection is mandatory to guarantee safe progress towards the destination. For instance, You Only Look Once (YOLO) [1,2] is an award-winning popular and efficient real-time CBOD system already used in industrial autonomous driving systems such as NVIDIA Drive [3] and Baidu's Apollo [4], a popular industrial-quality autonomous driving software framework used in several prototype vehicles (including autonomous trucks and robotaxies).

YOLO implements a computationally-intensive function based on a Convolutional Neural Network. It is developed and trained based on IEEE754 floating-point 32-bit arithmetic and has been proven highly effective and efficient.

For each frame (image), YOLO delivers the list of objects detected with their individual *confidence levels*. By default, only objects with detection confidence above 50% are regarded as real objects for YOLO. Analogous thresholds – even identical – are considered and implemented accordingly in frameworks that employ YOLO's architecture, such as Apollo.

CBOD in general, and YOLO in particular, have a stochastic nature due to building on neural networks [5], and by delivering results with associated confidence levels. Such stochastic nature has often been leveraged to use reduced-precision arithmetic (e.g. floatingpoint 16-bit instead of floating-point 32-bit), to reduce computation costs while maintaining accuracy and, in some cases, at the expense of negligible accuracy loss.

While the effectiveness of object detection can be assessed at the granularity of an image, where we can tell whether objects have been detected, identifying false positives and false negatives, the semantic implications of false positives/negatives are irrelevant at the granularity of individual images for automotive object detection if decisions are taken averaging results across multiple frames, as it is the case of YOLO. Instead, detections are combined across multiple continuous frames captured from cameras, since only when an object is detected across multiple frames – not necessarily strictly consecutive frames – with sufficiently high confidence is regarded as a true object. For instance, an object detected in a single frame, but not in the neighbouring ones can be simply disregarded. Analogously, missing to detect with sufficient confidence an object in a single frame out of a sequence of frames where it is detected is very unlikely to change the outcome, and hence the object is regarded as a real object.

AD frameworks leverage detections across multiple frames and across multiple sensors (cameras, LiDARs, and radars) for the sake of robustness. Thus, a false positive/negative in a frame causes no semantic error on its own. This brings an additional dimension for error tolerance that can be exploited to further reduce hardware complexity.

This paper performs a case study for the trade-off between accuracy and complexity for CBOD in the context of AD systems. In particular, we show for the first time in an automotive case study that several approximation domains can be leveraged simultaneously with negligible impact on accuracy:

- A) Reduced precision arithmetic (e.g. floating-point 16-bit), in line with existing literature.
- B) Approximate arithmetic, such as approximate additions and multiplications, where results for some input operands may be systematically inaccurate.
- C) Random faults due to, for instance, low-voltage operation, which may make some operations produce arbitrarily erroneous results sporadically.

Hence, this work shows that opportunities exist for the future design of lower-power higher-frequency AD-specific CBOD accelerators by implementing low-voltage reduced-precision approximate arithmetic, without impacting on the semantics of the whole object-detection process.

The rest of the paper is organized as follows. Related work is provided in Section 2. Section 3 provides background on object detection systems as well as state-of-the-art on different approaches to improve the efficiency of deep learning models. Section 4 presents the different approximation domains and the expected impact on the CBOD outcome. Section 6 presents and evaluates the case study. Finally, Section 7 summarizes this work.

2 Related Work

Achieving efficient object detection has been the target of multiple works from different angles, including the reduction of the amount of data fetched and operated, and reducing the cost to fetch and operate such data, being both sets of solutions orthogonal and complementary. In the former area, we find works performing real-time video segmentation to compress spatio-temporal redundancy within and across frames [6], and devising new CNN architectures for low-cost salient object detection [7], and enabling CNN model compaction [8]. Those works ultimately lead to requiring less data to be fetched from memory, and to performing a lower number of operations on the data fetched. In the latter area, we find works related to using alternative data representations, approximate arithmetics and low voltage operation, which we discuss next.

There is a plethora of related works in the area of reduced precision for neural networks. The work from Tang et al. [9] proposes MLPAT, a power, area, and timing model framework for machine learning accelerators. The work explores the design space of precision and architecture tradeoffs for autonomous driving accelerators assessing the accuracy, power, and area of floating-point 16-bit, brain floating-point 16-bit, integer 16-bit, and integer 8-bit on a TPU-v1 [10], an ASIC developed by Google. Wu et al. [11] propose a custom low-precision (8-bit) floating-point quantization method for FPGA-based acceleration without re-training and study their approach on VGG16 [12] and tiny-YOLO [13]. In our work, we use the full version of YOLO v3 [2] and the standard floating-point formats. Mellempudi et al. [14] propose the use of 8-bit floating-point representation for weights, activations, errors, and gradients, although the work focuses on training a DNN.

Several works study other possible number formats, such as fixed-point, during training and inference [15], [16], [17], [18], concluding that it is feasible using alternative number format representations in the context of CNNs without significant accuracy impact. However, in practice, it is currently more feasible to use the standard floating-point number representation, since the hardware is highly optimized, while other formats may require further optimization.

Approximate arithmetic applied to CNNs has also been studied in several works. The work from Hammad et al. [19] studies the use of a dynamically configurable approximate multiplier for CNN inference on the VGG19 [20], Xception [21], and DenseNet201 [22] networks using the ImageNetV2 dataset [23]. Ibrahim et al. [24] discuss the use of approximate computing by means of employing approximate multipliers and adders for machine learning. Further related works can be found on the comprehensive survey from Manikandan [25]. To the best of our knowledge, there are no related works focusing on approximate arithmetic for CBOD in AD systems, and other works that study approximate arithmetic employ a lower complexity network. In our work, we have employed the SFA (simplified full adder) based adder [26] and a configuration of the UDM (underdesigned multiplier) [27] as illustrative examples of approximate arithmetic units. Other works study different approximate units.

The impact of faults on CNNs has been analyzed from different angles. Some other works study the impact of aggressive low voltage operations applied to CNNs. Salami et al. [28] perform an experimental study of the power-performance-accuracy characteristics of CNN accelerators with aggressive reduced supply voltage capability implemented in real FPGAs, although the study is not focused on the AD domain. Chandramoorthy [29] also performs an evaluation of low-voltage operation, but for character recognition, which is a different application to CBOD.

A. Ruospo et al. [30] characterize the impact of software-injected permanent faults in the CNN weights of the LeNet-5 CNN [31] with the MNIST dataset [32] for handwritten digit classification. Authors provide a characterization for 32-bit floating-point arithmetic and different fixed-point arithmetic configurations (32, 18, 16, 10, and 6 bits). Similarly, F. Libano et al. [33] study the impact of radiation induced hardware faults on different reduced precision models. In particular, the work analyses 32-bit and 16-bit floating-point versions, and an 8-bit integer version of a minimalist CNN (7 layers) also evaluated against the MNIST dataset [32] for handwritten digit classification, which is a simpler task than CBOD.

Other works [34,35] focus on the detection of faults exploiting spatio-temporal redundancy in the context of AD. Authors build on the fact that consecutive frames have a very high degree of visual similarity to detect faults. Authors compare the outcome of consecutive frames explicitly to detect hardware faults. Instead, we use the default implementation provided in the Darknet framework, which averages the confidence of the detections across the last three frames to mitigate some sporadic misdetections, but without including any specific mechanism for fault detection or correction. Instead, the very same mechanism used to mitigate model inaccuracies is exploited for fault tolerance in our case.

