CRVQ: Channel-Relaxed Vector Quantization for Extreme Compression of LLMs

Yuzhuang Xu Shiyu Ji Qingfu Zhu Wanxiang Che*
Research Center for Social Computing and Interactive Robotics (SCIR)
Harbin Institute of Technology, Harbin, China
{xyz,car}@ir.hit.edu.cn

Abstract

Powerful large language models (LLMs) are increasingly expected to be deployed with lower computational costs, enabling their capabilities on resource-constrained devices. Post-training quantization (PTQ) has emerged as a star approach to achieve this ambition, with best methods compressing weights to less than 2 bit on average. In this paper, we propose **C**hannel-**R**elaxed **V**ector **Q**uantization (CRVQ), a novel technique that significantly improves the performance of PTQ baselines at the cost of only minimal additional bits. This state-of-the-art extreme compression method achieves its results through two key innovations: (1) carefully selecting and reordering a very small subset of critical weight channels, and (2) leveraging extended codebooks to relax the constraint of critical channels. With our method, we demonstrate a 38.9% improvement over the current strongest sub-2-bit PTQ baseline, enabling nearer lossless 1-bit compression. Furthermore, our approach offers flexible customization of quantization bit-width and performance, providing a wider range of deployment options for diverse hardware platforms. Code and checkpoints are available at https://github.com/xuyuzhuang11/CRVQ.

1 Introduction

Large language models (LLMs) have made remarkable strides, with open-source models like LLaMA (Touvron et al., 2023a,b), Phi (Abdin et al., 2024), and Qwen (Bai et al., 2023; Yang et al., 2024) achieving exceptional performance. However, their massive sizes present challenges for deployment and usability. For instance, the widely adopted and powerful LLaMA2-70B model requires over 140GB when loaded in FP16 format, which makes application on most single-GPUs infeasible. This computational and memory overhead becomes even more prohibitive for resource-constrained devices. As the demand for edge-device LLM inference grows, especially on mobile platforms (Hu et al., 2024; Abdin et al., 2024), researchers have turned to compression techniques (Frantar et al., 2022; Shao et al., 2024; Xu et al., 2024) to enable efficient deployment without compromising performance noticeably.

Post-Training Quantization (PTQ) efficiently compresses LLMs by converting pre-trained models to lower-bit formats with relatively few computations (Frantar et al., 2022; Tseng et al., 2024; Huang et al., 2024). Its effectiveness has driven broad research and practical adoption. Notably, recent advances achieve 3-bit compression while astonishingly maintaining lossless performance (Tseng et al., 2024). Unfortunately, the performance of powerful PTQ methods rapidly deteriorates in extremely low bit-widths scenarios. For instance, GPTQ (Frantar et al., 2022), the most popular method for INT4 quantization, completely fails when weights are compressed to 2-bit. A possible reason is that these methods treat each weight as an isolated and uniformly distributed element, independently quantizing them while ignoring the inherent patterns among weights (Egiazarian et al., 2024; Tseng et al., 2024). This oversight results in suboptimal bit utilization, causing these methods to quickly reach their performance limits.

Recently, researchers have turned to vector quantization (VQ) to address this challenge in PTQ (Chee et al., 2023; Tseng et al., 2024; Egiazarian et al., 2024). By treating weights as vectors rather than isolated scalars (see Figure 1a and 1b), VQ-based approaches appear to show promising results. For example, AQLM (Egiazarian et al., 2024) enables seamless adoption of 1-bit settings while surpassing previous strong baselines (Yuan et al., 2024; Huang et al., 2024). However, noticeable performance degradation of VQ-based methods remains unsatisfactory at 1-bit level.

Inspired by LLM.int8() (Dettmers et al., 2022) and SpQR (Dettmers et al., 2024), we propose

*Corresponding author.

Transactions of the Association for Computational Linguistics, vol. 13, pp. 1488–1506, 2025. https://doi.org/10.1162/TACL.a.45
Action Editor: Taro Watanabe. Submission batch: 2/2025; Revision batch: 6/2025; Published 11/2025.

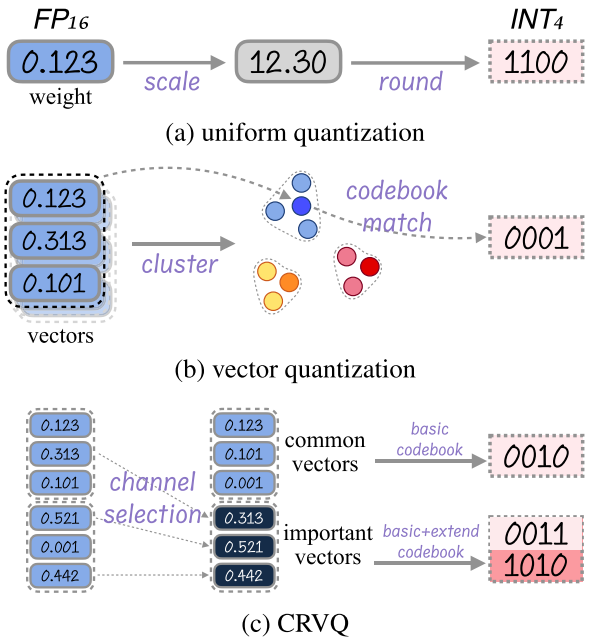


Figure 1: Comparison of different quantization methods. Uniform quantization treats each weight to be quantized as a scalar, whereas vector quantization considers weight segments as vectors. Our proposed CRVQ introduces distinct importance across weight channels, with the more critical channels selected and highlighted in dark colors. We use additional extended codebooks to quantize the vectors formed by these critical channels.

Channel-**R**elaxed **V**ector **Q**uantization (CRVQ) in this paper, aiming to treat different weight channels with varying importance in VQ for better quantization. Traditional VQ generally assumes equal importance across all weight channels (Tseng et al., 2024; Egiazarian et al., 2024). In contrast, we argue that even in VQ, a small subset of critical channels can play a pivotal role in maintaining model performance. CRVQ enables these critical channels to break the limits of the current quantization level, see the comparison in Figure 1. This involves two simple yet effective steps: First, we explore useful criteria for evaluating the importance of channels and reorder them, grouping channels of similar importance closer. Then, a few critical channels are selected and quantized with additional codebooks to enhance their representation. CRVQ significantly improves 1-bit PTQ, reducing perplexity by 38.9% and boosting zero-shot accuracy by 12.9% on LLaMA2-7B, with only a negligible bit-width cost. In summary, our contributions are 3-fold:

- We propose CRVQ, a novel and effective 1-bit PTQ method that selects critical weight channels and applies extended codebook fitting. Punching above its weight, it excels in extreme compression.
- Based on an in-depth analysis of the factors influencing performance, we highlight that CRVQ is a more efficient and flexible solution for achieving superior quantization performance on resource-constrained devices compared to traditional VQ.
- Extensive experiments demonstrate that our approach performs well in different models of varying sizes, ranging from 1.3B to 13B. On LLaMA2-7B, it reduces perplexity by 39% over AQLM with only a 0.06-bit overhead, showcasing its effectiveness.

2 Related Work

Similar to model pruning (Frantar and Alistarh, 2023; Sun et al., 2024) and knowledge distillation (Agarwal et al., 2024; Hsieh et al., 2023), model quantization is a key technique for compressing LLMs and reducing computational overhead. It can be classified into quantization-aware training (QAT) and post-training quantization (PTQ) based on its integrating stage.

QAT integrates quantization into the intermediate computations of model training, reducing performance degradation caused by low precision and leading superior results (Liu et al., 2024; Dettmers et al., 2023; Chen et al., 2024; Xu et al., 2024; Jo et al., 2024). While effective for extreme quantization, it still suffers from certain performance limitations and high computational costs. As this paper primarily focuses on PTQ, the remainder of this section will concentrate on the features and representative methods of PTQ.

PTQ transforms LLMs into their low-bit counterparts using solvers and limited calibration data, without requiring extensive training. On the one hand, PTQ has made remarkable progress in extreme compression scenarios. GPTQ (Frantar et al., 2022) iteratively quantizes columns while adjusting other weights to preserve accuracy, achieving 4-bit quantization. QuIP# (Tseng et al., 2024) employs compact E8P codebooks and incoherence processing to enable high-precision 2-bit VQ. Similarly, AQLM (Egiazarian et al., 2024) leverages learnable codebooks that can effectively adapt to the 1-bit setting. Additionally, BiLLM (Huang et al., 2024) achieves binarization by

combining weight selection and residual approximation. On the other hand, the idea of mixed precision has been widely applied in quantization research. LLM.int8() (Dettmers et al., 2022) is an early approach that proposed mixed-precision decomposition to preserve the functionality of critical channels. Other notable works employing special handling of outlier channels include SmoothQuant (Xiao et al., 2023), AWQ (Lin et al., 2024), SpQR (Dettmers et al., 2024), etc. Our work pushes the boundaries of 1-bit PTQ by enhancing VQ. Unlike existing methods, we are the first to explore the varying importance of channels within VQ, even though these channels are not adjacent. This novel perspective sets our approach apart from prior work.

