Quant-dLLM: Post-Training Extreme Low-Bit Quantization for Diffusion Large Language Models

Tianao Zhang[1,2*], Zhiteng Li[1*], Xianglong Yan[1], Haotong Qin[3], Yong Guo[4], Yulun Zhang[1†]

[1]Shanghai Jiao Tong University, [2]Zhiyuan College, SJTU, [3]ETH Zürich, [4]Huawei

Binarization / INT1 Quantization

Symmetric and Asymmetric Binarization

$$\text{sign}(x) = \begin{cases} +1, & \text{if } x \geq 0, \\ -1, & \text{others.} \end{cases} \xrightarrow{\text{sym.}} \mathbf{W}_b = \alpha \cdot \text{sign}(\mathbf{W}_f)$$

$$\xrightarrow{\text{asym.}} \mathbf{W}_b = \alpha \cdot \text{sign}(\mathbf{W}_f - \mu) + \mu$$

- Typically, we define $\mu = \text{mean}(\mathbf{W}_f)$.
- Let $\mathbf{B} = \text{sign}(\mathbf{W}_f - \mu)$, $\widetilde{\mathbf{W}} = \mathbf{W}_f - \mu$.
- **Objective of Binarization:**

$$\arg\min_{\alpha,\mathbf{B}} \|\widetilde{\mathbf{W}} - \alpha\mathbf{B}\|_F^2$$

- Least square solution for the scaling factor:

$$\alpha = (\mathbf{B}^\top\mathbf{B})^{-1}\mathbf{B}^\top\widetilde{\mathbf{W}}$$

- Typically, for row-wise binarization (also referred to as per-channel binarization), we have

$$\alpha = \frac{1}{m}\mathbf{B}^\top\widetilde{\mathbf{W}}, \qquad \text{where } \mathbf{B} \in \mathbb{R}^m$$

Binarization / INT1 Quantization

Binarization alone row-column axes

$$\text{sign}(x) = \begin{cases} +1, & \text{if } x \geq 0, \\ -1, & \text{others.} \end{cases} \qquad \mathbf{W}_{\text{b}} = \alpha_{\text{r}} \alpha_c^{\text{T}} \cdot \text{sign}(\mathbf{W}_{\text{f}})$$

- Here, we assume symmetric quantization, so we define $\mu = 0$.
- Let $\mathbf{B} = \text{sign}(\mathbf{W}_f)$.
- **Objective of Binarization:**

$$\arg\min_{\alpha_{\text{r}}, \alpha_{\text{c}}, \mathbf{B}} \left\| \widetilde{\mathbf{W}} - (\alpha_{\text{r}} \alpha_{\text{c}}^{\text{T}}) \odot \mathbf{B} \right\|_{\text{F}}^{2}$$

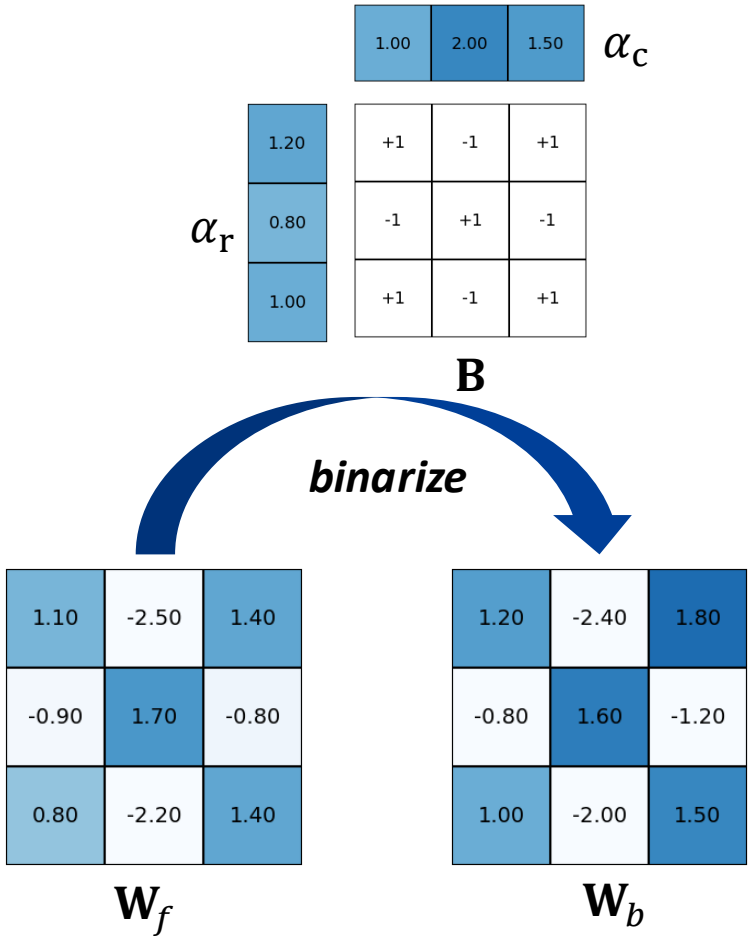
- Typically, for Binarization alone row-column axes, we have

$$\alpha_r = \frac{1}{m} \mathbf{B}^{\top} \widetilde{\mathbf{W}}, \qquad \text{where } \mathbf{B} \in \mathbb{R}^m$$

$$\alpha_{\text{c}} = \frac{1}{\text{n}} \mathbf{B}^{\top} (\text{diag}(\alpha_{\text{r}})^{-1} \widetilde{\mathbf{W}}), \qquad \text{where } \mathbf{B} \in \mathbb{R}^{\text{n}}$$

Binarization / INT1 Quantization

A toy example



$\mathbf{R} \neq 0$, Higher-order Binarization is required to further minimize the quantization error!