Overall, our paper is the first one providing a complete assessment of different approximation domains applied incrementally in a real-life case study for AD systems taking into account the fault-tolerance that those systems provide due to time and space redundancies.

3 Background

This section, provides background on the CBOD process, YOLO, and precision and approximation in arithmetic.

3.1 Camera-based Object Detection

The perception process in AD frameworks, such as Apollo, builds upon complementary and redundant sensors for scene understanding. This includes sensors of different types, such as cameras, LiDARs, and radars, but also sensors of the same type that may cover some surrounding areas redundantly (e.g. cameras with overlapping angles of view). This is



Figure 1: Perception process in AD.

illustrated in Figure 1. Each sensor process generates a new object list periodically (e.g. every 40 ms for CBOD at a rate of 25 *frames per second*, FPS), which are fused to generate a single object list.

The fusion process needs to provide a coherent identification of the objects surrounding the car, which implies matching objects identified by different sensors, taking into account their type, size, location, and confidence in their detection. This process is intrinsically faulttolerant since if one sensor discards a real object due to low confidence in its detection, but the other sensors covering the same area detect it with high confidence, the object will be considered for the fusion process.

The fusion process periodically generates a new list of objects (e.g. at times T_{i-2}, T_{i-1}, T_i). Then, such sequences of lists of objects need to be processed to track objects over time so that their trajectories can be properly predicted, thus allowing the AD system to take safe driving decisions. The processing of consecutive object lists is, again, intrinsically fault-tolerant since detections are leveraged across lists over time, so sporadic false positives/negatives have no semantic impact on the AD object detection process. As an illustrative example, if a car is detected in the front view of the vehicle (e.g. in object lists at times $T_{i-k}, T_{i-k+1}, ..., T_{i-1}$) but suddenly it is not detected in one object list (e.g. T_i), the trajectory prediction module will still be capable of considering such car as long as the non-detection of that car occurs



Figure 2: YOLO v3 architecture overview.

sporadically. Analogously, if a car is erroneously detected in front of the vehicle in just one object list (e.g. T_i), the trajectory prediction process will be capable of detecting it since it is impossible for such car getting in front of the vehicle in a too short time interval (e.g. in 40 ms), by being missing in T_{i-3} , T_{i-2} , and T_{i-1} , and instead, being in a specific location (e.g. close to the car) in T_i .

Overall, the perception process of an AD framework is a fault-tolerant process where independent false positive/negative detections for one sensor, or jointly for multiple sensors instantly (e.g. in a single object list produced by the fusion process), do not have any semantic impact affecting the driving decisions.

In this work, we analyse several videos captured by a single camera in an autonomous vehicle to be processed by YOLO v3, as CBOD representative. Hence, we do not exploit fault tolerance from redundant sensors due to lack of appropriate data for that purpose (e.g., data from multiple cameras from a driving sequence). However, we illustrate tolerance to errors of the process due to the intrinsic stochastic nature of the detection process in YOLO, and due to the fault tolerance of object detections across consecutive video frames.

3.2 YOLO Real-time Object Detection System

YOLO is a state-of-the-art real-time CBOD system used as the main component for CBOD in several industrial autonomous driving software frameworks [3,4,36,37]. YOLO operates on floating-point 32-bit data, so we assume this number representation as the baseline case in the rest of the paper despite our findings could be applied analogously to other representations (e.g., integer numbers).

Figure 2 shows an overview of the architecture of YOLO v3. YOLO v3 consists of 106 layers that include Convolutional, Upsample, Shortcut, Route, and Detection layers. The Residual Blocks consist of several convolutional layers and skip connections that are used for feature extraction. The main unique feature of the architecture is that object detections are performed at three different scales. In particular, the detection processes are performed



Figure 3: YOLO case study evaluated.

on layers 82, 94, and 106, and after each detection processes, the input image is upscaled by a factor of 2, allowing for better detection of objects of different sizes. The image is divided into a grid, and each cell is responsible for predicting three bounding boxes (i.e. a rectangle delimiting the area of an object). For each of the three bounding boxes, its coordinates, an objectness score (i.e. the probability that the cell includes an object) and class scores (i.e. a score for each class indicating the probability that the object belongs to that class) are calculated. This model defines 80 classes that include animals, food, vehicles, and pedestrians among others. Many bounding boxes may not contain an object, overlap with other bounding boxes that contain the same object, or have a very low probability of containing an object and/or belonging to a specific class. Therefore, a filtering process is applied to obtain the predictions with the highest confidence level and above a set threshold - set to 50% by default - and discard the remaining predictions. As illustrated in Figure 3, this can be done for the internal object list (iT_i) (i.e. an independent analysis of the frame), or at a higher abstraction level, for the resulting output list (T_i) that combines the detections of the last three frames in order to determine if a detection needs to be considered (i.e. the average confidence level is above the 50% threshold).

3.3 Precision and Approximation in Arithmetic

Reduced Precision Arithmetic. Reduced precision arithmetic is a widely-adopted model compression approach [10, 11, 15, 38]. Reducing the number of bits employed for the number representation, by using for example floating-point 16-bit or integer 16-bit, allows the immediate reduction of the model size. The process to map values of a higher precision arithmetic into a lower precision arithmetic is called quantization. For instance, a given floating-point 32-bit number that cannot be represented fully precisely with floating-point 16-bit is *quantized* by mapping it to a different – yet very close – floating-point 16-bit number. Reduced precision arithmetic can bring area, power, and timing savings due to the reduced hardware complexity.

Approximate Arithmetic. Approximate arithmetic [19, 24, 39] consists of implementing simplified arithmetic units, such as adders and multipliers, whose results are less precise compared to fully-precise arithmetic with a similar number of bits. The advantage of approximate designs is that the hardware complexity is significantly reduced, which leads to area, power, and timing savings, at the cost of – typically slightly – reduced precision.

The aforementioned approaches may be combined together to obtain further model compression and energy improvements at the expense of potentially lower accuracy.

4 Approximation Domains and Impact

In this section, we analyse how the YOLO camera-based object detection in particular, and deep learning-based object detection in general, perform object detection to easily tolerate errors, we review the three approximation domains considered in this paper, and how these can affect the object detection outcome.

4.1 Impact of Approximation in YOLO

YOLO v3 implements object detection with 100+ layers, out of which the most computingintensive ones are 53 Convolutional layers devoting most of their time to matrix-matrix multiplications, which is at the heart of most Deep Learning frameworks [40]. Inference occurs with the sequential processing of a number of matrices. Therefore, any inaccuracy or computing error occurring in the first layers is propagated to the following layers.

Because of the nature of the inference process, computationally speaking, contents of one cell of one matrix are used for the computation of multiple cells of another matrix which, in turn, is processed similarly. Therefore, while an error or an inaccuracy in the result of one cell is propagated to an increasing number of cells in the following layers of the network, such propagation occurs with decreasing intensity due to the use of weights, that lead only to partial propagation of errors to each individual cell. Overall, a local error in one cell becomes a series of much smaller errors in multiple cells in the following layers. Moreover, those errors can compensate the effect of each other partially. For instance, a slightly overapproximated value (e.g. 0.4 instead of 0.43) may mitigate inaccuracies (e.g. 0.5 + 0.4 = 0.9 instead of 0.48 + 0.43 = 0.91). Similar compensation effects have already been observed in other domains such as, for instance, critical path delay of circuits where, the larger the number of gates in the path, the lower the variation due to statistical compensation [41].

