electronics

MDPI

Article

Flexible Quantization for Efficient Convolutional Neural Networks

Federico Giordano Zacchigna [1,*], Sergio Lew [2,3] and Ariel Lutenberg [1,3]

1 Universidad de Buenos Aires, Facultad de Ingeniería (FIUBA), Laboratorio de Sistemas Embebidos (LSE), Buenos Aires C1063ACV, Argentina; alutenb@fi.uba.ar
2 Universidad de Buenos Aires, Facultad de Ingeniería (FIUBA), Instituto de Ingeniería Biomédica (IBYME), Buenos Aires C1063ACV, Argentina; slew@fi.uba.ar
3 Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Buenos Aires C1425FQB, Argentina
* Correspondence: fzacchigna@fi.uba.ar

Abstract: This work focuses on the efficient quantization of convolutional neural networks (CNNs). Specifically, we introduce a method called non-uniform uniform quantization (NUUQ), a novel quantization methodology that combines the benefits of non-uniform quantization, such as high compression levels, with the advantages of uniform quantization, which enables an efficient implementation in fixed-point hardware. NUUQ is based on decoupling the quantization levels from the number of bits. This decoupling allows for a trade-off between the spatial and temporal complexity of the implementation, which can be leveraged to further reduce the spatial complexity of the CNN, without a significant performance loss. Additionally, we explore different quantization configurations and address typical use cases. The NUUQ algorithm demonstrates the capability to achieve compression levels equivalent to 2 bits without an accuracy loss and even levels equivalent to $\sim$1.58 bits, but with a loss in performance of only $\sim$0.6%.

Keywords: CNN; quantization; uniform; non-uniform; mixed-precision; FPGA; ASIC; edge devices; embedded systems

check for updates

Citation: Zacchigna, F.G.; Lew, S.; Lutenberg, A. Flexible Quantization for Efficient Convolutional Neural Networks . *Electronics* **2024**, *13*, 1923. https://doi.org/10.3390/electronics13101923

Academic Editor: Chen Yang

Received: 16 March 2024
Revised: 11 April 2024
Accepted: 14 April 2024
Published: 14 May 2024

1. Introduction

The computational complexity of convolutional neural networks (CNNs) has been increasing since their inception, mainly due to their usage in solving more challenging problems, which, in turn, leads to the incorporation of a higher number of parameters. Furthermore, applications now have more stringent requirements, and the platforms where CNNs are deployed face tighter constraints in terms of resource budgets. These constraints include factors such as the size and bandwidth (BW) of external memory, on-chip memory capacity, the number of processing elements (PEs), and maximum power consumption. When considering edge devices, which are gaining popularity, these constraints become even more restrictive. In general, deploying a CNN on one of these embedded platforms is a challenging task, if not impossible. However, this issue can be addressed by applying a technique known as network compression, which allows the CNN to fit within the platform's constraints. One possible classification for compression techniques, starting from a high level and going down to a lower level, is as follows:

1. Architecture modifications: The goal of this compression technique is to define an efficient high-level architecture for the NN. Some examples of architectural modifications include Inception [1] and residual blocks [2], which propose novel nonlinear networks as alternatives to linear networks like VGG [3]. Additionally, MobileNets [4] utilize depthwise convolutions to enhance computational efficiency, while EfficientNets [5] offer a method to uniformly scale the depth, width, and resolution of a network. Another method for optimizing the architecture is network architecture search (NAS) [6],

which falls under the subfield of Auto-ML. NAS automates the search process. Knowledge distillation (KD) [7] is another method that contributes to NN compression. It enables the transfer of knowledge from a larger model to a smaller one, resulting in a more compact network with an improved architecture tailored to address the specific problem effectively. Co-design is another technique that falls within this category [8].

2. Pruning: This technique is based on the idea of removing unimportant parts of the NN [9,10]. Typically, it is applied after training, when it becomes possible to determine what is important and what is not. Pruning can be performed in various ways, including layer-wise, filter-wise, shape-wise, channel-wise, and element-wise [10]. It can be classified into two types: structured and unstructured pruning [10].
3. Quantization: This approach aims to find an efficient way of representing or storing the weights and activations [8,11–24]. Popular quantization techniques include uniform quantization (UQ), non-uniform quantization (NUQ) or weight-sharing, and matrix/tensor decomposition. The KD process may also be applied to optimize the quantization parameters.

Most of these methods can be used together to jointly compress the NN.

The main objective of this work is to address the quantization challenges of CNNs in order to improve CNN efficiency for embedded systems and edge devices; this leads us to 1. develop a flexible quantization approach that balances non-uniform and uniform quantization benefits, aiming for high compression without losing accuracy; and 2. state and solve the optimization problem that lets us find the best quantization parameters for a CNN.

To achieve the objective, it is proposed to apply a hardware-aware quantization. In this work, this means that the resulting compressed network satisfies two key conditions: 1. It can be efficiently implemented using fixed-point (FxP) hardware, and 2. it helps reduce the hardware resource needs, such that the constraints are met. More specifically, our contributions are as follows:

1. Non-uniform uniform quantization (NUUQ): The main idea behind NUUQ is to decouple the number of quantization levels from the number of quantization bits, combined with a clustered based quantization and efficient data clipping. Consequently, there is an extra optimization parameter: the number of quantization levels (or clusters). A mixed-precision (MP) approach is also considered in NUUQ, but with a significant advantage, which is that the granularity of the MP quantization may be different for quantization bits and quantization levels. Moreover, the impact in the hardware complexity when increasing the quantization levels granularity is very little. So it is possible to increase it, but with little impact in the hardware complexity. This results in more quantization flexibility compared to the classical quantization. The overall result is a compressed network that can be efficiently implemented using FxP hardware.
2. Mid-level metrics: As a consequence of decoupling the number of quantization bits and levels, new more representative metrics are needed. We introduce three mid-level complexity metrics that are more hardware-focused than the usual metrics, yet remain implementation-independent. These metrics are utilized to constrain our optimization problem, and they effectively relate the platform constraints to the application requirements.
3. Exploration of quantization parameters and practical applications: We conduct an analysis of how various quantization parameters influence CNN performance, providing insights for their optimal selection. Additionally, we present real-world use cases, showcasing our method's practical benefits in deploying efficient CNNs

When implementing quantization in CNN, there are numerous possibilities to consider. For instance, the choice between UQ or NUQ for each data source, such as kernels, biases, and activations, introduces various possibilities. Additionally, granularity can be adjusted both in terms of the number of bits and quantization levels. It is also possible to combine different types of quantization schemes and granularities for each layer, further expanding the options. In this work, we aim to explore and compare several of these configurations, selecting those that we deem most representative and suitable for real life applications These configurations are detailed in Table 1.

Table 1. Summary of the quantization configurations explored in this study. Some cases might involve fine-tuning (FT). The abbreviated names for each case are displayed in the last column.

Description	Arithmetic	FT	Name
Original model (baseline)	FlP	No	`OM`
UQ with layer-wise MP [12,16,17,20,25–30]	FxP	No	`UQ-L`
	FxP	Yes	`UQ-L-FT`
NUUQ with layer-wise MP [this work]	FxP	No	`NUUQ-LL`
	FxP	Yes	`NUUQ-LL-FT`
NUUQ with channel-wise level selection and layer-wise bit selection [this work]	FxP	No	`NUUQ-LC`
	FxP	Yes	`NUUQ-LC-FT`

This work is structured as follows: Firstly, we present related work in Section 2. Secondly, our proposal is developed in Section 3. In Section 4, we present the results, and finally, we draw conclusions in Section 5.

2. Related Work

Neural networks are implemented on various platforms today, including classic microprocessors with or without machine learning (ML) support, GPUs (graphics processing Units), GPGPUs (general-purpose graphics processing Units), FPGAs (field-programmable gate arrays), ML-specific FPGAs, and ASICs (application-specific integrated circuits). Each of these platforms is suitable for different applications or domains. Additionally, each has distinct advantages, disadvantages, requirements, and limitations [8,31,32].

