The 21st Annual IFIP International Conference on Network and Parallel Computing（NPC 2025）

HOHAI UNIVERSITY

Cloud-Guided LLM Quantization for Resource-Constrained Edge Devices

Qinxiao Deng[1], Tianfu Pang[1], Benteng Zhang[1],Nie Bingbing[2], Xiaoming He[3], Yingchi Mao[1],Jie Wu[4]

[1] College of Computer Science and Software Engineering, Hohai University, Nanjing, China

[2] Huaneng Lancang River Hydropower Inc., China

[3] College of Internet of Things, Nanjing University of Posts and Telecommunications, Nanjing, China

[4] Center for Networked Computing, Temple University, Philadelphia, USA

November 15, 2025

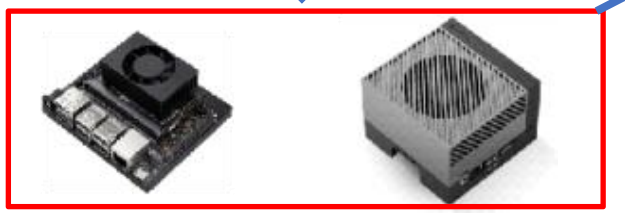
1.1 Background

Large Language Model (LLM)
GPT3 (175B)
Storage: 350 GB(FP16)

Edge device (ED)

- **Dialogue Generation**
- **Information Retrieval**
- **Intelligent Decision-Making**

- Insufficient Computing Power
- Storage Capacity Limitations

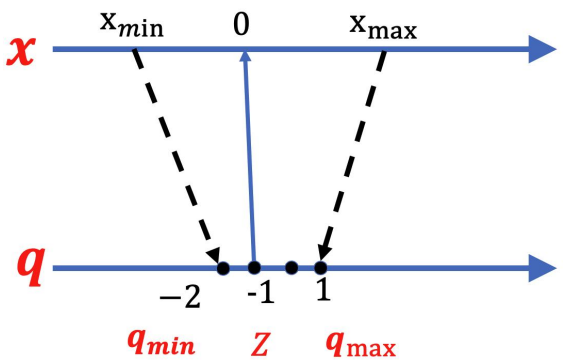


Jetson AGX Orin
Jetson Orin NX

Model Compression → **Solutio** → **Edge Deployment**

1.2 What is Quantization?

Model Compression → ➢ Pruning ➢ Distillation ➢ Quantization → **Post-Training Quantization(PTQ)**



Weights
32bit Full Precision

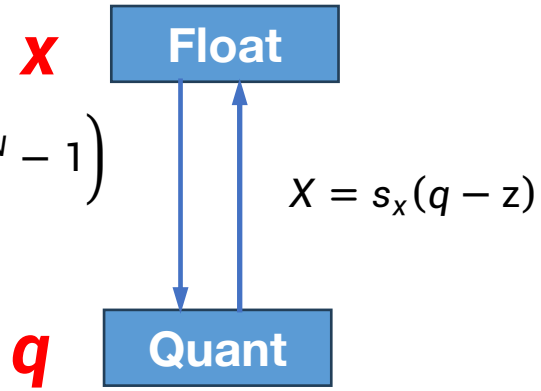
2.09	-0.96	1.48	0.09
0.05	-0.14	-1.06	2.12
-0.91	1.92	0	-1.03
1.57	0	1.53	1.49

Quantization
2bit INT

1	-2	0	1
1	1	2	1
2	1	1	2
1	1	0	0

$$q = clamp\left(\left[\frac{x}{s_x}\right] + z, 0, 2^N - 1\right)$$

$$s_x = \frac{q_{max} - q_{min}}{2^N - 1}$$



$$X = s_x(q - z)$$

- **Pruning**
  - ➢ Eliminate the unimportant weights
  - ➢ Requires Retraining the Model(High Cost)
- **Distillation**
  - ➢ Transfer large-model knowledge to student model
  - ➢ The student model will still exceed the device's burden
- **Quantization**
  - ➢ Quantization-Aware Training(QAT): Requires Retraining the Model (High Cost)
  - ➢ Post-Training Quantization(PTQ): Quantize pre-trained floating-point models ✓

