Leveraging Mixed-Precision CNN Inference for Increased Robustness and Energy Efficiency

Tim Hotfilter, Julian Hoefer, Philipp Merz, Fabian Kreß, Fabian Kempf, Tanja Harbaum, Jürgen Becker
Karlsruhe Institute of Technology
{hotfilter, julian.hoefer, fabian.kress, kempf, harbaum, becker}@kit.edu

***Abstract*—Convolutional Neural Networks (CNNs) show tremendous performance in many Computer Vision (CV) tasks like image segmentation crucial to autonomous driving. However, they are computationally demanding and usually not robust to image corruptions like weather influences. In this paper, we introduce our mixed-precision inference method to overcome these two challenges. Therefore, we enable mixed-precision CNN execution on modern embedded system on chips (SoC) that feature a DNN accelerator and a reconfigurable fabric. In case of a weather change, we can quickly adjust the inference precision to maintain model accuracy, while benefitting from fewer off-chip memory accesses compared to full precision. Therefore, we identify optimal quantization schemes for different weather conditions that maximize model accuracy and minimize data offloading. To enable mixed-precision inference, we present our dynamic number conversion architecture for data going back and forth to the off-chip memory, hosted on the reconfigurable tile of the SoC. Using a DeepLabV3+ model with a Resnet-101 backbone for image segmentation, for example, our evaluation yields 60% less off-chip movements under clear weather conditions. Applying rain, fog, and brightness to the input of various models, we can report an up to 26%, 23.8% and 45% reduction in data transactions, respectively, while maintaining the baseline model accuracy. We finally demonstrate that our architecture does not impact the throughput of the CNN inference and consumes very few resources of the reconfigurable fabric.**

***Index Terms*—Mixed-precision CNN inference, Robust CNNs, CNN accelerator, high-performance heterogeneous SoCs**

I. Introduction

The last 15 years were marked by numerous breakthroughs in Deep Neural Networks (DNN). Today, those algorithms are indispensable for applications, from robotics to consumer products, and even for safety-critical tasks like autonomous driving [1]. However, inference of modern algorithms requires many million energy-intensive computations [2]. As a result, applications like image segmentation are today computed on hardware platforms with dedicated DNN accelerators to enable computation in a real-time and more efficient manner.

Nevertheless, two major challenges in computing DNNs both efficiently and safely persist. First, numerous large intermediate results in Convolutional Neural Networks (CNNs) have to be kept as local as possible to the hardware accelerator to ensure fast and efficient inference; secondly, CNNs need to have a high robustness, e.g., towards unforeseen weather condition changes. While human vision is very robust to those influences, CNNs are still struggling with model accuracy. This becomes especially apparent are tasks like autonomous driving. To address the first issue, approaches like quantization and pruning were introduced, however, they tend to worsen the model accuracy. While this can be tolerated or mitigated in many applications, the robustness of a CNN may be troubled by the imprecision those approaches superadd.

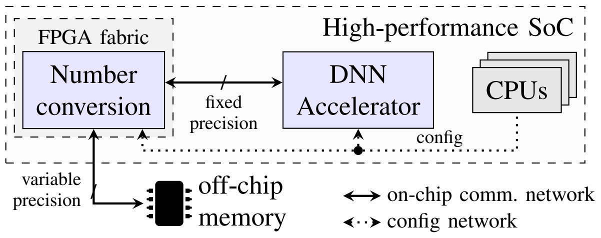


Fig. 1: Our proposed dataflow to enable mixed-precision to increase robustness and energy efficiency through compressed off-chip memory transactions without the need to alter the DNN accelerator itself.