In terms of the architectures employed, we can highlight that the use of either unrolled or iterative architectures is possible. The iterative approach offers greater flexibility but tends to be less hardware optimized. On the other hand, the unrolled approach is more specific, providing optimizations that result in less flexibility. Moreover, for scenarios where low latency is needed, typically with a batch size of 1, the unrolled method is advantageous compared to its iterative counterpart [33].

In terms of the implementation, there are two major approaches: the use of floating-point (FlP) or FxP arithmetic. FlP arithmetic provides greater flexibility, while FxP, being optimized for a specific application, achieves higher efficiency levels in terms of energy consumption, hardware size and other resource utilization. The use of different FxP arithmetic for different parts of the CNN is proposed in order to improve the network performance. This approach is known as MP [25–27,34,35]. And this is the main reason why unrolled architectures are better using hardware resources. The MP approach has also reached the iterative architectures. The work [36] serves as an example, where an iterative, FxP, ASIC architecture is proposed, seeking efficient resource utilization. The FxP arithmetic, is limited to 2, 4, 8, or 16 bits, and the number of PEs is 8, 4, 2, or 1, respectively.

Works like [32,37–45] propose to use the weight sharing (WS) approach to increase the weight reusability. This enables the storage of a reduced set of weights along with a set of indices, rather than storing all the individual weights separately. As the indices representation requires fewer bits than the weights, it results in a reduction in memory footprint. To implement WS, clustering algorithms are usually used, with the most popular one being the K-means clustering. The WS approach is commonly employed in conjunction with FlP arithmetic. As the probability of each weight is not even along all of them, coding results are useful for further reducing the memory footprint.

The most widely used coding scheme is Huffman coding (HC). This has been used in works like [46–49].

The utilization of FT in order to recover accuracy loss has become essential whenever feasible; this is when training data are available [50–52]. The FT technique has also been proposed in [16] to recover performance after applying quantization, which is especially relevant for this work.

Quantization algorithms can primarily be classified into UQ and NUQ. UQ algorithms hold a significant advantage in being easily implementable using FxP arithmetic, which reduces hardware complexity, thereby enhancing performance and lowering latency and power consumption [30,36,53]. On the other hand, NUQ algorithms typically utilize hardware with FlP arithmetic, which has greater complexity compared to FxP hardware, but they offer the advantage of achieving higher levels of accuracy [13,37].

Nowadays, state-of-the-art hardware for large-scale deployment of ML algorithms for inference is primarily dominated by FxP hardware, spanning across CPUs, GPUs, TPUs, FPGAs, and ASICs [8,31–33,53]. This puts NUQ algorithms at a disadvantage. To overcome this challenge, Ref. [18] proposes a combination of UQ and NUQ. Similarly, this work proposes NUUQ, which also combines both approaches; however, in contrast to [18], this study is not confined to 2^n quantization levels, nor does it enforce equidistant levels, which resembles NUQ more closely.

When using UQ, it is very important to properly choose the quantization precision, specially for the lower-value regions [9,15,54–58]. This process depends on several variables: the scale factor, the range of values, and the number of allocated bits. In [15], it is also proposed to clip the higher values in order to improve the lower-value resolution. Moreover, this clipping parameter, α-value, is optimized during training.

3. Proposed Quantization Method and Metrics

3.1. Quantization Method

The NUUQ approach proposes the combination of NUQ and UQ techniques to achieve non-uniform quantization while remaining efficient for implementation using FxP representation. This is accomplished through a layer-wise approach involving clipping, NUQ (utilizing K-means, although other methods are also viable), followed by subsequent UQ applied to the cluster centers. Additionally, the cluster centers are encoded using HC to improve memory footprint efficiency. Finally, FT is optionally applied to recover any performance lost during the quantization process. Figure 1 provides an example of how the NUUQ quantization function (Q^{NUU}) is formed, and Figure 2 demonstrates the outcome of applying NUUQ to the weights of a specific layer.

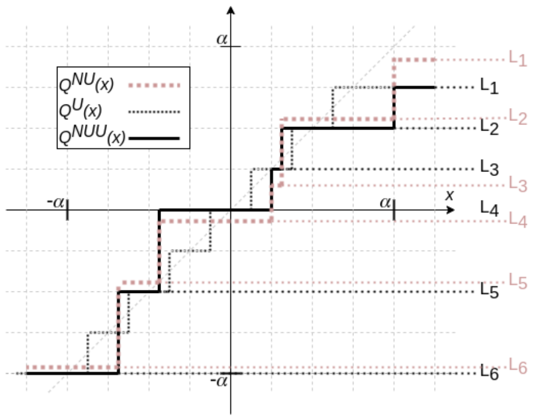


Figure 1. The image serves as an illustration of how to derive the quantization function Q^{U} by combining three functions: clipping, Q^{α} (not shown in the image), non-uniform quantization, Q^{NU}, and uniform quantization, Q^{U}. The valid quantization levels for FxP arithmetic are indicated by the arrows on the right side of the image, alongside with the original and quantized levels. As we can see, the levels for QNU are not valid, making it inviable for FxP hardware implementation. Conversely, the levels for Q^{NUU} are valid since they were quantized using Q^{U}. As observed, Q^{NUU} serves as an approximation of Q^{NU}, and this approximation improves with an increase in the number of bits used in Q^{U}.

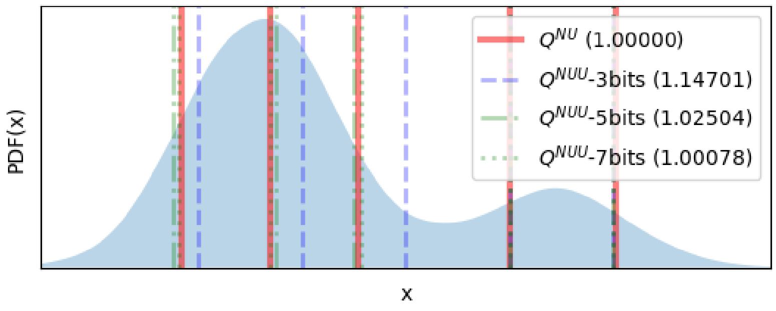


Figure 2. This image displays the probability density distribution for the original weights (represented by the light blue filled curve) along with the values of the cluster centers (indicated by vertical lines) after quantization. We applied Q^{NU} and Q^{NUU} with 3, 5, and 7 bits, while maintaining a fixed number of clusters (5 clusters) across all cases. In the case of Q^{NUU}, the centers, x^{NUU}, are adjusted to match FxP valid values, resulting in displacement from the original centers, x^{NU}. The number between brackets in the legend indicates the mean of the variance of the clusters after normalization by the variance of the original cluster, this is $\mathrm{mean}[\mathrm{var}(x_i^{\mathrm{NU}})/\mathrm{var}(x_i^{\mathrm{NU}})]$. As expected, the variance decreases with an increase in the number of bits.

It is important to note that both the allocation of bits and number of clusters is performed on a layer-wise basis. This simplifies hardware implementation since PEs and decoders (in the case of hierarchical coding usage) can be utilized across all channels. While bit and number of cluster assignment is conducted on a layer-wise basis, clustering itself can be executed either on a layer-wise or channel-wise basis, with the expectation that the latter approach may yield improved results in terms of performance and spatial complexity.

3.2. Mid-Level Metrics and Optimization Problem

During the implementation process, it is important to have notion of how our design variables relate to our constraints. For example, if we assume that the weights of the CNN are stored in external memory, the number of bits used for weights quantization are directly related to the BW needs. In this work, the number of quantization bits and quantization levels have no longer a tight relationship, so it is important to see that the usual metrics are no longer representative. For example, the spatial complexity of the weights, now depends has a tight dependence with the number of clusters, while the size (in bits) of the activations depends on the number of bits. For this reason, in this work, we introduce three mid-level complexity metrics that are more hardware-focused and yet remain implementation-independent.

1. Binary operations complexity (BOC): This metric represents the binary time complexity (hardware complexity) of the layer, measured in binary operations. Is represented by c^{T}.
2. Binary weight complexity (BWC): This metric quantifies the spatial complexity of the weights (K and B), measured in bits. Is represented by c^{W}.
3. Binary activation complexity (BAC): This metric measures the spatial complexity of the layer activation, expressed in bits. Is represented by c^{A}.