Higher-order Binarization

1st-order and knd-order Binarization

- **Definition**
  - **1st-order** binarization: $\widehat{\mathbf{W}} = (\alpha_r \alpha_c^T) \odot \mathbf{B}$
  - **knd-order** binarization: $\widehat{\mathbf{W}} = \sum (\alpha_{r,k} \alpha_{c,k}^T) \odot \mathbf{B_k}$

For outliers, it's more accurate but uses more memory.

- **Optimization**
  - **1st-order** binarization: $\mathbf{B} = \text{sign}(\mathbf{W_f})$, $\alpha_r = \frac{1}{m}\mathbf{B}^T\widetilde{\mathbf{W}}$, and $\alpha_c = \frac{1}{n}\mathbf{B}^T(\text{diag}(\alpha_r)^{-1}\widetilde{\mathbf{W}})$
  - **knd-order** binarization: how to obtain the optimal $\alpha_{r,1}, \alpha_{r,k}, \alpha_{c,1}, \alpha_{c,k}, \mathbf{B_1}$, and $\mathbf{B_k}$?
    - **Objective of Binary Residual Approximation**:

$$\arg\min_{\alpha_{r,1},\alpha_{c,1},\mathbf{B_1}} \left\|\mathbf{W} - (\alpha_{r,1}\alpha_{c,1}^T)\odot\mathbf{B_1}\right\|_F^2, \qquad \arg\min_{\alpha_{r,k},\alpha_{c,k},\mathbf{B_k}} \left\|\mathbf{R_{k-1}} - (\alpha_{r,k}\alpha_{c,k}^T)\odot\mathbf{B_k}\right\|_F^2$$

where $\mathbf{R_0} = \mathbf{W}$, $\mathbf{R_k} = \mathbf{R_{k-1}} - (\alpha_{r,k}\alpha_{c,k}^T)\odot\mathbf{B_k}$.

Diffusion LLM

LLaDA

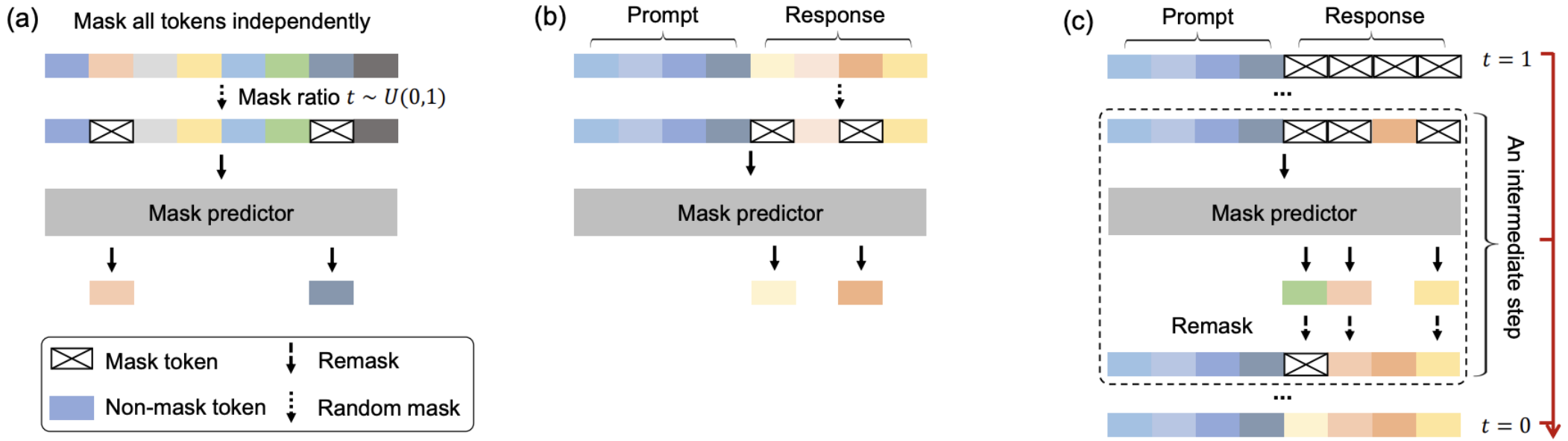


Figure 2: **Overview of LLaDA.** (a) Pre-training. LLaDA is trained on text with random masks applied independently to all tokens at the same ratio $t \sim U[0, 1]$. (b) SFT. Only response tokens are possibly masked. (c) Sampling. LLaDA simulates a diffusion process from $t = 1$ (fully masked) to $t = 0$ (unmasked), predicting all masks simultaneously at each step with flexible remask strategies.