In semantic terms, such error propagation translates into some minor variations in the confidence levels and object locations for multiple objects detected, rather than creating large deviations affecting just one or very few objects. This is true when using lower precision arithmetic, approximate arithmetic, and even when having sporadic erroneous values. Therefore, one could expect that only those detections whose confidence levels are close to the threshold of acceptance (e.g. 50%) may change enough to lead to a different semantic output. For instance, if confidence levels change by up to $\pm 5\%$, only detections with confidence levels originally in the range [45%, 55%] could be classified differently.

If we further consider that the fusion process leverages such information from different redundant or partially-redundant sensors¹, and then such information regarding object detec-

 $^{^{1}}$ As indicated before, fault-tolerance due to the use of multiple redundant sensors is not analysed quantitatively in the use case in this paper.

tions is also leveraged across periodic detection processes (e.g. every 40 ms), we can expect that objects in the critical confidence level range change across sensors, and over time for the same sensor. Moreover, we can also expect that inaccuracies do not always affect in the same direction a given object (e.g. moving it from slightly above the threshold to below the threshold, or vice versa, across several sensors and repeatedly over time).

In summary, the stochastic nature of the deep learning-based object detection process, thus with estimated confidence values and thresholds, and with sensor and time redundancies, is intrinsically fault-tolerant and hence, amenable to the use of techniques that trade off some accuracy to save power and design complexity.

4.2 Approximation Domains in Object Detection

Our target for approximation are the arithmetic units based on the assumption that they generally account for most of the area and power in systolic arrays. Also, we build on the assumption that their latency determines, or at least has large impact, on the operating frequency of the accelerator.

Reduced precision arithmetic. Reduced precision (e.g. 16-bit operands instead of 32-bit ones) brings several benefits such as the use of less area to store data and for the arithmetic units operating such data, lower latencies for arithmetic units operating those smaller operands, and lower power to perform computations among others. On the other hand, reduced precision can encode fewer values. Hence, whenever the value to be stored cannot be represented exactly with the number of bits available for the particular representation and precision used, a close value that can be represented is used instead. In general, the higher the precision, the higher the number of values that can be represented exactly, and the lower the error introduced due to rounding a non-representable value to a similar representable one.

In the particular case of YOLO, similarly to many deep learning models, the model parameters are represented with floating-point arithmetic. By default, 32-bit floating-point IEEE754 compliant values (also referred to as floats) are used by YOLO, such that 1 bit is used to represent the sign (s), 8 bits for the exponent (e), and 23 bits for the mantissa (m). Instead, standard 16-bit floating point values (also referred to as half-floats) devote 1 bit to s, 5 to e, and 10 to m. Other than denormalized values and other special cases, values represented have the form:

$$half_{float}(s, e, m) = (-1)^{s} \cdot 2^{e-15} \cdot (1.m_2)$$

For instance, the half-float 00110101010100_2 would correspond to the following real number:

$$half_{-}float(0,01101,01010100) = (-1)^{0} \cdot 2^{13-15} \cdot 1.0101010100_{2} = 0.333007813$$

Note that in the case of floats, values represented have the form $float(s, e, m) = (-1)^s \cdot 2^{e-127} \cdot (1.m_2)$.

In general, using b bits for the representations implies that up to 2^{b} values can be represented. Therefore, by decreasing b, fewer values can be represented. Whether this has a large or small impact strictly depends on the values represented. For instance, if their exponents are typically in the range [-14, 15], then the number of e bits of half-floats suffices, and all m bits can be used for meaningful digits, as opposed to denormalized and out-of-range values, thus limiting precision loss to be below 0.05% (i.e. $\frac{1}{2048}$). In the case of YOLO, we note that most values are in this range. Moreover, values out of that range are normally added to values with higher exponents, thus diluting to some extent their impact in precision.

Overall, reduced precision arithmetic is expected to cause a limited impact in YOLO object detection if it is not reduced too aggressively (e.g. using only 8-bit floating-point representation).

Approximate arithmetic. There are a plethora of arithmetic unit designs for approximate arithmetic, such as approximate adders and multipliers. Those can be used to operate the mantissa of the operands, since exponents are much less tolerant to errors due to their exponential impact in the value represented, as shown in [30, 42, 43], and instead, the mantissa may tolerate errors particularly in its lower-order bits, which could cause effects similar to those of reduced precision arithmetic.

The rationale behind approximate arithmetic is that a significant part of the logic is devoted to providing precision for the computation of a reduced number of values. Thus, such logic can be removed or reduced so that the cost reduction is significant while the impact in the result is limited to some inputs, and in those cases, the impact is also low by using approximation for the lowest order bits only.

As discussed for the case of reduced precision, YOLO uses values such that, in most cases, the highest order bits of the mantissa are relevant. Hence, the impact of approximation is very limited. Analogously to the case of reduced precision, inaccuracies due to approximate arithmetic impact the lowest order bits thus mitigating the relative impact of those inaccuracies. And also, as in the case of reduced precision, errors due to approximate arithmetic can compensate each other partially.

Overall, as for reduced precision arithmetic, approximate arithmetic is expected to cause a limited impact in YOLO object detection.

On the other hand, approximate arithmetic provides power reductions and a reduction of the critical path length (i.e., minimum affordable cycle time). For instance, for the approximate adder considered in our evaluation, authors show a 28% power reduction and 50% delay reduction [26] (see further details in Section 6). In the case of the multiplier, power reduction may be as high as 31% [27].

Aggressive low voltage operation. Dynamic energy is the dominant source of energy consumption in combinational logic such as those arithmetic units used for compute-intensive workloads such as object detection. It is well-known that dynamic energy is quadratic on supply voltage (V_{DD}) and hence, one of the most effective ways to save energy in this type of units is decreasing V_{DD} , despite the negative impact that lower V_{DD} operation can have on latency [44].

Circuits are usually powered and timed with some guardbands to ensure that they can complete their function by the end of the cycle. However, given a pair $\langle V_{DD}, T_{cycle} \rangle$, where T_{cycle} stands for the cycle time, such that correct operation is ensured, one can decrease V_{DD} while keeping T_{cycle} so that lower energy is consumed at the expense of some sporadic timing errors that will lead to erroneous outputs. Such errors can occur for some specific inputs triggering specific delay paths in the circuits, affected negatively by process variations, under some specific temperature ranges, or under particular combinations of those effects. Alternatively, a simpler approach consists of using lower power – yet slower – transistors without changing V_{DD} (e.g. synthesizing those parts of the circuit targeting a slightly lower operating frequency while keeping the original frequency during operation), so that power savings can be obtained while keeping V_{DD} unchanged. Slower paths due to slower transistors may lead to some errors under specific operation conditions and input data transitions, with the impact of errors and power savings being completely implementation dependent. The impact of such timing errors is generally arbitrary because typically there are many critical paths across several corner conditions with a delay close to T_{cycle} .