These metrics are derived for our approach as follows. First, it is assumed that the output of each layer can be efficiently computed using multiply and accumulate (MAC) operations. In this situation, we are interested in knowing the sizes of I, K, B, and A (n^{I}, n^{K}, n^{B} y n^{A}), as well as the total number of MACs (n^{M}) for each layer. In [59], these values are derived for the Conv2D and dense layers. We will also assume that PEs implement MAC operations using FxP arithmetic according to Figure 3, which is very representative for both ASIC and FPGA).

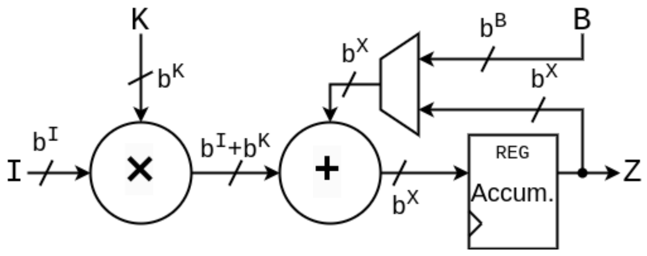


Figure 3. This figure illustrates the multiply–accumulate (MAC) architecture with inputs I, K, and B, and the output Z. The inputs have bit widths of b^{I}, b^{K} and b^{B}, respectively, while the accumulator bit width is $b^{\mathrm{X}} = \max(b^{\mathrm{I}} + b^{\mathrm{K}} + b^{\mathrm{E}}, b^{\mathrm{B}})$, where b^{E} represents the number of extra bits used to prevent overflow during the successive sums. For instance, in a conservative implementation, it may be set the value of b^{E} to $\lceil \log_2(n^{\mathrm{M}}) \rceil$ to prevent any potential overflow.

Next, it is common to consider that the binary complexity for a sum of two words with m and n bits, respectively, is equal to $\max(m, n)$, and the binary complexity for the multiplication of those two words is equal to $(m\,n)$. Finally, we are in a position to derive the formulas for the proposed metrics:

$$c^{\mathrm{T}} = n^{\mathrm{M}}\, b^{\mathrm{K}}\, b^{\mathrm{I}} + n^{\mathrm{M}}\, b^{\mathrm{X}} \tag{1}$$

$$c^{\mathrm{W}} = n^{\mathrm{K}}\, L^{\mathrm{K}} + k^{\mathrm{K}}\, b^{\mathrm{K}} + n^{\mathrm{B}}\, L^{\mathrm{B}} + k^{\mathrm{B}}\, b^{\mathrm{B}} \tag{2}$$

$$c^{\mathrm{A}} = n^{\mathrm{A}}\, b^{\mathrm{A}} \tag{3}$$

According to our NUUQ quantization approach, in each layer, we have these variables: clipping values (α^{K}, α^{B}, α^{A}), number of bits (b^{K}, b^{B}, b^{A}), and number of clusters (k^{K}, k^{B}, k^{A}). Even though it is possible to use custom quantization levels for the activations, its implementation needs special hardware in the datapath. Thus, the k^{A} is discarded, which better matches real-life platforms. The optimization problem we address in this work becomes

$$\begin{aligned} \mathcal{Q}^{\star} &= \underset{\mathcal{Q}}{\operatorname{argmax}}\ \text{accuracy}(\mathcal{W}, \mathcal{Q}; \mathcal{I}) \\ &\text{s.t.}\quad c^{\mathrm{T}} < c^{\text{T-max}}, \quad c^{\mathrm{W}} < c^{\text{W-max}}, \quad c^{\mathrm{A}} < c^{\text{A-max}} \end{aligned} \tag{4}$$

with

$$\mathcal{Q} = \left\{ \alpha_i^{\{\mathrm{K,B,A}\}}, b_i^{\{\mathrm{K,B,A}\}}, k_i^{\{\mathrm{K,B}\}} \right\} \tag{5}$$

where $\mathcal{W}$ and $\mathcal{Q}$ are the model parameters and the quantization parameters of the CNN, respectively, and $\mathcal{I}$ is the validation data.

In this work, we want the most even results regardless of the quantization approach used; thus, a random search algorithm is used, even if the search is very slow. In [59], the search is described in detail.

4. Results

In this section, we will present the results obtained in two subsections: In Section 4.1, the exploration conducted across various cases is depicted, while Section 4.2 shows the outcomes of applying `NUUQ` to two typical use cases. In both scenarios, a downsized version of the VGG (See [59] for more detail) and the CIFAR10 dataset were employed.

For all cases, before optimizing quantization parameters ($\mathcal{Q}$), the constraints for the problem in (4) must be set. These maximum values for complexities are derived from a uniformly quantized model, enabling the standardization of these values. In other words, the numbers of bits and clusters for weights and activations (as applicable) are chosen to quantize a model uniformly and the resulting complexities of this model are used as the constraints in (10). This approach allows us to link a single tuple with number of bits and clusters, $(b^{\mathrm{K}}, c^{\mathrm{K}}, b^{\mathrm{B}}, c^{\mathrm{B}}, b^{\mathrm{A}})$ to complexity values, providing a more intuitive understanding for the reader. For instance, a uniformly quantized model with 4 bits and 4 clusters, $(b^{\mathrm{K}}, c^{\mathrm{K}}, b^{\mathrm{B}}, c^{\mathrm{B}}, b^{\mathrm{A}}) = (4, 4, 4, 4, 4)$, has a given $(c^{\mathrm{T}}, c^{\mathrm{W}}, c^{\mathrm{A}})$. This allows us to claim that a model has been heterogeneously quantized with 4 bits and 4 clusters, which might have no meaning as the number of bits and clusters varies for each layer. However, it actually refers to the complexities of this model being less than or equal to those of the uniformly quantized model with these bit and cluster values. This provides a more intuitive way of expressing the complexities.

For the remaining figures in this section, when referring to the number of bits and clusters, it should be understood that there is a direct relationship with complexities, as explained in the previous paragraph.

4.1. Exploration

In this section exploration results are analysed. Observing the heatmaps in Figure 4, it is evident that accuracy is highly sensitive to both the number of activation bits and the average number of clusters. However, concerning the latter, it is also noticeable that once a certain threshold for accuracy is reached, the sensitivity diminishes. As for the number of bits, while there is a modest improvement as they increase, it is also noteworthy that accuracy shows little sensitivity to this value, at least within the range of 3 to 8 bits. On the other hand, the significance of applying FT is clear, as this process significantly enhances accuracy. This outcome is well-known, and therefore, not surprising. The application of FT typically depends solely on data availability.