Diffusion LLM

Dream

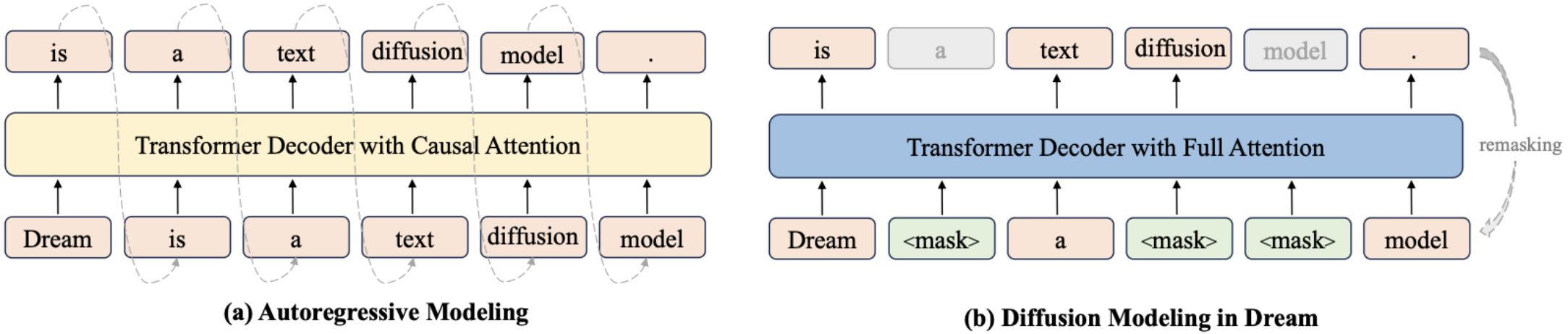


Figure 2: Comparison of autoregressive modeling and diffusion modeling in Dream. Dream predicts all the masked tokens in a shifted manner, allowing for maximumly architectural alignment and weight initialization with AR models.

Introduction

Observation

1. Static calibration creates a **distribution mismatch** with the dynamic masked activations of diffusion LLMs.
2. The optimization of binarization parameters α_r and α_c does not adequately reflect the real-world scenario, as it does not account for the **input data $\mathbf{X}$**.
3. Uniform bit-width allocation ignores the **column-wise clustering** of highly salient weights.

Solution

1. **Masked Calibration Simulation (MCS)**: To reduce the distribution mismatch, MCS generates timestep-aware, partially visible calibration inputs that mirror the model's native diffusion denoising process.
2. **Data-aware Any-order Quantizer (DAQ)**: To incorporate input data $\mathbf{X}$, DAQ optimizes a data-aware objective that directly minimizes output activation error.
3. **Adaptive Blockwise Mixed Precision (ABMP)**: To address uneven salience, ABMP calculates block-wise importance scores to dynamically assign mixed precision.

Method

Overview

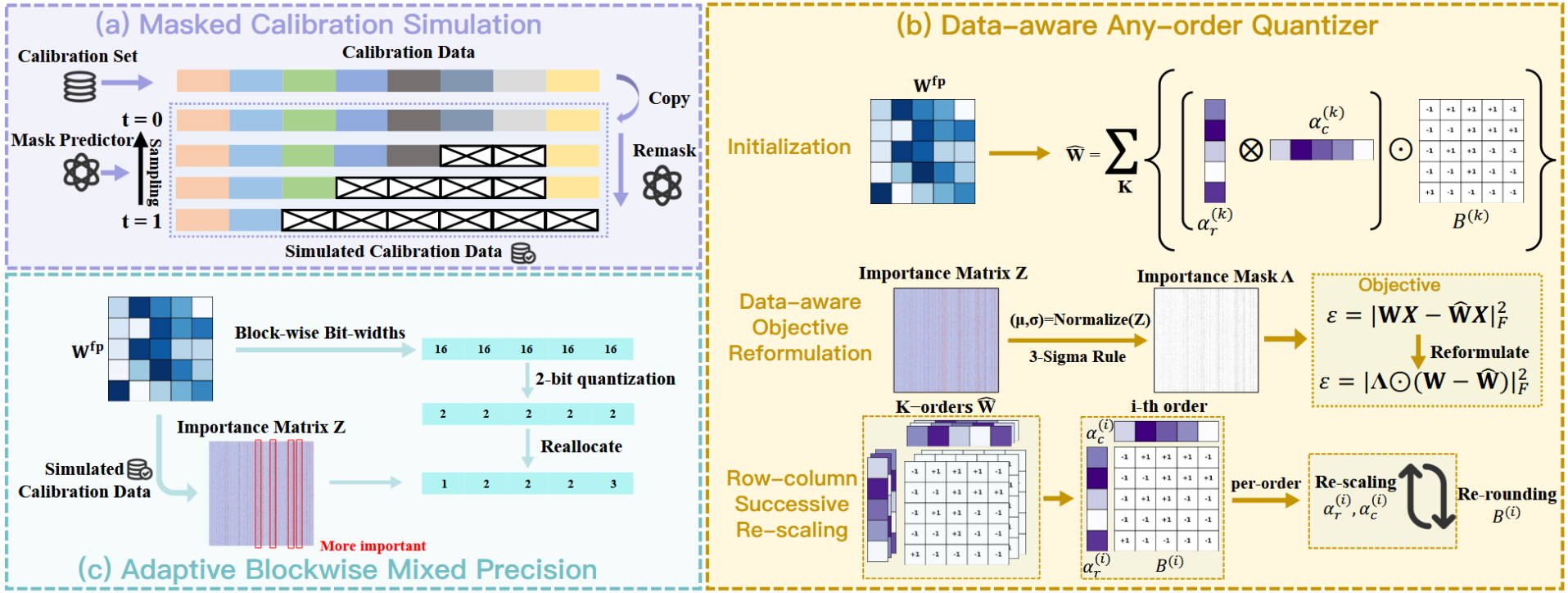


Figure 2: Overview of our Quant-dLLM. **Masked Calibration Simulation**: Aligns calibration with diffusion by simulating masked, timestep-aware inputs. **Adaptive Blockwise Mixed Precision**: Assigns binary orders by importance under a 2-bit average. **Data-aware Any-order Quantizer**: Builds multi-binary RC forms with data-aware optimization.