In this paper, we present our mixed-precision approach to bring inference efficiency of CNNs and robustness against wrong predictions on corrupted inputs closer together, while maintaining a high model accuracy. Even though mixed-precision is known to support the energy efficiency [3], it is, however, rarely supported by DNN accelerators. Therefore, we show our dataflow that enables mixed-precision CNN inference in modern high-performance System on Chips (SoC) that use an off-the-shelf DNN accelerators. Instead of changing the DNN accelerator itself, we use a reconfigurable fabric, which is present in most modern SoCs, to convert data moving between the DNN accelerator and the off-chip memory during runtime, as shown in Figure 1. This allows us to dynamically adjust the computational precision, which we project to increase robustness and energy efficiency by compressing off-chip memory transactions. For example, we can work with a higher degree of quantization when facing clear weather conditions to benefit from higher compression. In turn, we can apply less quantization when dealing with uncertain situations, e.g. fog or heavy rain, to regain model accuracy. Our evaluation of different CNNs, with various datasets and weather conditions shows that we save up to 59.8% off-chip accesses in clear conditions compared to bfloat16 inference, and maintain the accuracy under different weather conditions, while still saving up to 26% energy. Hereby, our number converter adds only little latency, requires little resources and can keep up with the throughput requirements of CNNs.

In summary, our contributions are as follows:

- We present our low-latency and resource efficient hardware architecture that enables mixed-precision inference.
- We provide an extensive evaluation of multiple CNNs under various weather conditions and intensities to get valuable insights on how the model accuracy behaves concerning compression of off-chip memory transfers.
- Our exploration tool and hardware architecture is available to the public to foster further research[1].

II. Motivation and Related Work

Robustness towards perturbed or corrupted inputs is one of the main challenges that still exists before CNNs can be deployed in safety-critical systems such as computer vision tasks in autonomous driving vehicles. Besides that, energy efficiency, especially for large segmentation CNNs, plays an important role. Since local on-chip memory cannot keep all intermediate results and weights of large CNN workloads, offloading is required, which contributes a great portion of the total inference energy. Mixed-precision inference is a well-known method to optimize the energy efficiency of CNN inference by compressing those off-chip memory transfers [3]. CNN models are to some extent already resilient to image corruptions. However, this ability drops with further quantization, as computational noise has a higher influence on the result. Here, mixed-precision can help us to reestablish resiliency by increasing the computational precision in case of corrupted inputs. For example, we can use more precise weights and intermediate results for inputs influenced by weather conditions to regain model accuracy, and in the same way more quantized weights and intermediate results to benefit from energy savings in clear weather conditions.

A. Robustness-aware Quantization of CNNs

Lately, various approaches have been proposed to address robustness issues of CNNs using different strategies at several system levels. In particular, many recent studies have been conducted to increase the robustness of CNN models against adversarial attacks. Wijayanto et al. explored the impact of compression on CNN robustness through quantization [4]. Thereby, the presented compression algorithm achieves high compression rates for classification tasks and also reduces the error rate of the model compared to other methods. Also, Khalid et al. use quantization to increase the robustness of CNNs against adversarial attacks [5]. They propose to insert an additional quantization layer at the input of a given CNN. Depending on the chosen method, the quantization levels are constant or learned during training. Apart from straightforward quantization of CNNs, there are other methods to increase the robustness, such as the one proposed by Zhao et al. [6]. The authors perform an outlier-aware quantization during training on the non-maximum suppression stage of oriented object detection networks. Based on the results, the authors can show an improved robustness of the evaluated network compared to common quantization approaches.

[1]https://github.com/itiv-kit/dnn-model-exploration

B. Hardware Architectures for Mixed-Precision Inference

Common CNN hardware accelerators typically only support a fixed bit width and therefore cannot put the energy efficiency gains achieved by variable quantization into effect. A few hardware accelerators have been proposed to support arbitrary precision CNN inference. Envision [7] uses dynamic-voltage-accuracy-frequency scaling (DVAFS) which allows to disable submultipliers based on the targeted bit width. However, this approach suffers from a significant area overhead. BitBlade [8] addresses this issue by using bitwise summation and a tiling scheme to improve the utilization of the multipliers. Besides that, accelerators with reconfigurable precision were presented [9], but they only support a limited number of bit precision and add onto the area budget. Moreover, bit-serial accelerators [3], [10] were introduced to provide mixed-precision inference, thus ensuring high utilization of the hardware blocks used. Although these architectures provide a simple interface for different bit widths, they introduce additional hardware overhead and, in particular, suffer from lower energy efficiency compared to non-bit-serial architectures [11].