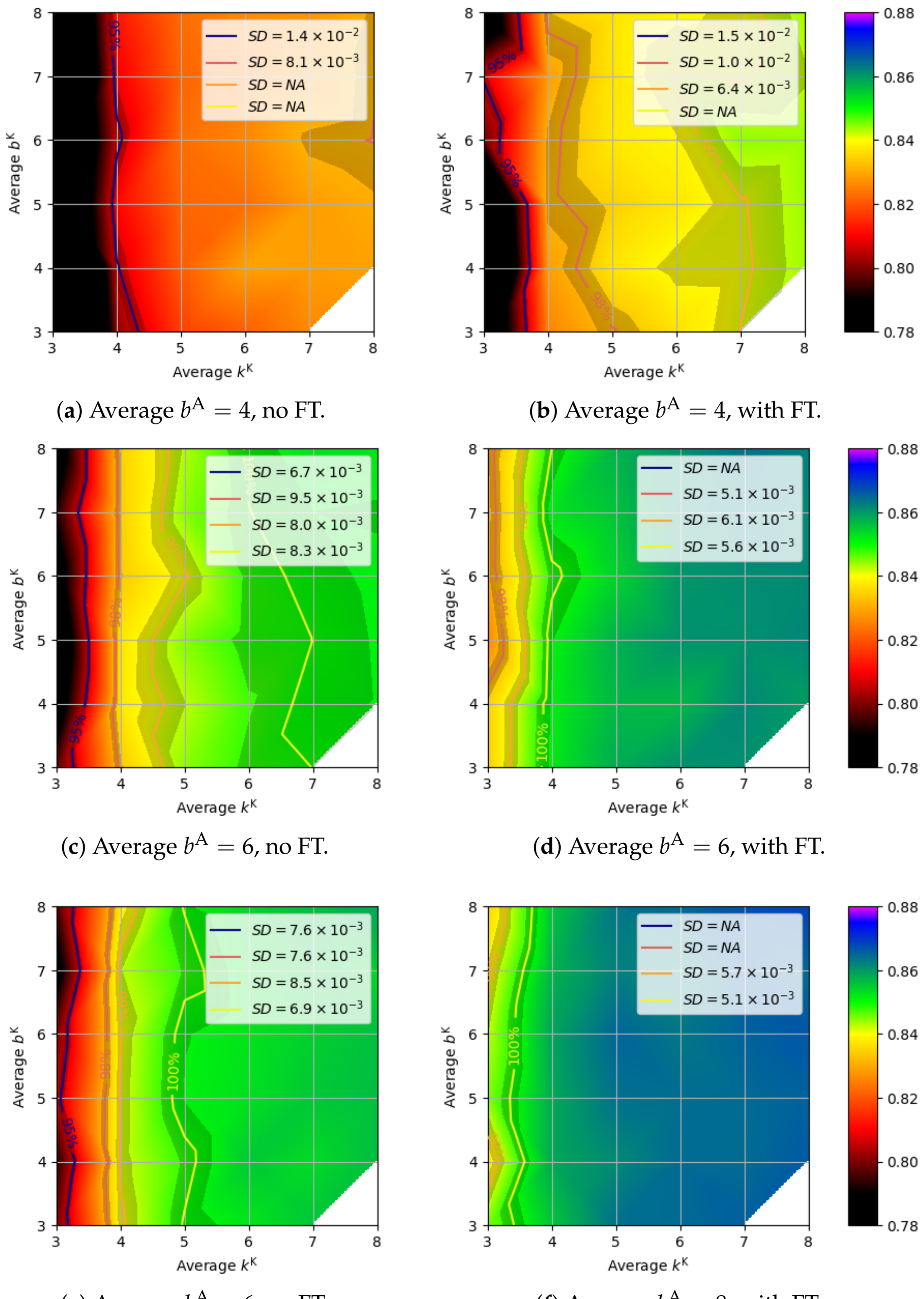
For a deeper analysis, we consider the conditions under which 100% accuracy can be recovered in the context of NUUQ. These findings are segmented based on activation bits, the number of clusters, and the number of bits for weights, providing a comprehensive understanding of their respective impacts. The possibility of regaining full accuracy is significantly influenced by the value of b^A. Our analysis indicates a critical range between 4 and 6 bits for b^A, within which the feasibility of achieving 100% accuracy is determined, regardless of the application of FT. This value should be higher for the case without FT than in those with FT, but unfortunately it is not possible to define this value without further simulations. In the case of the number to recover full accuracy, the required value varies in the presence of FT and for different values of b^A. Table 2 showcases this the ranges for k^K:

Table 2. Approximate ranges for the number of clusters (k^K) required to achieve 100% accuracy in CNN quantization, categorized by activation bits (b^A) and the application of fine-tuning (FT).

Average Activation Bits (b^A)	**Number of Clusters**	
	Without FT	**With FT**
8	-	-
4	6.5 to 7.0	3.8 to 4.1
6	4.7 to 5.4	3.4 to 3.7

Our findings suggest that the specific value of b^K is less critical, provided it exceeds 3 (lower values were not analysed). However, in scenarios with $b^A = 4$ (both with and without FT) and $b^A = 6$, an increase in b^K appears to compensate for the limitations associated with lower b^A or the absence of FT.

Understanding these critical values is not just academic; it is crucial for optimising CNN quantization. By identifying specific ranges and thresholds affecting accuracy recovery, we can define the parameter exploration space more effectively. This strategic approach tailors quantization parameters to meet accuracy requirements while optimising hardware resources. By avoiding values that lead to under-quantization, which wastes hardware resources, and over-quantization, which may lead to inadequate performance, the design space is smaller, enabling more efficient parameter exploration. These insights guide the quantization process, ensuring computational resources are allocated judiciously for optimal balance between accuracy and efficiency.



(**a**) Average $b^{\mathrm{A}} = 4$, no FT. (**b**) Average $b^{\mathrm{A}} = 4$, with FT.

(**c**) Average $b^{\mathrm{A}} = 6$, no FT. (**d**) Average $b^{\mathrm{A}} = 6$, with FT.

(**e**) Average $b^{\mathrm{A}} = 6$, no FT. (**f**) Average $b^{\mathrm{A}} = 8$, with FT.

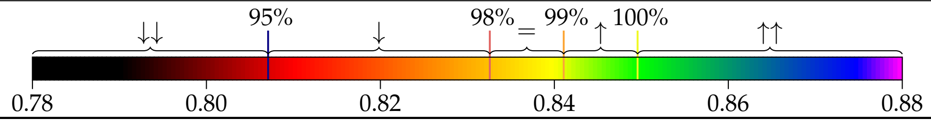
Figure 4. This figure presents the test accuracy for various base complexity values (BOC, BWC and BAC) when utilizing the NUUQ-RS search flow (**a**–**c**) and NUUQ-RS-FT (**d**–**f**) ; this is without and with Fine-Tuning (FT), respectively. The base complexity values are obtained using the base model when globally quantized with $b^{\{\mathrm{K,\,B,\,A}\}}$ bits and $k^{\{\mathrm{K,\,B}\}}$ clusters. During the search, the bits and clusters are allocated layer-wise while always keeping the complexity below the base complexity values. The base complexity values are calculated using $b^{\mathrm{K}} = \{3, \ldots, 8\}$, $k^{\mathrm{K}} = \{3, \ldots, 8\}$, $b^{\mathrm{B}} = 8$, $k^{\mathrm{B}} = 32$ and $b^{\mathrm{A}} = \{4, 6, 8\}$ (4 bits for (**a**,**d**); 6 bits for (**b**,**e**); 8 bits for (**c**,**f**)). The contours in the graph represent constant accuracy slices at levels equivalent to {95%, 98%, 99%, 100%} of the FlP model accuracy. The not-available (NA) text in the legend means that the x% of accuracy is not reached for that heatmap.

In Table 3, some heatmap points are presented in numerical form. Based on these, some noteworthy results include the following:

- With `NUUQ`, it is possible to achieve accuracies above 99% (marked as ↑) with reductions of 10.7 times ($b^A = 4$), 16 times ($b^A = 6$), and even up to 20.2 times ($b^A = 8$).
- In some cases using `NUUQ` surpass the original accuracy (marked as ↑↑) with reductions of 1.7 times ($b^A = 6$) and up to 16 times ($b^A = 8$).
- In all cases, comparing `UQ-MP` against `NUUQ-MP` shows that it is feasible to sacrifice some BOC for a significant improvement in BWC. For instance:
  - Compressed by 10.7 times, `NUUQ-MP` achieves improvements of up to 1.6% compared to `UQ-MP`.
  - Compressed by 16.0 times, `NUUQ-MP` achieves improvements of up to 9.3% compared to `UQ-MP`.
- `NUUQ` achieves compressions of around 20.2 times, losing only 0.6% accuracy for the bA=8 case. This compression level is equivalent to using approximately 1.58 bits for the quantization in terms of the BWC. These compression levels are prohibitive for `UQ-MP`.
- Regarding the BAC, there are no significant differences observed between `UQ-MP` and `NUUQ-MP`.

Table 3. Accuracies and complexities for noteworthy configurations and cases.

