RAVN: Reinforcement Aided Adaptive Vector Quantization of Deep Neural Networks

Anamika Jha †, Aratrik Chattopadhyay †, Mrinal Banerji †, Disha Jain †
†Mercedes-Benz Research & Development India
{anamika.jha, aratrik.chattopadhyay, mrinal.banerji, disha.jain}@mercedes-benz.com

Abstract

In the expanding field of deep learning, deploying deep neural networks (DNNs) in resource-constrained environments presents daunting challenges due to their complexity. Existing methodologies try to reduce the model complexity through the quantization of the DNNs. Adaptive quantization (AQ) is one such quantization technique for reducing model complexity. The drawbacks of current adaptive quantization techniques include limited adaptability to different datasets and models, suboptimal codebook generation, high computational complexity, and limited generalization to unseen scenarios. In contrast, we propose to address these issues through a sophisticated AQ methodology which incorporates vector quantization (VQ) of weights and Quantization-Aware Training (QAT) in tandem with reinforcement learning (RL). The above-mentioned approach facilitates dynamic allocation of quantization parameters of the DNN models, thereby reducing complexity, power utilization and ease of deployment on edge devices. We evaluated our proposed approach on three publicly available benchmark datasets namely, CIFAR-10, CIFAR-100 and ImageNet on state-of-the-art floating-point DNN architectures and showed a boost of up to 4% in their respective quantized counterparts. The source code of the proposed approach will be available here upon acceptance of the work.

1. Introduction

The field of deep learning has seen a significant expansion, with deep neural networks (DNNs) becoming central to a range of applications, from image recognition to natural language processing. Despite their success, deploying these networks in environments with limited computational resources, such as mobile devices or IoT sensors, presents significant challenges. The core issues stem from the substantial network sizes and the intensive computational demands of DNNs, which are not ideally suited for such resource-constrained platforms. Traditional approaches to mitigating these challenges primarily involve quantization techniques that convert floating-point weight variables of DNNs into lower precision formats. While effective in reducing network sizes and computational requirements, these conventional methods often result in significant accuracy loss and exhibit limited adaptability to different data sets and architectures. Furthermore, they struggle with high computational complexity and suboptimal codebook generation, leading to diminished performance in unfamiliar scenarios. In response to these limitations, adaptive quantization (AQ) has emerged as a promising solution. AQ distinguishes itself by dynamically adjusting the quantization levels to suit specific data distributions and network characteristics. This approach not only aims to preserve the original network's accuracy but also adapts to various operational contexts. However, existing AQ techniques still face challenges regarding adaptability and efficiency. Our work addresses these issues by introducing a sophisticated AQ methodology that integrates vector quantization (VQ) with reinforcement learning (RL). This novel combination allows for the dynamic allocation of quantization parameters, thereby streamlining the complexity and energy consumption of DNNs while facilitating their deployment on edge devices. By leveraging RL, our approach optimizes codebook generation and reduces computational demands, leading to better generalization across diverse datasets and scenarios. We have thoroughly evaluated our proposed method on publicly available benchmark datasets, including CIFAR-10, CIFAR-100, and ImageNet, using state-of-the-art DNN architectures. The results demonstrate significant improvements in network's compactness and performance, showcasing the potential of our approach in making DNNs more accessible for devices with limited computational capabilities. Moreover, we provide insights into the implications of our findings for future research and practical applications, highlighting the versatility and efficiency of our proposed AQ framework.

2. Related Works

The field of neural network quantization has been a focal point of research due to the increasing need to deploy deep learning networks on resource-constrained devices. This section outlines seminal works and recent advancements underpinning our proposed methodology, highlighting contributions in the realms of vector quantization, reinforcement learning in quantization, and adaptive quantization applications

2.1. Vector Quantization in Neural Networks

Vector Quantization (VQ) has long been employed in signal processing and image compression. The adaptation of VQ for neural network weights was pioneered by [6], who demonstrated significant network size reductions without commensurate losses in accuracy. Subsequent research expanded upon these findings, exploring various clustering techniques to optimize the balance between compression and performance. For instance, [7] integrated VQ with pruning and Huffman coding, achieving unprecedented compression rates. Our approach builds upon this foundation, applying VQ through K-means clustering to quantize neural network weights more efficiently.

2.2. Reinforcement Learning for network Compression

The integration of Reinforcement Learning (RL) in our approach introduces a dynamic aspect to quantization. One of the pioneers in the field to leverage RL for determining the sparsity levels in neural networks was done by [27], paving the way for more nuanced compression strategies. More recently, [5] applied RL to automate the selection of quantization parameters, demonstrating enhanced performance over static quantization schemes. Our methodology extends this paradigm by employing RL, not only to adjust quantization levels but also to dynamically assign bit-widths across different layers, an approach that has not been extensively explored in prior studies.

