ARB-LLM: Alternating Refined Binarizations For Large Language Models

Zhiteng Li[1*], Xianglong Yan[1*], Tianao Zhang[1], Haotong Qin[2], Dong Xie[3], Jiang Tian[3], Zhongchao Shi[3], Linghe Kong[1†], Yulun Zhang[1†], Xiaokang Yang[1]

[1]Shanghai Jiao Tong University, [2]ETH Zürich, [3]Lenovo Research

Introduction

Observation

1. **Distribution shift** occurs after asymmetric binarization.
2. The optimization of binarization parameters μ and α does not adequately reflect the real-world scenario, as it does not account for the **input data $\mathbf{X}$**.
3. Row-wise binarization is not effective at handling **column-wise deviations**.

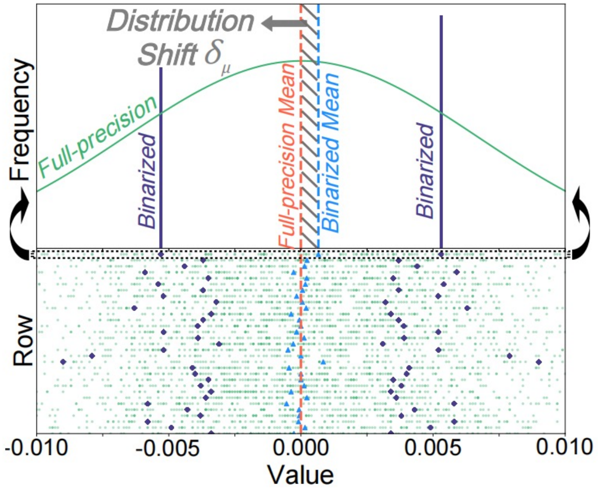


*Figure: **Top:** distribution shift of one row. **Bottom:** distribution shifts of multiple rows.*

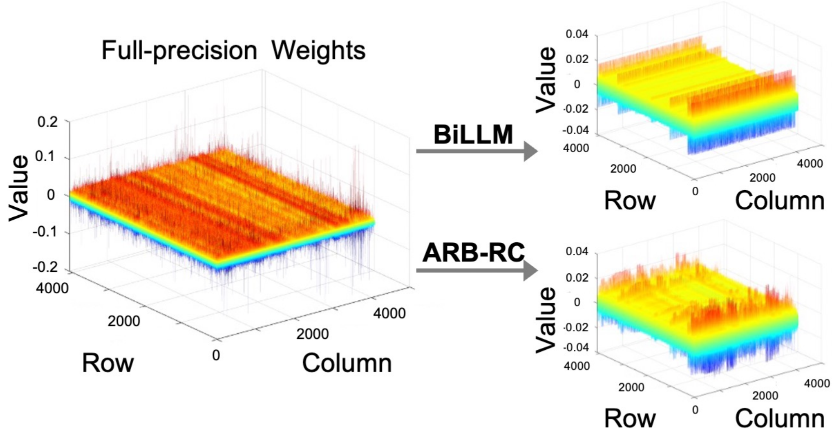
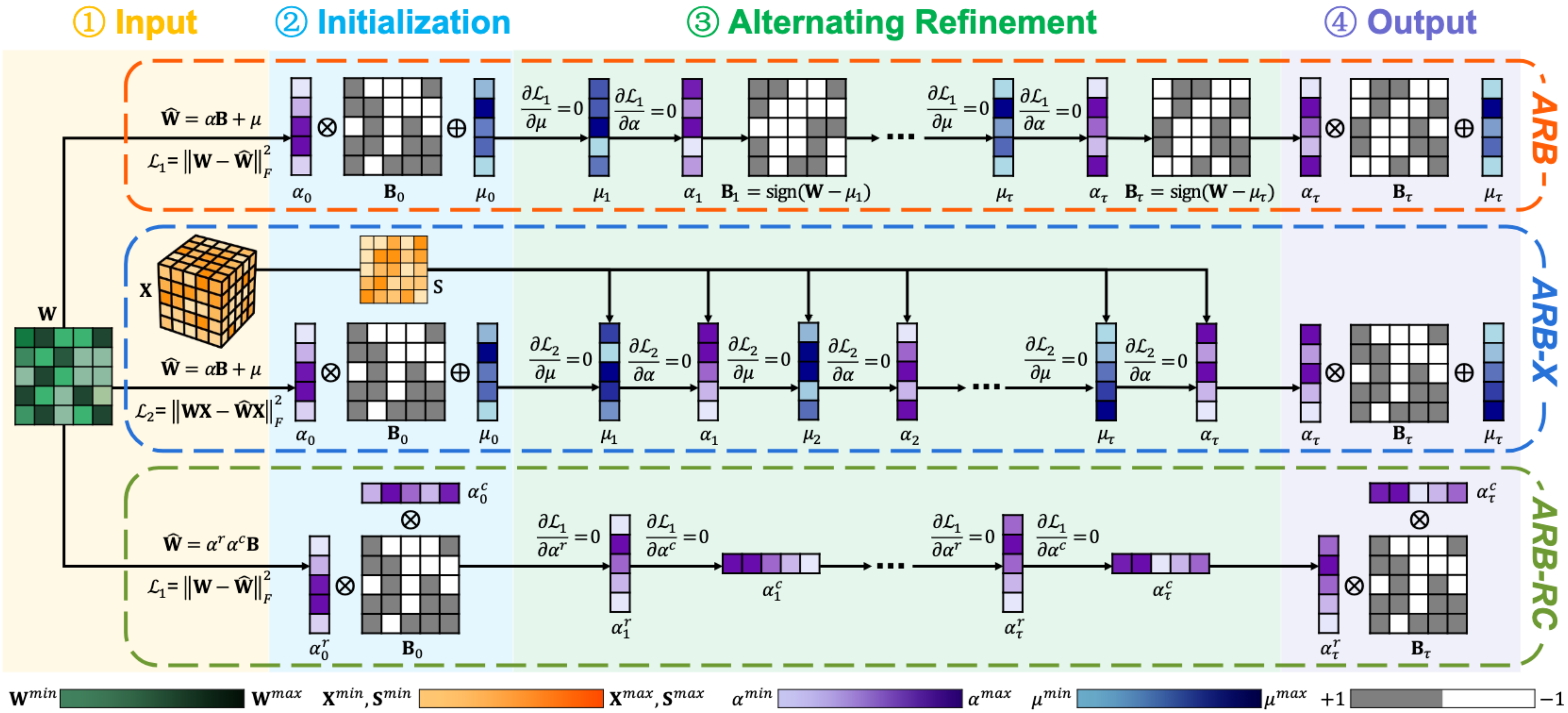