Case			Accuracy (SD)		Complexity		
			Without FT	With FT	BOC	BWC	BAC
OM (FlP-32)			0.852 (0.013)	– (–)	NA	1.0	1.0
OM (FlP-16)			0.852 (0.013)	– (–)	NA	2.0	2.0
UQ-MP (FxP)	b^A	b^K	Without FT	With FT	BOC	BWC	BAC
	4	2	0.422 (0.131) ↓↓	0.644 (0.088) ↓↓	2.4	16.0	29.3
		3	0.791 (0.026) ↓↓	0.821 (0.016) ↓	2.2	10.7	29.3
	6	2	0.528 (0.078) ↓↓	0.751 (0.047) ↓↓	2.2	16.0	19.5
		3	0.815 (0.023) ↓	0.844 (0.010) ↑	1.9	10.7	19.5
	8	2	0.567 (0.085) ↓↓	0.765 (0.055) ↓↓	2.0	16.0	14.7
		3	0.825 (0.018) ↓	0.851 (0.009) ↑↑	1.7	10.7	14.7
NUUQ-MP (FxP)	b^A	$b^K(k^K)$	Without FT	With FT	BOC	BWC	BAC
	4	3 (3)	0.713 (0.065) ↓↓	0.769 (0.059) ↓↓	2.2	20.2	29.3
		8 (3)	0.761 (0.028) ↓↓	0.807 (0.022) ↓	1.5	20.2	29.3
		3 (4)	0.807 (0.010) ↓↓	0.829 (0.005) ↓	2.2	16.0	29.3
		8 (4)	0.812 (0.008) ↓	0.834 (0.004) =	1.5	16.0	29.3
		4 (8)	0.830 (0.003) ↓	0.845 (0.005) ↑	2.0	10.7	29.3
		8 (8)	0.831 (0.003) ↓	0.844 (0.005) ↑	1.5	10.7	29.3
	6	3 (3)	0.801 (0.018) ↓↓	0.834 (0.019) =	1.9	20.2	19.5
		8 (3)	0.776 (0.042) ↓↓	0.828 (0.017) ↓	1.2	20.2	19.5
		3 (4)	0.835 (0.009) =	0.853 (0.008) ↑↑	1.9	16.0	19.5
		8 (4)	0.837 (0.008) =	0.851 (0.007) ↑↑	1.2	16.0	19.5
		4 (8)	0.852 (0.006) ↑↑	0.861 (0.006) ↑↑	1.7	10.7	19.5
		8 (8)	0.853 (0.005) ↑↑	0.864 (0.003) ↑↑	1.2	10.7	19.5
	8	3 (3)	0.797 (0.021) ↓↓	0.846 (0.009) ↑	1.7	20.2	14.7
		8 (3)	0.798 (0.034) ↓↓	0.846 (0.008) ↑	1.0	20.2	14.7
		3 (4)	0.842 (0.005) ↑	0.858 (0.004) ↑↑	1.7	16.0	14.7
		8 (4)	0.839 (0.005) =	0.856 (0.006) ↑↑	1.0	16.0	14.7
		4 (8)	0.857 (0.003) ↑↑	0.867 (0.003) ↑↑	1.5	10.7	14.7
		8 (8)	0.859 (0.003) ↑↑	0.869 (0.004) ↑↑	1.0	10.7	14.7



4.2. Use Cases

Let us explore two typical scenarios where using NUUQ quantization presents advantages. The first use case is common in a configurable iterative architecture where arithmetic is fixed, such as at 4, 8, or 16 bits. This situation is typically encountered in CNN implementations on FPGA, GPU, or CPU. While ASIC implementations are possible, they are less common since they would not fully leverage their advantages. In the second case, we consider unrolled architectures, typically found in both ASIC and FPGA. In such architectures, it is common to encounter different arithmetic applied in each layer.

Figures 5 and 6 showcase the results for the first and second cases, respectively. In both scenarios, applying NUUQ demonstrates the potential for a reduction in BWC without sacrificing accuracy, although achieving this requires a trade-off in BOC. This is evident in Figure 5, where a NUUQ case without sacrificing BOC (blue curve) does not yield superior accuracy, as applying NUUQ without sacrificing BOC is equivalent to UQ.

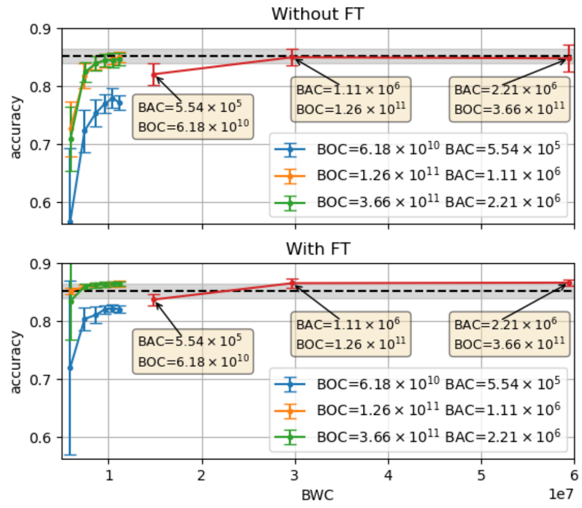


Figure 5. This chart depicts accuracy versus binary weights complexity (BWC) for the first use case. It illustrates that using NUUQ allows for a lower BWC without sacrificing accuracy compared to UQ. The upper graph displays results without applying FT, while the lower one incorporates FT.

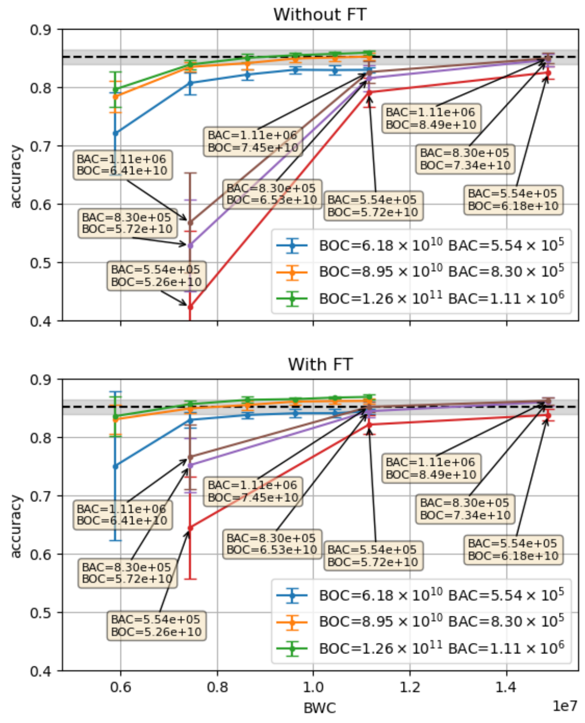


Figure 6. This graph presents findings similar to those depicted in Figure 5, but for the second use case.

In closing this exploration of use cases, we want to highlight how they apply to real situations. The examples we have talked about are based on real challenges that people face when using convolutional neural networks in edge devices [30,33,36,53]. We have shown how our NUUQ method can be adjusted and used effectively in different situations, which helps close the gap between research and actual use. In this way, our work is also useful for making AI a reality.

4.3. Comparison to Other Works

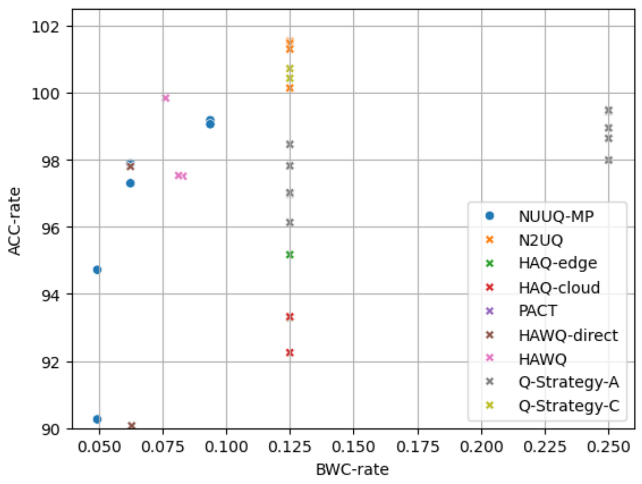
In Figure 7, we present a comparative analysis of the NUUQ strategy alongside methodologies from other studies. We compare the accuracy loss rate against the weight compression rate for a comprehensive evaluation across these works The metrics are defined as follows:

$$\text{ACC-rate} = \frac{\text{ACC}}{\text{ACC-FlP}} \qquad \text{BWC-rate} = \frac{\text{BWC}}{\text{BWC-FlP}} \tag{6}$$