2.3. Adaptive Quantization

Adaptive quantization (AQ) represents a significant advancement over traditional quantization approaches by introducing mechanisms that dynamically adjust the quantization parameters during the training process. This adaptability allows the quantization process to account for the unique characteristics of each network and dataset, potentially leading to better preservation of accuracy. Techniques such as BinaryConnect[1] and Ternary Weight Networks[16] have explored binary and ternary quantizations, respectively, but with limited adaptability. Recent studies have focused on more sophisticated adaptive quantization frameworks. HAWQ[3], HAWQ-V2[4], and AdaQuant[14] introduce novel methods for adaptive quantization that consider the Hessian information of the network parameters, enabling more effective bit-width assignment without significant loss of accuracy. In contrast to the previous methods, the proposed method synergizes various elements to provide a cohesive framework which dynamically tunes quantization parameters, facilitating an optimal balance between efficiency and performance.

- Integrates Vector Quantization (VQ) and Reinforcement Learning (RL): The approach dynamically adjusts bit-widths during the Post-Training Quantization (PTQ) process, improving the trade-off between network compactness and performance.
- Preservation of network Accuracy: Despite the reduction in network size and computational demands, the approach maintains high fidelity to the original full-precision networks, thereby ensuring minimal loss in accuracy.
- Efficient Deployment on Edge Devices: Reduces complexity, power utilization, and increases ease of deployment on edge devices, making advanced AI solutions more accessible for real-world applications.
- Addresses Adaptability Issues: Overcomes the limitations of current adaptive quantization techniques such as limited adaptability to different datasets and networks, suboptimal codebook generation, high computational complexity, and limited generalization.

Through this, we aim to advance the frontier of efficient deep learning network deployment, particularly in environments where computational resources are at a premium.

3. Methods

Let us consider an image $\mathtt{I}$ which is propagated through an N layered neural network $\mathtt{\Phi} = \mathbf{\Phi}_1 \circ \mathbf{\Phi}_2 \circ \ldots \mathbf{\Phi}_N$ parameterized by floating-point weights $\mathtt{W} = \{\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_N\}$ and generates features $\mathtt{F} = \{\mathbf{F}_1, \mathbf{F}_2 \ldots \mathbf{F}_N\}$ such that the feature generated by layer $\mathbf{\Phi}_i$ is $\mathbf{F}_i = \mathbf{\Phi}_i(\mathbf{F}_{i-1})$. Next, we will elaborate on the important constituents of our proposed approach as illustrated in Fig. 1.

3.1. Vector Quantization

Traditional quantization frameworks entails a careful choice of precision of weights in a deep neural network. Neural network weight precision (bit-width) generally dictates the network size and complexity. VQ is one such approach which proposes to reduce network complexity by assigning similar precision (bit-widths) to similar weight variables in a network. This selective approach clusters weight variables in different groups and assigns each group the same bit-width instead of allocating different bit-widths to individual weight variables which leads to an increase in the size of the network. Hence, this approach allows for a more nuanced network compression thereby reducing network complexity. Our proposed approach develops on the VQ framework

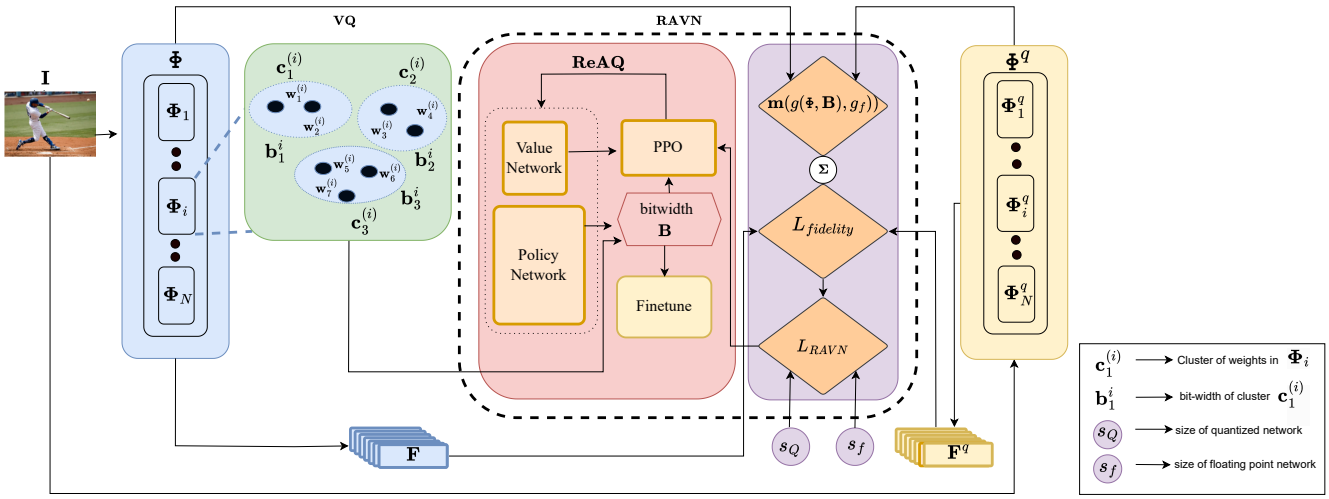

Figure 1. Overview of the proposed **RAVN** framework. The bit-widths are initialised from VQ which is passed as input to our proposed RAVN framework. The network weights of Φ are quantised via backpropagation through our proposed loss L_{RAVN} using our ReAQ optimisation framework to produce final network weights Φ^q.

as discussed below.