*Figure: BiLLM [1] with row-wise binarization smooths the deviations. Our ARB-RC with **row-column-wise binarization** effectively preserves them.*

[1] BiLLM: Pushing the Limit of Post-Training Quantization for LLMs. ICML, 2024.

Method

Overview



Method

Alternating Refined Binarization (ARB)

- **1st-order ARB**

Alternating Refine

① **Refine zero-point** (compensate *distribution shift*):

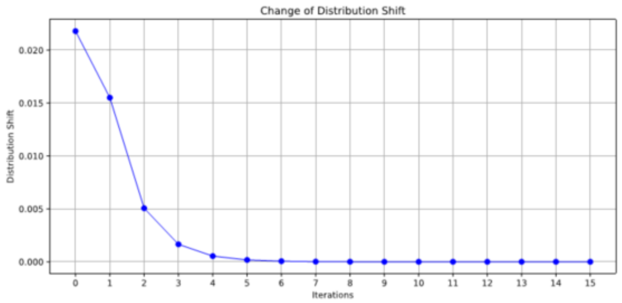
$$\mu \leftarrow \mu + \delta_\mu, \qquad \text{where } \delta_\mu \coloneqq \text{mean}(\mathbf{R})$$

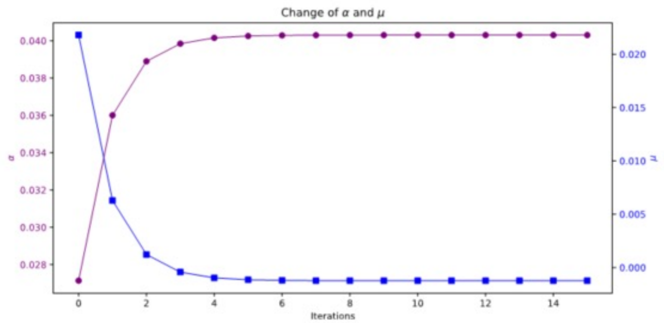
② **Refine scale:**

$$\alpha \leftarrow \frac{1}{m}\mathbf{B}^\top(\mathbf{W} - \mu)$$

③ **Refine binary matrix:**

$$\mathbf{B} \leftarrow \text{sign}(\mathbf{W} - \mu)$$





Theorem 1. *For any $\tau \geq 0$, Algorithm 1 achieves a quantization error $\mathcal{L}_1^\tau$ satisfying*