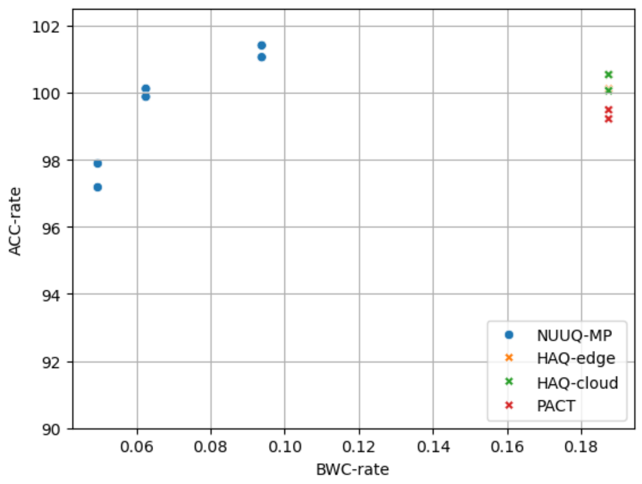
here, ACC-FlP represents the accuracy of the original model using FlP arithmetic with 32 bits, while BWC-FlP denotes the BWC under the same conditions. Ideally, we aim for a high ACC-rate together with a low BWC-rate. Figure 7 shows these metrics for different number of activation bits, 4, 6 and 8, in Figures 7a, 7b and 7c respectively. Figure 7 aggregates all the previous cases alongside new values for the average activation bits. The effect of changing the average number of activation bits and the average number of bits used for weights (only for this work) can be seen as vertically aligned points, this is, points with the same BWC-rate but different ACC-rate. From this figure, it can be seen that our NUUQ approach consistently surpasses other methodologies in a broad range of scenarios with different average numbers of bits for activations and weights. However, it is important to acknowledge that comparisons across different studies are inherently complex and may not always be directly comparable due to various underlying factors:

1. Unlike the method described in [18], our approach employs standard quantization for activations. This decision helps us avoid the need for specialized hardware in the datapath, ensuring better alignment with fundamental design principles. Nevertheless, exploring NUUQ for activations will be performed in future research to enhance design flexibility and enable comparative analysis.
2. Our methodology inherently allows for more bits allocated to weights for a given BWC due to its unique ability to decouple the number of quantization levels from the bit count. This flexibility also means that for a fixed number of bits, our approach results in fewer quantization levels, making direct comparisons challenging.
3. The CNN architectures employed vary across studies. Our analysis exclusively utilizes the VGG16 structure, while other research includes models like VGG, ResNet, and MobileNet, adding another layer of complexity to the comparison.
4. Different datasets are used across different works. Our study utilizes CIFAR-10, whereas many others employ ImageNet. The choice of CIFAR-10 was made by the limitations of our search algorithm's efficiency with larger datasets.
5. Contrary to most studies that focus on search optimization, our work employs a random search strategy to ensure fairness and comprehensiveness in parameter exploration.
6. Our FT methodology is applied globally across the entire network, diverging from other studies that FT sequentially layer by layer. While the latter may yield more refined FT results, our goal is to maintain fairness in parameter search.

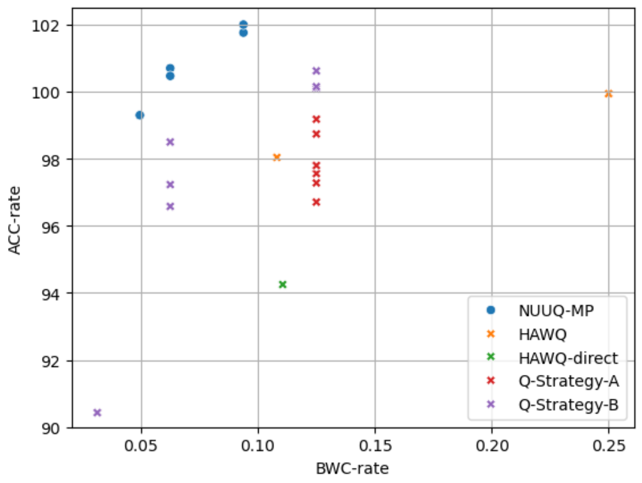
These points reveal that our comparison is both detailed and complex, with some factors benefiting our approach and others presenting disadvantages.



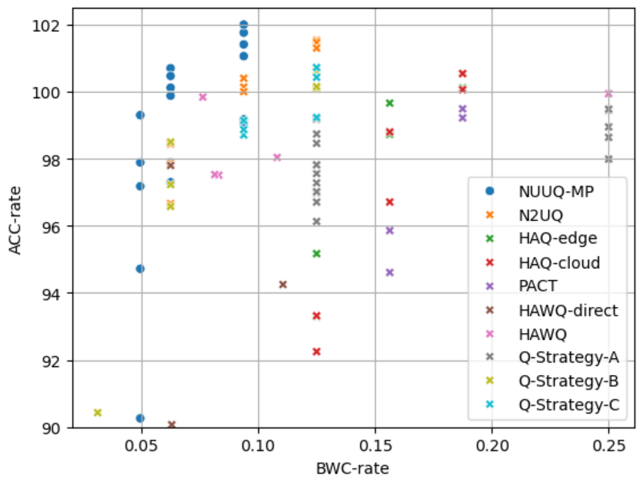
(**a**) Average activation bits $b^A = 4$.



(**b**) Average activation bits $b^A = 6$.



(**c**) Average activation bits $b^A = 8$.



(**d**) All values for the average activation bits, $b^A \in \{2.0, 2.5.0, 3.0, 4.0, 5.0, 6.0, 8.0\}$.

Figure 7. This figure provides a comparison of the accuracy and BWC rate between our work and others. Subfigures delineate various scenarios with average activation bits set to 4, 6, and 8 (see Subfigures **a**–**c**). Additionally, Subfigure (**d**) aggregates results from all previous scenarios alongside new ones. The name NUUQ-MP is the quantization proposed in this work, N2UQ in [18], HAQ-edge and HAQ-cloud in [12], PACT in [15], HAWQ-direct and HAWQ in [16], Q-Strategy-A in [17], Q-Strategy-B in [26] and Q-Strategy-C in [29].

4.4. Limitations

While the proposed NUUQ method demonstrates significant advancements in the efficient quantization of convolutional neural networks, particularly in terms of achieving high compression levels without substantial accuracy loss, it is important to recognize that are some limitations:

- Random search efficiency: The utilization of a random search algorithm for the optimization of quantization parameters, while effective in exploring a fair solution space, presents scalability challenges. This method's computational intensity limits its applicability to larger network architectures or more extensive datasets.
- Network architecture scope: During our parameter exploration, we specifically focused on sequential CNN models because of their straightforward structure, which is conducive to clear and effective result demonstration. This choice enabled us to transparently showcase the advantages of our methodology. Nonetheless, it is important to note that concentrating on these simpler architectures may limit how our

findings apply to more intricate or non-sequential network designs, such as those incorporating skip connections or parallel processing paths.

- Layer diversity: the current implementation of NUUQ has been applied to a select subset of neural network layers, primarily those commonly found in conventional CNNs used for classification tasks. The quantization techniques presented herein should be extended to other layer types.

By addressing these limitations in future iterations of our research, we aim to enhance the performance, applicability, and scalability of the NUUQ method.

5. Conclusions and Future Work

Firstly, the metrics proposed in this study provide more insightful information than classical metrics used in high-level models, making them more suitable for characterizing quantized models operating in FxP arithmetic.

Secondly, the key conclusion of our work is the notable advantages of applying NUUQ. The proposed technique introduces the number of clusters as a quantization parameter, providing an extra degree of freedom when applying mixed-precision quantization. Consequently, a trade-off relationship between BOC and BWC metrics is established, allowing for the possibility to sacrifice BOC for a reduction in BWC without facing a loss in accuracy. As a result, the number of potential quantization configurations increases, allowing for more efficient configurations.

Regarding future direction, we want to highlight a few aspects. Our immediate focus will be on refining the search methodology used for optimizing quantization parameters. The current random search approach, while comprehensive, limits the scalability of NUUQ to larger CNN architectures and more extensive datasets.