1.3 The Challenges of PTQ(Post Training Quantization)

Challenge1: Quantization below 8-bit causes severe accuracy loss

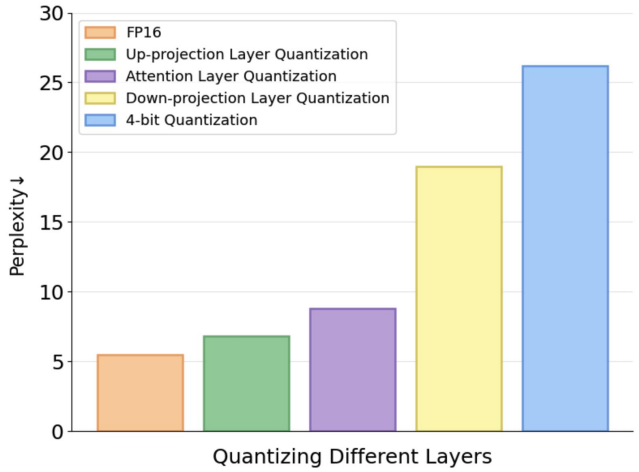
Outliers cause severe accuracy degradation in low-bit quantization → ZeroQuant[1]、SmoothQuant[2]、AWQ [3]、OS+[4] perform special handling for outliers

Challenge2: Outliers across channels and tokens in KV cache quantization

KVquant[5]、IntactKV[6] smooth key-value outliers via static equivalent parameters → Fail to capture dynamic token/channel distributions.

Challenge3: Different modules exhibit varying sensitivity to quantization

- They rely on global **static equivalent transformation**.
- Inability to precisely adapt to diverse channel distributions → unhandled outliers in some channels.
- 4-bit LLM quantization: severe accuracy loss.



[1] *Yao Z, Yazdani Aminabadi R, Zhang M, et al. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. In: NeurIPS, pp. 35: 27168-27183 (2022).*
[2] *Xiao G, Lin J, Seznec M, et al.: Smoothquant: Accurate and efficient post-training quantization for large language models. In: PMLR, pp. 38087-38099 (2023).*
[3] *Ji Lin, Jiaming Tang , et al. AWQ: activation-aware weight quantization for LLM compression and acceleration. CoRR, abs/2306.00978.*
[4] *Xiuying Wei, Yunchen Zhang, Yuhang Li, et al.: Outlier suppression+: Accurate quantization of large language models by equivalent and effective shifting and scaling. In: Empirical Methods in Natural Language Processing, pp. 1648–1665 (2023).*
[5] *Hooper C, Kim S, Mohammadzadeh H, et al. Kvquant: Towards 10 million context length llm inference with kv cache quantization. In NeurIPS, pp. 37: 1270-1303(2024).*
[6] *Liu R, et al. IntactKV: Improving Large Language Model Quantization by Keeping Pivot Tokens Intact. In: ACL 2024.*