Let $\mathbf{W}_i = \{\mathbf{w}_1^{(i)}, \mathbf{w}_2^{(i)}, \dots \mathbf{w}_k^{(i)}\}$ be the set of weights for a given layer $\mathbf{\Phi}_i$, where $\mathbf{w}_j^{(i)} \in \mathbf{W}_i, 1 \leq j \leq k$ are individual weight variables associated with network layer $\mathbf{\Phi}_i$. In order to cluster together similar weights across N layers in Φ, we apply K-means [8] to group the weights into M clusters with cluster centroids $\mathbf{c}_m^{(i)}$, where $1 \leq m \leq M$. The clustering process can be mathematically represented as below:

$$L_C^i = \min_{\mathbf{c}_1^{(i)}, \mathbf{c}_2^{(i)}, \dots, \mathbf{c}_m^{(i)}} \sum_{m=1}^{M} \sum_{\mathbf{w}_j^{(i)} \in \mathbf{W}_i} \|\mathbf{w}_j^{(i)} - \mathbf{c}_m^{(i)}\|^2 \quad (1)$$

where L_C^i denotes the clustering loss for the layer $\mathbf{\Phi}_i$. Next, following [8] each weight $\mathbf{w}_j^{(i)}$ is assigned to the closest computed cluster centroid $\mathbf{c}_m^{(i)}$ and a corresponding bit-width value b_j^i following the rule as mentioned below. Let $\mathbf{b}_i$ be a set of values whose each element denotes the bit-widths assigned to the weight variables $\mathbf{w}_j^{(i)} \in \mathbf{W}_i$. Then the elements in $\mathbf{b}_i$ can be constructed as

$$\begin{aligned} b_j^i = b_k^i \quad & \text{if} \{\mathbf{w}_j^i, \mathbf{w}_k^i\} \in \mathbf{c}_m^i, \forall j \neq k \\ \neq b_k^i \quad & \text{if } \mathbf{w}_j^i \in \mathbf{c}_m^i \& \mathbf{w}_k^i \notin \mathbf{c}_m^i, \forall j \neq k \end{aligned} \quad (2)$$

Hence we assign equal bit-width values to the weight variables $\mathbf{w}_j^i \in W_i$ if they belong to the same cluster centroid $\mathbf{c}_m^i$, otherwise we assign different values to them. The allocation of bit-widths can be extended for the network Φ as $\mathbf{B} = \{\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_N\}$ where $\mathbf{b}_i$ denotes the bit-widths allocated to a network layer $\mathbf{\Phi}_i$. Since the assigned bit-width variables are responsible for controlling the network precision and it's performance, we will associate network precision with bit-width in the rest of the paper. The proposed dynamic bit-width allocation approach through VQ laid the stepping stones for a robust quantization framework and forms the backbone of our proposed approach.

3.2. Reinforced Adaptive Quantization (ReAQ)

The integration of the proposed bit-width aware VQ approach (refer Sec.3.1) in our framework provided a robust method of bit-width allocation between similar and dissimilar floating-point (FP) network weight variables. However a random bit-width allocation may lead to a reduction in the performance of a floating-point (FP) network and introduce quantization loss which we want to minimise. Hence, we seek for an optimal distribution of bit-widths across all the network weights in Φ which will not only minimise the quantization loss but also help us achieve a given target performance. We propose to satisfy the above constraints via our L_{ReAQ} loss which we will describe subsequently.

Let us consider a bit-width assignment $\mathbf{b}_i$ to Φ in the layer $\mathbf{\Phi}_i$. Since assignment of a new bit-width changes the weights of Φ, we refer to the changed network layer as $\mathbf{\Phi}_i^q$. The change in the networks-weights will introduce a quantization loss which we denote as $L_{quant} = f(\mathbf{\Phi}_i, \mathbf{\Phi}_i^q, \mathbf{b}_i)$ (Sec 3.3). Additionally, let $g(\mathbf{\Phi}_i, \mathbf{b}_i)$ and g_f denote the measure of performance of the network and target performance respectively whose difference we want to minimise. Since, we want to reduce our quantisation loss L_{quant} and the difference between $g(\mathbf{\Phi}_i, \mathbf{b}_i)$ and g_f, the formulation of our final loss L_{ReAQ} can be represented as:

$$L_{ReAQ} = f(\mathbf{\Phi}_i, \mathbf{\Phi}_i^q, \mathbf{b}_i) + \lambda(\mathbf{m}(g(\mathbf{\Phi}_i, \mathbf{b}_i), g_f)) \quad (3)$$

where $\mathbf{m}(.)$ is a distance metric (e.g MSE) which represents the loss incurred in network performance and λ is a trainable hyperparameter. However, the formulation of the loss function in Eq. 3 has one caveat. We can choose to maximise the network performance $g(.)$ by using higher precision network parameters(bigger elements in b_i) and still reduce L_{ReAQ}. In other words, theoretically we can reduce the loss in Eq. 3 to zero by copying the weights in $\mathbf{\Phi}_i$ to the weights in $\mathbf{\Phi}_i^q$ and maximise the network performance $g(.)$ to match g_f . However, this in turn increases the network size, thereby nullifying the benefits of network quantization. Hence, we propose to address this shortcoming by adding another constraint in the formulation of our loss function.