Another intriguing direction involves the quantization of activations in addition to weights and biases. Although this extension may pose challenges for hardware implementation due to the dynamic nature of activations, it holds potential for comprehensive comparisons and deeper insights into quantization impacts across different network components.

It is also important to address non-sequential networks, which will allow us to address a broader spectrum of CNN architectures, including those with complex topologies such as skip connections and parallel branches.

Incorporating depth-wise convolutions, known for their efficiency in CNN architectures, will be a significant step forward. Adapting NUUQ to support this layer type could enhance its utility, especially in mobile and embedded applications where computational resources are limited.

Finally, a vital aspect of our future work involves translating the theoretical and simulation-based outcomes of this research into real-world applications. Implementing NUUQ-optimized CNNs in hardware platforms such as ASICs, FPGAs, or microprocessors will validate its practical viability. The use cases presented in this study serve as a foundation for this transition, with the ultimate goal of deploying efficient, quantized CNNs in actual embedded systems.

Author Contributions: Conceptualization, F.G.Z., S.L. and A.L.; Methodology, F.G.Z.; Software, F.G.Z.; Validation, F.G.Z.; Investigation, F.G.Z.; Resources, F.G.Z.; Writing—original draft, F.G.Z.; Writing—review & editing, F.G.Z., S.L. and A.L.; Supervision, S.L. and A.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data used in this study are well known and available online. The dataset name is https://www.cs.toronto.edu/~kriz/cifar.html (accessed on 13 April 2024).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going Deeper with Convolutions. *arXiv* **2014**, arXiv:cs.CV/1409.4842.
2. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. *arXiv* **2015**, arXiv:cs.CV/1512.03385.
3. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2015**, arXiv:cs.CV/1409.1556.
4. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**, arXiv:cs.CV/1704.04861.
5. Tan, M.; Le, Q.V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *arXiv* **2020**, arXiv:cs.LG/1905.11946.
6. White, C.; Safari, M.; Sukthanker, R.; Ru, B.; Elsken, T.; Zela, A.; Dey, D.; Hutter, F. Neural Architecture Search: Insights from 1000 Papers. *arXiv* **2023**, arXiv:cs.LG/2301.08727.
7. Gou, J.; Yu, B.; Maybank, S.J.; Tao, D. Knowledge Distillation: A Survey. *Int. J. Comput. Vis.* **2021**, *129*, 1789–1819. [CrossRef]
8. Campos, J.; Dong, Z.; Duarte, J.; Gholami, A.; Mahoney, M.W.; Mitrevski, J.; Tran, N. End-to-end codesign of Hessian-aware quantized neural networks for FPGAs and ASICs. *arXiv* **2023**, arXiv:cs.LG/2304.06745.
9. Tung, F.; Mori, G. CLIP-Q: Deep Network Compression Learning by In-parallel Pruning-Quantization. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 7873–7882. [CrossRef]
10. Liang, T.; Glossner, J.; Wang, L.; Shi, S.; Zhang, X. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing* **2021**, *461*, 370–403. [CrossRef]
11. Gholami, A.; Kim, S.; Dong, Z.; Yao, Z.; Mahoney, M.W.; Keutzer, K. A Survey of Quantization Methods for Efficient Neural Network Inference. *arXiv* **2021**. [CrossRef]
12. Wang, K.; Liu, Z.; Lin, Y.; Lin, J.; Han, S. HAQ: Hardware-Aware Automated Quantization with Mixed Precision. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 8604–8612. [CrossRef]
13. Choi, Y.; El-Khamy, M.; Lee, J. Towards the Limit of Network Quantization. *arXiv* **2017**, arXiv:cs.CV/1612.01543.
14. Gong, Y.; Liu, L.; Yang, M.; Bourdev, L. Compressing Deep Convolutional Networks using Vector Quantization. *arXiv* **2014**, arXiv:cs.CV/1412.6115.
15. Choi, J.; Wang, Z.; Venkataramani, S.; Chuang, P.I.J.; Srinivasan, V.; Gopalakrishnan, K. PACT: Parameterized Clipping Activation for Quantized Neural Networks. *arXiv* **2018**. [CrossRef]
16. Dong, Z.; Yao, Z.; Gholami, A.; Mahoney, M.; Keutzer, K. HAWQ: Hessian AWare Quantization of Neural Networks with Mixed-Precision. *arXiv* **2019**, arXiv:cs.CV/1905.03696.
17. Banner, R.; Nahshan, Y.; Soudry, D. Post training 4-bit quantization of convolutional networks for rapid-deployment. In Proceedings of the 33rd International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; Curran Associates Inc.: Red Hook, NY, USA, 2019.
18. Liu, Z.; Cheng, K.T.; Huang, D.; Xing, E.; Shen, Z. Nonuniform-to-Uniform Quantization: Towards Accurate Quantization via Generalized Straight-Through Estimation. *arXiv* **2022**, arXiv:cs.CV/2111.14826.
19. Martinez, J.; Shewakramani, J.; Liu, T.W.; Bârsan, I.A.; Zeng, W.; Urtasun, R. Permute, Quantize, and Fine-tune: Efficient Compression of Neural Networks. *arXiv* **2020**. [CrossRef]
20. Bablani, D.; Mckinstry, J.L.; Esser, S.K.; Appuswamy, R.; Modha, D.S. Efficient and Effective Methods for Mixed Precision Neural Network Quantization for Faster, Energy-efficient Inference. *arXiv* **2024**, arXiv:cs.LG/2301.13330.
21. Liu, Z.; Oguz, B.; Zhao, C.; Chang, E.; Stock, P.; Mehdad, Y.; Shi, Y.; Krishnamoorthi, R.; Chandra, V. LLM-QAT: Data-Free Quantization Aware Training for Large Language Models. *arXiv* **2023**, arXiv:cs.CL/2305.17888.
22. Zhu, K.; He, Y.Y.; Wu, J. Quantized Feature Distillation for Network Quantization. *arXiv* **2023**, arXiv:cs.CV/2307.10638.
23. Sayed, R.; Azmi, H.; Shawkey, H.; Khalil, A.H.; Refky, M. A Systematic Literature Review on Binary Neural Networks. *IEEE Access* **2023**, *11*, 27546–27578. [CrossRef]
24. Yamamoto, K. Learnable Companding Quantization for Accurate Low-bit Neural Networks. In Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 5027–5036. [CrossRef]
25. Capotondi, A.; Rusci, M.; Fariselli, M.; Benini, L. CMix-NN: Mixed Low-Precision CNN Library for Memory-Constrained Edge Devices. *IEEE Trans. Circuits Syst. II Express Briefs* **2020**, *67*, 871–875. [CrossRef]
26. Latotzke, C.; Ciesielski, T.; Gemmeke, T. Design of High-Throughput Mixed-Precision CNN Accelerators on FPGA. In Proceedings of the 2022 32nd International Conference on Field-Programmable Logic and Applications (FPL), Belfast, UK, 29 August–2 September 2022; pp. 358–365. [CrossRef]
27. Nguyen, D.T.; Kim, H.; Lee, H.J. Layer-Specific Optimization for Mixed Data Flow With Mixed Precision in FPGA Design for CNN-Based Object Detectors. *IEEE Trans. Circuits Syst. Video Technol.* **2021**, *31*, 2450–2464. [CrossRef]
28. Huang, Y.; Chen, K.; Shao, Z.; Bai, Y.; Huang, Y.; Du, Y.; Du, L.; Wang, Z. LSMQ: A Layer-Wise Sensitivity-Based Mixed-Precision Quantization Method for Bit-Flexible CNN Accelerator. In Proceedings of the 2021 18th International SoC Design Conference (ISOCC), Jeju Island, Republic of Korea, 6–9 October 2021; pp. 256–257. [CrossRef]