Overall, decreasing V_{DD} aggressively brings significant dynamic energy savings at the expense of causing some sporadic errors with arbitrary impact in the output result of arithmetic units. However, if those errors are rare enough, the overall impact in the object detection process is expected to be tiny in semantic terms.

5 Evaluation Framework

In this section, we present our evaluation framework, discussing the datasets used for the evaluation, the approximation setups evaluated, and the precision metrics employed to assess the accuracy of the model.

5.1 Datasets

To assess the impact of different approximation domains in the CBOD process, we consider the default YOLO v3 [2] version with a pre-trained set of weights that is implemented within the Darknet framework [45], processing a set of images with labelled objects – the COCO dataset [46] – as well as several representative driving video recordings [47–52].

The labelled dataset consists of a set of images that have been tagged with labels accurately. In the case of object detection, a bounding box and an associated class are set for every object contained in each image. This information is referred to as the ground truth and is later used for comparison against the output of the CBOD process. We use the testing subset of the COCO dataset for our case study, as this dataset contains thousands of high-quality images. We can expect relatively highly accurate results with this dataset, since we use a pre-trained version of YOLO that was trained using the training subset of COCO. The COCO dataset contains images of common objects in context. However, due to its generality, the dataset includes many images completely unrelated to driving scenarios. Therefore, we filtered out those images not containing either a person or a vehicle to reduce the number of unrelated images. However, even if the images contain vehicles, this does not imply that the images strictly belong to driving scenarios.

To complement our experimental analysis, we have studied and included a set of unlabelled videos recorded by collection cars explicitly for the development of CBOD. Due to the cost of performing object detection with software-implemented arithmetic units (i.e. 45 minutes per frame and configuration in our case), we only processed a subset of the frames in each video (≈ 30 seconds for each video).

We considered the use of several autonomous driving labelled datasets, such as, CityPersons [53], KITTI [54] and ApolloScape [55], among others (up to 12 datasets in practice). However, to the best of our knowledge, there are no publicly available datasets for which to evaluate the pre-trained YOLO v3 model with sufficient accuracy. We discarded each of those datasets due to at least one of the following reasons: (i) not being publicly available, (ii) not being sufficiently representative for our case study, due to not having the most relevant object classes found in the context of autonomous driving (i.e., at least vehicles and pedestrians), (iii) missing 2D annotations, which are required since the pre-trained YOLO v3 model produces 2D bounding box object detections, and (iv) using similar annotation guidelines to those of the pre-trained YOLO v3 model, which is trained with the COCO dataset, to guarantee meaningful results.

5.2 Approximation setups

We have integrated the SoftFloat library [56] in the Darknet framework to implement floatingpoint numbers at software-level, replacing native floating-point operations by their software emulation, which allows us to use arbitrary representations and arithmetics. We evaluate four different scenarios: the IEEE754 floating-point 32-bit fully precise version (YOLO32full), the IEEE754 floating-point 16-bit fully precise version (YOLO16full), the floating-point 16-bit version using approximate multipliers and adders (YOLO16approx), and YOLO16approx with some faults injected to mimic erroneous outputs due to aggressive low voltage operation (YOLO16appfault).

As said, YOLO16approx corresponds to YOLO16full, but using approximate multipliers and adders for the mantissa, which have been implemented on top of the SoftFloat library to emulate their behavior. In particular, YOLO16 approx implements the addition operation without forwarding the carry from the lowest half of the addition to the highest half of the addition [26]. This optimization significantly decreases the latency of the most critical delay path, as shown in [26], which enables the use of lower power gates within a given V_{DD} and T_{cycle} envelope, while still preserving full precision for the addition of the most significant bits, hence limiting the impact of approximation. Regarding the approximate multiplier, we use the design in [27], which provides a method to build 2x2 approximate multipliers and also provides a method to build larger multipliers using blocks of 2x2 smaller multipliers. Using this method, given 2 operands (A and B), split into their half uppermost bits (AH and BH respectively) and their half lowermost bits (AL and BL respectively), we have implemented an approximate multiplier where the blocks ALxBL, AHxBL and ALxBH are approximate, and the block AHxBH is accurate since this block has a higher impact in the result of the multiplication. While other approximate arithmetic units could be considered, the units considered prove to be effective, as shown later in this section, and hence, they already serve the purpose of illustrating that the use of approximate arithmetic is suitable path for CBOD in AD systems.

Finally, YOLO16appfault implements YOLO16approx, but mimicking the impact of faults due to low voltage operation at software level, inside the SoftFloat library. We model low voltage related faults by randomly flipping one bit of the result of floating-point multiplications or additions with a given probability of fault injection per result, p_{faulty} , which suffices to corrupt the result on an arbitrary manner. How specific p_{faulty} values relate to particular

 V_{DD} values is technology dependent and beyond the scope of this work. Bits are only flipped in the mantissa. Given that the exponent and sign have fewer bits, we assume that the logic to operate them has lower delay (e.g. in terms of fanout-of-4, FO4, gates) and can easily tolerate some delay increase to deliver correct results despite low voltage operation without impacting cycle time. Instead, we assume that mantissa bits are in the critical path for delay, and hence, any increase in the latency of the logic operating mantissa bits due to lower voltage operation could lead to erroneous outputs. Therefore, it is possible decreasing V_{DD} to some extent with no impact in the exponent and sign bits, and experiencing faults only in the mantissa bits. Moreover, as shown later, mantissa bits are relatively fault tolerant, and hence, there are opportunities to trade off some faults in the mantissa for power savings. We evaluate some p_{faulty} values to assess the sensitivity of different tradeoffs between V_{DD} values (or degrees of aggressiveness using lower power transistors for the mantissa operation) and fault rates generated by raising p_{faulty} until it had a visible – yet affordable – impact in results. Note that, processing each frame involves operating in the order of 10^{10} 16-bit floating-point values, so 10¹¹ mantissa bits (the mantissa has exactly 10 bits for 16-bit floating point numbers). Hence, the number of faults injected is around $10^{10} * p_{faulty}$ per frame. Therefore, we injected between 10^4 ($p_{faulty} = 10^{-6}$) and 10^7 ($p_{faulty} = 10^{-3}$) faults per frame.

5.3 Precision metrics

To assess the impact on the accuracy of the different approximation domains, we build on the Intersection over Union (IoU), Average Precision (AP) and Mean Average Precision (mAP) metrics as described in [57]. The IoU is obtained as follows:

$$IoU = \frac{Area_{gt} \cap Area_p}{Area_{qt} \cup Area_p} \tag{1}$$

where $Area_{gt}$ is the area of the ground truth bounding box for the corresponding object and $Area_p$ is the area of its predicted bounding box. The IoU gives a value between 0 (if the bounding boxes do not intersect) and 1 (if the bounding boxes are exactly the same in terms of area and position). Using IoU we can set a threshold t such that only if $IoU \ge t$, the prediction is regarded as correct, otherwise it is regarded as incorrect. In our case, we assume t = 0.5, in line with the PASCAL VOC challenge [58]. With IoU we can classify objects as correct detections (true positives, TP), erroneous detections (false positives, FP), and missed detections (false negatives, FN).

We have used a publicly available framework [57] to measure the mAP and to obtain the number of TP, FP, and FN of the different experimental setups.