Method

Masked Calibration Simulation (MCS)

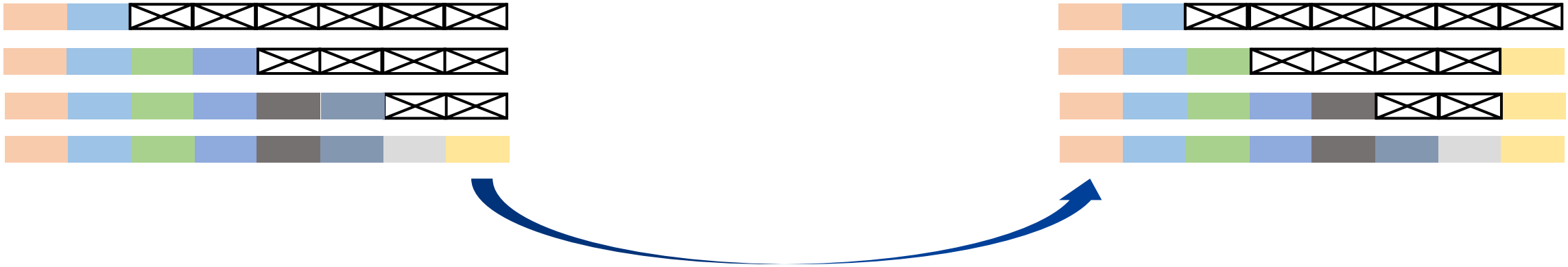
Conventional PTQ calibration assumes:

- Autoregressive activations.
- Fully visible tokens.
- Static token distribution.

distribution mismatch

In diffusion LLM:

- Token visibility depends on timestep.
- Activation distribution changes over denoising steps.



simulate the timestep-dependent masking pattern

Method

Masked Calibration Simulation (MCS)

- **Pseudocode of Masked Calibration Simulation**

Algorithm 1 MCS: Masked Calibration Simulation

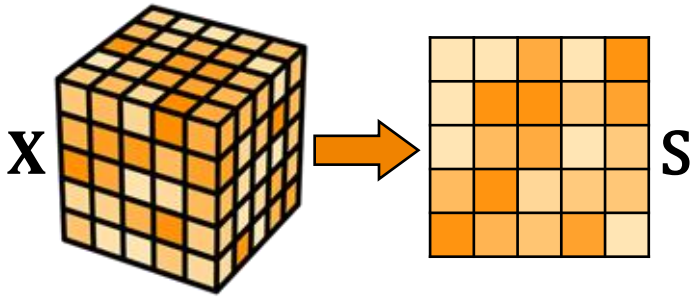
func MCS($\mathcal{X}, T, \gamma, \alpha(t)$)
Input: $\mathcal{X}$ — full sequences; T — total timesteps;
γ — prefix ratio; $\alpha(t)$ — visibility schedule
Output: Masked calibration set $\mathcal{D}_{\text{calib}}$

$\mathcal{D}_{\text{calib}} \leftarrow \emptyset$
for all $x \in \mathcal{X}$ **do**
  $L \leftarrow$ length of x
  $\mathcal{P} \leftarrow \{1, \ldots, \lfloor \gamma \cdot L \rfloor\}$
  for $t = 1$ to T **do**
    $\alpha \leftarrow \alpha(t)$
    for $i = 1$ to L **do**
      $r_i \leftarrow 1$ if $i \in \mathcal{P}$ else Bernoulli(α)
      $\tilde{x}_i(t) \leftarrow x_i$ if $r_i = 1$ else `[MASK]`
    end for
    $\mathcal{D}_{\text{calib}} \leftarrow \mathcal{D}_{\text{calib}} \cup \{\tilde{x}(t)\}$
  end for
end for
return $\mathcal{D}_{\text{calib}}$

Method

Data-aware Any-order Quantizer (DAQ)

- **Knd-order binarization**
  - **knd-order** binarization: $\widehat{\mathbf{W}} = \sum(\alpha_{\mathrm{r,k}}\alpha_{\mathrm{c,k}}^{\mathrm{T}}) \odot \mathbf{B_k}$
- **Quantization Error**
  - **weight-activation:** $\mathcal{L}_2 = \left\|\mathbf{WX} - \widehat{\mathbf{W}}\mathbf{X}\right\|_F^2$, where $\mathbf{X} \in \mathbb{R}^{B\times L\times m}$ is a small calibration set.
- **Reformulation of $\mathcal{L}_2$**
  - Let $\mathbf{S} = \sum_b \mathbf{X}_b^{\mathsf{T}}\mathbf{X}_b$ and $\mathbf{R} = \mathbf{W} - \widehat{\mathbf{W}}$, where $\mathbf{S} \in \mathbb{R}^{m\times m}$.
  - Then $\mathcal{L}_2 = \left\|\mathbf{WX} - \widehat{\mathbf{W}}\mathbf{X}\right\|_F^2 = \langle \mathbf{S}, \mathbf{R}^{\mathsf{T}}\mathbf{R}\rangle_F = \mathrm{Tr}(\mathbf{RSR}^{\mathsf{T}})$.
- **Consider MCS**
  - Then $\mathcal{L}_2 = \mathrm{Tr}((\mathbf{W} - \widehat{\mathbf{W}})\mathbf{S}_{\mathrm{MCS}}(\mathbf{W} - \widehat{\mathbf{W}})^{\mathsf{T}})$.



