Reliability-Aware Performance Optimization of DNN HW Accelerators through Heterogeneous Quantization

Samira Nazari[1*], Mahdi Taheri[2,3*], Ali Azarpeyvand[1,2], Mohsen Afsharchi[1], Christian Herglotz[3], and Maksim Jenihhin[2]

[1]University of Zanjan, Zanjan, Iran
[2]Tallinn University of Technology, Tallinn, Estonia
[3]Brandenburgische Technische Universität Cottbus, Cottbus, Germany

***Abstract*—The paper introduces a framework for implementing heterogeneous quantization tailored to individual layers of DNN accelerators. The proposed method, evaluated across DNN benchmarks AlexNet, VGG-11 and ResNet-18, enhances both accuracy drop and performance. On average, the accuracy drop at a BER of 10^{-4} was reduced by nearly 4 times compared to homogeneous quantization, demonstrating a considerable improvement in reliability under fault-prone conditions. Furthermore, the memory overhead introduced by the method for protected heterogeneous networks is less than 0.1%, highlighting that these reliability gains come with negligible resource costs.**

***Index Terms*—deep neural networks, parallel processing, memory overhead, reliability, DNN accelerator**

I. Introduction

DNNs' sizes are growing and getting more complex [1]. This growth is making them computationally intensive and requires substantial energy even for a single inference [2]. A promising strategy to mitigate power and latency issues in DNNs is the use of quantization, which reduces computational complexity by representing data with lower precision. Techniques such as precision reduction, pruning, and functional approximation have demonstrated their efficacy in improving power and area efficiency while maintaining acceptable accuracy levels [3]. Despite these advancements, most existing approaches rely on homogeneous quantization, where a uniform level of precision is applied across all components of the network [4], [5]. However, DNN components—such as channels, filters, layers, and neurons—exhibit varying levels of resiliency to quantization [6]–[8]. This variability opens the door to heterogeneous quantization, which adapts the precision level based on the resiliency of each component. By tailoring the degree of quantization to the specific needs of individual components, heterogeneous quantization offers a more effective pathway to balancing power efficiency, memory utilization, and performance in DNN hardware accelerators.

*Authors contributed equally

This paper presents a framework for applying heterogeneous quantization at the layer level to DNN accelerators. The framework is based on identifying channels' resilience and applies a specific degree of quantization per layer. The contributions of this paper are as follows:

- **Methodology for vulnerability assessment of quantized DNNs:** Developed a hybrid methodology combining GA with fault injection for accurate and fast Layer Vulnerability analysis in quantized DNNs.
- **Technique for heterogeneous vulnerability-aware quantization:** Achieved higher throughput, accuracy, and reliability with a lower memory utilization Based on identifying layers´ vulnerability using GA, and applying a specific degree of quantization per layer.
- **Design space exploration framework**: Developed an open-source framework to facilitate the methodology. https://github.com/nilay1400/Heterogenous-Quantization

The rest of the paper is structured as follows. Section II describes the proposed methodology. Section III presents the experimental setup and discusses the obtained results. Finally, Section IV concludes the paper.

II. Proposed Methodology

The methodology for heterogeneous vulnerability-aware quantization is illustrated in Figure 1 and outlined in Algorithm 1. Vulnerability-aware quantization optimizes DNNs by reducing bit-width across layers while maintaining robustness against potential faults. The goal is to balance quantization with model vulnerability, particularly in layers critical for maintaining accuracy under challenging conditions. In the proposed quantization strategy, layers in the DNN are selectively assigned different bit-widths based on their sensitivity to perturbations. Figure 2 presents a sample heterogeneous quantization of a DNN. Conducting a vulnerability analysis on each quantized layer helps identify the layers that have a significant impact on model accuracy under fault conditions. Layers demonstrating high Vulnerability retain higher bit-width values to ensure network accuracy, whereas less vulnerable layers adopt lower bit-width values to improve efficiency.