Let s_Q and s_f denote the sizes of the quantized and floating-point network respectively. We choose to maximise the difference between s_Q and s_f since we want the quantized network to be lesser in size compared to the floating-point counterpart. Hence the final formulation of L_{ReAQ} along with the proposed constraint can be mathematically represented as:

$$L_{ReAQ} = f(\mathbf{\Phi}_i, \mathbf{\Phi}_i^q, \mathbf{b}_i) + \lambda(\mathbf{m}(g(\mathbf{\Phi}_i, \mathbf{b}_i), g_f)) \quad \text{subject to} \quad s_Q \leq s_f \tag{4}$$

Next, we will discuss the formulation of our quantization loss.

3.3. Fidelity Loss

In this section, we explain the choice of the quantization loss $f(.)$ (mentioned in Sec 3.2) of our proposed approach. A robust formulation of a quantization loss should ensure that the semantics in the floating-point network is preserved in the quantized counterpart. Hereby, we propose to leverage the structural similarity between feature maps obtained from the respective networks as a cue to quantify the semantic difference between both which we will describe below.

Let a bit-width allocation $\mathbf{B}$ (refer Sec 3.1) to the weights in Φ inject a quantization loss $f(.)$ resulting in a quantized network Φ^q. Let the features generated from the quantized network $\Phi^q = \mathbf{\Phi}_1^q \circ \mathbf{\Phi}_2^q \circ \dots \mathbf{\Phi}_N^q$ be denoted as $\mathbf{F}^q = \{\mathbf{F}_1^q, \mathbf{F}_2^q, \dots, \mathbf{F}_N^q\}$ where feature generated from $\mathbf{\Phi}_i^q$ is $\mathbf{F}_i^q = \mathbf{\Phi}_i^q(\mathbf{F}_{i-1}^q)$. Similarly the features generated from the floating-point network Φ for layer $\mathbf{\Phi}_i$ can be represented as $\mathbf{F}_i = \mathbf{\Phi}_i(\mathbf{F}_{i-1})$ (refer to Section 3). The structural similarity between $\mathbf{F}_i$ and $\mathbf{F}_i^q$ can be measured using a distance metric $\mathbf{d}(.)$ (eg. L2) which we want to minimise. This is equivalent to minimising the loss function as mentioned below:

$$L_{fidelity}^i = \mathbf{d}(\mathbf{F}_i, \mathbf{F}_i^q) \tag{5}$$

We can extend the above formulation for N number of layers in the network which yields our proposed loss $L_{fidelity}$.

$$L_{fidelity} = \frac{\sum_{i=1}^{N} \mathbf{d}(\mathbf{F}_i, \mathbf{F}_i^q)}{N} \tag{6}$$

Integration of the proposed loss function ($L_{fidelity}$) into our method forces the network to not only maintain the performance, but also preserve the internal dynamics and representation thus ensuring a more faithful and effective quantization.

Hence substituting $L_{fidelity}$ in Eq.4 and generalising $\mathbf{b}_i$, Φ_i for the whole network our final loss can be formulated as below:

$$L_{RAVN} = L_{fidelity} + \lambda(\mathbf{m}(g(\Phi, \mathbf{B}), g_f)) \quad \text{subject to} \quad s_Q \leq s_f \tag{7}$$

where $\mathbf{B}$ denotes the bit-width assignment for the whole network Φ. It is to be noted that, L_{RAVN} integrates VQ within a formalized framework, while L_{ReAQ} is dedicated exclusively to Adaptive Quantization using Reinforcement Learning, without VQ

We propose to minimise the loss-function in Eq. 7 through an iterative optimisation framework in a reinforcement learning paradigm. In our proposed framework, the assigned reinforcement learning agent receives a reward for minimising L_{RAVN} and increasing the network performance through an imposed constraint such that s_Q is lesser than the floating-point network size s_f. In doing so, the agent ensures that not only the quantized network has competitive performance by minimising L_{RAVN} but gets penalised if the network size increases through the imposed constraint in Eq. 7. Hence, the proposed approach results in less complex and competitive quantized networks Φ^q thereby proving its efficacy as a robust quantization algorithm. The steps of the training procedure of the proposed algorithm is elucidated through a pseudocode in Algorithm. 1.

4. Experiments and Analysis

We evaluated our proposed approach RAVN against state-of-the-art floating-point(FP) and 4-bit architectures on three publicly available benchmark datasets for image classification tasks namely, CIFAR-10, CIFAR-100 and a widely chosen subset of the ImageNet dataset. We chose the widely used average precision as our accuracy metric and used it for the evaluation of classification tasks. We will refer to average precision as accuracy throughout the rest of our paper.