Outline

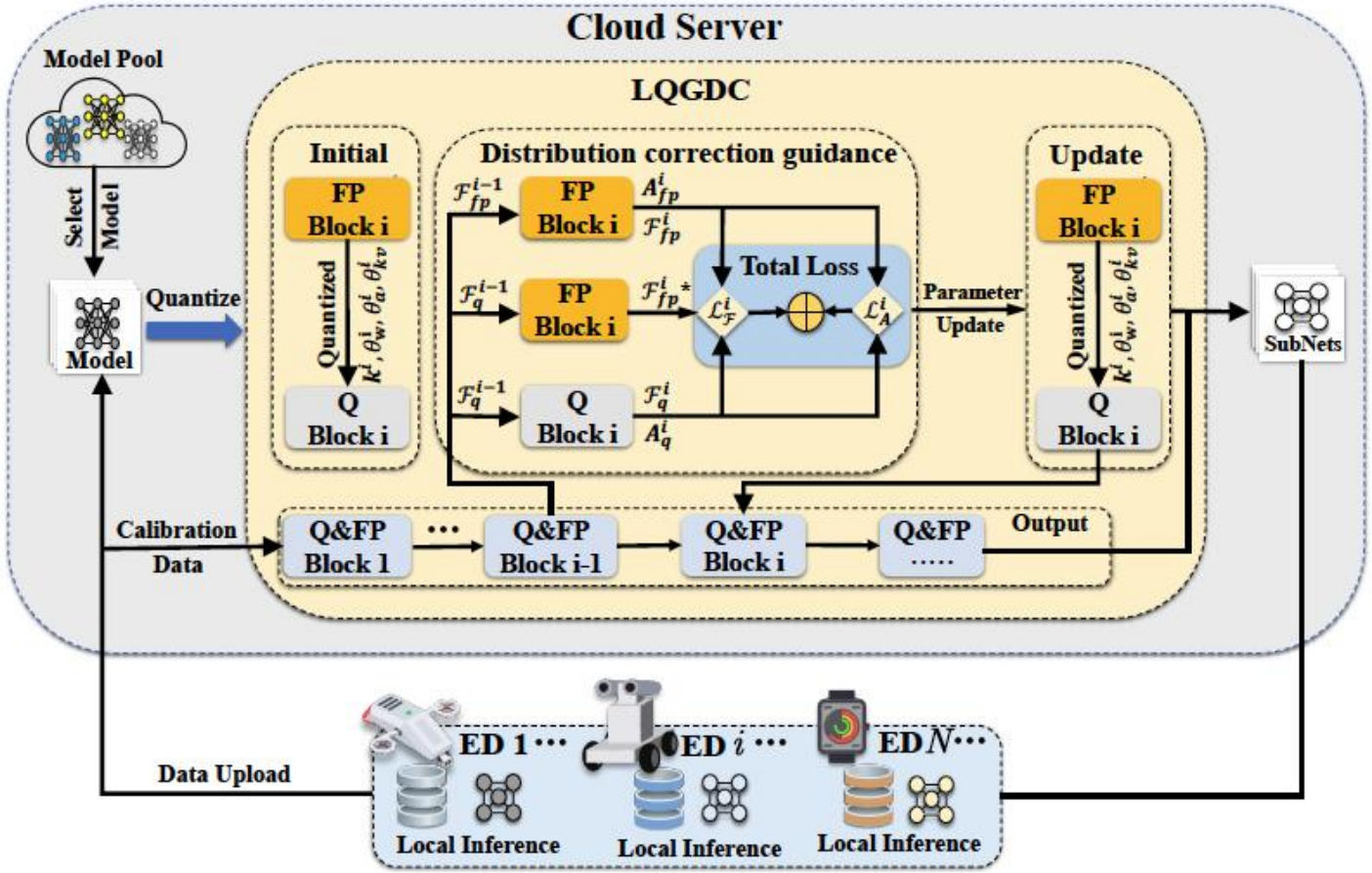
1. Introduction

2. Approach

3. Experimental Evaluation

4. Conclusion

2.1 Approach: LQGDC(*Learnable Quantization Guided by Distribution Correction*)



Phase 1:

Model Selection and Data Upload.

Phase 2:

Cloud-Side Quantization and Optimization

1. *Learnable Transformation*
2. *Loss Calculation and Optimization*

Phase 3:

Model Distribution and Local Inference

2.1 Approach: LQGDC(*Learnable Quantization Guided by Distribution Correction*)

Challenge2: Quantization below 8-bit causes severe accuracy loss

Challenge3: Outliers across channels and tokens in KV cache quantization

Challenge4: Different modules exhibit varying sensitivity to quantization

Transformer Block Parameter & Inference

Challenge2

Learnable Transformation

- **LWAS: Learnable weights-activations smooth**
- **LWAC: Learnable Weight-Activation Clipping**

Challenge3

Learnable Scale and Offset Quantization for KV Cache

- ***LKVTS: Learnable KV Cache Transformation and Smoothing***
- **HKVQ:*Historical KV Quantization***

Parameter Initial

Parameter Update

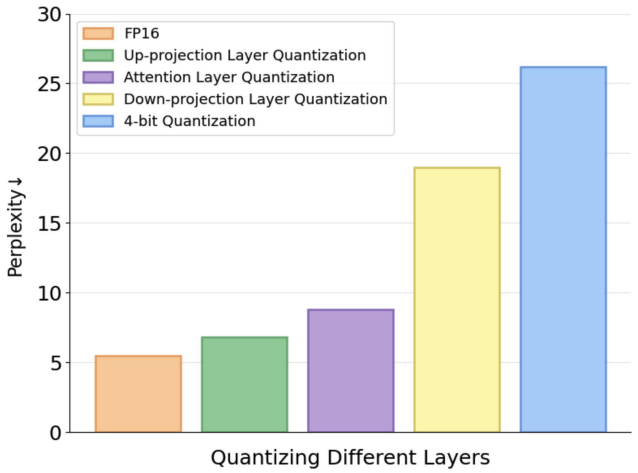
Challenge4

Optimization Module

- **DCG: Distribution Correction-Guided**

2.1 Approach: LQGDC(*Learnable Quantization Guided by Distribution Correction*)

Address accuracy degradation of low-bit quantized models caused by static quantization parameters