3 Background

3.1 Vector Quantization

VQ partitions the weight matrix into vectors and uses the vector as the basic quantization unit. Let $\mathcal{S} = \{\mathbf{w}_i \in \mathbb{R}^d\}$ represent the set of d-dimensional vectors obtained by partitioning $\mathbf{W}^{M\times N}$.

$$\mathcal{S} = \{\mathbf{W}_{i,j:j+d} \mid i \in [\,0, M\,), j \in [\,0, N\,)\,, \; j \bmod d = 0\,\}. \quad (1)$$

VQ applies the K-means algorithm to return the cluster centers $\mathcal{C} = \{\mathbf{c}_i \in \mathbb{R}^d\}$, which are subsequently used as the codebook:

$$\mathcal{C} = K\text{-means}(\mathcal{S}). \quad (2)$$

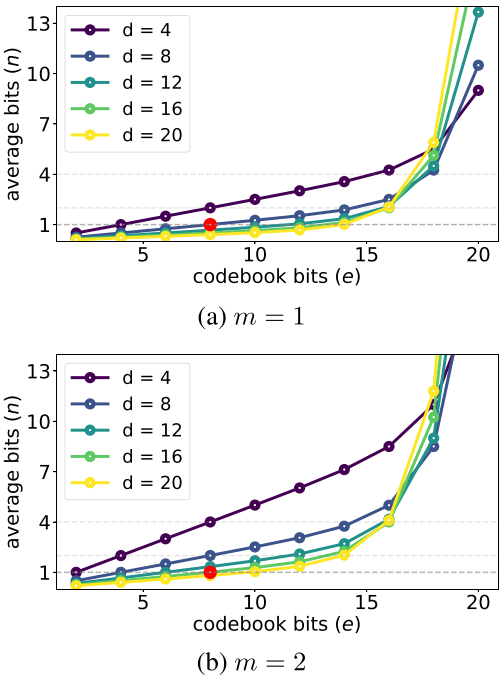
If each vector in $\mathcal{C}$ is encoded using e-bit binary number, the codebook $\mathcal{C}$ contains 2^e vectors, i.e. $|\mathcal{C}| = 2^e$. After the codebook is built, each vector in $\mathcal{S}$ is replaced with its closest vector from $\mathcal{C}$.

$$\mathcal{S} \approx \left\{\hat{\mathbf{w}}_i = \arg\min_{\mathbf{c}_j \in \mathcal{C}} \|\mathbf{w}_i - \mathbf{c}_j\|_2\right\}. \quad (3)$$

Now, the original d-dimensional vector $\mathbf{w}$ is encoded into an e-bit binary code q as follows:

$$q_i = \arg\min_j \|\mathbf{w}_i - \mathbf{c}_j\|_2, \quad q_i \in 0, 1, \ldots, 2^e - 1. \quad (4)$$

Naturally, the capacity of the codebook increases with e, leading to better approximation of $\mathbf{W}$. To reduce approximation error, we can iteratively refine the process above by quantizing the residual $\Delta\mathbf{W} = \mathbf{W} - \hat{\mathbf{W}}$, introducing more codebooks and codes. Beam search is typically used for the optimal multi-step encoding (Egiazarian et al., 2024).



(a) $m = 1$

(b) $m = 2$

Figure 2: Visualization of Equation 6. The curves represent bit-width based on the vector dimension d and the codebook bit-width e. The top (a) and bottom (b) plots correspond to $m = 1$ and $m = 2$, respectively.

To further mitigate the model sensitivity to extreme compression, block or end-to-end fine-tuning a few parameters is considered highly beneficial (Tseng et al., 2024; Egiazarian et al., 2024).

$$\mathcal{L} = \|\text{block}(\mathbf{X}) - \text{quantized\_block}(\mathbf{X})\|_2. \quad (5)$$

Let $\mathcal{L}$ be the loss and $\mathbf{X}$ the block input here. While end-to-end fine-tuning slows PTQ, it remains more efficient than most QAT methods.

3.2 From AQLM to 1-bit Quantization

Compared to QuIP#, AQLM is more adaptable for the 1-bit quantization level, as it avoids the challenge of relying on predefined dense codebooks. Let the number of codebooks be m, the average bit-width n of a quantized 4096×4096 matrix can be calculated as follows (see Section A.1 for details):

$$n = \frac{me}{d} + 2^{e-20}md. \quad (6)$$

1-bit quantized models can be built by selecting different combinations of hyper parameters. Figure 2 illustrates the bit-width curve for

various e, d corresponding to two different codebook number m, respectively. It can be observed that when the codebook bit-width e exceeds 14, the average bit-width rises sharply. It is caused by the excessive codebook storage overhead, reducing the proportion of bit-width allocated to weights. Consequently, larger values of e are avoided.

We adopt a commonly used $e = 8$, which limits the feasible settings for 1-bit quantization to $m = 1, d = 8$ or $m = 2, d = 16$. In this work, we choose the configuration with a smaller bit-width, $m = 1, d = 8, e = 8$. As we will demonstrate in Section 6.4, CRVQ also performs as expected under the $m = 2, d = 16$ setting.

4 Approach

This section shows our design for channel-relaxed quantization. We start with a discussion of how to measure the channel importance in Section 4.1 and then demonstrate how to adapt different importance into VQ in Section 4.2 and 4.3. Finally, we demonstrate the overall algorithm in Section 4.4.

4.1 Measuring Channel Importance

It is widely recognized in recent years that different weight channels exhibit varying levels of sensitivity (Yao et al., 2022; Xiao et al., 2023; Dettmers et al., 2024). Consequently, the contributions of these channels in the quantization process are not uniform. A considerable number of modern PTQ studies (Dettmers et al., 2024, 2022; Xiao et al., 2023) leverage this insight by adopting different treatment of weights, leading to improved results. However, the criteria for measuring weight sensitivity vary across different works.

The Hessian metric serves as a widely used benchmark for evaluating weight sensitivity. Inspired by SpQR (Dettmers et al., 2024), we first employ an importance criterion that jointly considers quantization error and activation magnitude, with the latter derived from the Hessian proxy. Consequently, the importance I_i of a weight channel $\mathbf{W}_{:,i}$ can be formulated as:

$$I_i = \max_j \frac{\left[w_{ji} - \mathrm{VQ}\left(w_{ji}\right)\right]^2}{2\left[\mathbf{X}\mathbf{X}^{\mathrm{T}}\right]^{-1}_{ii}}, \tag{7}$$

where $\mathbf{X}$ denotes as the activation that multiply weight $\mathbf{W}$ to be quantized, see Section A.2 for details. When calculating the importance I_i, the quantization error is first obtained by pre-quantizing the weights using VQ and then multiplying activation.

Furthermore, some studies (Yao et al., 2022; Huang et al., 2024) also observe that weights exhibit row-wise salients, leading to varying sensitivity to quantization. Intuitively, the presence of salient weights may pose greater challenges for effective quantization. Thus, channel importance may be defined as $I_i = \max_j w_{ji}^2$ as well.

The optimal selection in CRVQ will be discussed in Section 6.3. Once the metric I_i for each weight channel is computed, the importance of each channel can be determined.

4.2 Channel Reordering

Although the importance of all channels is computed, we are not yet able to handle the more critical channels independently by VQ framework. This is because the distribution of important channels is highly sparse, and their neighboring channels are often not critical. In other words, critical channels are not inherently contiguous. To break this obstacle, we reorder all channels based on their importance. As shown in the left of Figure 3, the scattered distribution of important channels (index 1, 3, and 5) in the weight matrix is reorganized such that all important channels become adjacent (in the left vector group).

4.3 Extended Codebook Fitting

After channel reordering, all channels are grouped column-wise, with each group containing many d-dimensional vectors. We define the important channel ratio λ and divide all groups into two types, important and non-important. Now, all vectors within a group belong to either important or non-important channels. This grouping allows for special treatment of the important channels.

There are two primary paths to enhance quantization precision, using more codebooks or employing codebooks with larger capacity. However, increasing codebook capacity also increases the encoding bit-width e, which in turn inflates the code for non-critical vectors. Therefore, we opt to use additional codebooks instead of expanding the capacity of existing ones. In CRVQ, we designate the codebooks applied to all vector groups as **basic codebooks** and those specifically used for critical vector groups as **extended codebooks**. As shown in the middle of Figure 3, codebook 1 is the basic codebook, while codebooks 2 to 4 are extended