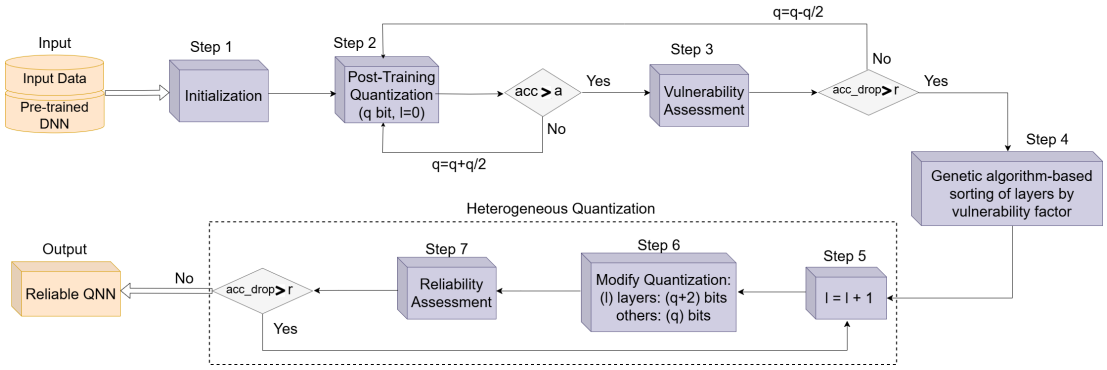


Fig. 1: Proposed methodology flow

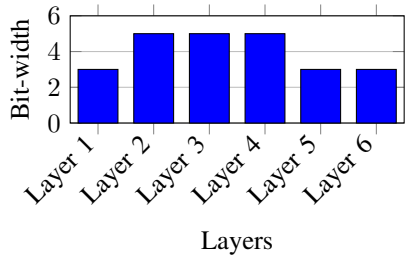


Fig. 2: Heterogeneous quantization bit-widths across layers of a sample network.

This heterogeneous allocation allows for an overall reduction in memory usage without compromising model accuracy under fault-prone conditions.

The input to the process includes a DNN with floating-point parameters and a designated input dataset for evaluation. The user specifies initial parameters, including the golden model accuracy threshold (a), as well as an acceptable accuracy drop threshold in the presence of faults (r) tailored to the specific application requirements. The starting bit-width for quantization (q) is set to 8 bits as it is the optimum quantization without accuracy degradation [9]. This step establishes the initialization parameters for the quantization process (Algorithm 1 lines 3-7).

After the initialization step, the DNN is quantized to the initial bit-width q, and the quantized model's accuracy is evaluated. If the accuracy falls below the specified threshold, the bit-width q is incrementally adjusted upward according to binary search ($q = q + \frac{q}{2}$) to ensure that the accuracy meets the application's minimum requirement.

Once the accuracy evaluation confirms that the model's accuracy is within an acceptable range, a fault injection process is performed to simulate specific Bit Error Rates (BERs). The goal here is to identify a bit-width q at which the model achieves adequate accuracy but experiences an accuracy drop greater than the defined threshold under fault injection. This condition indicates that the model, while accurate, remains vulnerable to errors due to quantization (Algorithm 1 lines 10-22, Figure 1 steps 2 and 3). This bit-width q serves as the starting point for heterogeneous quantization, where specific layers are selectively assigned different bit-widths to address and mitigate the observed vulnerability.

The next step in the methodology involves sorting the layers of the quantized q-bit DNN according to their vulnerability factors (Algorithm 1 lines 23-28, Figure 1 step 4). To accomplish this, an initial random set of weights is selected from each layer in the DNN. Faults are injected into these weights, and the resulting accuracy drop is measured, providing an initial estimate of each layer's susceptibility to quantization-induced errors.

Subsequent generations of weights are then selected through a genetic algorithm (GA) approach [10], with the accuracy drop serving as the fitness function. The GA iteratively selects weights with the goal of maximizing the fitness function, meaning that weights are chosen to produce the highest accuracy drop upon injection.

This GA operates as a search-based optimization method designed to identify vulnerable layers in DNNs by injecting faults and observing the resulting impact on model performance. The GA begins with an initial population of candidate solutions, each representing a potential configuration of injected faults within the network. Each candidate is evaluated based on its "fitness," defined as the impact of its fault configuration on the accuracy or output stability of the DNN.