Mitigate the issue of varying quantization sensitivity across different modules

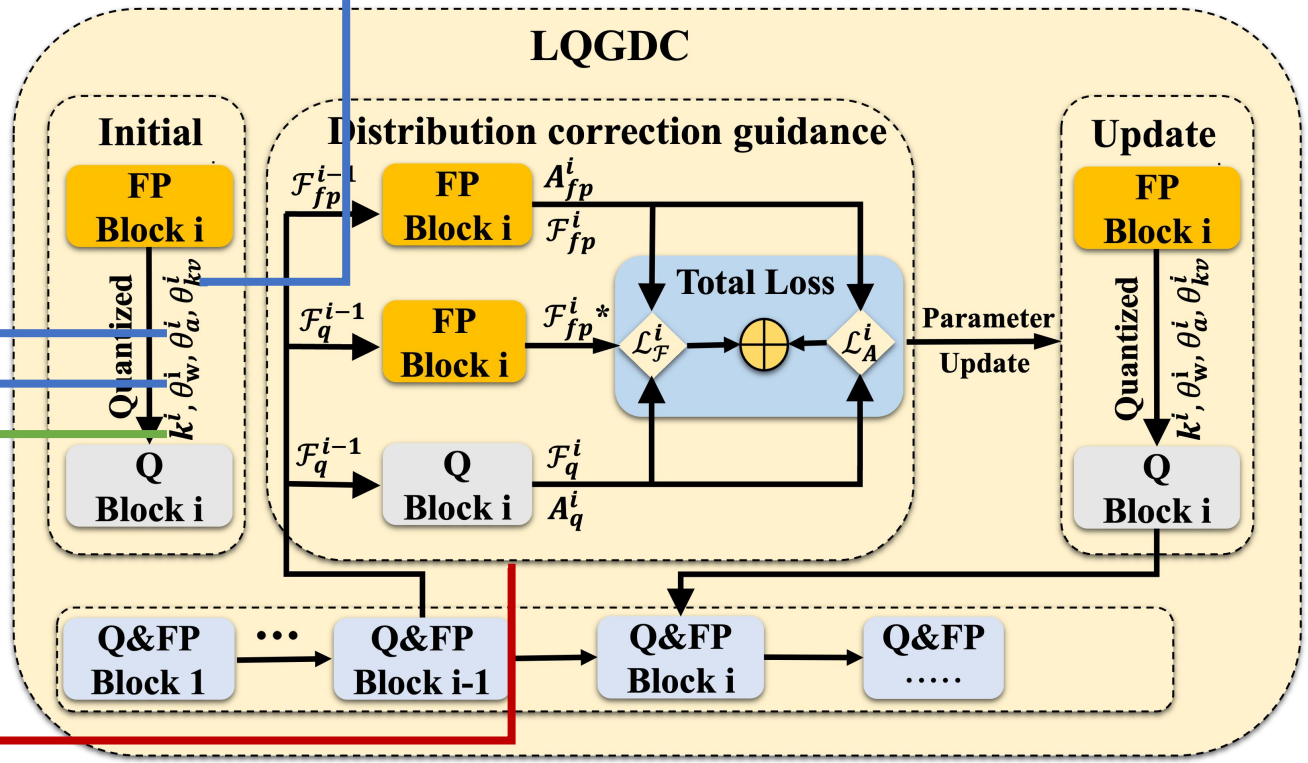
LWAC

LWAC

LWAS

LKVTS

Outliers across channels and tokens in KV cache quantization



DCG

2.2 Learnable Transformations

Quantization Target

Main calculations:

$Y = XW$

challeng

outlier s → low precision

Various parameter distributions

solution

Learnable Transformations

1.LWAS: Learnable weights-activations smooth

$$Y = \left(X \cdot diag(k)^{-1}\right)(diag(k) \cdot W) + B = \widehat{X} \cdot \widehat{W} + B$$