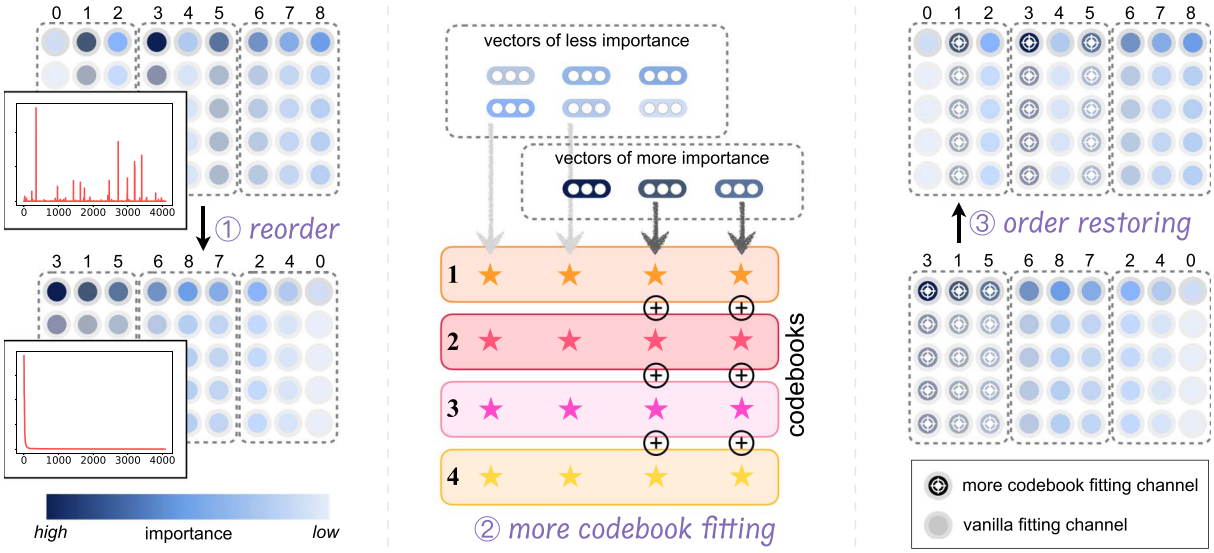


Figure 3: Illustration of the proposed CRVQ, consisting of three phases. On the left, channels are reordered by importance, with circle shading indicating weight significance. Next, vectors are fitted to codebooks, where critical weights are represented as the sum of corresponding vectors from all codebooks (1 to 4). The right sub-figure shows channels reordered to their original sequence.

codebooks. We first perform quantization on all groups using the basic codebook. Next, we supplement the quantization of critical vector groups by fitting them with the extended codebooks using additive VQ. During inference, the weights are restored to their original order through a reverse reordering process. In this paper, we use notation like ''$8 + 8 \times 3$'' to denote the codebook configuration, where one basic codebook and three extended codebooks are used.

4.4 Overall Algorithm

Now we demonstrate our CRVQ with the help of Algorithm 1. Similar to most PTQ methods, the quantization process is performed layer-by-layer. For each weight matrix $\mathbf{W}$ in a specific layer, we first compute the column-wise channel importance and reorder these channels (Line $3 \sim 5$). Then, we construct the basic codebook $\mathcal{C}_{\text{base}}$ and search the binary encoding $\mathcal{B}_{\text{base}}$ as well as the quantized weight $\mathbf{W}_{\text{encoded}}$ (Line $6 \sim 7$). Here we use only one basic codebook and please refer to Section 3.2 for more details. Next, we recur to $m-1$ extended codebooks $\{\mathcal{C}^1_{\text{ext}}, \ldots, \mathcal{C}^{m-1}_{\text{ext}}\}$ to fitting the λ important channels in additive manner (Line $8 \sim 13$). Additive VQ iteratively *fits the residual* $\mathbf{E}^t$ from the previous quantization step, allowing important vector groups to benefit from multiple rounds of refinement. Finally, we follow previous works (Egiazarian et al., 2024; Tseng et al., 2024) by finetuning the codebook and beam-searching the optimal code (Line $14 \sim 17$). Moreover, block and end-to-end finetuning is leveraged for enhancing the quantization performance (Line 19,21), which

Algorithm 1 CRVQ: Channel-Relaxed Vector Quantization for LLMs

Require: LLM to be quantized $\mathcal{M}$ with blocks $\{B_1, B_2, \ldots, B_k\}$, number of codebooks m, ratio of important channels λ, loss threshold ϵ, and precomputed $\mathbf{X}, \mathbf{X}\mathbf{X}^{\mathrm{T}}$ for each layer.

1: **for** `block` $B_i \in \{B_1, B_2, \ldots, B_k\}$ **do**
2: **for** `weight` $\mathbf{W} \in$ `layer_weights`(B_i) **do**
3: $\mathbf{W}_{\text{quant}} \leftarrow$ `Prequantize`$(\mathbf{W})$
▷ prequantize the weight $\mathbf{W}$ using VQ
4: $\mathbf{I} \leftarrow$ `ComputeImportance`$(\mathbf{W}, \mathbf{W}_{\text{quant}}, \mathbf{X}\mathbf{X}^{\mathrm{T}})$
5: $\mathbf{W}_{\text{sorted}} \leftarrow$ `ReorderChannels`$(\mathbf{W}, \mathbf{I})$
6: $\mathcal{C}_{\text{base}} \leftarrow$ `KMeansCodebook`$(\mathbf{W}_{\text{sorted}})$
7: $\mathbf{W}_{\text{encoded}}, \mathcal{B}_{\text{base}} \leftarrow$ `VectorQuant`$(\mathbf{W}_{\text{sorted}}, \mathcal{C}_{\text{base}})$
▷ $\mathcal{B}$ is the encoding of $\mathbf{W}$ on codebook $\mathcal{C}$
8: **for** $t = 1, \ldots, m-1$ **do**
9: $\mathbf{E}^t \leftarrow$ `ComputeError`$(\mathbf{W}^{\lambda}, \mathbf{W}^{\lambda}_{\text{encoded}})$
▷ λ indicates the important channels
10: $\mathcal{C}^t_{\text{ext}} \leftarrow$ `KMeansCodebook`$(\mathbf{E}^t)$
11: $\mathbf{E}^t_{\text{encoded}}, \mathcal{B}^t_{\text{ext}} \leftarrow$ `VectorQuant`$(\mathbf{E}^t, \mathcal{C}^t_{\text{ext}})$
12: $\mathbf{W}^{\lambda}_{\text{encoded}} \leftarrow$ `Update`$(\mathbf{W}^{\lambda}_{\text{encoded}}, \mathbf{E}^t_{\text{encoded}})$
13: **end for**
14: **while** `QuantLoss`$(\mathbf{W}, \mathbf{W}_{\text{encoded}}, \mathbf{X}) \geq \epsilon$ **do**
▷ loss is $\|\mathbf{W}\mathbf{X} - \mathbf{W}_{\text{encoded}}\mathbf{X}\|_2^2$
15: `FineTuneCodebook`$(\mathcal{C}_{\text{base}}, \{\mathcal{C}^1_{\text{ext}}, \ldots, \mathcal{C}^{m-1}_{\text{ext}}\})$
16: `BeamSearchOptimize`$(\mathbf{W}, \mathcal{C}_{\text{base}}, \{\mathcal{C}^1_{\text{ext}}, \ldots, \mathcal{C}^{m-1}_{\text{ext}}\}, \mathcal{B}_{\text{base}}, \{\mathcal{B}^1_{\text{ext}}, \ldots, \mathcal{B}^{m-1}_{\text{ext}}\})$
17: **end while**
18: **end for**
19: `FineTuneBlock`(B_i)
20: **end for**
21: **return** $\mathcal{M}_{\text{quant}} \leftarrow$ `E2E_FineTune`$(\mathcal{M})$

is widely used in Egiazarian et al. (2024) and Tseng et al. (2024).

5 Experiment

5.1 Settings

Models and Data We follow contemporaneous works (Xu et al., 2024; Jo et al., 2024) and perform experiments on OPT-1.3B/2.7B, LLaMA-7B/13B, and LLaMA2-7B/13B to evaluate our method. Results on LLaMA3, Qwen2, and Mistral series are in Section 5.3 and 5.4. Since the baselines used for comparison also require calibration data, we follow the experimental setup of AQLM by randomly sampling fixed-length data from the `Red_Pajama` dataset to support algorithm execution. For OPT models, the data length is set to 2048, while for the other models, calibration is performed with a text length of 4096.

Baselines We compare CRVQ with several representative extreme PTQ methods, including PB-LLM (Yuan et al., 2024), BiLLM (Huang et al., 2024), and AQLM (Egiazarian et al., 2024). PB-LLM is integrated with GPTQ as its backbone and 10% of critical weights quantized to 8 bits to ensure an average bit-width below 2 bits. In BiLLM, we use a block size of 128 and 128 calibration samples. In AQLM, we configure the codebook count $m = 1$, vector dimension $d = 8$, and codebook bit-width $e = 8$. We adopt its optimal setup by using 2048 calibration samples. For our proposed CRVQ, the ratio of critical channels is set to $\lambda = 2\%$, with **one** basic codebook and **three** extended codebooks. Similar to AQLM, CRVQ employs $d = 8$, $e = 8$, and 2048 calibration samples. All experiments are conducted on 1×NVIDIA A100-80GB. Further details on the baselines are provided in Section A.3.

Evaluation Metrics To evaluate baseline performance, we calculate perplexity on datasets like WikiText2 (Merity et al., 2017) and C4 (Raffel et al., 2020), using randomly sampled evaluation data of the same length as the calibration data. Lower perplexity indicates better preservation of the original output distribution. Additionally, we report zero-shot accuracies on commonsense reasoning tasks, including Winogrande (Sakaguchi et al., 2021), HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), BoolQ (Clark et al., 2019), and ARC-e/ARC-c (Clark et al., 2018). Evaluations are conducted using the open-source toolkit LM-Evaluation-Harness.[1]

5.2 Main Results

As shown in Table 1, CRVQ consistently outperforms baselines across model sizes. While PB-LLM has the highest average bit-width, it shows no clear advantage in perplexity or zero-shot tasks, likely due to suboptimal bit utilization. In contrast, BiLLM, AQLM, and CRVQ perform comparably. Extreme PTQ incurs some performance loss compared to QAT (Xu et al., 2024; Jo et al., 2024), but its benefits scale with model size. Smaller models like OPT show performance degradation, whereas larger LLaMA models achieve results close to QAT (Xu et al., 2024) or even unquantized models (Touvron et al., 2023b).