Method

Data-aware Any-order Quantizer (DAQ)

- **Data-aware Objective Reformulation (DOR)**
  - Quantization error is not uniformly distributed but concentrates on a small, critical subset of weights.
  - Importance matrix $\mathbf{Z} \in \mathbb{R}^{n \times m} = (\mathbf{W}\mathbf{D}^{-1}) \odot (\mathbf{W}\mathbf{D}^{-1})$, where $\mathbf{D} = \mathrm{diag}((\mathbf{S}_{\mathrm{MCS}} + \gamma\mathbf{I})^{-1})$.
  - Then standardize $\tilde{\mathbf{Z}} = (\mathbf{Z} - \mu) \oslash \sigma$.
  - Importance mask $\mathbf{\Lambda} = \mathbf{1} + (\lambda - 1)\mathbf{\Pi}$, where $\mathbf{\Pi} = \mathbf{I}(|\tilde{Z}| > 3)$.
- **Reformulation of $\mathcal{L}_\Lambda$**
  - $\mathcal{L}_\Lambda = \left\| \mathbf{\Lambda} \odot (\mathbf{W} - \widehat{\mathbf{W}}) \right\|_F^2 = \left\| \mathbf{\Lambda} \odot (\mathbf{W} - \sum(\alpha_{\mathrm{r,k}} \alpha_{\mathrm{c,k}}^{\mathrm{T}}) \odot \mathbf{B_k}) \right\|_F^2$.

Method

Data-aware Any-order Quantizer (DAQ)

- **Row-column Successive Re-scaling (RSR)**

⓪ **Initialize**

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

$$\alpha_r = \frac{1}{m}\mathbf{B}^{\top}\widetilde{\mathbf{W}}, \text{where } \mathbf{B} \in \mathbb{R}^{m}$$

$$\alpha_c = \frac{1}{n}\mathbf{B}^{\top}(\mathrm{diag}(\alpha_r)^{-1}\widetilde{\mathbf{W}}), \text{where } \mathbf{B} \in \mathbb{R}^{n}$$

$$\mathcal{L}_{\Lambda} = \left\|\mathbf{\Lambda} \odot (\mathbf{W} - (\alpha_r \alpha_c^{\mathrm{T}}) \odot \mathbf{B})\right\|_{\mathrm{F}}^{2}$$

Successive Re-scaling

① **Re-scale row factor:**

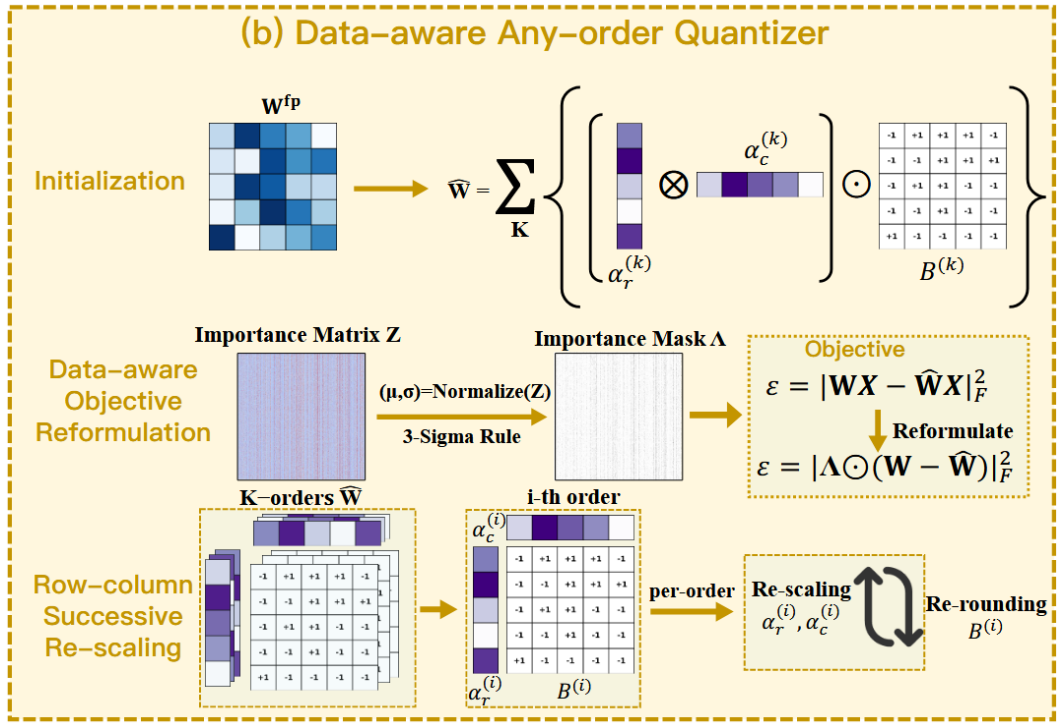
$$\alpha_r \leftarrow \frac{(\mathbf{\Lambda}^2 \odot \mathbf{W} \odot \mathbf{B})\alpha_c}{\mathbf{\Lambda}^2(\mathrm{diag}(\alpha_c)\alpha_c) + \varepsilon\mathbf{I}} \quad \longleftarrow \quad \frac{\partial \mathcal{L}_{\Lambda}}{\partial \alpha_r} = 0$$

② **Re-scale column factor:**

$$\alpha_c \leftarrow \frac{(\mathbf{\Lambda}^2 \odot \mathbf{W} \odot \mathbf{B})^{\mathrm{T}}\alpha_r}{\mathbf{\Lambda}^2(\mathrm{diag}(\alpha_r)\alpha_r) + \varepsilon\mathbf{I}} \quad \longleftarrow \quad \frac{\partial \mathcal{L}_{\Lambda}}{\partial \alpha_c} = 0$$

Method

Data-aware Any-order Quantizer (DAQ)