4.1. Datasets

The CIFAR-10 [15] dataset has 60000 color images of shape 32x32 and consists of 10 classes where each class consists of 6000 images. The CIFAR-100 [15] dataset has 50,000 training images and 10,000 test images. The dataset

Network	CIFAR-10 (%)			CIFAR-100 (%)			ImageNet (%)		
	Float	4bit	Proposed	Float	4bit	Proposed	Float	4bit	Proposed
ResNet-20[12]	90.12[11]	88.07	**89.35**	72.14[17]	70.67	**71.03**	81.15[11]	**80.78**	80.19
ResNet-50[12]	85.21[31]	81.04	**82.97**	80.93[31]	77.03	**78.39**	80.42[30]	77.31	**79.50**
ResNet-152[12]	86.22[12]	83.81	**84.39**	83.31[12]	81.02	**81.92**	80.62[9]	78.18	**78.57**
EfficientNet[26]	89.52[22]	**88.14**	87.64	73.82[23]	71.11	**72.84**	77.01[29]	**76.42**	76.39
MobileNet[13]	89.12[24]	87.81	**88.22**	70.73[25]	69.06	**69.98**	75.20[28]	71.03	**73.97**
ResNet-200[10]	83.12[10]	82.17	**82.79**	77.79[10]	75.92	**76.98**	83.83[28]	82.08	**82.09**

Table 1. Performance comparison of different architectures across datasets for varied precision levels. Results in bold indicate better performance.

Algorithm 1 Training of the Proposed RAVN approach

Input:
- Input image ($\mathtt{I}$)
- Floating-point Network weights ($\mathbf{W}_i$)
- Floating-point Feature Maps ($\mathbf{F}_i$)
- Floating-point network size (s_f)
- Number of layers in Floating-point network (N)
- Floating-point network performance (g_f)
- Initial random bit-width assigned $\mathbf{B} = \{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_N\}$

Output: Quantized network weights (Φ^q)

while not converged **do**
  while $i < N$ **do**
    Assign $\mathbf{b}_i$ to $\mathbf{\Phi}_i$ following Eq. 2
    and produce $\mathbf{\Phi}_i^q$
    $\mathbf{F}_i^q \leftarrow \mathbf{\Phi}_i^q(\mathtt{I})$
    $L_{fidelity}^i \leftarrow \mathbf{d}(\mathbf{F}_i, \mathbf{F}_i^q)$
    $r^i \leftarrow g(\mathbf{\Phi}_i^q, \mathbf{b}_i)$
    $L_{RAVN}^i \leftarrow L_{fidelity}^i + r^i$ subject to $s_Q \leq s_f$
  end while
  Compute L_{RAVN} following Eq. 7
  Update weights Φ^q and bit-widths $\mathbf{B}$
end while

consists of a wide variety of images from 100 classes where each class consists of 600 images including 500 training images and 100 test images. Additionally, the dataset also provides 20 superclass tags grouped from the 100 fine classes. The images present in the dataset have a shape of 32×32 and similar to CIFAR-10 have three channels representing color dimensions. The chosen ImageNet dataset [2] contains around 50,000 training images, and 10,000 validation images annotated following the WordNet hierarchy. The dataset is used as a benchmark in image classification and object detection tasks and consists of a wide variety of objects from 100 diverse classes with manually annotated object bounding boxes for each image.

4.2. Implementation details

The implementation of the proposed approach uses the popular Pytorch framework [20]. We used a training batch size of 64 and learning rate is set at 1e-3 using a cosine scheduler [18] with a 0.1 annealing factor. We minimize our proposed loss functions using the Adam-W optimizer [19], which enhances generalization through effective regularization by incorporating a weight decay of 5e-4. We chose our training hyperparameter λ as 1 to maintain a consistent scale across various tests, ensuring comparability of results. The bit-widths $\mathbf{B}$ are initialized randomly in the range [2,8]. The scale factor s_f is set to match the floating-point accuracy targets, optimizing the conversion from floating-point to fixed-point formats while preserving computational precision. Finally, we quantize the floating-point (FP) architectures iteratively using the proximal-policy optimization algorithm (PPO) [21] through our novel RAVN framework, which supports robust and efficient optimization strategies.

4.3. Results

In this work, we introduce a novel quantization technique for deep neural networks aimed at optimizing both computational efficiency and environmental impact, without sacrificing the accuracy of the networks. The objective of the proposed method is not to compare the accuracies across different architectures; instead, it aims to evaluate the performance differences between floating-point networks and their corresponding quantized counterparts. Our proposed method is evaluated across three benchmark datasets namely, CIFAR-10, CIFAR-100, and ImageNet, and compared with traditional 32-bit floating point (Float) and 4-bit quantized networks through average precision metric as shown in Table 1. We will refer to the average precision metric as accuracy in the rest of our paper. Additionally, we also summarise the average bit-width consumed for each of the architectures quantized through our framework in Table 3. Since the average bit-width consumed in our framework is approximately 4.5, we chose the networks quantized with bit-width of 4 as our closest baseline to demonstrate the ef-

ficacy of the proposed method. Moreover, we compare the efficacy of the proposed method in reducing network complexity and carbon emission through Table. 2 and Table. 4 respectively.