Through iterative steps, the GA applies selection, crossover, and mutation to generate new candidate solutions, focusing on those that demonstrate a high potential for exposing vulnerability. Over successive generations, the algorithm effectively "zooms in" on regions where fault injection yields significant impacts, enabling a targeted and efficient exploration of the network's vulnerability profile. This approach adapts dynamically, converging towards critical points in the model's architecture, making it both efficient in terms of inferences and effective at pinpointing weak layers in large, parameter-dense networks.

The layer vulnerability factor (LVF) is calculated as shown in Equation 1 and reflects the extent to which each layer contributes to the model's overall sensitivity to quantization errors. The genetic algorithm continues generating new sets of weights and recalculating LVFs until successive generations show minimal change in LVF values so that sorting of layers remain consistent. This convergence indicates stability in the

Algorithm 1 Heterogeneous Quantization Algorithm

1: **Input:** Trained FP32 DNN model, Input data, Accuracy threshold a, Accuracy drop threshold r, Quantization bit-width q, No. protected layers $l = 0$
2: **Output:** Reliable QNN model
3: Load the trained DNN
4: SET accuracy threshold TO a
5: SET reliability threshold TO r
6: SET quantization bit-width TO $q = 8$
7: SET no. protected layers TO $l = 0$
8: quantized_weights← QuantizeWeights(model.weights, bit_width)
9: Evaluate vulnerability
10: process = True
11: **while** process **do**
12: **if** $acc\_drop < r$ AND $acc > a$ **then**
13: $q = q - \frac{q}{2}$
14: **else**
15: $q = q + \frac{q}{2}$
16: **end if**
17: quantized_weights← QuantizeWeights(model.weights, bit_width)
18: Evluate vulnerability, update acc and acc_drop
19: **if** $q = 8$ or $q = 0$ **then**
20: process = False
21: **end if**
22: **end while**
23: **while** $LayerSorting_{old} != LayerSorting_{new}$ **do**
24: Choose a random set of weights
25: Inject faults in selected weights
26: Evaluate Fitness Function
27: Calculate LVF
28: **end while**
29: Report sorted layers
30: **while** $acc\_drop > r$ **do**
31: $l = l + 1$
32: ModifyQuantization(l): l layers $q + 2$ bits, other layers q bits
33: Evaluate reliability
34: **end while**
35: Report the final QNN

ranking of layers by their vulnerability.

$$LVF(l) = \frac{N_{vulnerable}(l)}{N_{total}(l)}, \tag{1}$$

where $N_{vulnerable}(l)$ is the number of neurons in layer l identified as the most vulnerable by the GA and $N_{total}(l)$ is the total number of neurons in layer l.

Upon completion of this step, the layers of the quantized q-bit DNN are effectively ranked according to their LVFs, establishing a hierarchy of vulnerability. This ranking is critical for the subsequent heterogeneous quantization step, where higher LVF layers are assigned higher bit-widths to enhance model resilience while maintaining efficiency.

At this stage, all layers are quantized to q bits. To enhance the DNN's resilience, the bit-width of the most vulnerable layers is selectively increased. Let l denote the number of layers assigned a higher bit-width for increased fault tolerance. l is incremented by 1 ($l = 1$), meaning only the most vulnerable layer undergoes modification.

For each selected layer, two bits are added to its quantization, with these additional bits replicating the most significant bit (MSB) of the weight. This configuration protects the MSB, the most critical bit for weight accuracy, by implementing triple modular redundancy (TMR), thereby enhancing the layer's resilience to faults. When a layer is quantized to a higher bit-width, it undergoes a voting procedure during inference to determine the correct value for each weight, ensuring that the majority vote across the replicated bits sets the final value.

The modified quantization (Step 6) is then evaluated to ensure the accuracy drop remains within acceptable limits (Algorithm 1 lines 31-33). If the accuracy drop now meets the predefined requirements, the DNN with the current heterogeneous quantization configuration is finalized and reported as the output (Step 7). Otherwise, l is incremented, and the process repeats until the accuracy drop threshold is satisfied.