$$\mathcal{L}_1^\tau = \mathcal{L}_1^0 - m((\alpha^\tau)^2 - (\alpha^0)^2 - (\mu^\tau - \mu^0)^2) \leq \mathcal{L}_1^0, \tag{5}$$

where α^0 and μ^0 denote the initial scaling factor and mean respectively, α^τ, μ^τ, and $\mathcal{L}_1^\tau$ represent the scaling factor, mean, and quantization error after the τ-th iteration respectively.

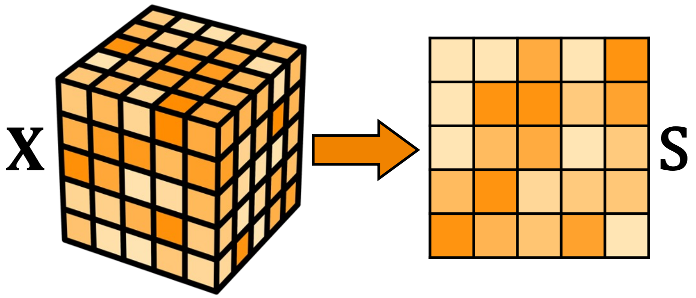
Method

ARB with Calibration Data (ARB-X)

- **Quantization Error**
  - **weight-only:** $\mathcal{L}_1 = \left\|\mathbf{W} - \widehat{\mathbf{W}}\right\|_F^2$
  - **weight-activation:** $\mathcal{L}_2 = \left\|\mathbf{W}\mathbf{X} - \widehat{\mathbf{W}}\mathbf{X}\right\|_F^2$, where $\mathbf{X} \in \mathbb{R}^{B\times L\times m}$ is a small calibration set.

Decoupling is needed to avoid re-computation and save memory!

- **Reformulation of $\mathcal{L}_2$**
  - Let $\mathbf{S} = \sum_b \mathbf{X}_b^\top \mathbf{X}_b$ and $\mathbf{R} = \mathbf{W} - \mu - \alpha\mathbf{B}$, where $\mathbf{S} \in \mathbb{R}^{m\times m}$.
  - Then $\mathcal{L}_2 = \left\|\mathbf{W}\mathbf{X} - \widehat{\mathbf{W}}\mathbf{X}\right\|_F^2 = \langle \mathbf{S}, \mathbf{R}^\top\mathbf{R}\rangle_F = \mathrm{Tr}(\mathbf{R}\mathbf{S}\mathbf{R}^\top)$.



Theorem 2. *The speedup ratio η of the reformulation compared to the original method is*

$$\eta \propto \frac{1}{k \cdot \left(\frac{1}{n\cdot T} + \frac{1}{B\cdot L}\right)}, \tag{10}$$

where n is the hidden dimension of $\mathbf{W}$, k is the block size, and T is the number of iterations.

Method

ARB along Row-Column Axes (ARB-RC)

- **1st-order ARB-RC**

  ⓪ **Initialize binary matrix (*unchanged*):**

$$\mathbf{B} \coloneqq \operatorname{sign}(\mathbf{W})$$

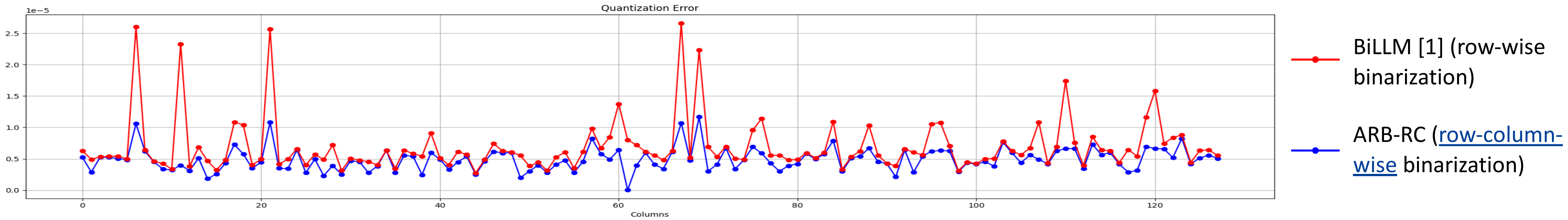
Alternating Refine

① **Refine row scale:**

$$\alpha^r \leftarrow \frac{\operatorname{diag}(\mathbf{W}(\alpha^c\mathbf{B})^\top)}{\operatorname{diag}((\alpha^c\mathbf{B})(\alpha^c\mathbf{B})^\top)} \Longleftarrow \frac{\partial \mathcal{L}_1}{\partial \alpha^r} = 0$$