III. Tooling and HW-Support for Robust Mixed-Precision CNNs

Our approach enables mixed-precision CNN inference on state-of-the-art heterogeneous platforms like AMD's Versal platform. Typically, such systems feature highly performant accelerators for DNN workloads, along with multiple processors and a reconfigurable fabric, which allows for further system extension. However, those DNN accelerators usually do not support mixed-precision inference, rather they offer the common bfloat16 number format, which is a good tradeoff between precise inference and the possibility for online or on-chip learning. To benefit from memory compression through quantized inference on such systems, we harness the reconfigurable fabric to host a number converter that has access to the on-chip communication network, as shown in Figure 1. Our number converter dynamically quantizes and dequantizes data flowing between the on-chip DNN accelerator and the off-chip memory. In particular, the off-chip memory stores integer values with arbitrary precision between two and 16 bits and the DNN accelerator operates using bfloat16. With our setup, we can apply any precision on a layer-granularity, individually for weights and intermediate results. This allows us to benefit from off-chip memory transfer compression and from a high model accuracy, since quantization sensitive layers are computed more precisely. Compression allows us to save numerous off-chip memory stores and reads. If a weather change corrupts the input, we can benefit from this fine-granular quantization approach as well. In that case, we can use a set of weights with a higher precision and apply less quantization to intermediate results going to the off-chip memory. Since the memory consumption of CNN weights in comparison to its intermediate results is moderate [12], we can easily store multiple sets of weights in the off-chip memory.

The goal of our approach is to dynamically and quickly respond to changing conditions. Compared to traditional ap-

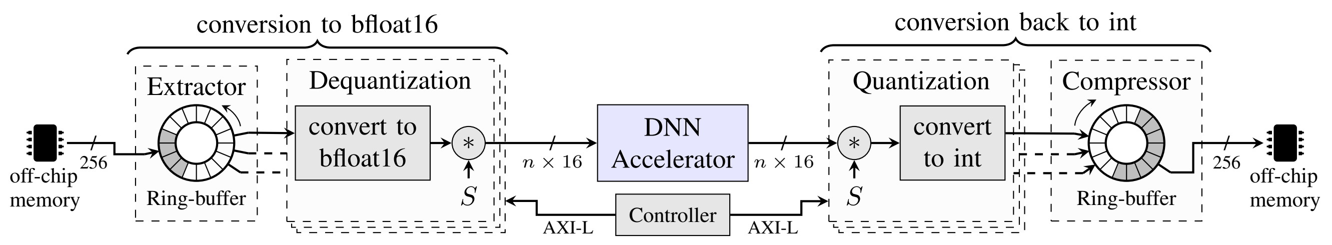


Fig. 2: Overview of our number converter architecture. Weights and intermediate results from the off-chip memory are passed through an extractor and multiple dequantization units to convert integers into bfloat16 values. Results from the DNN accelerator are converted back to integer, which is realized with multiple quantization units and a compressor to pack the values for offloading. All components in dashed boxes are placed on the reconfigurable fabric.

proaches like deraining techniques, which add many more computations to the workload [13], we project that we can leverage a higher inference precision to increase the network robustness. It has to be noted, that in the case of very severe weather conditions, we may have to add a deraining method as well. However, in this case, we may also be able to save energy in the computations of less quantization sensitive layers.

A. Quantization Aware Robustness

Mixed-precision inference was proven beneficial in saving energy by compressing off-chip memory transfers, which are necessary since the intermediate results of modern CNNs exceed local memory capacity. Our claim is that we can, apart from compression, leverage mixed-precision inference to react to corrupted input images and regain the model accuracy dynamically by increasing the precision of computations. An increased precision leads to less quantization noise and hence can support the model accuracy. For example, when an autonomous driving vehicle faces rainy or foggy weather conditions, which obstruct the view, we can gradually ramp up the precision to maintain the model accuracy, while still benefitting from compression of off-chip memory transfers.