By applying TMR to the MSB and implementing a voting mechanism, this approach targets critical vulnerabilities in the most sensitive layers, thereby achieving a balanced trade-off between model robustness and quantization efficiency.

To evaluate the impact of the proposed methodology on performance, two key metrics are used [5]. Reliability-Aware Performance (RAP) is a metric that evaluates the overall performance of a network by combining accuracy drop, memory overhead, and execution time overhead. It quantifies the trade-off between reliability and resource usage, where smaller RAP values indicate more reliable networks that incur less memory and performance overhead. It is calculated as:

$$RAP = acc\_drop \times mem\_ovh \times perf\_ovh \tag{2}$$

where acc_drop represents the accuracy drop, $perf\_ovh$ refers to the execution time overhead, and mem_ovh denotes the memory utilization overhead.

P_{drop} calculates the probability of experiencing an accuracy drop over the device's lifetime due to faults. It takes into account various factors such as the number of parameters, bit width, device lifetime, fault probability, and the observed accuracy drop under different fault conditions. A lower P_{drop} value indicates a more resilient network with a reduced likelihood of significant accuracy degradation during operation. P_{drop} is defined as:

$$P_{drop} = N^2 \times W^2 \times T/t \times P_{single} \times BER \times acc\_drop \tag{3}$$

where N is the number of parameters, W is their bit width, T is device life time, t is test time interval, P_{single} is the probability of one bit flip during t and acc_drop is the reported accuracy drop in the BER. This metric is based on the probability of a one-bit flip stated in [11].

III. Experimental Results

A. Experimental Setup

To simulate cumulative memory faults, random faults are injected into the network weights during evaluation using various BERs. This fault injection enables a detailed analysis of how accuracy degrades under different fault levels, highlighting the resilience gained through heterogeneous quantization.

Table I shows different types of quantization applied to each DNN benchmark. To maintain consistency across all models, approximately 25% of layers are selected for protection in heterogeneous quantization. hom (n,-) represents homogeneous quantization with n bits; while hom (-,n+2) is homogeneous quantization with $n+2$ bits, in which for all weights, two most significant bits are used for protection and in het (n,n+2) heterogeneous quantization is applied, combining both n and $n+2$ bits. The base quantization bit-width is denoted by n, with results reported for three sample n values: 3, 4, and 5. Table II provides an overview of each model's base accuracy for homogeneous quantization without protection, serving as a reference point for interpreting the results of the heterogeneous quantization. All experiments are conducted on an NVIDIA GeForce GTX 1050 Ti GPU, using the PyTorch framework for training, testing, and fault injection.

B. Quantization and reliability evaluation

To demonstrate the impact of heterogeneous quantization on reliability, Figures 3, 4 and 5 present the fault injection results for both homogeneous and heterogeneous quantizations of each model. Across all models and quantization types, heterogeneous quantization consistently exhibits less accuracy degradation under fault injection at varying BERs, indicating enhanced resilience.

In Figure 3, AlexNet with heterogeneous quantization (5,7) shows the highest resilience. Similarly, in Figure 4, VGG-11 achieves the greatest reliability in the heterogeneous (5,7) configuration; however, it also demonstrates significant improvement in reliability when converted from homogeneous (4,-) to heterogeneous (4,6). In Figure 5, ResNet exhibits similar behavior across all homogeneous and heterogeneous quantizations, although at larger bit-widths, such as $n = 5$, the curve smooths out, indicating that accuracy drops tend to occur at higher BERs.

The proposed heterogeneous quantization results in less accuracy drop and more reliable models. However, it comes with increased memory utilization and longer execution time during inference compared to its homogeneous counterpart. To evaluate the performance of the proposed method, it is compared with state-of-the-art (SOTA) quantized and protected DNNs in [5]. In [5], homogeneous quantization is applied to models, with protection added to all layers.

Tables III, IV and V compare the results for homogeneous quantization with n-bit width and no protection (hom (n,-)), homogeneous quantization with increased protection ($n+2$-bit width, hom (-,n+2) [], and heterogeneous quantization with n-bit width and $n+2$-bit protection (het (n, n+2)) for AlexNet, VGG-11 and ResNet-18 respectively. The accuracy drop is reported for a sample BER, along with memory utilization, execution time, RAP, and P_{drop} values.