② **Refine column scale:**

$$\alpha^c \leftarrow \frac{\operatorname{diag}(\mathbf{W}^\top(\alpha^r\mathbf{B}))}{\operatorname{diag}((\alpha^r\mathbf{B})^\top(\alpha^r\mathbf{B}))} \Longleftarrow \frac{\partial \mathcal{L}_1}{\partial \alpha^c} = 0$$



Method

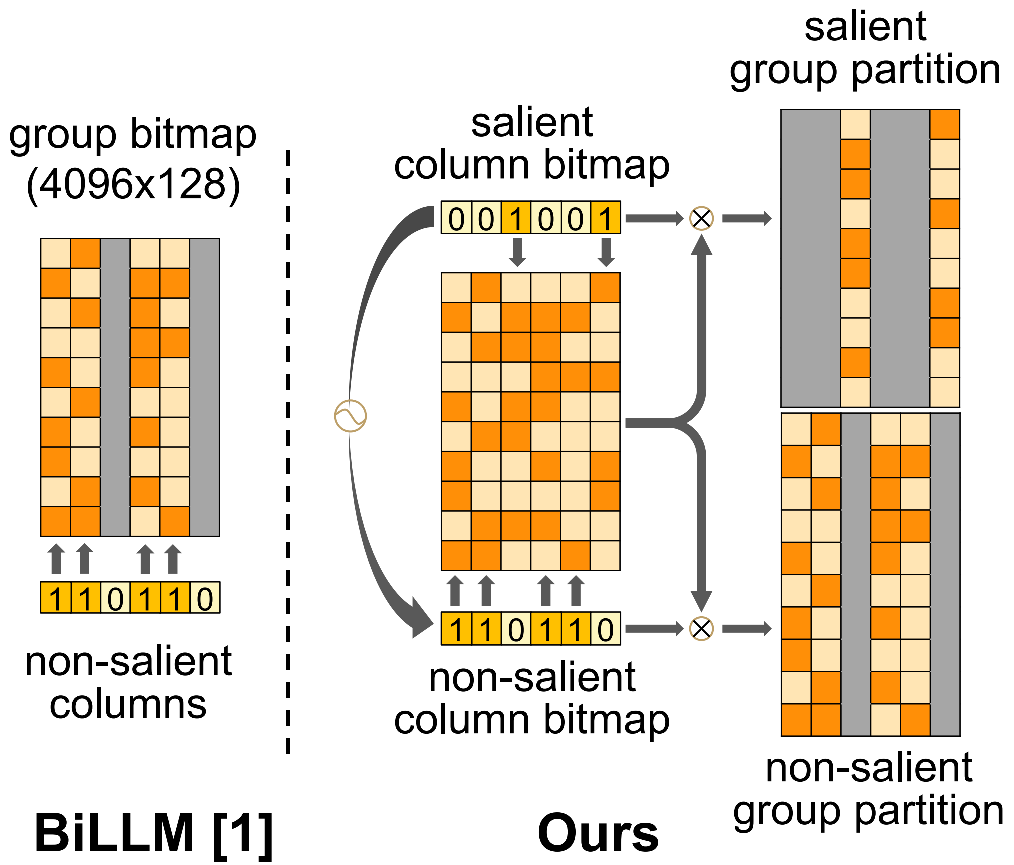
Column-Group Bitmap (CGB)

- **Salient and Non-salient Weights (column bitmap)**
  - **Hessian matrix** is used to identify salient weights:

$$s_i = \frac{w_i^2}{[\mathbf{H}^{-1}]_{ii}^2}.$$

  - Columns with higher s (salient columns) are quantized by *2nd-order binarization*, while non-salient columns are quantized by *1st-order binarization*.
- **Sparse and Concentrated Groups (group bitmap)**
  - Both salient and non-salient weights are further divided into 2 groups by their **magnitude**:

$$\mathbf{G_s} = \mathbf{1}_n\mathbf{C}_\mathbf{s}^\top \odot \mathbf{G}, \qquad \mathbf{G_{ns}} = \mathbf{1}_n\mathbf{C}_\mathbf{ns}^\top \odot \mathbf{G}$$



Experiments

Text Generation

Table 2: Perplexity of RTN, GPTQ, PB-LLM, BiLLM, and our methods on LLaMA family. The columns represent the perplexity results on **WikiText2** dataset with different model sizes. N/A: LLaMA-2 lacks a 30B version, and LLaMA-3 lacks both 13B and 30B versions. *: LLaMA has a 65B version, while both LLaMA-2 and LLaMA-3 have 70B versions.