As stated before, our approach enables to run a given CNN with different precision levels for weights and intermediate results for each layer. Therefore, we first need to identify quantization sensitive layers that have to be computed with higher precision and those layers that benefit from a higher compression from further quantization. Similarly, we have to do this identification process for inputs that are corrupted by different weather conditions with various intensities, to find a Pareto-optimal precision for each layer that maximizes off-chip memory transfer compression and model accuracy. To accomplish this, we run a design space exploration (DSE) using a genetic algorithm (GA) that tests different sets of bit-width combinations for each layer's intermediate results and weights. To obtain the model accuracy regarding arbitrary precision, we use fake quantization and bfloat16 precision, which reflects the way our approach works in real-world later. The exploration evaluates the model accuracy as well the amount of data transferred between the accelerator and the off-chip memory, to find those Pareto-optimal solutions. Since not necessary all intermediate results have to be offloaded, we use Timeloop [14] to model the underlying hardware architecture during exploration time. Timeloop maps the workload on the specified hardware accelerator and reports, among others, the number of off-chip memory transactions and the thereby consumed energy, considering a given technology. Based on Timeloop's results and the degree of quantization, we can compute the energy and transaction savings a certain combination of bit-widths yields in contrast to the baseline. To get comparable results for various weather conditions, our GA always starts with a fixed starting population and applies the same crossover and mutation algorithms in all DSEs.

B. Hardware Support for Mixed-Precision Inference

The main component to enable mixed-precision CNN inference is a hardware architecture for number conversion that alters the dataflow between the off-chip memory and the DNN accelerator. This architecture has to support a fast conversion and should be resource efficient. An external controller should be able to dynamically adjust conversion formats, either in between layers or when weather conditions change. The controller might trigger a more precise computation when sensors detect a weather change or the model accuracy worsens. Each conversion has to be supported from an integer format ranging from 2 to 16 bit to bfloat16 for data going to the accelerator, as well as the other way around for data being offloaded. From an arithmetic viewpoint, each input and weight has to undergo a conversion according to the following equations, in which x_q is the quantized and x_{float} the dequantized value. S and Z denote a scale and zero-offset factor.

$$x_q = \frac{x_{float}}{S} + Z \qquad\qquad x_{float} = (x_q - Z) * S$$

To make the computation simpler, zero-centered quantization is applied and the zero-offset factor set to 0. Moreover, we can transform the division in the quantization operation into another multiplication. In this case, the conversion in both directions can be realized in hardware using a multiplier, which can be parallelized and pipelined in a straightforward manner. Details of our data flow involving our hardware architecture along with all the necessary components are given in Figure 2. All dashed boxes are placed onto the reconfigurable fabric. For an online conversion of data, we have direct access to the on-chip communication network, allowing us to redirect the data stream between the DNN accelerator and off-chip memory through our number converter architecture. Here, connection to the on-chip network is realized using an AXI-Stream interface with a width of 256 bit. Inside the extractor, incoming data is first buffered in a ring buffer, from which we can read multiple chunks of any bit length to feed to the dequantization units. Each cycle, multiple dequantization units convert integer inputs into bfloat16. The conversion follows a straightforward nine-stage approach, of which one checks the sign bit and the other eight shift the exponent. Since bfloat16 has eight exponent bits, we need exactly these nine stages, which can be realized in a pipelined structure. Then the scale factor is multiplied onto the converted number. Integer precision and scale factors, which were determined in advance by our DSE, can be updated during runtime via an AXI-Lite interface from an external control unit, e.g., a CPU. Once the conversion has finished, we can pack multiple bfloat16 values and send them to the DNN accelerator. The accelerator receives its inputs and weight without noticing they underwent a conversion.

Intermediate results and weights may be stored in a local memory that directly holds values in bfloat16 format. All other intermediate results, which have to be off-loaded, are now passed through the second part of our architecture: the quantization part. First, the individual bfloat16 values are unpacked and the inverse scale factor is applied in multiple quantization units, running in parallel. Then, a conversion into arbitrary integer format is performed. Processed quantized values are then given into the compressor that packs converted values into 256 bit packets for writing to the off-chip memory. We realize this using another ring buffer, that can store arbitrary converted integer values and reads 256 bit at once. This helps us to keep the data always aligned, implement correct AXI handshaking, and to use all bits of the memory bus efficiently.