29. Tang, C.; Ouyang, K.; Wang, Z.; Zhu, Y.; Wang, Y.; Ji, W.; Zhu, W. Mixed-Precision Neural Network Quantization via Learned Layer-wise Importance. *arXiv* **2023**, arXiv:cs.LG/2203.08368.
30. Umuroglu, Y.; Rasnayake, L.; Sjalander, M. BISMO: A Scalable Bit-Serial Matrix Multiplication Overlay for Reconfigurable Computing. *arXiv* **2018**. [CrossRef]
31. Zacchigna, F.G. Methodology for CNN Implementation in FPGA-Based Embedded Systems. *IEEE Embed. Syst. Lett.* **2023**, *15*, 85–88. [CrossRef]
32. Garland, J.; Gregg, D. Low Complexity Multiply-Accumulate Units for Convolutional Neural Networks with Weight-Sharing. *ACM Trans. Archit. Code Optim.* **2018**, *15*, 1–24. [CrossRef]
33. Zhang, X.; Ye, H.; Wang, J.; Lin, Y.; Xiong, J.; Hwu, W.M.; Chen, D. DNNExplorer: A Framework for Modeling and Exploring a Novel Paradigm of FPGA-based DNN Accelerator. In Proceedings of the 2020 IEEE/ACM International Conference on Computer Aided Design (ICCAD), Virtual, 2–5 November 2020; pp. 1–9.
34. Zhe, W.; Lin, J.; Aly, M.S.; Young, S.; Chandrasekhar, V.; Girod, B. Rate-Distortion Optimized Coding for Efficient CNN Compression. In Proceedings of the 2021 Data Compression Conference (DCC), Snowbird, UT, USA, 23–26 March 2021; pp. 253–262. [CrossRef]
35. Gajjala, R.R.; Banchhor, S.; Abdelmoniem, A.M.; Dutta, A.; Canini, M.; Kalnis, P. Huffman Coding Based Encoding Techniques for Fast Distributed Deep Learning. In Proceedings of the 1st Workshop on Distributed Machine Learning (DistributedML'20), Barcelona, Spain, 1 December 2020; Association for Computing Machinery: New York, NY, USA, 2020; pp. 21–27. [CrossRef]
36. Sharma, H.; Park, J.; Suda, N.; Lai, L.; Chau, B.; Chandra, V.; Esmaeilzadeh, H. Bit Fusion: Bit-Level Dynamically Composable Architecture for Accelerating Deep Neural Network. In Proceedings of the 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA), Los Angeles, CA, USA, 1–6 June 2018; pp. 764–775. [CrossRef]
37. Han, S.; Mao, H.; Dally, W.J. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. *arXiv* **2016**, arXiv:cs.CV/1510.00149.
38. Dupuis, E.; Novo, D.; O'Connor, I.; Bosio, A. CNN weight sharing based on a fast accuracy estimation metric. *Microelectron. Reliab.* **2021**, *122*, 114148. [CrossRef]
39. Dupuis, E.; Novo, D.; O'Connor, I.; Bosio, A. Fast Exploration of Weight Sharing Opportunities for CNN Compression. *arXiv* **2021**, arXiv:cs.LG/2102.01345.
40. Dupuis, E.; Novo, D.; O'Connor, I.; Bosio, A. A Heuristic Exploration of Retraining-free Weight-Sharing for CNN Compression. In Proceedings of the 2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC), Taipei, Taiwan, 17–20 January 2022; pp. 134–139. [CrossRef]
41. Wang, X.; Bao, A.; Yu, Q. Weight-sharing multi-stage multi-scale ensemble convolutional neural network. *Int. J. Mach. Learn. Cybern.* **2019**, *10*, 1631–1642. [CrossRef]
42. Meng, Z.; Zhao, F.; Liang, M.; Xie, W. Deep Residual Involution Network for Hyperspectral Image Classification. *Remote Sens.* **2021**, *13*, 55. [CrossRef]
43. Ouyang, K.; Hou, Y.; Zhou, S.; Zhang, Y. Convolutional Neural Network with an Elastic Matching Mechanism for Time Series Classification. *Algorithms* **2021**, *14*, 192. [CrossRef]
44. Takahashi, R.; Matsubara, T.; Uehara, K. A Novel Weight-Shared Multi-Stage CNN for Scale Robustness. *IEEE Trans. Circuits Syst. Video Technol.* **2019**, *29*, 1090–1101. [CrossRef]
45. Chavan, A.; Bamba, U.; Tiwari, R.; Gupta, D. Rescaling CNN Through Learnable Repetition of Network Parameters. In Proceedings of the 2021 IEEE International Conference on Image Processing (ICIP), Anchorage, AK, USA, 19–22 September 2021; pp. 754–758. [CrossRef]
46. Cheng, W.; Lin, I.C.; Shih, Y.Y. An Efficient Implementation of Convolutional Neural Network with CLIP-Q Quantization on FPGA. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2022**, *69*, 4093–4102. [CrossRef]
47. Chandra, M. Data Bandwidth Reduction in Deep Neural Network SoCs using History Buffer and Huffman Coding. In Proceedings of the 2018 International Conference on Computing, Power and Communication Technologies (GUCON), Greater Noida, India, 28–29 September 2018; pp. 1–3. [CrossRef]
48. Tariq, R.; Khawaja, S.G.; Akram, M.U.; Hussain, F. Reconfigurable Architecture for Real-time Decoding of Canonical Huffman Codes. In Proceedings of the 2022 2nd International Conference on Digital Futures and Transformative Technologies (ICoDT2), Rawalpindi, Pakistan, 24–26 May 2022; pp. 1–6. [CrossRef]
49. Chen, T.; Liu, H.; Shen, Q.; Yue, T.; Cao, X.; Ma, Z. DeepCoder: A deep neural network based video compression. In Proceedings of the 2017 IEEE Visual Communications and Image Processing (VCIP), St. Petersburg, FL, USA, 10–13 December 2017; pp. 1–4. [CrossRef]
50. Zheng, H.; Shen, L.; Tang, A.; Luo, Y.; Hu, H.; Du, B.; Tao, D. Learn From Model Beyond Fine-Tuning: A Survey. *arXiv* **2023**, arXiv:cs.AI/2310.08184.
51. Arnab, A.; Xiong, X.; Gritsenko, A.; Romijnders, R.; Djolonga, J.; Dehghani, M.; Sun, C.; Lučić, M.; Schmid, C. Beyond Transfer Learning: Co-finetuning for Action Localisation. *arXiv* **2022**, arXiv:cs.CV/2207.03807.
52. Tian, K.; Mitchell, E.; Yao, H.; Manning, C.D.; Finn, C. Fine-tuning Language Models for Factuality. *arXiv* **2023**, arXiv:cs.CL/2311.08401.
53. Wang, C.; Luo, Z. A Review of the Optimal Design of Neural Networks Based on FPGA. *Appl. Sci.* **2022**, *12*, 10771. [CrossRef]

54. Dupuis, E.; Novo, D.; O'Connor, I.; Bosio, A. On the Automatic Exploration of Weight Sharing for Deep Neural Network Compression. In Proceedings of the 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 9–13 March 2020; pp. 1319–1322. [CrossRef]
55. Zhou, S.; Wang, Y.; Wen, H.; He, Q.; Zou, Y. Balanced Quantization: An Effective and Efficient Approach to Quantized Neural Networks. *arXiv* **2017**, arXiv:cs.CV/1706.07145.
56. Deng, C.; Deng, Z.; Han, Y.; Jing, D.; Zhang, H. GradQuant: Low-Loss Quantization for Remote-Sensing Object Detection. *IEEE Geosci. Remote Sens. Lett.* **2023**, *20*, 1–5. [CrossRef]
57. Chen, Q.; Teng, Y.; Zhang, H.; Jiang, K.; Duan, Q.; Li, X.; Zhao, X.; Li, R. Post-Training Quantization for Longformer with Chunkwise Quantization Granularity and Optimized Percentile. In Proceedings of the 2022 7th International Conference on Computer and Communication Systems (ICCCS), Wuhan, China, 22–25 April 2022; pp. 27–31. [CrossRef]
58. Chen, L.; Lou, P. Clipping-Based Post Training 8-Bit Quantization of Convolution Neural Networks for Object Detection. *Appl. Sci.* **2022**, *12*, 12405. [CrossRef]
59. Zacchigna, F.G. NUUQ Repository. 2024. Available online: https://github.com/colorete87/nuuq (accessed on 13 April 2024).