TABLE I: Different types of quantization for DNN models.

Model	Name	Type of Quantization
AlexNet	hom $(n,-)$	all layers: n bits
	hom $(-,n+2)$	all layers: $n+2$ bits 2 bits for protection
	het $(n,n+2)$	6 layers: n bits 2 layers: $n+2$ bits
VGG-11	hom $(n,-)$	all layers: n bits
	hom $(-,n+2)$	all layers: $n+2$ bits 2 bits for protection
	het $(n,n+2)$	8 layers: n bits 3 layers: $n+2$ bits
ResNet-18	hom $(n,-)$	all layers: n bits
	hom $(-,n+2)$	all layers: $n+2$ bits 2 bits for protection
	het $(n,n+2)$	13 layers: n bits 5 layers: $n+2$ bits

TABLE II: Base accuracies of the benchmark NNs (%)

Type	VGG-11	ResNet-18	AlexNet
hom (5,-)	92.19	92.81	95.46
hom (4,-)	92.18	92.66	94.91
hom (3,-)	89.83	90.84	93.26

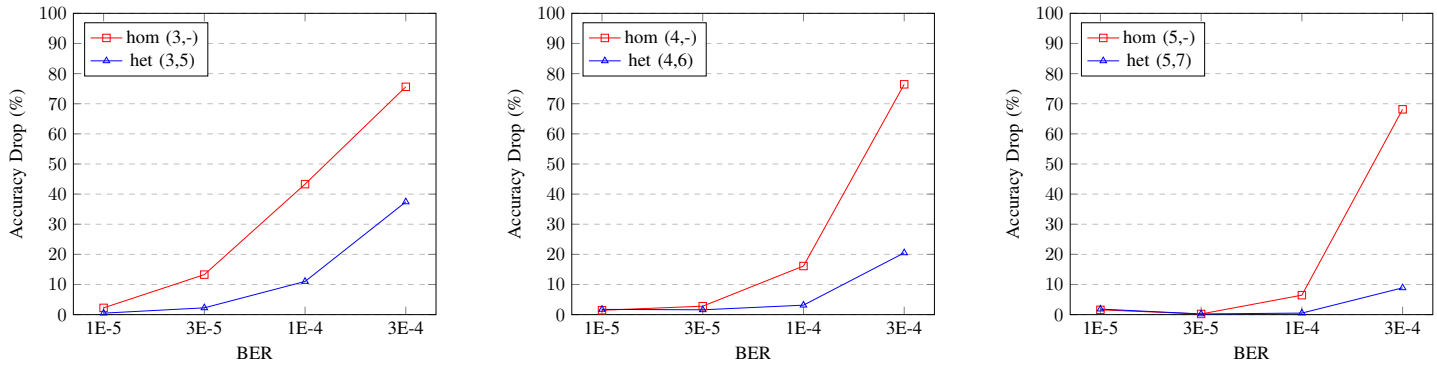
Figure 6 illustrates RAP values for AlexNet, VGG-11, and ResNet-18 across various quantization types. Since the quantized 8-bit models are used as a reference for RAP calculation, the values are reported as percentages. For AlexNet, heterogeneous quantization shows consistently lower RAP values, underscoring its efficiency and resilience over homogeneous quantization methods. In VGG-11 and ResNet-18, heterogeneous quantization generally achieves lower RAP values across most q values outlined in the algorithm, suggesting the need for further exploration to identify the optimal quantization configurations for these models. This difference in RAP for q values stems from varying accuracy drops, which could fluctuate at different BERs, highlighting how quantization efficiency and resiliency are BER-dependent and may vary across fault injection conditions.