IV. Results and Discussion

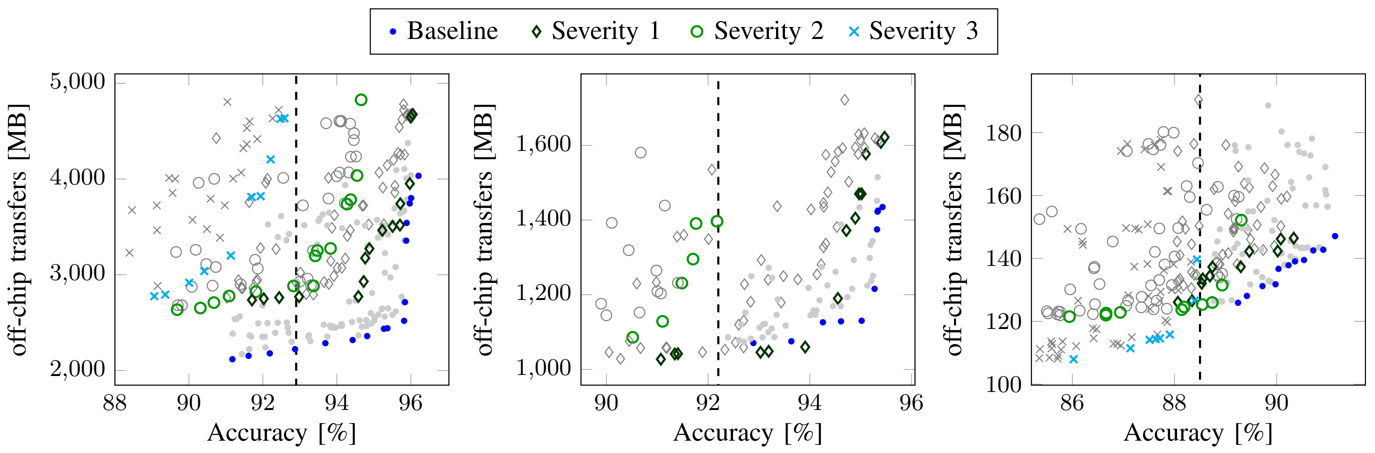
For our evaluation, we look at three large image-segmentation CNNs that infer three distinct datasets using our proposed architecture and report the latency and area consumption as well as the off-chip transfer savings. The datasets are corrupted by different weather conditions for robustness evaluation. The goal of our evaluation is to find precision levels for each network's layer and image corruption type that maximizes the trade-off between fewer off-chip memory accesses through quantization and model accuracy. Thereby, we try to maintain the baseline accuracy of the model. As evaluation datasets, we use a variant of CityScapes [15] with rain and fog, as well as a version of BDD-100k with rain, fog, brightness and snow added through the image corruption tool by Michaelis et al. [16]. Both datasets are well-established in image segmentation, which is crucial for, e.g., autonomous driving. CityScapes is evaluated with DeepLabV3+ [17] for semantic segmentation, achieving a 95.9% and 95.2% pixel accuracy with a ResNet-101 and a MobileNetV2 backbone, respectively. For BDD-100k, we apply YOLOP [18] to segment drivable area, which achieves a 91.5% baseline accuracy. In general, our tool and architecture can evaluate any PyTorch model to identify to what degree mixed-precision can increase the efficiency and robustness.

Finding an optimal precision level for each layer's intermediate result and weights is crucial to benefit the most from mixed-precision inference. As stated before, we find those levels using a DSE and a GA. Our GA evaluates 20 generations with a population size of 15. The objective is set to both maximize the model accuracy and the off-chip memory compression through quantization. An accuracy constraint is added to dismiss solutions with a very low model accuracy. This constraint is obtained based on the baseline model accuracy. From here, we allow for a 3% accuracy degradation. To have a broad evaluation, we perform multiple DSEs: each model and dataset combination is tested with all the above-mentioned weather conditions and intensities. In total, we evaluate over 4500 precision combinations.