Learnable per-channel factors *k* mitigate outliers

- Balance difficulty of quantization between weights and activations
- Dynamically adjusts the distribution of weights and activation

Even after smoothing, residual outliers may persist

2. LWAC: Learnable Weight-Activation Clipping

$$s_x = \frac{\alpha_w \max(\widehat{W}) - \beta_w \min(\widehat{W})}{2^N - 1}$$

$$z = -\left[\frac{\beta_w \min(\widehat{W})}{s_x}\right], \theta_w = \{\alpha_w, \beta_w\}$$

Dynamically adjusts the quantization range

$$W_q = clamp\left(\left[\frac{\widehat{W}}{s_x}\right] + z, 0, 2^N - 1\right)$$

Preventing residual outliers from consuming low-bit width representation

summary

Learnable parameters — **Based on distribution**

① smooth activation and weights
② adjust the quantization clipping range

2.3 *Learnable Scale and Offset Quantization for KV Cache*

Quantization challenge

- *Demand for long-text inference*

Solution

Learnable KV Cache Transformation and Smoothing (LKVTS)

3. *LKVTS:Learnable KV Cache Transformation and Smoothing*

- Challenge: Quantization errors due to dynamic distribution of KV cache
- **Solution**: Smooth the KV Cache through **Learnable parameter $\boldsymbol{\theta_{kv} = \{\sigma, \gamma\}}$:**

$$Y_{kv} = \frac{Y_{kv} - \sigma}{\gamma} \cdot \gamma + \sigma \rightarrow \widetilde{Y_{kv}}$$

- Scale and translate the KV Cache , quantize $\widetilde{Y_{kv}}$ maintain equivalence

①

① Symmetrical channel distribution to **quantize easier**

② Adjust numerical distribution of KV to symmetrical distribution based on *M*, reducing quantization error

②

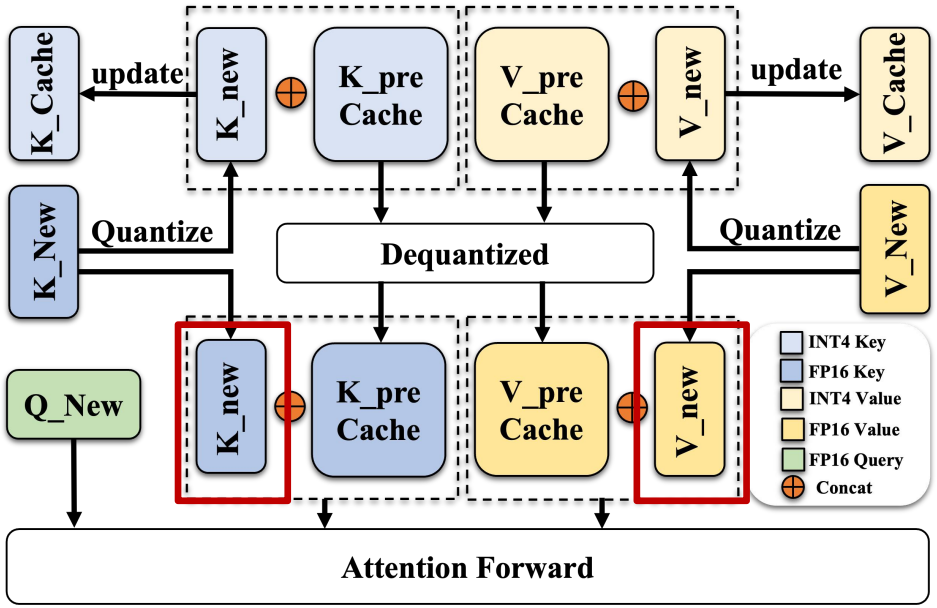
3. *LKVTS:Learnable KV Cache Transformation and Smoothing*

- **Adjust the quantization range of each token** to reduce the error due to outliers in KV Cache

$$\widetilde{Y_q} = clamp\left(\left[\frac{\widetilde{Y_{kv}} - M}{s_{kv}}\right], 0, 2^N - 1\right),$$

$$s_{kv} = \frac{\max\left(\left|\widetilde{Y_{kv}} - M\right|\right)}{2^N - 1}, \quad M = \text{Median}\left(\widetilde{Y_{kv}}\right).$$