IV. Conclusion

The proposed heterogeneous quantization framework demonstrates substantial improvements in both reliability and performance compared to traditional homogeneous methods. By adapting bit-widths to the resiliency of individual DNN layers, the approach reduced the accuracy drop under fault

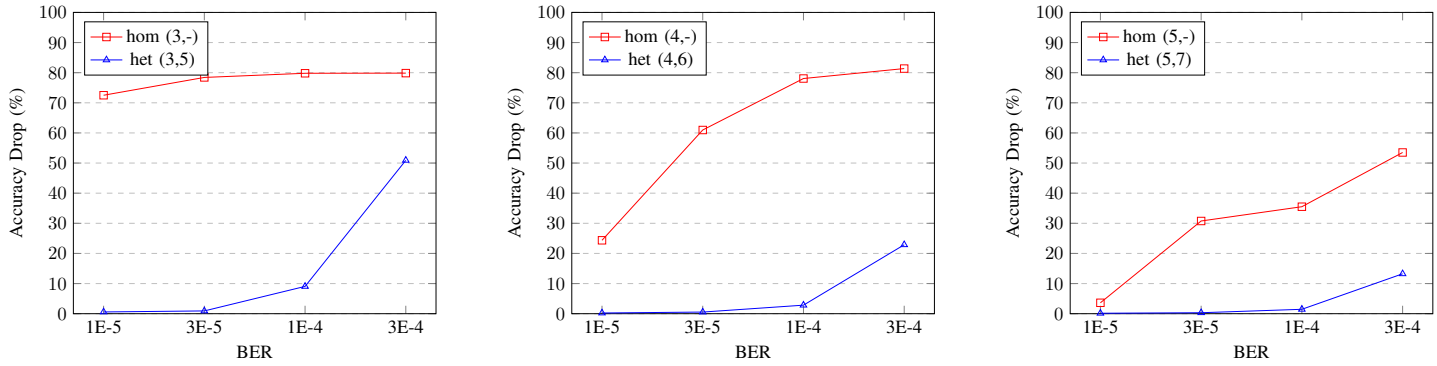
TABLE III: Accuracy drop, memory utilization, execution time and other metrics of AlexNet. Accuracy drop is reported at BER=1E-4.

Type	Accuracy Drop (%)	Memory Utilization	Execution Time (%)	RAP	P_{drop}
hom (3,-)	43.31	174,868,504	100	43.31	1.76E-1
hom (-,5)	0.05	291,447,520	40860.34	34.05	5.65E-4
het (3,5)	11.01	174,973,656	112.43	12.38	4.48E-2
hom (4,-)	16.13	233,158,016	85.40	18.36	1.17E-1
hom (-,6)	1.99	349,737,024	29046.62	1156.05	3.24E-2
het (4,6)	3.14	233,263,168	150.67	6.31	2.27E-2
hom (5,-)	6.46	291,447,520	85.74	9.23	7.30E-2
hom (-,7)	2.52	408,026,528	27418.04	1612.18	5.58E-2
het (5,7)	0.51	291,552,672	151.41	1.28	5.77E-3