As stated before, we use Timeloop to get an estimation of the actual memory transactions and how much traffic we can save using our compression using mixed-precision inference. As underlying model in Timeloop we use an Eyeriss-v2 architecture. We adjusted the original parameters of Eyeriss-v2 to match modern state-of-the-art accelerators: the local memory is increased to 3 MB, the bus connection to the main memory to 256 bit and the amount of processing elements to 48x48 (2,304 in total).

A. Evaluation of Mixed-Precision to Enhance Robustness

The results of our DSE are depicted in Figure 3. Corresponding Pareto-optimal solutions are marked in color, while dominated solutions are shown in gray. Because we evaluate various weather conditions with different intensities for different networks, we can only show a selection, which, however, presents our most significant findings. Figure 3a shows how the model accuracy of a DeepLabV3+ model with a ResNet-101 backbone is impacted under rainy conditions. The figure shows the model accuracy concerning the amount of off-chip memory transactions, with three severity levels of rain and the baseline in blue. This network running with entirely bfloat16 requires in total 6,279.88 MB of data being offloaded. From the figure, we can now see that we can reduce the off-chip transfers to 2,519.23 MB (59.8% less), while maintaining the baseline accuracy. Influenced by light rain (severity level 1, dark green diamond marks), we can leverage a higher computational precision and keep up with the accuracy. Here we can save 1,632.2 MB (26%) of data moved to and fetched from the off-chip memory. If the rain gets heavier (severity level 2, green circle marks), our approach can reach an accuracy of 94.5% with more precision. However, at this severity level, the model accuracy starts to decline, as even the full-precision algorithm is unable to mitigate heavy rainy conditions. As reference, we test very heavy rain conditions (severity level 3, cyan cross marks) using 32 bit floating-point precision, which only yields a 92.6% pixel accuracy that our approach can also achieve, but with 26.1% less off-chip memory transfers.



(a) DeepLabV3+ with ResNet101 backbone (Baseline accuracy 95.9%, off-chip transfers without compression: 6,279.8 MB), which is subject to corruption by rain.

(b) DeepLabV3+ with a MobileNetV2 backbone (Baseline accuracy 95.2%, off-chip transfers without compression: 2,068.9 MB). Data is subject to corruption by fog.

(c) YOLOP (Baseline accuracy 90.5%, off-chip transfers without compression: 266.5 MB) with BDD100k. Data is corrupted with brightness.

Fig. 3: Model accuracy in relation to the off-chip memory movements required for a single inference of three different CNNs under various weather conditions. A dashed line shows a lower accuracy constraint.

Similarly, Figure 3b gives the accuracy impact regarding the precision of a DeepLabV3+ with a MobileNetV2 backbone under foggy conditions. This network loads and retrieves 2,068.9 MB of data per inference, of which we can save 41.2% (1,215.75 MB) under clear conditions. At severity level 1, we can reduce the number of off-chip memory transfers by 23.8% (492.4 MB) and maintain baseline model accuracy through mixed-precision inference. With very severe fog, the model itself cannot maintain the baseline accuracy. Considering the full precision accuracy of 91.9%, we still reduce off-chip data movement by 32.4% (672.2 MB).

Figure 3c shows YOLOP, a comparably lightweight model, for drivable area detection corrupted by heightened brightness. Running at full precision, the model creates 266.47 MB memory traffic. With mixed-precision inference, we can reduce the transfers to 142.77 MB (46% less). At severity level 1, we can achieve a model accuracy of 90.3% with 146.45 MB of off-chip movements, 45% less compared to entire bfloat16 precision. From level 2 the model accuracy starts to degrade, here we can reach 89.3% model accuracy with 152.19 MB of data transfers, 42.9% less.