Algorithm 2 DAQ: Data-aware any-order Quantizer

func DAQ($\mathbf{W}, \mathbf{Z}, K, T, \lambda$)
Input $\mathbf{W}, \mathbf{Z} \in \mathbb{R}^{n\times m}$, $K, T \in \mathbb{N}$, $\lambda > 1$
Output

$$\hat{\mathbf{W}} = \sum_{k=1}^{K} \left(\boldsymbol{\alpha}_r^{(k)} \boldsymbol{\alpha}_c^{(k)\top}\right) \odot \mathbf{B}^{(k)}$$

$\boldsymbol{\Lambda} \coloneqq$ build_importance_mask($\mathbf{Z}, \lambda$)
$\hat{\mathbf{W}} \coloneqq \mathbf{0}_{n\times m}$
for $k = 1, 2, \ldots, K$ **do**
  $\mathbf{R} \leftarrow \mathbf{W} - \hat{\mathbf{W}}$
  $\boldsymbol{\alpha}_r^{(k)}, \boldsymbol{\alpha}_c^{(k)}, \mathbf{B}^{(k)} \leftarrow$ binary_rc_init($\mathbf{R}$)
  $\mathbf{S}^{(k)} \leftarrow \boldsymbol{\alpha}_r^{(k)} \left(\boldsymbol{\alpha}_c^{(k)}\right)^\top$
  $\hat{\mathbf{W}} \leftarrow \hat{\mathbf{W}} + \mathbf{S}^{(k)} \odot \mathbf{B}^{(k)}$
end for
for $t = 1, 2, \ldots, T$ **do**
  for $k = 1, 2, \ldots, K$ **do**
    $\hat{\mathbf{W}}_{\setminus k} \leftarrow \sum_{q\neq k} \left(\mathbf{S}^{(q)} \odot \mathbf{B}^{(q)}\right)$
    $\mathbf{R}^{(k)} \leftarrow \mathbf{W} - \hat{\mathbf{W}}_{\setminus k}$
    $\boldsymbol{\alpha}_r^{(k)} \leftarrow$ update_$\alpha_r(\mathbf{R}^{(k)}, \mathbf{B}^{(k)}, \boldsymbol{\alpha}_c^{(k)}, \boldsymbol{\Lambda})$
    $\boldsymbol{\alpha}_c^{(k)} \leftarrow$ update_$\alpha_c(\mathbf{R}^{(k)}, \mathbf{B}^{(k)}, \boldsymbol{\alpha}_r^{(k)}, \boldsymbol{\Lambda})$
    $\mathbf{S}^{(k)} \leftarrow \boldsymbol{\alpha}_r^{(k)} \left(\boldsymbol{\alpha}_c^{(k)}\right)^\top$
  end for
  $\{\mathbf{B}^{(k)}\}_{k=1}^K \leftarrow$ update_B$\left(\mathbf{W}, \{\boldsymbol{\alpha}_r^{(k)}\}_{k=1}^K, \{\boldsymbol{\alpha}_c^{(k)}\}_{k=1}^K\right)$
  $\hat{\mathbf{W}} \leftarrow \sum_{k=1}^K \mathbf{S}^{(k)} \odot \mathbf{B}^{(k)}$
end for
return $\hat{\mathbf{W}}$

func binary_rc_init($\mathbf{X}$)
$\boldsymbol{\alpha}_r \leftarrow \frac{1}{m} |\mathbf{X}| \mathbf{1}_m$
$\boldsymbol{\alpha}_c \leftarrow \frac{1}{n} |\mathbf{X}|^\top \operatorname{diag}(\boldsymbol{\alpha}_r)^{-1} \mathbf{1}_n$
$\mathbf{B} \leftarrow \operatorname{sign}(\mathbf{X})$
return $\boldsymbol{\alpha}_r$, $\boldsymbol{\alpha}_c$, $\mathbf{B}$

func update_α_r($\mathbf{X}$,$\mathbf{B}$,$\boldsymbol{\alpha}_c$,$\boldsymbol{\Lambda}$)
$\mathbf{u} \leftarrow (\boldsymbol{\Lambda}^2 \odot \mathbf{X} \odot \mathbf{B})\, \boldsymbol{\alpha}_c$
$\mathbf{v} \leftarrow (\boldsymbol{\Lambda}^2 (\operatorname{diag}\boldsymbol{\alpha}_c)\, \boldsymbol{\alpha}_c) + \varepsilon \mathbf{1}_n$
return $\mathbf{u} \oslash \mathbf{v}$

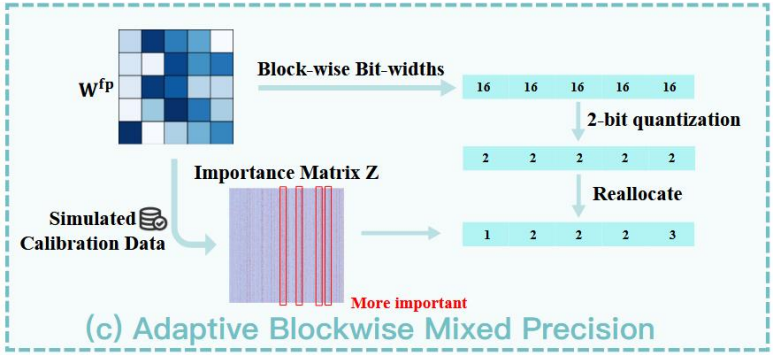
func update_α_c($\mathbf{X}$,$\mathbf{B}$,$\boldsymbol{\alpha}_r$,$\boldsymbol{\Lambda}$)
$\mathbf{u} \leftarrow (\boldsymbol{\Lambda}^2 \odot \mathbf{X} \odot \mathbf{B})^\top \boldsymbol{\alpha}_r$
$\mathbf{v} \leftarrow (\boldsymbol{\Lambda}^2)^\top (\operatorname{diag}(\boldsymbol{\alpha}_r)\, \boldsymbol{\alpha}_r) + \varepsilon \mathbf{1}_m$
return $\mathbf{u} \oslash \mathbf{v}$