4.3.1 Comparison with 4-bit

Our proposed method demonstrates significant improvements in accuracy compared to the conventional 4-bit quantized versions across all the mentioned networks and datasets. For instance, on the CIFAR-10 dataset, our method achieves a 1.45% increase in accuracy for ResNet-20 over the 4-bit counterpart. Notably, on the more challenging ImageNet dataset, our approach enhances the performance of ResNet-50 by a margin of 2.83% compared to its 4-bit counterpart. On both the networks we are able to achieve and average performance gain of up to 2.28% with a 0.4 increase in bit-width. This translates to a minor increase of 2.5MB in network size while using only an additional 0.1kg CO2 footprint. Moreover, the proposed method is capable of increasing the performance of deeper neural network architectures like ResNet-152 by a margin of 1.11% on the CIFAR-100 dataset while consuming an additional 6MB network size and 0.2kg CO2 footprint. We show a similar improvement in performance on the 4-bit counterpart of the ResNet-200 architecture with our proposed method outperforming by a margin of up to 1.4% on the CIFAR-100 dataset while keeping the carbon-footprint levels under check. These enhancements in accuracy, particularly for complex datasets, assures that the proposed quantization approach not only maintains the integrity of the predictive power of the network but also keeps the network complexity and environmental impact at a check.

4.3.2 Comparison with Float (32-bit)

We also compare the performance of the proposed method with floating-point counterparts on different architectures as depicted in Table. 1. However, since the performance of the quantized networks can be as good as their floating-point counterpart, we will focus on the reduction achieved in network complexity and carbon-footprint as our baseline for comparison. Our proposed method reduces the size of the ResNet-20 architecture from 52 MB to 22 MB, a staggering 136% reduction along with an 89% reduction in carbon footprint. However, the proposed approach still manages to achieve almost similar performance on the CIFAR-100 and ImageNet datasets showing an average minor performance drop of 0.5% over both the datasets. This highlights the capability of the proposed method in maintaining the performance of the floating-point architectures with minimal loss but significantly reducing computational burden and complexity through network size reduction. In ResNet-200, our method manages to reduce the network size from 230MB to 58 MB showing a 75% reduction in size and generating 85% less carbon footprint. However, the proposed method still maintains the accuracy loss to a minimal value of 1% even for complex datasets like ImageNet. This trend of modest decrease in performance for substantial gains in network size reduction and carbon footprint is consistent across all evaluated architectures, indicating the practicality of our method in resource-constrained environments.

Network	Float (32bit) (MB)	4-bit (MB)	Proposed (MB)
ResNet-20	52	19	22
ResNet-50	110	28	30
ResNet-152	185	48	54
EfficientNet	150	29	32
MobileNet	125	24.5	28
ResNet-200	230	37	58

Table 2. Network sizes for various bit-widths.

Additionally, we also extend our evaluation in terms of the computational resources, where we observe significant reductions in RAM usage on the NVIDIA A100 as shown in Table 5. Notably, in ResNet-50 architecture the RAM usage shows a considerable decrease of 82.5% from 12 GB to 2.1GB. These reductions in memory requirements are crucial for deploying advanced neural network networks on edge devices and in data centres, where memory is often a limiting factor.

In summary, our proposed quantization method provides a compelling balance between accuracy, storage efficiency, environmental sustainability, and computational resource utilization. It presents a significant step forward in the development of efficient and environmentally friendly neural network quantization techniques. The strategic trade-offs between a slight increase in network size and bit-width from 4-bit to our proposed method result in substantial performance boosts and efficiency gains, making it an optimal solution for real-world applications that demand high performance with computational and environmental efficiency

Network	Average Bit-Width
ResNet-20	4.5
ResNet-50	4.3
ResNet-152	4.4
EfficientNet	4.5
MobileNet	4.3
ResNet-200	4.4

Table 3. Average Bit-Width of different quantized architectures using the proposed RAVN framework

Network	Float (32-bit) (kg CO2)	4-bit (kg CO2)	Proposed (kg CO2)
ResNet-20	2.1	0.24	0.23
ResNet-50	7.3	1.0	1.2
ResNet-152	4.2	1.0	1.2
EfficientNet	1.0	0.08	0.1
MobileNet	0.5	0.11	0.11
ResNet-200	8.3	1.2	1.3

Table 4. Carbon Footprint emission of different architectures for different bit-widths

Network	Float (32-bit) (GB)	4-bit (GB)	Proposed (GB)
ResNet-20	5.2	1.4	1.9
ResNet-50	12	2.0	2.1
ResNet-152	13.8	3.25	3.28
EfficientNet	7	1.22	1.22
MobileNet	3	0.57	0.58
ResNet-200	20	4.2	4.6

Table 5. RAM usage of different networks with varying bit-widths on NVIDIA A100.