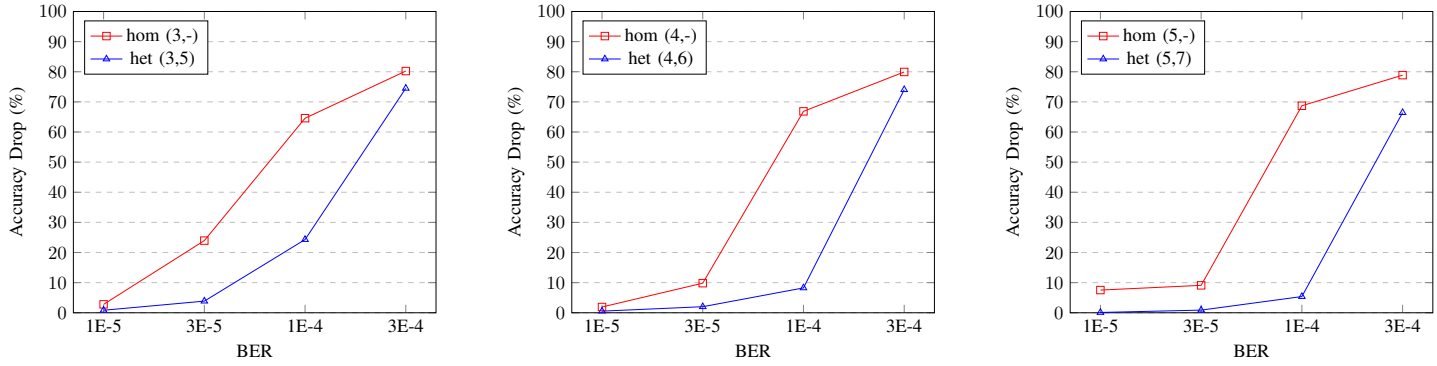
(a) $q = 3$ (b) $q = 4$ (c) $q = 5$

Fig. 3: Fault injection results for different quantization types of AlexNet



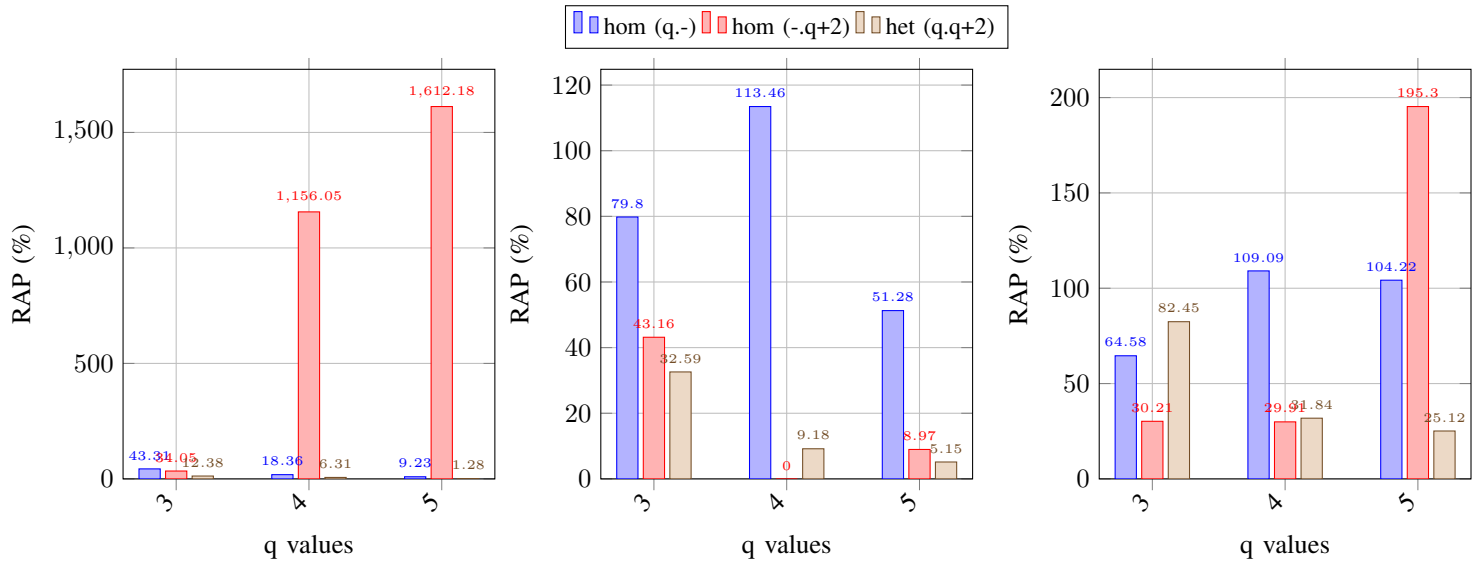
(a) $q = 3$ (b) $q = 4$ (c) $q = 5$

Fig. 4: Fault injection results for different quantization types of VGG-11



(a) $q = 3$ (b) $q = 4$ (c) $q = 5$

Fig. 5: Fault injection results for different quantization types of ResNet-18



(a) AlexNet (b) VGG-11 (c) ResNet-18

Fig. 6: RAP values in different quantization types of DNNs (at $BER = 1E-4$)

TABLE IV: Accuracy drop, memory utilization, execution time and other metrics of VGG-11. Accuracy drop is reported at BER=1E-4.