For perplexity, CRVQ achieves the lowest across all models, with AQLM as the second-best. For example, on WikiText2 with LLaMA-13B, CRVQ achieves a perplexity of 9.81, compared to 15.25 for AQLM, demonstrating their effectiveness in reducing bit-width. More importantly, CRVQ achieves this with negligible additional bit-width cost.

For zero-shot accuracy, CRVQ achieves the best results in most cases, particularly within the LLaMA series, except for a few tasks on LLaMA-13B (e.g., Winogrande, ARC-e). On most LLaMA models, CRVQ outperforms the second-best baseline by 7% to 13%. We also provide generation cases and analysis in Section A.4.

5.3 Results on Different Model Series

We compare CRVQ with strong baselines on the up-to-date LLaMA3, Qwen2, and Mistral models, to demonstrate the wide applicability of our method on different model series. The results are shown in Table 2. Except for the differences in model series, all experimental settings remain consistent with those in Section 5.1, including the use of `Red_Pajama` as the calibration dataset. Additionally, the evaluation datasets follow the same configuration as described in Section 5.1. As we have observed, CRVQ still shows a clear

[1] `https://github.com/EleutherAI/lm-evaluation-harness`.

Model	Method	Wbits	Perplexity(↓)		Zero-shot Accuracy(↑)						
			Wiki2	C4	Wino.	Hella.	PIQA	BoolQ	ARC-e	ARC-c	Avg.
OPT-1.3B	FP16	16.0	14.67	14.76	59.67	53.69	72.25	56.85	51.43	29.86	53.96
	PB-LLM	1.73	912.41	834.57	50.27	27.14	52.10	38.10	29.46	22.51	36.59
	BiLLM	1.11	59.24	67.71	52.33	32.87	58.00	52.17	35.52	23.04	42.32
	AQLM	1.02	59.68	45.82	50.75	29.94	55.71	**52.26**	33.21	23.29	40.87
	CRVQ	1.10	**39.73**	**33.29**	**52.80**	**32.87**	**58.43**	52.11	**36.15**	**23.63**	**42.67**
OPT-2.7B	FP16	16.0	12.46	13.17	60.69	60.64	74.59	59.60	54.34	31.23	56.85
	PB-LLM	1.73	574.13	520.86	50.27	28.07	52.80	41.14	30.41	22.51	37.53
	BiLLM	1.11	46.40	47.95	51.85	33.82	59.41	**52.17**	35.61	23.46	42.73
	AQLM	1.01	38.73	32.51	50.43	31.61	57.56	51.77	35.61	23.81	41.80
	CRVQ	1.08	**28.91**	**26.11**	**52.88**	**36.09**	**60.94**	51.99	**39.35**	**25.09**	**44.40**
LLaMA-7B	FP16	16.0	5.68	7.08	69.93	76.16	79.16	74.98	72.90	44.62	69.62
	PB-LLM	1.71	231.79	272.66	50.38	28.89	53.26	44.55	31.82	23.17	38.67
	BiLLM	1.09	45.61	75.27	52.72	35.94	58.05	56.42	35.52	22.53	43.53
	AQLM	1.01	17.95	19.20	51.30	38.10	57.67	62.20	39.23	25.09	45.60
	CRVQ	1.07	**13.68**	**15.48**	**55.25**	**43.77**	**61.48**	**62.29**	**44.15**	**25.43**	**48.73**
LLaMA-13B	FP16	16.0	5.09	6.61	73.24	79.09	80.30	77.92	74.54	47.78	72.14
	PB-LLM	1.71	83.17	98.82	50.27	35.52	56.41	58.61	33.92	20.35	42.51
	BiLLM	1.09	13.98	17.97	**61.01**	51.26	67.19	62.29	**53.11**	28.61	**53.91**
	AQLM	1.01	12.93	14.67	58.25	45.61	63.11	63.43	46.51	25.17	50.35
	CRVQ	1.06	**10.73**	**12.56**	60.22	**52.04**	**67.25**	**64.37**	50.76	**28.75**	53.90
LLaMA2-7B	FP16	16.0	5.12	6.63	69.22	75.93	79.11	77.68	74.75	46.16	70.47
	PB-LLM	1.71	186.06	223.54	49.48	31.10	52.64	40.82	30.41	24.20	38.11
	BiLLM	1.08	26.52	39.71	50.59	31.46	55.01	60.10	35.16	20.44	42.12
	AQLM	1.01	21.28	24.44	50.59	32.04	55.66	61.41	33.00	20.73	42.24
	CRVQ	1.07	**13.01**	**15.72**	**55.09**	**42.28**	**61.48**	**62.57**	**39.31**	**25.51**	**47.71**
LLaMA2-13B	FP16	16.0	4.57	6.05	71.90	79.37	80.63	80.86	77.53	49.23	73.25
	PB-LLM	1.71	228.22	259.69	50.56	30.82	53.98	42.18	29.94	22.80	38.38
	BiLLM	1.08	20.52	32.01	54.85	38.95	60.94	65.60	45.96	25.94	48.71
	AQLM	1.01	15.25	18.35	52.49	35.14	56.42	62.29	38.01	24.32	44.78
	CRVQ	1.06	**9.81**	**12.48**	**58.09**	**50.88**	**65.40**	**67.89**	**49.49**	**28.07**	**53.30**

Table 1: Main results of evaluation experiment. We report the perplexity and zero-shot accuracy. The results of AQLM and CRVQ are from the models after e2e-finetuning. The **best** scores are in **bold**.

advantage in perplexity across models from different developers, and it remains superior on most downstream tasks.

5.4 Effectiveness on MoE Architecture

Although our extremely low-bit compression algorithm is not specifically designed for the Mixture-of-Experts (MoE) architecture, we still try to extend the applicability of CRVQ on the MoE architecture. Here we choose a classic MoE model, Mistral-8×7B,[2] as our experimental model, and compare the actual effects of CRVQ and AQLM. In the experimental setting that is exactly the same as the main experiment, we treat each expert (FFN module) in the MoE in the same way as the dense model. The results are shown in Table 3. First, it showns that our method is still effective and maintains its advantages on the different MoE architecture. Second, it also shows that CRVQ is still applicable with a larger number of weights (about 50B).

[2]https://mistral.ai/en/models.

6 Discussion

This section delves into the effectiveness and its inner components of the proposed method on LLaMA2-7B, with the goal of enhancing understanding of its underlying mechanisms.

6.1 Ratio of Important Channels

CRVQ leverages extra codebooks for a few critical channels to maximize quantization benefits with minimal overhead. A natural question arises: *how many channels are sufficient for CRVQ to perform*

Model	Method	Wbits	Perplexity(↓)		Zero-shot Accuracy(↑)						
			Wiki2	C4	Wino.	Hella.	PIQA	BoolQ	ARC-e	ARC-c	Avg.
LLaMA3.2-1B	FP16	16.0	9.07	12.04	60.14	63.62	74.59	63.98	60.52	36.01	59.81
	BiLLM	1.09	394.87	398.45	**50.43**	**28.72**	53.26	58.13	29.50	**23.89**	40.65
	AQLM	1.02	115.25	90.85	47.91	26.95	53.92	50.00	30.85	20.65	38.38
	CRVQ	1.10	**70.03**	**64.72**	48.70	28.15	**54.95**	**61.47**	**31.19**	21.59	**41.01**
LLaMA3.2-3B	FP16	16.0	7.28	10.01	69.38	73.61	77.42	72.94	71.59	45.90	68.47
	BiLLM	1.10	122.19	139.83	**51.14**	**32.01**	54.90	61.80	**32.87**	21.25	42.31
	AQLM	1.01	83.03	63.99	50.04	28.16	53.97	61.62	30.89	21.16	40.97
	CRVQ	1.07	**49.17**	**40.23**	50.91	30.98	**56.04**	**62.45**	32.03	**21.42**	**42.31**
LLaMA3.1-8B	FP16	16.0	5.84	8.43	73.48	78.85	81.18	82.02	81.19	53.50	75.03
	BiLLM	1.10	38.79	57.07	50.43	**36.16**	57.94	**62.57**	35.19	23.04	44.22
	AQLM	1.01	57.90	48.44	49.57	28.77	54.19	57.71	30.13	20.22	40.09
	CRVQ	1.07	**27.36**	**29.53**	**51.62**	36.10	**59.92**	61.53	**37.25**	**23.29**	**44.95**
Qwen2-0.5B	FP16	16.0	12.35	16.18	57.46	49.07	69.31	60.86	50.25	28.67	52.60
	BiLLM	1.09	1549.81	1587.81	**51.54**	26.79	51.90	45.32	28.03	**23.21**	37.79
	AQLM	1.06	96.34	84.86	48.54	**27.26**	52.18	54.68	31.06	21.33	39.17
	CRVQ	1.22	**77.72**	**69.06**	50.59	26.87	**54.46**	**61.28**	**31.36**	22.78	**41.22**
Qwen2-1.5B	FP16	16.0	8.87	12.20	66.38	65.37	75.35	72.42	60.77	36.26	62.75
	BiLLM	1.09	522.46	618.99	50.36	28.45	53.37	52.81	28.45	**23.72**	39.52
	AQLM	1.02	52.24	51.97	**53.35**	29.10	56.26	62.05	31.02	21.59	42.22
	CRVQ	1.11	**44.62**	**43.66**	51.70	**29.68**	**56.47**	**62.23**	**31.40**	22.18	**42.28**
Qwen2-7B	FP16	16.0	6.67	9.57	72.45	78.85	81.01	84.98	74.49	49.91	73.61
	BiLLM	1.09	28.80	31.68	56.27	46.01	64.85	62.81	**49.54**	**28.16**	51.27
	AQLM	1.01	20.57	23.03	52.88	42.30	62.46	62.02	41.62	25.34	47.77
	CRVQ	1.07	**17.39**	**19.91**	**57.54**	**47.18**	**65.13**	**64.04**	46.25	27.56	**51.28**
Mistral-7B	FP16	16.0	4.91	7.42	74.11	81.11	82.15	83.76	79.59	54.18	75.81
	AQLM	1.01	16.28	19.84	52.25	36.33	57.62	61.74	37.88	23.81	44.94
	CRVQ	1.07	**13.01**	**16.23**	**54.70**	**44.11**	**60.83**	**62.75**	**42.26**	**25.51**	**48.36**

Table 2: Perplexity and zero-shot accuracy on LLaMA3, Qwen2, and Mistral models. The results of AQLM and CRVQ are based on the models after e2e-finetuning. The **best** scores are highlighted in **bold**.