The assessment of object detectors is mainly based on the *Precision* (P), an indication of the accuracy for detecting only relevant objects (i.e., objects represented in the ground truth), and *Recall* (R), an indication of the accuracy for detecting all relevant objects. Precision and recall are defined as follows:

$$P = \frac{TP}{All \,detections} = \frac{TP}{TP + FP} \tag{2}$$

$$R = \frac{1T}{All \, groundtruths} = \frac{1T}{TP + FN} \tag{3}$$

A good detector should identify all ground-truth objects and avoid predicting non-existent objects. Therefore, FP and FN should be close to 0, and P and R close to 1.

Building on these two concepts, the Average Precision (AP) is measured by an all-point interpolation method for a given object class, and a set IoU. The AP is defined as follows:

$$AP_{0.5} = \sum_{n} (R_{n+1} - R_n) P_{interp}(R_{n+1})$$
(4)

where $AP_{0.5}$ indicates that the AP is measured with IoU = 0.5 and

$$P_{interp}(R_{n+1}) = \max_{\{\tilde{R}:\tilde{R}\geq R_n+1\}} P(\tilde{R})$$
(5)

If AP values are obtained for different object classes, they can be averaged to obtain a single value called Mean Average Precision (mAP) that weights all object classes equally disregarding the object frequency for each type [57]. The mAP is defined as follows:

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{6}$$

6 Results

In this section, we discuss the labelled dataset evaluation, the video case study evaluation, the confidence range of the detections, and we provide some complementary experiments analysing the area of the misdetections and the misclassification of vehicle types.

6.1 Labelled Dataset Results

We have computed the AP values for the labelled dataset for all types of vehicles (cars, trucks, motorbikes, and buses) and for the person class of object. Results are shown in Table 1 (top data rows), where we show the number of TP, FP and FN, the mAP for the four vehicle classes, and for the four vehicle classes and persons together. In particular, we show results for YOLO32full, YOLO16full, YOLO16approx, and YOLO16appfault varying p_{faulty} between 10^{-6} and 10^{-3} . As shown, reducing precision slightly reduces both the number of TP and FP, and slightly increases FN, in this case, which still leads to near-identical mAP values to those of YOLO32full. If we further introduce approximate arithmetic (YOLO16approx), results remain nearly constant. If we introduce faults, results change marginally up to $p_{faulty} = 10^{-4}$, and higher fault rates lead to noticeable differences due to the increased number of FP.

Overall, these results indicate that CBOD is highly tolerant to multi-domain approximations such as those of reduced precision, approximate arithmetic, and even some moderate fault rates due to aggressive low voltage operation.

6.2 Case Study (Video) Results

The main drawback of the labelled dataset analysis is that the COCO dataset does not contain many images relevant for driving conditions (e.g. images of road scenes taken by



Figure 4: Example of a frame with different (but true) object detections by YOLO32full (left), YOLO16full (center) and the original frame (right).

in-vehicle cameras), so the value of the results of this analysis in the context of autonomous driving are limited.

We include the analysis of videos of real driving conditions that contain relevant data despite not being labelled, as discussed in Section 5.1. The main drawback of using unlabelled videos is that they lack groundtruth labelled objects to assess the accuracy of the model. Therefore, the only means of assessing the accuracy of the model is to use the baseline YOLO32 full as the reference configuration. However, the baseline configuration is not 100%accurate. Therefore, when comparing different configurations, it cannot be stated whether discrepancies among them are due to erroneous detections in one or the other configuration, other than by visual inspection of the corresponding frame. For instance, this is illustrated in the frame in Figure 4, where we see that both, YOLO32full and YOLO16full, have 5 TP and at least 1 FN. However, without visual inspection and taking YOLO32full as a reference, we would assume that YOLO32full has 5 TP and no FN, whereas YOLO16full has 4 TP, 1 FN, and 1 FP. It is important to clarify that all the quantitative results provided in this work are calculated strictly from the output of YOLO, and the metrics have been obtained with a scientific framework to avoid any kind of subjective visual inspection. Therefore, visual inspection is only employed for qualitative discussions. This may lead to pessimistic accuracy results, to some extent. However, we believe that the analysis of unlabelled videos is still very relevant in this context because they allow considering real driving conditions.

Table 1 (center and bottom rows) shows analogous results to those of the labelled dataset but for the unlabelled video. In the table, iT_i corresponds to the object detection per individual frame (internal YOLO results), whereas T_i corresponds to the object detection results averaged across the last three frames, as illustrated in Figure 3.

Note that YOLO32full is omitted in the table since it is used as a reference to compare the other setups. Results are shown for both, the internal object list for a single frame (central set of rows) and for the list averaging confidence levels across 3 consecutive frames (bottom set of rows). The first observation is that mAP values are very high for all configurations and object types (i.e. vehicles and persons) with respect to YOLO32full, except when $p_{faulty} \geq 10^{-4}$. This holds for both iT_i and T_i object lists.

When comparing results for both lists, we observe that T_i has lower TP, FP and FN values. Lower TP, rather than indicating a higher number of undetected objects, it indicates that even YOLO32full detects fewer objects because the intrinsic fault tolerance of YOLO averaging confidence levels across frames allows removing sporadic marginal detections (e.g. objects

Table 1: Results for the labelled data set (top), and both iT_i and T_i for the video use case. Gray background rows indicate the most aggressive configurations in terms of approximation that cause negligible (below 1%) accuracy degradation.

Labelled data set					
Configuration	TP	FP	FN	mAP	mAP vehicle
				vehicle	and person
YOLO32full	5269	1076	4135	59.73	60.47
YOLO16full	5226	1038	4178	59.79	60.39
YOLO16approx	5230	1023	4174	59.49	60.16
$YOLO16appfault(10^{-6})$	5238	1030	4166	59.26	60.00
YOLO16 appfault (10^{-5})	5214	1018	4190	59.29	59.99
YOLO16 appfault (10^{-4})	5227	1034	4177	59.20	59.95
$YOLO16appfault(10^{-3})$	5210	1219	4194	58.53	59.39
Video use case: iT_i – single frame detections					
Configuration	TP	FP	FN	mAP	mAP vehicle
				vehicle	and person
YOLO16full	51297	857	1472	94.78	95.00
YOLO16approx	51142	797	1627	93.69	93.96
YOLO16 appfault (10^{-6})	51123	842	1646	95,50	95,50
YOLO16 appfault (10^{-5})	51061	839	1708	$94,\!07$	94,19
YOLO16 appfault (10^{-4})	50804	1223	1965	90,71	91,21
YOLO16 appfault (10^{-3})	48853	3562	3916	83,14	83,77
Video use case: T_i – confidence averaged across frames					
Configuration	TP	FP	FN	mAP	mAP vehicle
				vehicle	and person
YOLO16full	49720	462	1184	96.25	96.21
YOLO16approx	49552	395	1352	97.29	97.14
$\mathbf{YOLO16appfault}(10^{-6})$	49514	408	1390	97,06	96,92
$\mathbf{YOLO16appfault}(10^{-5})$	49448	446	1456	95,42	95,62
$\mathbf{YOLO16appfault}(10^{-4})$	49307	710	1597	92,27	92,84
$\mathbf{YOLO16appfault}(10^{-3})$	47659	2422	3245	89,35	88,88

whose confidence is slightly above 50% only sporadically). This very same effect also occurs for the other configurations. FP and FN values also decrease due to the filtering of sporadically different confidence values for all configurations. Overall, globally, mAP values for iT_i and T_i do not differ much. However, if we look at how fault-tolerant are both lists, we realize that T_i is much more tolerant to approximation since, as we introduce further approximation domains and as we increase p_{faulty} , mAP degrades slowly. For instance, the two last configurations for iT_i bring mAP drops of around 3% and 7% respectively, whereas for T_i those drops are in the order of 3-4% in both cases.