Method	Dataset	Accuracy(%)	Network Size
VQ	CIFAR-10	78.32	45
AQ	CIFAR-10	82.86	42
RL+AQ	CIFAR-10	81.25	37
Proposed	CIFAR-10	**83.97**	**30**
VQ	CIFAR-100	76.41	47
AQ	CIFAR-100	80.44	41
RL+AQ	CIFAR-100	79.85	35
Proposed	CIFAR-100	**81.39**	**34**
VQ	ImageNet	74.51	44
AQ	ImageNet	77.0	47
RL+AQ	ImageNet	77.91	**37**
Proposed	ImageNet	**79.5**	38

Table 6. Ablation Studies

5. Ablation Studies

In this section, we delve into an ablation study conducted to evaluate the individual contributions of the main components in our RAVN framework namely, Vector Quantization (VQ), Reinforcement Learning (RL), and Adaptive Quantization (AQ) with ResNet-50 as our baseline choice of network architecture. The purpose of this study is to understand the impact of each component of the proposed framework on the overall performance of the quantized deep neural network networks across different datasets and used Average Precision (AP) as our metric for comparison. The comparative analysis of the proposed framework is depicted in Table 6.

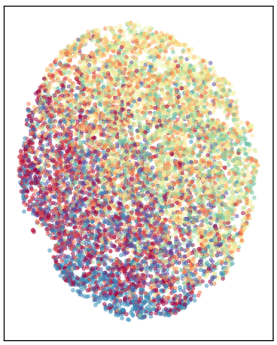

Figure 2. T-SNE plot of ResNet-50 weights before applying VQ. Different colours indicate different clusters

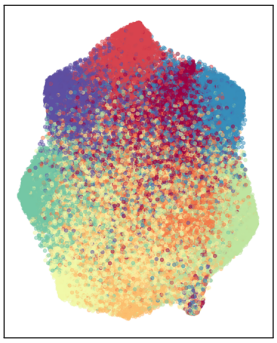

Figure 3. T-SNE plot of ResNet-50 weights after applying VQ. Different colours indicate different clusters

5.1. Vector Quantization(VQ)

Initially, we isolated the vector quantization component by disabling the RL and AQ elements from our proposed approach. This modification allows us to assess the effectiveness of VQ in reducing network size while maintaining network performance. On CIFAR-10, CIFAR-100, and ImageNet, neural networks quantized using VQ achieved accuracies of 78.32%, 76.41%, and 74.51% respectively. The efficacy of the VQ component in our framework is also elaborated through the T-SNE plots of weight variables for ResNet-50 architecture. While, it can be seen from Fig. 2 that the weight parameter values are scattered throughout, the ResNet-50 weight variables quantized through VQ as shown in Fig. 3 demonstrate better clustering, emphasizing the crucial role of VQ in our proposed approach. Hence, we chose VQ as our baseline setting for the comparison with the rest of the components of our proposed approach.

5.2. Adaptive Quantization (AQ)

In this configuration, we examined the role of AQ in a quantization framework by removing the VQ and RL components of our proposed approach. The specific component showed a remarkable boost in network performance by at most 5% in the CIFAR-10 dataset and 2% in the CIFAR-100 dataset respectively. Moreover, the network demonstrated a staggering boost of 8% in network performance compared to the standard RL setting achieving an accuracy of 77% on the ImageNet dataset. The general boost in performance achieved through the adoption of an adaptive quantization technique necessitates the requirement of a similar strategy in quantization frameworks.

5.3. RL+AQ

RL+AQ offers a refined perspective on the collaborative effectiveness of combining the two components of our proposed approach, namely RL and AQ. By applying this joint strategy, we attained significant network accuracies of 82.08% on CIFAR-10, 79.86% on CIFAR-100, and an enhanced 77.91% on the ImageNet dataset. The synergy between RL and AQ not only achieved a notable improvement in accuracy by 1% on the ImageNet dataset when compared with AQ setting, but also succeeded in dynamically quantizing the network, thus reducing the network size by 19% across all datasets. It is to be noted that there are minimal performance variations of up to 1% as observed in the CIFAR-10 and the CIFAR-100 datasets. However, the massive reduction in network size underscores the advantage of integrating RL with AQ through the RL+AQ component of our proposed approach. Additionally, this also leaves room for analysing the potential for Vector Quantization (VQ) to further enhance our approach through complementary synergy, thereby suggesting promising avenues for future enhancements.

5.4. Full RAVN Framework

The proposed framework integrates Vector Quantization (VQ), Reinforcement Learning (RL), and Adaptive Quantization (AQ), achieving substantial improvements in accuracy and network size. This configuration yielded the highest accuracies among all variants tested: 83.97% on CIFAR-10, 81.39% on CIFAR-100, and 79.5% on ImageNet. The incorporation of the VQ component alone enhanced performance by an average of 2%. Moreover, this framework significantly reduced network sizes by an average of 13% across these datasets, illustrating its effectiveness in both preserving accuracy and minimizing network footprint. These outcomes highlight the benefits of combining VQ with RL-guided AQ, demonstrating considerable size reductions while maintaining accuracy, particularly for complex datasets.

Finally, our ablation study delineates that each component of the proposed RAVN framework plays a vital role in enhancing the efficiency and effectiveness of adaptive quantization for deep neural networks. The integration of these components allows for navigating the trade-offs between network size, computational cost, and performance, particularly crucial in the context of deploying deep learning networks in resource-constrained environments.