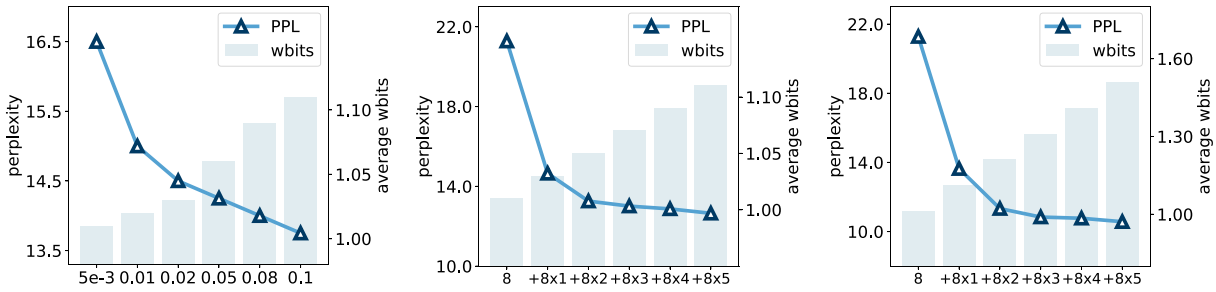
Model	Method	Wbits	Perplexity(↓)		Zero-shot Accuracy(↑)						
			Wiki2	C4	Wino.	Hella.	PIQA	BoolQ	ARC-e	ARC-c	Avg.
Mistral-8×7B	FP16	16.0	3.59	6.52	76.64	83.94	83.35	85.41	83.42	59.81	78.76
	AQLM	1.01	12.42	15.93	57.06	48.86	66.87	63.49	49.71	26.88	52.14
	CRVQ	1.07	**9.28**	**12.32**	**60.69**	**63.60**	**73.56**	**72.32**	**57.53**	**34.47**	**60.36**

Table 3: Perplexity and zero-shot accuracy on Mistral-8×7B model. The results of AQLM and CRVQ are based on the models after e2e-finetuning. The **best** scores are highlighted in **bold**.

effectively? We test this under codebook count $m = 2$, i.e., ''$8 + 8 \times 1$'', varying the channel ratio λ from 5e-3 to 0.1. As shown in Figure 4a, when $\lambda < 0.02$, the perplexity of the quantized model drops rapidly by 6.7 compared to the model without critical channels. Increasing λ to 0.1 further improves performance, even though the average bit-width remains as low as 1.1 bit. These results validate the significant impact of critical channels. Considering the performance-to-cost ratio at $\lambda = 0.02$, this setting maybe well-suited for deployment on extremely low-resource devices.

6.2 Number of Extended Codebooks

In CRVQ, we apply multiple extended codebooks to a few critical channels. There are also two important questions: *how many extended codebooks are necessary for the method to work effectively, and is more always better?* We investigate the impact of the extended codebooks under two critical



(a) different λ (b) different m ($\lambda = 0.02$) (c) different m ($\lambda = 0.1$)

Figure 4: Quantization performance varies with the critical channel ratio λ and codebook count m.

Ratio	Reorder Strategy	Perplexity(↓)		Zero-shot Accuracy(↑)						
		Wiki2	C4	Wino.	Hella.	PIQA	BoolQ	ARC-e	ARC-c	Avg.
$\lambda = 0.02$	*Random*	19.99	22.93	50.12	32.87	55.44	61.25	34.09	22.70	42.75
	W-only	17.72	20.19	53.51	34.69	56.04	57.43	36.95	24.74	43.89
	W-A	**13.01**	**15.72**	**55.09**	**42.28**	**61.48**	**62.57**	**39.31**	**25.51**	**47.71**
$\lambda = 0.1$	*Random*	14.89	17.88	53.83	37.73	58.00	57.09	37.21	23.98	44.64
	W-only	16.32	18.24	51.22	37.66	58.81	59.02	38.30	23.98	44.83
	W-A	**10.83**	**13.43**	**56.99**	**46.95**	**64.64**	**61.59**	**44.61**	**26.71**	**50.25**

Table 4: Performance on different reorderding strategy. Here we use $8 + 8 \times 3$ setting.

channel ratios, $\lambda = 0.02$ and $\lambda = 0.1$. As shown clearly in Figure 4b and 4c, using three extended codebooks provides the most notably performance improvement compared to using none in both settings. However, when the extended codebooks count exceeds three, the performance gains become marginal. Thus, adding more codebooks does not necessarily yield better results and we recommend using three extended codebooks for optimal performance.

6.3 Different Reorder Strategy

We discuss on different importance metrics to understand the contributions of different channels in CRVQ. Based on the analysis in Section 4.1, we evaluate three reordering strategies: random reordering (*Random*), reordering based on weight magnitude (*W-only*), and reordering using weight combined with Hessian proxy (*W-A*). The results in Table 4 indicate that the *W-A* consistently outperforms the others. We also find that the effectiveness of *Random* and *W-only* strategies varies with λ. Specifically, *W-only* performs better than *Random* at extremely low λ. It is likely because the *Random* strategy has low hit rates for critical channels under such condition. These results emphasize the necessity of explicitly catching critical channels in VQ to achieve superior performance.

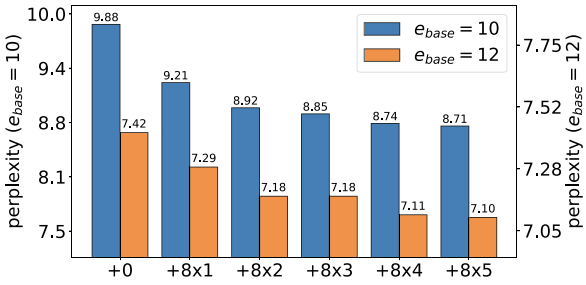
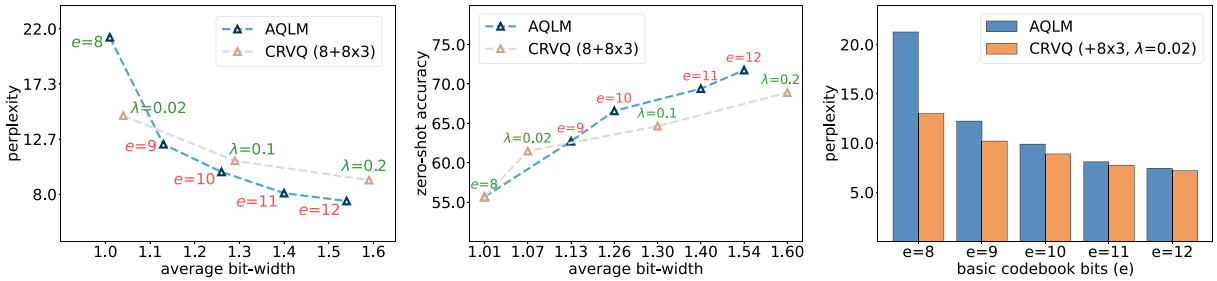


Figure 5: Effectiveness of CRVQ when combined with stronger basic codebook, i.e., e changes from 8 to 10 and 12. The performance consistently improves as the extended codebook count increases.

6.4 Flexible Codebook Settings

CRVQ is not limited to integrate with a single basic codebook and $e = 8$. On the contrary, it can be seamlessly integrated with any vanilla VQ method, applying multi-codebook fitting specially to critical channels. First, as the basic codebook bit-width increases, its fitting ability improves, but CRVQ still works well. Figure 5 shows the performance of CRVQ when applied to two larger codebooks ($e = 10$ and $e = 12$). Obviously, CRVQ consistently shows effectiveness as the extended codebooks increase. Moreover, our method also works well with involving more base codebooks. For instance, under the setting of $m = 2$, $d = 16$, $e = 8$, the perplexity of AQLM is



(a) (b) (c)

Figure 6: Bit scaling laws of AQLM and CRVQ on perplexity and zero-shot accuracy (PIQA). In Figure 6a and 6b, we increase the codebook bit-width of AQLM from 8 to 12 and compare it with CRVQ in different λ. Note that the basic codebook bit-width of CRVQ constantly equals 8 here. The hardware-unfriendly settings are noted in red. In Figure 6c, the codebook bit-width of AQLM and basic codebook bit-width of CRVQ synchronously increase from 8 to 12.