Model	**Method**	**Block Size**	**Weight Bits**	**7B/8B***	**13B**	**30B**	**65B/70B***
LLaMA	Full Precision	-	16.00	5.68	5.09	4.10	3.53
	RTN	-	3.00	25.54	11.40	14.89	10.59
	GPTQ	128	3.00	8.63	5.67	4.87	4.17
	RTN	-	2.00	106,767.34	57,409.93	26,704.36	19,832.87
	GPTQ	128	2.00	129.19	20.46	15.29	8.66
	RTN	-	1.00	168,388.00	1,412,020.25	14,681.76	65,253.24
	GPTQ	128	1.00	164,471.78	131,505.41	10,339.15	20,986.16
	PB-LLM	128	1.70	82.76	44.93	23.72	12.81
	BiLLM	128	1.09	49.79	14.58	9.90	8.37
	ARB-LLM$_{\mathbf{X}}$	**128**	**1.09**	**21.81**	**11.20**	**8.66**	**7.27**
	ARB-LLM$_{\mathbf{RC}}$	**128**	**1.09**	**14.03**	**10.18**	**7.75**	**6.56**

Q&A Datasets

- **ARB-LLM** outperforms *binary PTQ methods* PB-LLM and BiLLM.
- **ARB-LLM** also surpasses *2-bit quantization method* GPTQ.

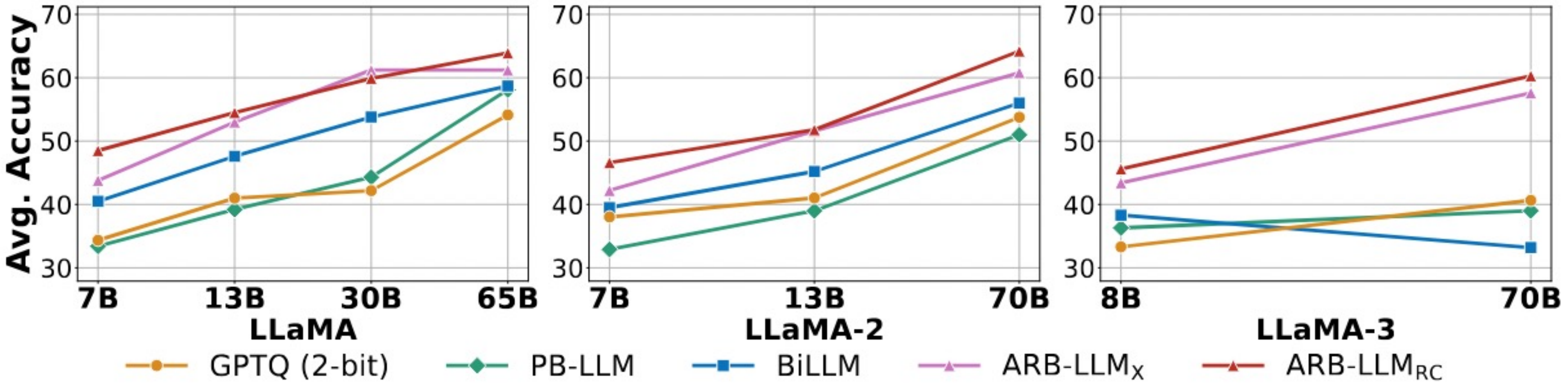


Figure 6: Average accuracy of 7 zero-shot QA datasets on LLaMA1&2&3 families.

Ablation Study

a) **ARB-X and ARB-RC** improve upon the base ARB.

b) **CGB** enhances the basic bitmap used in BiLLM.

c) **Column bitmap and group bitmap in CGB** significantly impact the performance.

d) **ARB-X** remains stable across **varying calibration set sizes**.

e) **ARB-LLM** performs well with **only a few iterations**.

f) **More groups** lead to better performance.