func update_B$\left(\mathbf{W}, \{\boldsymbol{\alpha}_r^{(k)}\}_{k=1}^K, \{\boldsymbol{\alpha}_c^{(k)}\}_{k=1}^K\right)$
for $i = 1, \ldots, n$ **do**
  for $j = 1, \ldots, m$ **do**
    $\mathbf{s} \leftarrow \left[(\alpha_r^{(k)})_i (\alpha_c^{(k)})_j\right]_{k=1}^K$
    $\mathbf{b} \leftarrow$ search$\left(W_{ij}, \mathbf{s}\right)$
    for $k = 1, \ldots, K$ **do**
      $(\mathbf{B}^{(k)})_{ij} \leftarrow \mathbf{b}_k$
    end for
  end for
end for
return $\{\mathbf{B}^{(k)}\}_{k=1}^K$

func build_importance_mask($\mathbf{Z}, \lambda$)
$\mu \leftarrow$ mean($\mathbf{Z}$), $\sigma \leftarrow$ std($\mathbf{Z}$)
$\mathbf{M} \leftarrow \mathbb{I}\left(|\mathbf{Z} - \mu| > 3\sigma\right)$
return $\mathbf{1}_{n\times m} + (\lambda - 1)\, \mathbf{M}$

Method

Adaptive Blockwise Mixed Precision (ABMP)



Algorithm 3 ABMP: Adaptive Blockwise Mixed Precision

1: Given block partition $\mathcal{G}$ and importance $\mathbf{Z}$
2: Compute $s_g = \sum_{(i,j)\in g} \mathbf{Z}_{ij}$ for all $g \in \mathcal{G}$
3: Set $k \leftarrow \lfloor 0.05\,|\mathcal{G}| \rfloor$ and sort blocks by s_g
4: Assign $b_g = 3$ to top-k, $b_g = 1$ to bottom-k, others $b_g = 2$ $\quad (\frac{1}{|\mathcal{G}|}\sum_g b_g = 2)$
5: Apply DAQ quantization to each block using its assigned order $K = b_g$.

Experiments

Q&A Datasets

Table 1: Results of GPTQ, GPTAQ, Slim-LLM, and our Quant-dLLM with 2-bit weight quantization among 7 tasks on LLaDA-Base, LLaDA-Instruct, LLaDA-1.5, Dream-Base, and Dream-Instruct. The numbers in parentheses represent the number used for evaluation. Best results are marked in **bold**.

Model	Method	MMLU(5)	Wino.(5)	PIQA(0)	ARC-C(0)	ARC-E(0)	Hella.(0)	BBH(0)	Avg
LLaDA-8B-Base	FP	65.76	74.59	74.70	44.11	74.20	54.27	42.60	61.46
	GPTQ	34.75	57.85	57.45	22.44	37.04	33.77	4.06	35.34
	GPTAQ	34.73	59.04	56.42	22.27	38.17	35.13	5.34	35.87
	Slim-LLM	47.98	60.85	62.46	26.11	53.41	39.25	6.64	42.39
	Quant-dLLM	**56.87**	**68.19**	**69.75**	**36.26**	**68.69**	**46.45**	**32.18**	**54.06**
LLaDA-8B-Instruct	FP	63.91	72.30	74.43	53.24	79.34	53.95	47.89	63.58
	GPTQ	42.15	58.88	61.21	27.22	51.18	37.46	3.93	40.29
	GPTAQ	44.94	60.22	61.04	29.01	52.44	37.18	4.55	41.34
	Slim-LLM	50.35	60.69	65.72	34.73	65.57	41.34	27.48	49.41
	Quant-dLLM	**54.07**	**65.67**	**70.73**	**43.86**	**71.97**	**47.39**	**35.02**	**55.53**
LLaDA-1.5	FP	64.21	54.38	74.54	54.27	79.80	54.38	56.27	62.55
	GPTQ	48.48	37.47	61.48	30.20	52.90	37.47	6.91	39.27
	GPTAQ	47.26	35.89	58.81	28.41	51.39	35.89	5.38	37.58
	Slim-LLM	44.99	41.86	65.89	33.70	64.86	41.86	24.90	45.44
	Quant-dLLM	**54.32**	**47.82**	**71.16**	**47.18**	**73.06**	**47.82**	**38.09**	**54.21**
Dream-7B-Base	FP	69.43	73.56	74.59	55.38	82.58	54.42	50.12	65.73
	GPTQ	23.84	50.51	53.86	19.88	28.70	29.69	3.08	29.94
	GPTAQ	23.90	52.49	55.88	20.82	30.43	30.98	4.00	31.21
	Slim-LLM	25.54	51.62	54.24	20.05	31.14	30.83	4.18	31.09
	Quant-dLLM	**40.22**	**59.19**	**63.71**	**29.35**	**54.29**	**40.87**	**25.59**	**44.75**
Dream-7B-Instruct	FP	69.71	72.93	75.14	57.25	83.80	54.43	57.92	67.31
	GPTQ	23.96	50.59	54.90	18.52	30.77	30.11	5.32	30.60
	GPTAQ	24.32	49.72	56.42	20.73	31.82	30.97	5.11	31.30
	Slim-LLM	25.29	51.93	56.09	20.56	36.99	31.55	7.61	32.86
	Quant-dLLM	**43.14**	**57.93**	**66.59**	**33.62**	**62.96**	**41.36**	**30.35**	**47.99**