Type	Accuracy Drop (%)	Memory Utilization	Execution Time (%)	RAP	P_{drop}
hom (3,-)	79.80	84,399,168	100	79.8	7.56E-2
hom (-,5)	0.07	140,665,280	37002.18	43.16	1.84E-4
het (3,5)	9.06	84,632,000	358.74	32.59	8.63E-3
hom (4,-)	78.06	112,532,224	109.01	113.46	1.31E-1
hom (-,6)	0	168,798,336	37512.16	0	0
het (4,6)	2.83	112,765,056	242.89	9.18	4.79E-3
hom (5,-)	30.77	140,665,280	100	51.28	8.10E-2
hom (-,7)	0.01	196,931,392	38461.07	8.97	5.16E-5
het (5,7)	1.46	140,898,112	211.47	5.15	3.85E-3

TABLE V: Accuracy drop, memory utilization, execution time and other metrics of ResNet-18. Accuracy drop is reported at BER=1E-4.

Type	Accuracy Drop (%)	Memory Utilization	Execution Time (%)	RAP	P_{drop}
hom (3,-)	64.58	33,493,056	100	64.58	9.64E-3
hom (-,5)	0.18	55,821,760	10070.57	30.21	7.46E-5
het (3,5)	24.30	38,452,672	295.57	82.45	4.78E-3
hom (4,-)	66.85	44,657,408	122.39	109.09	1.77E-2
hom (-,6)	0.14	66,986,112	10685.55	29.91	8.36E-5
het (4,6)	8.25	49,617,024	260.54	31.84	2.70E-3
hom (5,-)	68.71	55,821,760	91.01	104.22	2.85E-2
hom (-,7)	0.80	78,150,464	10462.76	195.30	6.50E-4
het (5,7)	5.38	60,781,376	257.29	25.12	2.64E-3

conditions by nearly 4 times on average, significantly enhancing reliability. Additionally, the framework improved the reliability-accuracy-performance (RAP) metric by over 80% on average across various networks, achieving these gains with negligible memory overhead. These results underscore the effectiveness of heterogeneous quantization in fault-prone environments without compromising resource efficiency.

V. Acknowledgements

This work was supported in part by the Estonian Research Council grant PUT PRG1467 "CRASHLESS", EU Grant Project 101160182 "TAICHIP" and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID "458578717".

References

[1] M. Taheri, "Dnn hardware reliability assessment and enhancement," *27th IEEE European Test Symposium (ETS).*, May 2022.

[2] M. Taheri *et al.*, "AdAM: adaptive fault-tolerant approximate multiplier for edge DNN accelerators," in *2024 IEEE European Test Symposium (ETS)*, 2024.

[3] ——, "Deepaxe: A framework for exploration of approximation and reliability trade-offs in dnn accelerators," in *2023 24th International Symposium on Quality Electronic Design (ISQED)*. IEEE, 2023, pp. 1–8.

[4] M. H. Ahmadilivani *et al.*, "Enhancing fault resilience of qnns by selective neuron splitting," in *2023 IEEE 5th AICAS*, 2023, pp. 1–5.

[5] S. Nazari *et al.*, "Fortune: A negative memory overhead hardware-agnostic fault tolerance technique in dnns," *Authorea Preprints*, 2024.

[6] N. Cherezova *et al.*, "Heterogeneous approximation of dnn hw accelerators based on channels vulnerability," in *EEE International Conference on Very Large-Scale Integration (VLSI-SoC)*. In press, 2024.

[7] M. Taheri *et al.*, "Exploration of activation fault reliability in quantized systolic array-based DNN accelerators," in *2024 25th International Symposium on Quality Electronic Design (ISQED)*, 2024.

[8] ——, "Saffira: a framework for assessing the reliability of systolic-array-based dnn accelerators," in *2024 27th International Symposium on Design & Diagnostics of Electronic Circuits & Systems (DDECS)*. IEEE, 2024, pp. 19–24.

[9] B. Jacob *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[10] S. Nazari *et al.*, "Genie: Genetic algorithm-based reliability assessment methodology for deep neural networks," in *11th International Conference on Computing and Artificial Intelligence*. In press, 2025.

[11] Z. Yan *et al.*, "When single event upset meets deep neural networks: Observations, explorations, and remedies," in *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2020, pp. 163–168.