We also observe that the impact of approximation is analogous across object types, as expected, since those are orthogonal concerns. Instead, a relationship exists between the dimensions of the objects, occupied area, and the impact of approximation, since detections



Figure 5: TP and FP+FN per confidence level for all object types with the T_i object list. The configurations are (a) YOLO16full, (b) YOLO16approx, and (c) YOLO16appfault(10⁻⁴).

of small objects tend to have lower confidence values and the impact of approximation could make those values drop below or grow above the acceptance threshold, hence with a semantic impact.

Finally, considering only whether predictions are above or below the threshold omits information about the confidence of those predictions. For this reason, we have measured the distribution of the confidence levels for both TP and FP+FN. We depict those values for all object types (not only vehicles and persons) in Figure 5, for the YOLO16full, YOLO16approx, and YOLO16appfault(10^{-4}) configurations, using the T_i object list. Note that the y-axis is in logarithmic scale.

All configurations show similar trends. As shown in Figure 5, TP distributes quite homogeneously across confidence levels. However, FP+FN concentrates in the low confidence range, thus indicating that discrepancies negligibly impact objects detected with high confidence. This indicates that the object detection process is naturally fault-tolerant in the context of autonomous driving. Moreover, visual inspection revealed that discrepancies for high-confidence objects relate to (1) objects correctly identified in terms of location and type, but whose bounding box differs noticeably because other objects have been included in the bounding box, either in the approximate configuration or in the reference one (YOLO32full), and (2) overlapping objects where each configuration detects a different subset.

Overall, despite only a subset of redundancy levels are exploited in this work, we already show that CBOD for automotive systems is an intrinsically fault-tolerant process, which eases the adoption of specific accelerators building on multiple approximation domains such



Figure 6: Area of all the objects detected and misdetected for the YOLO16Approx T_i video use case.

as reduced precision, approximate arithmetic, and aggressive voltage scaling for low power operation. In this work we have analysed the different sources of approximation in an incremental manner. However, we believe that the conclusions obtained with our analysis can be extrapolated to the other cases. For instance, reduced precision arithmetic and approximate arithmetic have some impact on the lowermost values of the mantissa, and rarely propagate to uppermost values. Hence, those effects are expected to hold regardless of the baseline used in each case. Similarly, fault injection in the mantissa has low impact when occurring in lowermost bits. If applied on YOLO32full, it is expected that a fraction of faults will impact the 10 uppermost bits of the mantissa (i.e., 10 every 23 faults on average), hence with similar effects to those of fault injection in YOLO16full, whereas the rest of the faults (i.e., 13 every 23 faults on average) will impact the 13 lowermost bits of the mantissa, hence with lower impact than that of injecting faults in the 10th bit of the mantissa of YOLO16full.

6.3 Complementary Experiments and Results

6.3.1 Area of the Misdetections

Another factor to take into account is the area of the misdetected objects of certain types. For instance, a car with a very small bounding box area corresponds to a car that is far away. Hence, it is not problematic if this car is misdetected for a few frames as long as the car is detected when the distance is reduced and the area of the bounding box of the car is much larger.

Figure 6 shows the area range of all vehicle detections (TP) and objects not detected (FN) for the YOLO16approx T_i video use case. The area of the objects is defined as the



Figure 7: Example of a misdetected truck with a large area using YOLO32full (left) and YOLO16full (right).

ratio of the bounding box area with respect to the total area of the image, hence the value of the area ranges from 0 to 1 and, for example, a 0.5 area corresponds to an object occupying half of the image. Note that the y-axis is in logarithmic scale. Note also that the x-axis increments in orders of magnitude every 10 steps.

We observe that the total number of misdetections increases for vehicles with an area smaller than 0.006, which are considered to be small objects. In particular, misdetections for small objects are typically one order of magnitude lower than total detections, whereas misdetections for large objects are two orders of magnitude lower. Hence, we can conclude that, the larger the object, the lower the probability of experiencing a misdetection.

Regarding reported misdetections for large area objects, we have inspected visually a number of them, and in all cases, we have found scenarios where arguably no misdetection occurred in practice. For instance, Figure 7 shows an example of a truck with large area and a confidence level of 0.98 that is considered to be a misdetection. In this case, both configurations detect the truck, but the bounding box *IoU* is below 0.5 because YOLO32full (left picture) includes the building behind the track in the bounding box. Hence, this YOLO16full prediction (right picture) is regarded as both FP and FN. In this particular example, YOLO16full predicts the object area more accurately than YOLO32full, but the opposite can also happen.

The percentage of correct detections (TP), as shown in Figure 8, decreases with object areas lower than 0.003, and decreases more steeply below 0.0006. Note that the smallest observed area has a 100% of TP, but there are only 3 objects of this size, as shown in Figure 6. Therefore, this specific percentage can be simply disregarded since, with a larger number of objects within this area range, one would expect to observe a lower percentage of TP than for larger areas.

Overall, these results show that misdetections come mainly from objects with a small area, and have, therefore, low importance in this context. Misdetections of objects of large area are mainly due to both configurations detecting the same object, but with the IoU between the bounding-boxes being lower than the set threshold, and hence, the detections are regarded as FP and FN.



Figure 8: Percentage of correct detections (True Positives) for the YOLO16Approx T_i video use case.

6.3.2 Misclassification of Vehicle Types

By visual inspection, we noticed that, in some cases, different configurations detect the same object but classify it differently. For example, one configuration may classify a vehicle as a car and another configuration may classify it as a truck, especially in the case of vans, as in this YOLO model there is no van object class. Misclassification of an object may have a significant impact in the mAP metric, since a misclassified object in these circumstances is regarded as both FP and FN, as the model failed to detect (classify in this case) an object, and detected a new object (in this case the same object but of a different object class).

To assess the impact of vehicle type misclassifications, we have merged the different vehicle classes present in the baseline YOLO model (car, motorbike, truck, and bus) into a single class called *Vehicle*. We refer to this configuration as the *Generalization* configuration.

Figure 9 shows the number of TP and FP of all vehicle types for the labelled dataset. In all configurations, the number of TP of the generalization increases a bit and the number of FP decreases by the same amount. Therefore, the accuracy of the object detector in these terms improves. On the other hand, the results for the mAP of *Vehicles* in Figure 10, show that the mAP of *Vehicles* with the generalization decreases for some configurations (YOLO16full and YOLO16Approx), therefore contradicting to some extent the previous observation of increased accuracy. This shows a weak point of the mAP metric. In particular, the calculation of the mAP of *Vehicles* for the baseline is the simple mean of the AP for the different types of vehicles without considering the number of observations of each class, whereas the AP of the generalization of vehicles is computed as a single value naturally weighting all objects. This implies that, if a type of vehicle has a very high/low AP but it accounts for a tiny fraction of the total objects with respect to the number of detections of other types of vehicles, the



Figure 9: Comparison of the total number of T/P and F/P for the baseline and the generalization configurations for the labelled dataset.