Experiments

Mathematical & Scientific Reasoning

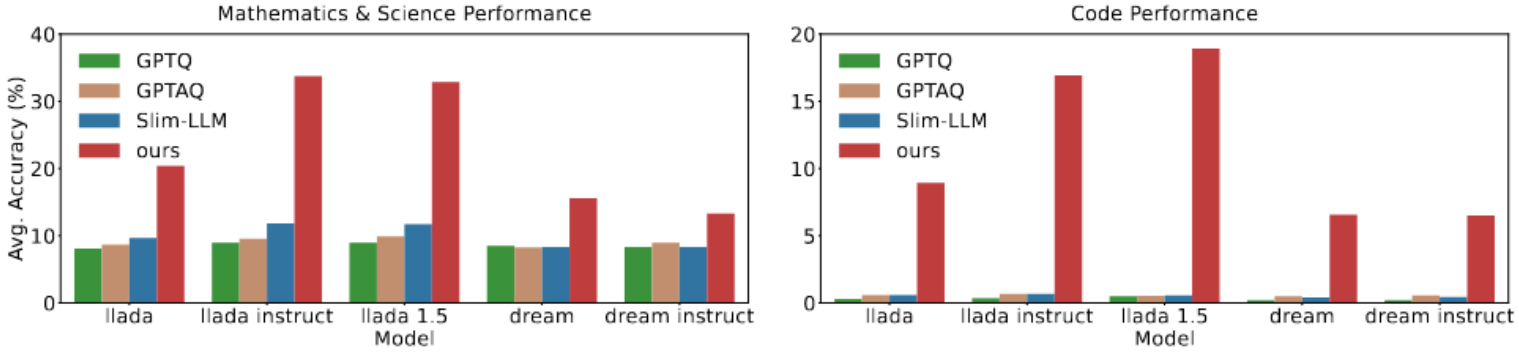


Figure 3: Average accuracy of mathematical & scientific reasoning, and code generation datasets on LLaDA series and Dream series.

Experiments

Ablation Study

a) Effectiveness of our **Masked Calibration Simulation.**

b) Effectiveness of **Adaptive Blockwise Mixed Precision.**

c) Both **Data-aware Objective Reformulation** and **Row-column Successive Re-scaling** can improve the performance.

d) Quant-dLLM remains stable across **varying calibration set sizes**.

Table 2: Ablation studies on LLaDA-8B-Base and Dream-7B-Base. We report MMLU in 5 shots.

(a) Effectiveness of MCS

Model	MCS	MMLU(5) ↑
LLaDA-8B-Base	✗	52.10
	✓	56.87
Dream-7B-Base	✗	37.81
	✓	40.22

(b) Effectiveness of ABMP

Model	ABMP	0%	5%	10%	15%
LLaDA-8B-Base	✗	54.32	–	–	–
	✓	–	56.87	55.87	53.01
Dream-7B-Base	✗	32.75	–	–	–
	✓	–	34.50	40.22	31.11

(c) Ablation Study for DAQ

Model	DAQ Component	MMLU(5) ↑
LLaDA-8B-Base	baseline	39.26
	RSR w/o DOR	48.32
	RSR w/ DOR	56.87
Dream-7B-Base	baseline	27.84
	RSR w/o DOR	34.73
	RSR w/ DOR	40.22

(d) Ablation Study for Calibration Set Size.

Model	Calib. Data Size	MMLU(5) ↑
LLaDA-8B-Base	64	54.49
	128	56.87
	256	55.59
Dream-7B-Base	64	38.24
	128	40.22
	256	39.64

Experiments

More Result

Table 1: Effectiveness of MCS on previous methods. We report MMLU in 5 shots.

((a)) LLaDA-8B-Base

Model	MCS	Method	MMLU(5) ↑
LLaDA-8B-Base	✗	GPTQ	34.75
	✓	GPTQ	36.75
	✗	GPTAQ	34.73
	✓	GPTAQ	37.92
	✗	Slim-LLM	47.98
	✓	Slim-LLM	50.11

((b)) Dream-7B-Base

Model	MCS	Method	MMLU(5) ↑
Dream-7B-Base	✗	GPTQ	23.84
	✓	GPTQ	24.08
	✗	GPTAQ	23.90
	✓	GPTAQ	24.68
	✗	Slim-LLM	25.54
	✓	Slim-LLM	26.01

Experiments

Memory

- **Compressed Memory**
  - Quant-dLLM achieves the smallest model size among all schemes, thanks to the memory-efficient row–column scaling factor design.

Table 3: Model size of LLaDA-8B-Base under different methods.

Model	Method	Bit	Memory (GB)
LLaDA-8B	FP16	16	16.09
	GPTQ	2	3.70
	Slim-LLM	2	3.72
	Quant-dLLM	2	3.69

Conclusion

Contribution

- We propose **Quant-dLLM** , a novel post-training extreme low-bit quantization framework for dLLMs.
  - We present **Masked Calibration Simulation**, reducing the distribution mismatch of calibration.
  - We introduce **Data-aware Any-order Quantizer**, including Data-aware Objective Reformulation (DOR) and Row-column Successive Re-scaling (RSR).
  - We propose **Adaptive Blockwise Mixed Precision**, concentrating representational capacity on the most critical model components.
- Quant-dLLM *outperforms SOTA* 2-bit PTQ methods while requiring *less memory*.

Thanks!