17.11, whereas CRVQ achieves 14.21 with three extended codebooks. This demonstrates the generality and flexibility of our approach across diverse settings.

6.5 Hardware-friendly Deployment

Although traditional VQ can also enhance its performance by increasing e, we recommend using $e = 8$ in combination with additional extended codebooks when deploying models. First, increasing e (e.g., to $e = 10$) may result in a larger overall bit-width, leading to lower bit-width utilization. It also hampers hardware alignment and computational acceleration. Furthermore, combining hardware-friendly basic codebooks with several extended codebooks and adjusting $\lambda\%$ critical channels can efficiently achieve better quantization performance without these drawbacks. Therefore, CRVQ is cost-effective on low-resource devices and easily adaptable to different devices. Please refer to Section A.5 for inference speed.

6.6 Bit Scaling Laws

We show that CRVQ is a solution more suitable for low-end hardware deployment. To further illustrate its bit scaling property and deployment efficiency, we evaluate its perplexity and zero-shot accuracy under different bit-width configurations.

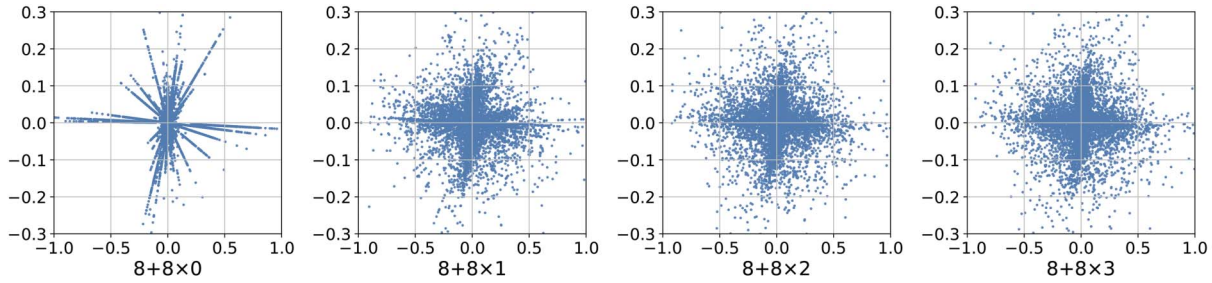
Compared to previous works, both AQLM and CRVQ allow flexible bit-width customization, enabling models to scale their capabilities according to hardware constraints. However, in extreme compression scenarios (sub-2-bit), AQLM can only adjust the codebook bit-width e to control bit-width, as increasing the number of codebooks m would double the total bit-width, exceeding 2. Yet, setting e to 9, 10, 11, or 12 is impractical for deployment, as it misaligns memory boundary with hardware-friendly configurations, leading to inefficiencies and limited acceleration. This approach, even if effective, is not practical for real deployment (see Figure 6a and 6b for details).

In contrast, CRVQ offers a more deployment-friendly solution by keeping the basic codebook bit-width fixed at $e = 8$ and adjusting the critical channel ratio λ to achieve bit scaling, thereby enabling capability scaling. This method achieves similar effectiveness while maintaining a hardware-efficient computation process.

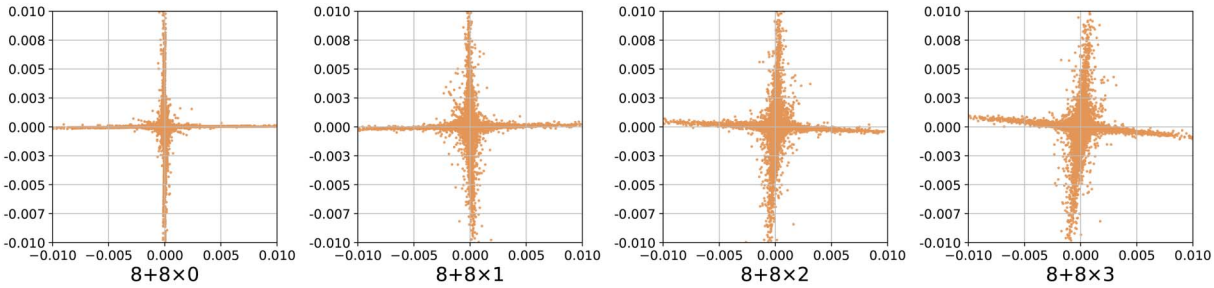
Additionally, if not considering hardware efficiency, we investigate how increasing the basic codebook bit-width in CRVQ affects its advantage over AQLM. In practice, as the basic codebook bit-width increases and model capacity improves, the advantage over AQLM gradually diminishes. This trend is illustrated in Figure 6c. The reason for this lies in the application scenario of CRVQ, which focuses on exhuming bit-width efficiency in extreme compression. When the basic codebook provides sufficient ability, the contribution of critical channels becomes negligible.

6.7 Codebook Fitting Visualization

We have demonstrated that a small subset of critical channels play a significant role in vector quantization. In our previous analysis, this is primarily reflected in overall LLM perplexity and downstream task accuracy. However, we



(a) vector samples from product of important weight channels and activation



(b) vector samples from product of non-important weight channels and activation

Figure 7: 2-D visualization of the element-wise product from quantized weight and activation. The weight matrix is `q_proj` from layer 0 of LLaMA2-7B and we partition the product into vectors with a length of $d = 8$. Here we use $\lambda = 0.02$. Figure 7a corresponds to important channels, while Figure 7b corresponds to non-important channels. We compare the impact of different numbers of extended codebooks (ranging from 0 to 3) and channel importance on the distribution of product vectors.

remain curious about the differences between applying extended codebooks to important versus non-important channels. To figuratively understand this distinction, we partition the element-wise product of the quantized weight matrix and an activation vector into vectors of $d = 8$, then apply PCA (Hotelling, 1933) for 2-D visualization. The results from important and non-important channels are shown in Figure 7a and 7b.

From Figure 7, it is evident that the product vectors from important channels and non-important channels exhibit distinct distribution patterns after dimensionality reduction. The former resembles a scattered point cloud, while the latter is a cross-like shape. As the number of extended codebooks increases, the product vectors of important channels gradually transition from a disordered spread to a cross-like structure, accompanied by a subtle rotational effect. In contrast, the product vectors of non-important channels primarily exhibit rotation without significant changes in shape as the number of extended codebooks increases.

These observations suggest that differences in channel importance manifest different quantization behaviors. Notably, the important weights change significantly with the increase in the number of extended codebooks.

6.8 Theoretical Analysis

Mixed-precision quantization is a well-established idea. However, we are the first to apply it within the vector quantization for LLMs and verify its effectiveness. Here we provide a simple analysis. We simply let $\mathbf{w}$ denote a weight vector to be quantized and $\mathbf{x}$ the corresponding activation. The quantization error incurred under a single codebook is

$$\mathcal{L} = \|\Delta\mathbf{w} \cdot \mathbf{x}\|_2^2 \leq \sum_i (\Delta w_i^{(1)} x_i)^2, \qquad (8)$$

where $\Delta w_i^{(1)}$ denotes the weight quantization error in the i-th channel. For simplicity, we denote the error upper-bound as $B_a = \sum_{i \in R_a} (\Delta w_i^{(1)} x_i)^2$, where R_a is the channel set. Now, suppose we apply $n-1$ extended codebooks to a random subset of channels $R_r \subset R_a$. The error bound can be formulated as

$$B_r = \sum_{i \in R_r} (\Delta w_i^{(n)} x_i)^2 + \sum_{j \in (R_a - R_r)} (\Delta w_j^{(1)} x_j)^2. \qquad (9)$$

Since the quantization error under multiple codebooks satisfies $\Delta w_i^{(n)} < \Delta w_i^{(1)}$, we clearly have $B_r \leq B_a$.

Further, from Gersho's vector quantization theory (Gersho and Gray, 2012), we know that channels with larger quantization error benefit more from multiple codebooks, due to the sparse distribution of codeword space in high-error regions. Formally, if $\Delta w_i^{(1)} > \Delta w_j^{(1)}$, the improvement satisfies $\Delta w_i^{(1)} - \Delta w_i^{(n)} > \Delta w_j^{(1)} - \Delta w_j^{(n)}$. In our setting, importance is estimated by a Hessian-based proxy, and $\Delta w_i^{(1)} x_i$ serves as an indicator. Since we only consider the most few critical channels, if $\Delta w_i^{(1)} x_i > \Delta w_j^{(1)} x_j$ holds, we can loosely derive $(\Delta w_i^{(1)} - \Delta w_i^{(n)}) x_i > (\Delta w_j^{(1)} - \Delta w_j^{(n)}) x_j$. Now consider only applying extended codebooks on the selected most critical channel set $R_m \subset R_a$. The error bound can be formulated as

$$B_m = \sum_{i \in R_m} (\Delta w_i^{(n)} x_i)^2 + \sum_{j \in (R_a - R_m)} (\Delta w_j^{(1)} x_j)^2. \tag{10}$$

Given that the channels in R_m have larger Hessian proxy values, we have $B_a - B_m > B_a - B_r$, although not strictly. Overall, we generally derive

$$B_m < B_r < B_a, \tag{11}$$

which justifies our approach of prioritizing critical channels for high-precision vector quantization via extended codebooks.