6. Discussions

The RAVN framework combines vector quantization with reinforcement learning for adaptive DNN weight quantization, enhancing layer-specific and overall network efficiency without significantly sacrificing accuracy, as shown in tests on CIFAR-10, CIFAR-100, and ImageNet. This approach facilitates DNN deployment in resource-limited settings like edge devices and mobile phones by reducing model size while retaining high accuracy. Our methodology, while promising, has few limitations. The use of reinforcement learning can extend the training duration. Success depends on the initial model quality, the reinforcement learning algorithm, and data diversity.

7. Future Works

Future research directions for the RAVN includes enhancing the clustering algorithms within the AQ framework, incorporating additional loss metrics to minimize quantization loss, and broadening its application to domains beyond image classification, such as natural language processing, speech recognition, and time-series analysis. RAVN's principles could significantly impact the deployment of efficient and high-performing deep learning models on resource-limited edge devices. The proposed method can be further adapted for hardware-embedded systems that support dynamic mixed precision, even at lower bit-widths. This adaptation is particularly beneficial for implementations on edge devices, where efficiency and compactness are crucial.

8. Conclusions

This work introduces RAVN, a novel framework merging vector quantization (VQ) and reinforcement learning (RL) for adaptive bit-width adjustment in Post-Training Quantization (PTQ), enhancing neural network quantization by reducing size and computational needs while maintaining accuracy. RAVN's RL-based dynamic quantization marks a significant shift from static methods, optimizing codebook generation and reducing complexity for improved performance and applicability. RAVN is a considerable advancement in adaptive quantization, offering a solution for deep neural network deployment in resource-limited settings and laying groundwork for future efficient model quantization research.

References

[1] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. *Advances in neural information processing systems*, 28, 2015. 2

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 5

[3] Zhen Dong, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 293–302, 2019. 2

[4] Zhen Dong, Zhewei Yao, Daiyaan Arfeen, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq-v2: Hessian aware trace-weighted quantization of neural networks. *Advances in neural information processing systems*, 33:18518–18529, 2020. 2

[5] Ahmed T Elthakeb, Prannoy Pilligundla, Fatemehsadat Mireshghallah, Amir Yazdanbakhsh, and Hadi Esmaeilzadeh. Releq: A reinforcement learning approach for automatic deep quantization of neural networks. *IEEE micro*, 40(5):37–45, 2020. 2

[6] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*, 2014. 2

[7] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015. 2

[8] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108, 1979. 3

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 630–645. Springer, 2016. 5

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5

[13] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 5

[14] Itay Hubara, Yury Nahshan, Yair Hanani, Ron Banner, and Daniel Soudry. Accurate post training quantization with small calibration sets. In *International Conference on Machine Learning*, pages 4466–4475. PMLR, 2021. 2

[15] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 4

[16] Fengfu Li, Bin Liu, Xiaoxing Wang, Bo Zhang, and Junchi Yan. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016. 2

[17] Wei Li. Cifar-zoo: Pytorch implementation of cnns for cifar dataset. https://github.com/BIGBALLON/CIFAR-ZOO, 2019. 5

[18] I Loshchilov and F Hutter. Stochastic gradient descent with warm restarts. In *Proceedings of the 5th Int. Conf. Learning Representations*, pages 1–16. 5

[19] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. 2018. 5

[20] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 5

[21] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 5

[22] Joao Paulo Schwarz Schuler, Santiago Romaní, Mohamed Abdel-nasser, Hatem Rashwan, and Domenec Puig. Grouped pointwise convolutions reduce parameters in convolutional neural networks. *Mendel*, 28:23–31, 2022. 5

[23] Joao Paulo Schwarz Schuler, Santiago Romaní, Mohamed Abdel-nasser, Hatem Rashwan, and Domenec Puig. Grouped pointwise convolutions reduce parameters in convolutional neural networks. *Mendel*, 28:23–31, 2022. 5

[24] Joao Paulo Schwarz Schuler, Santiago Romaní, Mohamed Abdel-nasser, Hatem Rashwan, and Domenec Puig. Grouped pointwise convolutions reduce parameters in convolutional neural networks. *Mendel*, 28:23–31, 2022. 5

[25] Joao Paulo Schwarz Schuler, Santiago Romaní, Mohamed Abdel-nasser, Hatem Rashwan, and Domenec Puig. Grouped pointwise convolutions reduce parameters in convolutional neural networks. *Mendel*, 28:23–31, 2022. 5

[26] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019. 5

[27] Karen Ullrich, Edward Meeds, and Max Welling. Soft weight-sharing for neural network compression. *arXiv preprint arXiv:1702.04008*, 2017. 2

[28] Ross Wightman. Pytorch image models. https://github.com/huggingface/pytorch-image-models, 2019. 5

[29] Ross Wightman, Hugo Touvron, and Hervé Jégou. Resnet strikes back: An improved training procedure in timm. *arXiv preprint arXiv:2110.00476*, 2021. 5

[30] Ross Wightman, Hugo Touvron, and Hervé Jégou. Resnet strikes back: An improved training procedure in timm. *arXiv preprint arXiv:2110.00476*, 2021. 5

[31] yukiyan. resnet-50-finetuned-eurosat. https://huggingface.co/yukiyan/resnet-50-finetuned-eurosat, 2024. 5