Figure 10: Comparison of the Vehicle mAP of the baseline and the generalization configurations for the labelled dataset.



Figure 11: Comparison of the Vehicle mAP of the baseline and the generalization configurations for the T_i video use case.

mAP of *Vehicles* can be significantly affected.

For the video use case, we also obtain that the number of TP of the generalization increases and the number of FP decreases. On the other hand, in this case the mAP of the vehicle generalization provides a higher mAP for all configurations, as shown in Figure 11 for the T_i configuration. The iT_i configuration follows analogous trends but it is omitted since it does not provide further insights. A careful analysis of the videos shows that, if we do not generalize and keep different types of vehicles in different classes, YOLO32full performs a relevant number of misclassifications. Hence, it is often the case that approximate configurations perform a correct object class classification whereas YOLO32full does not. Assuming that approximate configurations are always wrong upon a mismatch with YOLO32full is, therefore, highly pessimistic. However, since videos are not labelled, there is no automatic way to correct this issue. Generalizing vehicles into a single class is a way to mitigate this issue related to the lack of labelled driving videos.

From the analysis of both labelled and unlabelled data, we can conclude that different configurations in some cases detect the same object but with a different classification, thus introducing more variability in the results if the mAP is solely analysed. By considering all vehicle objects as a single class, we avoid issues related to unlabelled data misclassification by the baseline configuration. Moreover, in many cases – yet not necessarily always – classifying a vehicle object into the wrong subclass (e.g., classifying a car as a van) has irrelevant semantic impact.

7 Conclusions

Object detection in AD, is a stochastic process building upon deep learning, thus causing false positives/negatives. To tolerate errors, object detection is performed with sensor redundancy (multiple cameras, LiDARs, and radars), and time redundancy (leveraging detections over time), so that sporadic errors have no visible impact on the output.

This paper analyses the semantic impact of different approximation domains that can be used to save energy and complexity in the power-hungry CBOD process. In particular, in the context of an automotive case study, we show how abundant but minor errors caused by lower precision and approximate arithmetic, as well as sporadic arbitrary errors caused by aggressive low voltage operation have, ultimately, negligible semantic impact in object detection despite causing some impact in the confidence level for detections and even some seldom misdetections. Our results for both, a labelled dataset and real driving videos, show that accuracy (in terms of mAP) is within 1% of the original one even if we consider the three domains of approximate arithmetic, and considering floating-point precision from 32 to 16 bits, using approximate arithmetic, and considering fault rates of up to 10^{-5} in the mantissa of the data. Overall, our study provides strong ground to develop applicationspecific accelerators exploiting multiple approximation domains for cost-constrained domains such as the automotive one.

Acknowledgments

This work is partially funded by the DRAC project, which is co-financed by the European Union Regional Development Fund within the framework of the ERDF Operational Program of Catalonia 2014-2020 with a grant of 50% of total cost eligible. This work has also been partially supported by the Spanish Ministry of Science and Innovation under grant PID2019-107255GB-C21/AEI/10.13039/501100011033.

References

- [1] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, 2017.
- [2] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. ArXiv, abs/1804.02767, 2018.
- [3] NVIDIA DRIVE Autonomous Vehicle Development Platforms. https://developer. nvidia.com/drive. Accessed in Oct-2022.
- [4] Apollo, an open autonomous driving platform. https://apollo.auto/, 2018.
- [5] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. CoRR, abs/1511.08458, 2015.
- [6] H. Wang, X. Jiang, H. Ren, Y. Hu, and S. Bai. Swiftnet: Real-time video object segmentation. In 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 1296–1305, Los Alamitos, CA, USA, jun 2021. IEEE Computer Society.
- [7] Chaowei Fang, Haibin Tian, Dingwen Zhang, Qiang Zhang, Jungong Han, and Junwei Han. Densely nested top-down flows for salient object detection, 2021.
- [8] Guobin Chen, Wongun Choi, Xiang Yu, Tony Han, and Manmohan Chandraker. Learning efficient object detection models with knowledge distillation. In *Proceedings of the* 31st International Conference on Neural Information Processing Systems, NIPS'17, page 742–751, Red Hook, NY, USA, 2017. Curran Associates Inc.
- [9] Tianqi Tang, Shen Li, Yuan Xie, and Norm Jouppi. Mplat: A power, area, timing modeling framework for machine learning accelerators. In *DOSSA Workshop*, New York, NY, USA, 2018.
- [10] Norman P. Jouppi et al. In-datacenter performance analysis of a tensor processing unit. In Proceedings of the 44th Annual International Symposium on Computer Architecture, ISCA '17, page 1–12, New York, NY, USA, 2017. Association for Computing Machinery.
- [11] Chen Wu, Mingyu Wang, Xinyuan Chu, Kun Wang, and Lei He. Low-precision floatingpoint arithmetic for high-performance fpga-based cnn acceleration. ACM Trans. Reconfigurable Technol. Syst., 15(1), nov 2021.
- [12] Xiangyu Zhang, Jianhua Zou, Kaiming He, and Jian Sun. Accelerating very deep convolutional networks for classification and detection, 2015.
- [13] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 779–788, 2016.
- [14] Naveen Mellempudi, Sudarshan Srinivasan, Dipankar Das, and Bharat Kaul. Mixed precision training with 8-bit floating point, 2019.

- [15] Yunji Chen, Tao Luo, Shaoli Liu, Shijin Zhang, Liqiang He, Jia Wang, Ling Li, Tianshi Chen, Zhiwei Xu, Ninghui Sun, and Olivier Temam. Dadiannao: A machine-learning supercomputer. In 2014 47th Annual IEEE/ACM International Symposium on Microarchitecture, pages 609–622, 2014.
- [16] Zachariah Carmichael, Hamed F. Langroudi, Char Khazanov, Jeffrey Lillie, John L. Gustafson, and Dhireesha Kudithipudi. Performance-efficiency trade-off of low-precision numerical formats in deep neural networks. *Proceedings of the Conference for Next Generation Arithmetic 2019*, Mar 2019.
- [17] Darryl D. Lin, Sachin S. Talathi, and V. Sreekanth Annapureddy. Fixed point quantization of deep convolutional networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 2849–2858. JMLR.org, 2016.
- [18] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 1737–1746. JMLR.org, 2015.
- [19] Issam Hammad, Ling Li, Kamal El-Sankary, and W. Martin Snelgrove. Cnn inference using a preprocessing precision controller and approximate multipliers with various precisions. *IEEE Access*, 9:7220–7232, 2021.
- [20] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for largescale image recognition. In 3rd International Conference on Learning Representations (ICLR2015), 2015.
- [21] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1800– 1807, Los Alamitos, CA, USA, jul 2017. IEEE Computer Society.
- [22] Shuihua Wang and Yu-Dong Zhang. Densenet-201-based deep neural network with composite learning factor and precomputation for multiple sclerosis classification. ACM Transactions on Multimedia Computing, Communications, and Applications, 16:1–19, 06 2020.
- [23] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet?, 2019.
- [24] Ali Ibrahim, Mario Osta, Mohamad Alameh, Moustafa Saleh, Hussein Chible, and Maurizio Valle. Approximate computing methods for embedded machine learning. In 2018 25th IEEE International Conference on Electronics, Circuits and Systems (ICECS), pages 845–848, 2018.
- [25] N. Manikandan. Approximation computing techniques to accelerate cnn based image processing applications – a survey in hardware/software perspective. International Journal of Advanced Trends in Computer Science and Engineering, 9(3), 2020.