Table 4: Ablation study on LLaMA-7B, where all ARB methods are equipped with CGB except for ablation (b). Results are measured by perplexity, with final results highlighted in **bold**.

(a) Effectiveness of two advanced variants

Method	Calibration update	Row-column update	WikiText2↓	C4↓
BiLLM	-	-	49.79	46.96
ARB	✗	✗	22.67	26.44
ARB-LLM$_X$	✓	✗	**21.81**	**22.73**
ARB-LLM$_{RC}$	✗	✓	**14.03**	**17.92**

(b) Effectiveness of CGB

Method	CGB	WikiText2↓	C4↓
BiLLM	-	49.79	46.96
ARB-LLM$_X$	✗	26.29	27.11
ARB-LLM$_X$	✓	**21.81**	**22.73**
ARB-LLM$_{RC}$	✗	15.85	19.42
ARB-LLM$_{RC}$	✓	**14.03**	**17.92**

(c) Study of decoupling column and group bitmaps

Method	Column bitmap	Group bitmap	WikiText2↓	C4↓
ARB-LLM$_{RC}$	✗	✗	10,942.45	11,032.93
ARB-LLM$_{RC}$	✓	✗	369.20	205.56
ARB-LLM$_{RC}$	✗	✓	920.42	572.69
ARB-LLM$_{RC}$	✓	✓	**14.03**	**17.92**

(d) Study of ARB-LLM$_X$ calibration set size

Method	Calibration set size	WikiText2↓	C4↓
BiLLM	128	49.79	46.96
ARB-LLM$_X$	64	24.79	25.11
ARB-LLM$_X$	128	**21.81**	**22.73**
ARB-LLM$_X$	256	21.88	24.28

(e) Study of ARB-LLM iteration number

Method	#Iteration	WikiText2↓
BiLLM	0	49.79
ARB-LLM$_X$	1 / 3 / 15	22.59 / 21.12 / **21.81**
ARB-LLM$_{RC}$	1 / 3 / 15	15.23 / 14.34 / **14.03**

(f) Study of ARB-LLM group number

Method	#Group	WikiText2↓	C4↓
BiLLM	2	49.79	46.96
ARB-LLM$_X$	2 / 4	**21.81** / 6.55	**22.73** / 8.56
ARB-LLM$_{RC}$	2 / 4	**14.03** / 12.77	**17.92** / 16.06

Experiments

Time and Memory

- **Quantization Time**
  - ARB-LLM requires ***only a few additional minutes***, yet delivers significant improvements.
  - ARB-LLM$_X$ takes more time due to the integration of calibration data.

Table 5: Time comparison between BiLLM and our ARB-LLM methods on LLaMA-7B.

Method	CGB	#Iter=1	#Iter=3	#Iter=15
BiLLM	-	45 min (#Iter=0)		
ARB-LLM$_X$	✗	52 min	59 min	70 min
ARB-LLM$_X$	✓	72 min	78 min	88 min
ARB-LLM$_{RC}$	✗	48 min	49 min	53 min
ARB-LLM$_{RC}$	✓	67 min	68 min	76 min

- **Compressed Memory**
  - Although CGB requires more memory due to more scaling factors, the combination of **ARB-RC and CGB** still results in ***lower storage requirements than BiLLM.***

Table 6: Memory comparison between FP16, PB-LLM, BiLLM, and our ARB-LLM methods.

Method	CGB	LLaMA-7B	LLaMA-13B
FP16	-	13.48 GB	26.03 GB
PB-LLM	-	2.91 GB	5.33 GB
BiLLM	-	2.93 GB	5.36 GB
ARB-LLM$_X$	✗	2.93 GB	5.36 GB
ARB-LLM$_X$	✓	3.23 GB	5.95 GB
ARB-LLM$_{RC}$	✗	**2.63 GB**	**4.77 GB**
ARB-LLM$_{RC}$	✓	2.83 GB	5.17 GB

Conclusion

Contribution

- We propose **ARB-LLM**, a novel binarization framework with *theoretical analyses* for LLMs.
  - Based on **ARB**, we develop two advanced extensions: **ARB-X** and **ARB-RC**.
  - We propose **CGB**, which improves the bitmap utilization and enhances the performance.
- ARB-LLM$_{\mathrm{RC}}$ *outperforms SOTA* binary PTQ methods while requiring *less memory*.

Poster

- Time: Thu 24 Apr
  3 p.m. – 5:30 p.m. CST

About Me

Thanks!