6.9 Comparison to QAT Method

Our work focuses on extreme algorithms within the PTQ area. A related direction with similar goal is QAT, and the two methods have their respective advantages and limitations. QAT achieves relatively better performance through extensive training, whereas PTQ avoids the long training process but often falls short of QAT in performance. We compare our method with the classical 1-bit QAT approach, OneBit (Xu et al., 2024), as shown in Table 5. Notably, our algorithm outperforms OneBit in terms of computational efficiency while achieving comparable performance. Our study shows that with continued advancements, PTQ is approaching the performance levels of QAT.

6.10 Comparison with ARB-LLM

A contemporaneous work, ARB-LLM (Li et al., 2024), also explores 1-bit PTQ. It employs an alternating refined binarization approach to optimize the gap between binary and high-precision weights. While ARB-LLM also leverages the idea of certain weight sensitivity, it follows a different technical route from CRVQ, which is based on vector quantization. Tables 6 and 7 compares the performance of both methods on LLaMA series models. Here CRVQ uses $8+8\times3$, $\lambda = 0.02$, and ARB-LLM$_{\text{RC}}$ is reported. Although ARB-LLM has a clear advantage over CRVQ in terms of quantization runtime, CRVQ appears to achieve better performance in some models and tasks.

BiLLM and ARB-LLM are both improvements of the GPTQ algorithm in extreme compression scenarios, and they share a similar core idea. In contrast to uniform quantization like BiLLM and ARB-LLM, CRVQ takes a completely different approach, vector quantization. VQ may perform better in extreme compression settings, and recent studies have shown signs supporting this (Egiazarian et al., 2024; Tseng et al., 2024). We believe that the idea of VQ such as CRVQ will attract more in-depth research in the future.

Method	Wiki2	#GPUs	#Hours /GPU	#Hours
OneBit	8.76	16	448	7168
CRVQ	9.81	1	50	50

Table 5: Comparison between OneBit and CRVQ on LLaMA2-13B. Here we use $8 + 8 \times 3$ setting.

Model	ARB-LLM		CRVQ	
	Wbits	Wiki2	Wbits	Wiki2
LLaMA-7B	1.09	14.01	1.07	**13.68**
LLaMA-13B	1.09	**10.24**	1.06	10.73
LLaMA2-7B	1.08	15.58	1.07	**13.01**
LLaMA2-13B	1.08	11.49	1.06	**9.81**
LLaMA3.1-8B	1.06	**27.18**	1.07	27.36

Table 6: Perplexity of ARB-LLM and CRVQ.

Task	LLaMA2-7B		LLaMA2-13B	
	ARB-LLM	CRVQ	ARB-LLM	CRVQ
Wino.	54.85	**55.09**	58.01	**58.09**
Hella.	**45.61**	42.28	47.18	**50.88**
PIQA	60.94	**61.48**	**67.19**	65.40
ARC-c	25.17	**25.51**	**28.92**	28.07

Table 7: Comparison of zero-shot performance between ARB-LLM and CRVQ.

7 Conclusion

We introduce CRVQ, a simple yet effective extreme quantization algorithm that enhances performance by identifying critical weight channels and applying multi-codebook fitting via channel reordering. Extensive experiments on models of varying sizes demonstrate significant performance gains at minimal additional cost, outperforming representative strong baselines. We also analyze key factors affecting CRVQ and provide guidance for hardware resource adaptation. In addition, we conduct an in-depth and effective investigation into the underlying principles of the method.

Acknowledgments

We gratefully acknowledge the support of the National Natural Science Foundation of China (NSFC) via grants 62236004, 62206078 and 62476073. We are also deeply grateful to the reviewers and the action editor for their generous investment of time and their insightful comments, which have significantly contributed to refining and strengthening our work.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Qin Cai, Vishrav Chaudhary, Dong Chen, Dongdong Chen, Weizhu Chen, Yen-Chun Chen, Yi-Ling Chen, Hao Cheng, Parul Chopra, Xiyang Dai, Matthew Dixon, Ronen Eldan, Victor Fragoso, Jianfeng Gao, Mei Gao, Min Gao, Amit Garg, Allie Del Giorno, Abhishek Goswami, Suriya Gunasekar, Emman Haider, Junheng Hao, Russell J. Hewett, Wenxiang Hu, Jamie Huynh, Dan Iter, Sam Ade Jacobs, Mojan Javaheripi, Xin Jin, Nikos Karampatziakis, Piero Kauffmann, Mahoud Khademi, Dongwoo Kim, Young Jin Kim, Lev Kurilenko, James R. Lee, Yin Tat Lee, Yuanzhi Li, Yunsheng Li, Chen Liang, Lars Liden, Xihui Lin, Zeqi Lin, Ce Liu, Liyuan Liu, Mengchen Liu, Weishung Liu, Xiaodong Liu, Chong Luo, Piyush Madan, Ali Mahmoudzadeh, David Majercak, Matt Mazzola, Caio César Teodoro Mendes, Arindam Mitra, Hardik Modi, Anh Nguyen, Brandon Norick, Barun Patra, Daniel Perez-Becker, Thomas Portet, Reid Pryzant, Heyang Qin, Marko Radmilac, Liliang Ren, Gustavo de Rosa, Corby Rosset, Sambudha Roy, Olatunji Ruwase, Olli Saarikivi, Amin Saied, Adil Salim, Michael Santacroce, Shital Shah, Ning Shang, Hiteshi Sharma, Yelong Shen, Swadheen Shukla, Xia Song, Masahiro Tanaka, Andrea Tupini, Praneetha Vaddamanu, Chunyu Wang, Guanhua Wang, Lijuan Wang, et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219v4*.

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. 2024. On-policy distillation of language models: Learning from self-generated mistakes. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609v1*.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi. 2020. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 7432–7439. https://doi.org/10.1609/aaai.v34i05.6239

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher M. De Sa. 2023. QuIP: 2-bit quantization of large language models with guarantees.*Advances in Neural Information Processing Systems (NeurIPS)*, 36:4396–4429.

Mengzhao Chen, Wenqi Shao, Peng Xu, Jiahao Wang, Peng Gao, Kaipeng Zhang, and Ping Luo. 2024. EfficientQAT: Efficient quantization-aware training for large language models. *arXiv preprint arXiv:2407.11062v2*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 2924–2936.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? Try ARC, the AI2 Reasoning Challenge. *arXiv preprint arXiv:1803.05457v1*.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. LLM.int8(): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing System (NeurIPS)*, 35:30318–30332.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient finetuning of quantized LLMs. *Advances in Neural Information Processing System (NeurIPS)*, 36:10088–10115.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. 2024. SpQR: A sparse-quantized representation for near-lossless LLM weight compression.In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*.

Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. 2024. Extreme compression of large language models via additive quantization. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 12284–12303.

Elias Frantar and Dan Alistarh. 2023. SparseGPT: Massive language models can be accurately pruned in one-shot. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 10323–10337.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. GPTQ: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323v2*.

Allen Gersho and Robert M. Gray. 2012. *Vector Quantization and Signal Compression*, volume 159. Springer Science & Business Media.

Harold Hotelling. 1933. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417. https://doi.org/10.1037/h0071325

Cheng-Yu Hsieh, Chun-Liang Li, Chih-kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics (ACL Findings)*, pages 8003–8017. https://doi.org/10.18653/v1/2023.findings-acl.507

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, Xinrong Zhang, Zheng Leng Thai, Kaihuo Zhang, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai, Ning Ding, Chao Jia, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. MiniCPM: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395v3*.

Wei Huang, Yangdong Liu, Haotong Qin, Ying Li, Shiming Zhang, Xianglong Liu, Michele Magno, and Xiaojuan Qi. 2024. BiLLM: Pushing the limit of post-training quantization for LLMs. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 20023–20042.

Dongwon Jo, Taesu Kim, Yulhwa Kim, and Jae-Joon Kim. 2024. Mixture of scales: Memory-efficient token-adaptive binarization for large language models. *Advances in Neural Information Processing System (NeurIPS)*, 37:137474–137494.

Zhiteng Li, Xianglong Yan, Tianao Zhang, Haotong Qin, Dong Xie, Jiang Tian, Linghe

Kong, Yulun Zhang, and Xiaokang Yang. 2024. ARB-LLM: Alternating refined binarizations for large language models. *arXiv preprint arXiv:2410.03129v2*.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. *Proceedings of Machine Learning and Systems*, 6:87–100.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. 2024. LLM-QAT: Data-free quantization aware training for large language models. In *Findings of the Association for Computational Linguistics (ACL Findings)*, pages 467–484. https://doi.org/10.18653/v1/2024.findings-acl.26

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *Proceedings of the Fifth International Conference on Learning Representations (ICLR)*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106. https://doi.org/10.1145/3474381

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. 2024. OmniQuant: Omnidirectionally calibrated quantization for large language models. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. 2024. A simple and effective pruning approach for large language models. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971v1*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. LLaMA 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288v2*.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. 2024. QuIP#: Even better LLM quantization with hadamard incoherence and lattice codebooks. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 48630–48656.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 38087–38099.

Yuzhuang Xu, Xu Han, Zonghan Yang, Shuo Wang, Qingfu Zhu, Zhiyuan Liu, Weidong Liu, and Wanxiang Che. 2024. OneBit: Towards extremely low-bit large language models. *Advances in Neural Information Processing System (NeurIPS)*, 37:66357–66382.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang,

Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671v4*.

Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. 2022. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:27168–27183.

Zhihang Yuan, Yuzhang Shang, and Zhen Dong. 2024. PB-LLM: Partially binarized large language models. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4791–4800. https://doi.org/10.18653/v1/P19-1472

A Appendix

A.1 Average Bits Calculation

We first compute the average bit-width of quantized weights using the classical vector quantization method and explain the derivation of Equation 6, with a 4096×4096 matrix as an example. Following the notations in this paper, let m denote the number of codebooks, e the codebook bit-width, d the dimension of the short vectors obtained from matrix partitioning, and n the average bit-width. The total space consumption consists of two parts: the codebooks and the binary encodings.

Consider the total number of vectors. Each row is divided into $2^{12}/d$ vectors, and the entire matrix is partitioned into $2^{24}/d$ vectors. If each vector is represented using e-bit binary encoding, the total code space required is $2^{24}e/d$ bits. Since each codebook has its own code, the total code overhead is $2^{24}me/d$ bits. Additionally, each codebook contains 2^e vectors of length d, where each value is stored in FP16 format. Therefore, the total space occupied by m codebooks is $2^e md \times 2^4$ bits. Thus, the total storage requirement is given by $2^{24}me/d + 2^{e+4}md$. Since the matrix contains 2^{24} elements, the quantized average bit-width is:

$$n = \frac{me}{d} + 2^{e-20}md.$$

Assuming that the setting of AQLM is $m = 1, e = 8, d = 8$, the average bit-width here is $n \approx 1.002$.

CRVQ introduces the concept of extended codebooks. Suppose that among the m codebooks, there are $m - 1$ extended codebooks used to encode λ-fraction of important vectors. The total code space for these important vectors can be expressed as $(m-1)\lambda e/d$. Moreover, storing column reordering indices requires extra space. Each index occupies 16 bits (with 12 effective bits), resulting in an additional storage requirement of $2^{12} \times 2^4/2^{24}$ bits. Clearly, this overhead is negligible. Thus, the total storage requirement for CRVQ is given by:

$$n = \frac{e}{d} + \frac{(m-1)\lambda e}{d} + 2^{e-20}md. \tag{12}$$

Assuming that the setting of CRVQ is $8 + 8 \times 3$ (i.e., $m = 4$), $d = 8$ and $\lambda = 0.02$, the average bit-width here is $n \approx 1.062$.

A.2 Hessian-based Importance Metric

As shown in Equation 7, the importance metric based on the Hessian proxy considers both weight quantization error and activation magnitude. To understand the computation of importance, we unpack the matrix formulation. Let $\mathbf{W}^{M\times N}$ denote the weight matrix to be quantized, and the quantization error for each weight can be represented by $\Delta w_{ji} = w_{ji} - \text{VQ}(w_{ji})$. Suppose the activation multiplying with the weight is denoted as $\mathbf{X}^{N\times O}$. Then, the Hessian proxy can be computed as:

$$\mathbf{X}\mathbf{X}^{\mathrm{T}} = \begin{pmatrix} \sum_{k=1}^{O} x_{1k}^2 & \cdots & \sum_{k=1}^{O} x_{1k}x_{Nk} \\ \vdots & \ddots & \vdots \\ \sum_{k=1}^{O} x_{Nk}x_{1k} & \cdots & \sum_{k=1}^{O} x_{Nk}^2 \end{pmatrix} \tag{13}$$

Hence the Hessian-based importance metric can be reformulated as:

$$I_i = \max_j \frac{1}{2} \sum_{k=1}^{O} (x_{ik}\Delta w_{ji})^2. \tag{14}$$

Therefore, the selection of critical channels is guided by the outcome of the matrix product. Channels with both high quantization error and large corresponding activation magnitude have the greatest impact on the quantization results.

A.3 Details on Baselines and CRVQ

In this subsection, we provide the essential supplemental details of the baselines in this work and our proposed CRVQ:

- PB-LLM (Yuan et al., 2024): PB-LLM, which can be integrated with other classical PTQ algorithms, is an early approach to sub-2-bit PTQ, introduces mixed-precision quantization by applying higher bit-widths to a small subset of sensitive weights while binarizing the majority. We quantize the majority to 1-bit and maintain the sensitive weights to 8-bit, leading to a 1.7-bit algorithm. We use the hessian metric to find the salient weights. The other parameters are based on the code released by the authors.
- BiLLM (Huang et al., 2024): BiLLM enhances quantization precision by partitioning the weight distribution and assigning different quantization parameters to each partition. It also employs residual-based multi-step quantization to improve accuracy. We use the hessian metric to partition the weights and other settings are identical to the original paper.
- AQLM (Egiazarian et al., 2024): AQLM is an additive vector quantization method that combines layer-level, block-level, and end-to-end fine-tuning. We limit the fine-tuning of intra-layer codebooks to a maximum of 100 epochs, typically not exceeding 20 epochs in most cases. For block-level and end-to-end fine-tuning, we perform a maximum of 25 epochs, using a batch size of 64. The fine-tuning process are optimized by Adam with $\beta_1 = 0.90, \beta_2 = 0.95$.
- CRVQ: The fine-tuning parameters in our method are the same as those used in AQLM. Given the use of multiple codebooks, we set the beam search width (Egiazarian et al., 2024) to 1, which is the same as the project AQLM (Egiazarian et al., 2024).

A.4 Generation Cases

We quantize the LLaMA2-13B model using different methods and compare their generated outputs for given prompts, as shown in Table 8. The average bit-width of the models is 1.08, 1.01, and 1.06 bits respectively. The results indicate that CRVQ produces longer and more fluent text. However, despite its

Prompt	A cowboy rides a_
BiLLM	cowboy rides a cowboy
AQLM	troller. in the troller. The troller is a troller.
CRVQ	1950s Ford Mustang in the desert. The cowboy rides a 1950s Ford Mustang in the desert.
Prompt	There are a number of ways to reduce air pollution, such as_
BiLLM	100% of the air pollution. The air pollution is a number of the air pollution.
AQLM	10000000000000000000
CRVQ	1) reducing the amount of pollution from the burning of fossil fuels, 2) reducing the amount of pollution from the burning of coal, 3) reducing the amount of pollution from the burning of natural gas, 4) reducing the amount of pollution from the burning of wood, 5) reducing the amount of pollution from the burning of natural gas
Prompt	The capital of the state of New York is_
BiLLM	100% of the state of New York is 100% of the state of New York
AQLM	100% of the state of New York. of the state of New York
CRVQ	1100 miles away from the city of New York. The state of New York is divided into 10 counties.

Table 8: Generation cases of CRVQ and baselines. The three prompt we use is from Jo et al. (2024).

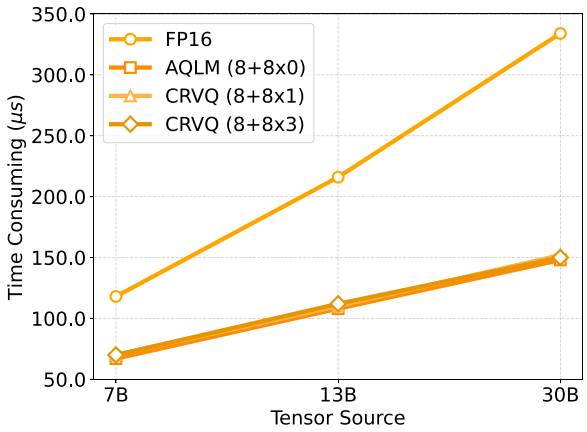


Figure 8: Speed of the `gate_proj` layer matrix-vector multiplication.

fluency, the generated content still contains many hallucinations. For example, in case 3, the capital of New York State is 150 miles from New York City, not 1,100 miles, and there are 62 counties, not 10.

A.5 Inference Speed

Similar to Egiazarian et al. (2024), we also evaluate the computational performance of CRVQ on the same A100 GPU, with the results of LLaMA2 series presented in Figure 8.

As shown in the Figure 8, although CRVQ introduces tensor channel reordering and extended codebooks compared to AQLM, the matrix computation time does not increase significantly. This is mainly due to two factors. First, the computational overhead of the extended codebooks is inherently small. Second, we replace reordering of the weight matrix with reordering of activation vectors. In resource-constrained devices, requests are often in small batches.

A.6 Limitations

While our method achieves obvious post-training extreme compression with minimal cost, offering a cost-effective solution for deploying LLMs on low-resource devices, some limitations remain. First, like other 1-bit baselines (Xu et al., 2024; Huang et al., 2024), our approach exhibits performance degradation compared to the original model and higher-bit quantization methods (e.g., 2-bit and above), highlighting the need for further exploration into the limits of extreme compression. Additionally, although our method significantly reduces computational demands compared to QAT-based 1-bit quantization (Xu et al., 2024), the quantization process itself remains time-consuming. Fortunately, this is a one-time cost. Finally, similar to many classic quantization studies, the role of a small subset of critical channels in the quantization process and its impact on overall performance remain insufficiently explored. This opens up exciting avenues for future research.

A.7 Ethics Statement

In this study, we employ models and data that are publicly available and open source. We affirm that the use of these models aligns with their original intended purposes. These models have been utilized strictly within the scope of academic and research-based activities, adhering to ethical guidelines and ensuring compliance with open-source licenses.