B. Hardware Considerations for Mixed-Precision Inference

Now that ideal bit-widths are found, we want to benefit from the mixed-precision inference in hardware. Therefore, this subsection will look at the performance of our hardware architecture, which enables mixed-precision CNN inference on modern heterogeneous SoCs. To evaluate our architecture, we map it onto an AMD VCK190 evaluation kit, which is populated with an AMD Versal XCVC1902 device. Table I shows a breakdown of the required resources to host our architecture on the reconfigurable fabric. Our evaluation design has 16 quantization and 16 dequantization units, together with one extractor and compressor. The design runs at 300 MHz clock speed. From the table, we can see that our number converter consumes less than 1% of the total available resources. The entire design, which comprises all necessary interfaces and peripherals for data movement, utilizes about 1% of the total resources. This allows us to scale the number of quantization and dequantization units up for more demanding designs and applications, while also keeping spare resources for other designs that need the reconfigurable fabric as well.

TABLE I: Resource utilization of our hardware prototype setup. The entire design features 16 quantizer and dequantizer units, with a 256 bit connection to the off-chip memory.

Component	LUTs	FFs	DSPs	BRAM
Entire Design	37,366	25,995	32	9
AXI-DMA & Interfaces	8,007	12,754	0	9
Number converter	29,359	13,241	32	0
Extractor	4,835	2,352	0	0
1x Quantizer	686	257	1	0
1x Dequantizer	556	212	1	0
Compressor	2,138	3,120	0	0

From a performance perspective, it is important for our design to match the throughput of the DNN accelerator and to be able to compute modern CNNs within a short time window. As stated before, we designed it in a highly parallelized and pipelined fashion. Based on a design with the aforementioned 16 units, we report a latency of two and ten cycles for the extractor and quantization unit, respectively. Similarly, a dequantization unit and the compressor have a latency of ten and two cycles. We tested the latency for different integer precision, showing a constant latency over the supported two to 16 bits. In total, we report a latency of 12 clock cycles per conversion direction, which equals an extra of 2 μs at 300 MHz considering a CNN workload with 50 layers. Since our number converter has an adjustable number of quantization and dequantization units and our design utilizes only a few resources, we can match the throughput requirements of the DNN accelerator. Considering, for example, YOLOP, which has 101 layers, Timeloop reports 139 million memory accesses. Adding the latency for adjusting the configuration of our architecture between layers for weights and intermediate results, we can reach a through-put of 4.8 billion conversions per second with 16 converters, yielding a total performance of 34.35 inferences of YOLOP per second. For the much larger DeepLabV3+ network with a ResNet-101 backbone, which requires 3.29 billion off-chip memory accesses, we can report a performance of 1.45 inferences per second. To achieve a similar inference rate for this complex workload, we can invest more resources for more conversion units running in parallel. Regarding energy consumption, Vivado reports 75 mW for our test setup with 16 converters running at 300 MHz. This equals 2.18 mJ of energy per YOLOP inference and 51.72 mJ for DeepLabV3+ with a ResNet-101 backbone, respectively.

C. Discussion

During our evaluation, we showed good results on various CNN workloads. However, we also find that an in-depth evaluation of the workload in advance is important for multiple reasons. First, it is crucial to identify sweet-spots between higher model accuracy and fewer off-chip memory accesses, since some few layers contribute a large share of the off-chip memory transactions. Therefore, a designer has to bear in mind and carefully consider acceptable accuracy degradation for a given application. Second, to figure out the throughput and latency requirements early on to tune the converter architecture to match performance requirements. Finally, to save inference energy from compression, it is crucial to understand the CNN workload, as selection of the right quantization degree plays an important role.

V. Conclusion

In this paper, we presented our architecture to add support for mixed-precision inference to modern state-of-the-art heterogeneous SoCs. We have shown that mixed-precision inference can not only increase the energy efficiency of CNN inference, but can also support the model's robustness towards various weather conditions, while still saving energy through targeted quantization of less quantization sensitive layers. Therefore, we proposed a hardware architecture with low resource requirements and latency, which takes care of dynamic number conversion of data going from the off-chip memory to the accelerator and back. We performed a wide range of experiments and design space evaluation and gained insightful results on how the model accuracy behaves under various weather conditions and degrees of quantization. We demonstrated, for example, up to 45% less off-chip memory transactions under rainy conditions while maintaining the same model accuracy as under clear weather conditions. Our source code and RTL is available as open-source to foster further research and extension. In the future, we plan to evaluate models that were already trained towards robustness. Furthermore, we want to look into other quantization schemes that leverage a lower quantization noise.