- [26] Doochul Shin and Sandeep K. Gupta. A re-design technique for datapath modules in error tolerant applications. In 2008 17th Asian Test Symposium, pages 431–437, 2008.
- [27] Parag Kulkarni, Puneet Gupta, and Milos Ercegovac. Trading accuracy for power with an underdesigned multiplier architecture. In 2011 24th International Conference on VLSI Design, pages 346–351, 2011.
- [28] Behzad Salami, Erhan Baturay Onural, Ismail Emir Yuksel, Fahrettin Koc, Oguz Ergin, Adrian Cristal Kestelman, Osman Unsal, Hamid Sarbazi-Azad, and Onur Mutlu. An experimental study of reduced-voltage operation in modern fpgas for neural network acceleration. In 2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pages 138–149, 2020.
- [29] Nandhini Chandramoorthy, Karthik Swaminathan, Martin Cochet, Arun Paidimarri, Schuyler Eldridge, Rajiv V. Joshi, Matthew M. Ziegler, Alper Buyuktosunoglu, and Pradip Bose. Resilient low voltage accelerators for high energy efficiency. In 2019 IEEE International Symposium on High Performance Computer Architecture (HPCA), pages 147–158, 2019.
- [30] Annachiara Ruospo, Alberto Bosio, Alessandro Ianne, and Ernesto Sanchez. Evaluating convolutional neural networks reliability depending on their data representation. In 2020 23rd Euromicro Conference on Digital System Design (DSD), pages 672–679, 2020.
- [31] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [32] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.
- [33] F. Libano, P. Rech, B. Neuman, J. Leavitt, M. Wirthlin, and J. Brunhaver. How reduced data precision and degree of parallelism impact the reliability of convolutional neural networks on fpgas. *IEEE Transactions on Nuclear Science*, 68(5):865–872, 2021.
- [34] Saurabh Jha, Shengkun Cui, Timothy Tsai, Siva Kumar Sastry Hari, Michael B. Sullivan, Zbigniew T. Kalbarczyk, Stephen W. Keckler, and Ravishankar K. Iyer. Exploiting temporal data diversity for detecting safety-critical faults in av compute systems. In 2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pages 88–100, 2022.
- [35] Lucas Klein Draghetti, Fernando Fernandes dos Santos, Luigi Carro, and Paolo Rech. Detecting errors in convolutional neural networks using inter frame spatio-temporal correlation. In 2019 IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS), pages 310–315, 2019.
- [36] Perception Apollo 3.0. https://github.com/ApolloAuto/apollo/blob/master/ docs/specs/perception_apollo_3.0.md, 2018.
- [37] Autoware. An open autonomous driving platform. https://github.com/Autoware-AI/ autoware.ai. Accessed in Oct-2022.

- [38] Yunchao Gong et al. Compressing deep convolutional networks using vector quantization, 2014.
- [39] Honglan Jiang, Francisco Javier Hernandez Santiago, Hai Mo, Leibo Liu, and Jie Han. Approximate arithmetic circuits: A survey, characterization, and recent applications. *Proceedings of the IEEE*, 108(12):2108–2135, 2020.
- [40] Hamid Tabani, Roger Pujol, Jaume Abella, and Francisco J Cazorla. A cross-layer review of deep learning frameworks to ease their optimization and reuse. In 2020 IEEE 23rd International Symposium on Real-Time Distributed Computing (ISORC), pages 144–145. IEEE, 2020.
- [41] Shekhar Borkar, Tanay Karnik, Siva Narendra, Jim Tschanz, Ali Keshavarzi, and Vivek De. Parameter variations and impact on circuits and microarchitecture. In *Proceedings* of the 40th Annual Design Automation Conference, DAC '03, page 338–342, New York, NY, USA, 2003. Association for Computing Machinery.
- [42] Guanpeng Li, Siva Kumar Sastry Hari, Michael Sullivan, Timothy Tsai, Karthik Pattabiraman, Joel Emer, and Stephen W. Keckler. Understanding error propagation in deep learning neural network (dnn) accelerators and applications. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '17, New York, NY, USA, 2017. Association for Computing Machinery.
- [43] Alberto Bosio, Paolo Bernardi, Annachiara Ruospo, and Ernesto Sanchez. A reliability analysis of a deep neural network. In 2019 IEEE Latin American Test Symposium (LATS), pages 1–6, 2019.
- [44] D. Ernst, S. Das, S. Lee, D. Blaauw, T. Austin, T. Mudge, Nam Sung Kim, and K. Flautner. Razor: circuit-level correction of timing errors for low-power operation. *IEEE Micro*, 24(6):10–20, 2004.
- [45] Joseph Redmon and Ali Farhadi. Darknet framework for yolov3. https://github.com/ pjreddie/darknet, 2018.
- [46] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing.
- [47] J. Utah. Rio 4K copacabana beach morning drive, [time: 2:50 3:20], [frames: 5094 5971]. https://www.youtube.com/watch?v=_hWCN1yV9TY, 2020.
- [48] J. Utah. San francisco 4k night drive, [time: 2:20 2:50], [frames: 4214 5091]. https://www.youtube.com/watch?v=jJ08h2cgWjI, 2020.
- [49] J. Utah. San francisco 4k driving downtown, [[time: 2:00 2:30], [frames: 3639 4516]. https://www.youtube.com/watch?v=E7t3QyEfyLA, 2020.

- [50] J. Utah. Los angeles 4k skyscraper metropolis wilshire boulevard, [time: 18:30 -19:00], [frames: 33237 - 34114]. https://www.youtube.com/watch?v=zfblxgasy-0, 2020.
- [51] J. Utah. Miami 4k gold coast scenic drive, [time: 13:25 13:55], [frames: 24142 25019]. https://www.youtube.com/watch?v=xj7abSp07w0, 2020.
- [52] J. Utah. Las vegas 4k sunset drive, [time: 1:10 1:40], [frames: 2097 2974]. https: //www.youtube.com/watch?v=X9U5DafT0d4, 2020.
- [53] Shanshan Zhang, Rodrigo Benenson, and Bernt Schiele. Citypersons: A diverse dataset for pedestrian detection, 2017.
- [54] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.
- [55] Xinyu Huang, Peng Wang, Xinjing Cheng, Dingfu Zhou, Qichuan Geng, and Ruigang Yang. The ApolloScape open dataset for autonomous driving and its application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(10):2702–2719, oct 2020.
- [56] John Hauser. Berkeley SoftFloat, 2018.
- [57] Rafael Padilla, Sergio L. Netto, and Eduardo A. B. da Silva. A survey on performance metrics for object-detection algorithms. In 2020 International Conference on Systems, Signals and Image Processing (IWSSIP), pages 237–242, 2020.
- [58] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2), 2010.