Acknowledgement

This project has received funding from the European High Performance Computing Joint Undertaking (JU) under Framework Partnership Agreement No 800928 and Specific Grant Agreement No 101036168 (EPI SGA2).

References

[1] J. Hoefer *et al.*, "SiFI-AI: A Fast and Flexible RTL Fault Simulation Framework Tailored for AI Models and Accelerators," in *Proceedings of the Great Lakes Symposium on VLSI 2023*, 5 2023, pp. 287–292.

[2] M. Horowitz, "1.1 Computing's energy problem (and what we can do about it)," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, Feb. 2014, iSSN: 2376-8606.

[3] H. Kim, Q. Chen, T. Yoo, T. T.-H. Kim, and B. Kim, "A 1-16b precision reconfigurable digital in-memory computing macro featuring column-mac architecture and bit-serial computation," in *ESSCIRC 2019 - IEEE 45th European Solid State Circuits Conference (ESSCIRC)*, 2019.

[4] A. W. Wijayanto, J. J. Choong, K. Madhawa, and T. Murata, "Towards robust compressed convolutional neural networks," in *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)*, 2019, pp. 1–8.

[5] F. Khalid *et al.*, "QuSecNets: Quantization-based defense mechanism for securing deep neural network against adversarial attacks," in *2019 IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2019, pp. 182–187.

[6] M. Zhao, K. Ning, S. Yu, L. Liu, and N. Wu, "Quantizing oriented object detection network via outlier-aware quantization and iou approximation," *IEEE Signal Processing Letters*, vol. 27, pp. 1914–1918, 2020.

[7] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "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)*, 2017, pp. 246–247.

[8] S. Ryu *et al.*, "Bitblade: Energy-efficient variable bit-precision hardware accelerator for quantized neural networks," *IEEE Journal of Solid-State Circuits*, vol. 57, no. 6, pp. 1924–1935, 2022.

[9] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H.-J. Yoo, "UNPU: An energy-efficient deep neural network accelerator with fully variable weight bit precision," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 1, pp. 173–185, 2019.

[10] P. Judd, J. Albericio, T. Hetherington, T. M. Aamodt, and A. Moshovos, "Stripes: Bit-serial deep neural network computing," in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2016, pp. 1–12.

[11] A. Parmar, K. Prasad, N. Rao, and J. Mekie, "An Automated Approach to Compare Bit Serial and Bit Parallel In-Memory Computing for DNNs," in *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2022, pp. 2948–2952, iSSN: 2158-1525.

[12] K. Guo, S. Zeng, J. Yu, Y. Wang, and H. Yang, "A Survey of FPGA-Based Neural Network Accelerator," *arXiv:1712.08934 [cs]*, Dec. 2018.

[13] D. Hendrycks and T. G. Dietterich, "Benchmarking Neural Network Robustness to Common Corruptions and Surface Variations," Apr. 2019, arXiv:1807.01697 [cs, stat].

[14] A. Parashar *et al.*, "Timeloop: A Systematic Approach to DNN Accelerator Evaluation," in *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, Mar. 2019, pp. 304–315.

[15] X. Hu, C.-W. Fu, L. Zhu, and P.-A. Heng, "Depth-Attentional Features for Single-Image Rain Removal," 2019, pp. 8022–8031.

[16] C. Michaelis *et al.*, "Benchmarking robustness in object detection: Autonomous driving when winter is coming," *arXiv preprint arXiv:1907.07484*, 2019.

[17] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation," Aug. 2018, arXiv:1802.02611 [cs].

[18] D. Wu *et al.*, "YOLOP: You Only Look Once for Panoptic Driving Perception," *Machine Intelligence Research*, vol. 19, no. 6, pp. 550–562, Dec. 2022, arXiv:2108.11250 [cs].