2.3 *Learnable Scale and Offset Quantization for KV Cache*



Inference in LLM:

- Pre-fill stage

$$X_K = XW_K, X_V = XW_V$$

- Decoding stage

$$t_K = tW_K, t_V = tW_V, X_K \leftarrow Concat(X_K, t_K), X_V \leftarrow Concat(X_V, t_V)$$

$$t_Q = tW_Q, A = Softmax(t_Q X_K^T), t_O = AX_V$$

- Introduce quantization errors and reducing the accuracy of attention calculations

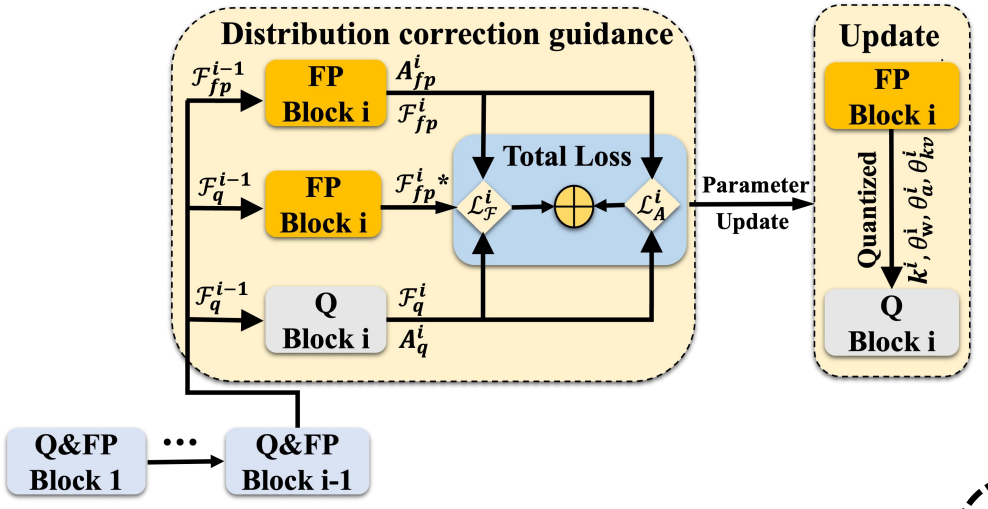
Traditional *KV Quantization*

- Quantize both the current and historical key-values simultaneously, then perform dequantization.
- **Introduce redundant truncation errors in Attention computation**.

4. HKVQ:*Historical KV Quantization*

- Only quantize past key-values; current key-values participate directly in forward computation.
- Eliminate redundant phase errors between quantization and dequantization of current key-values, **improving KV cache quantization accuracy**.

2.4 *Distribution Correction-Guided Parameter Update*



- **Value& Semantic :** MSE & Cosine Similarity
- Mitigates **inter-layer error accumulation** by aligning with an ideal output based on the previous block's quantized input.

Loss function 1

- **Feature Loss L_F**:Value & Semantic Direction

$$\mathcal{L}_{\mathcal{F}}^{i} = |\mathcal{F}_{fp}^{i} - \mathcal{F}_{q}^{i}|_2^2 - \log\left(\frac{\mathcal{F}_{q}^{i} \cdot \mathcal{F}_{fp}^{i}}{|\mathcal{F}_{q}^{i}||\mathcal{F}_{fp}^{i}|}\right) + |\mathcal{F}_{fp}^{i}{}^{*} - \mathcal{F}_{q}^{i}|_2^2 - \log\left(\frac{\mathcal{F}_{q}^{i} \cdot \mathcal{F}_{fp}^{i}{}^{*}}{|\mathcal{F}_{q}^{i}||\mathcal{F}_{fp}^{i}{}^{*}|}\right)$$

Loss function 2

- **Attention Loss L_A**: Restore Attention Distribution

$$\mathcal{L}_{\mathcal{A}}^{i} = D_{KL}\left(A_{q}^{i}|A_{fp}^{i}\right) + D_{KL}\left(A_{fp}^{i}|A_{q}^{i}\right)$$

- Symmetric Kullback-Leibler (KL) Divergence loss
- Performs a **bidirectional correction**, pulling the quantized attention distribution closer to the original and vice-versa for a more robust alignment.

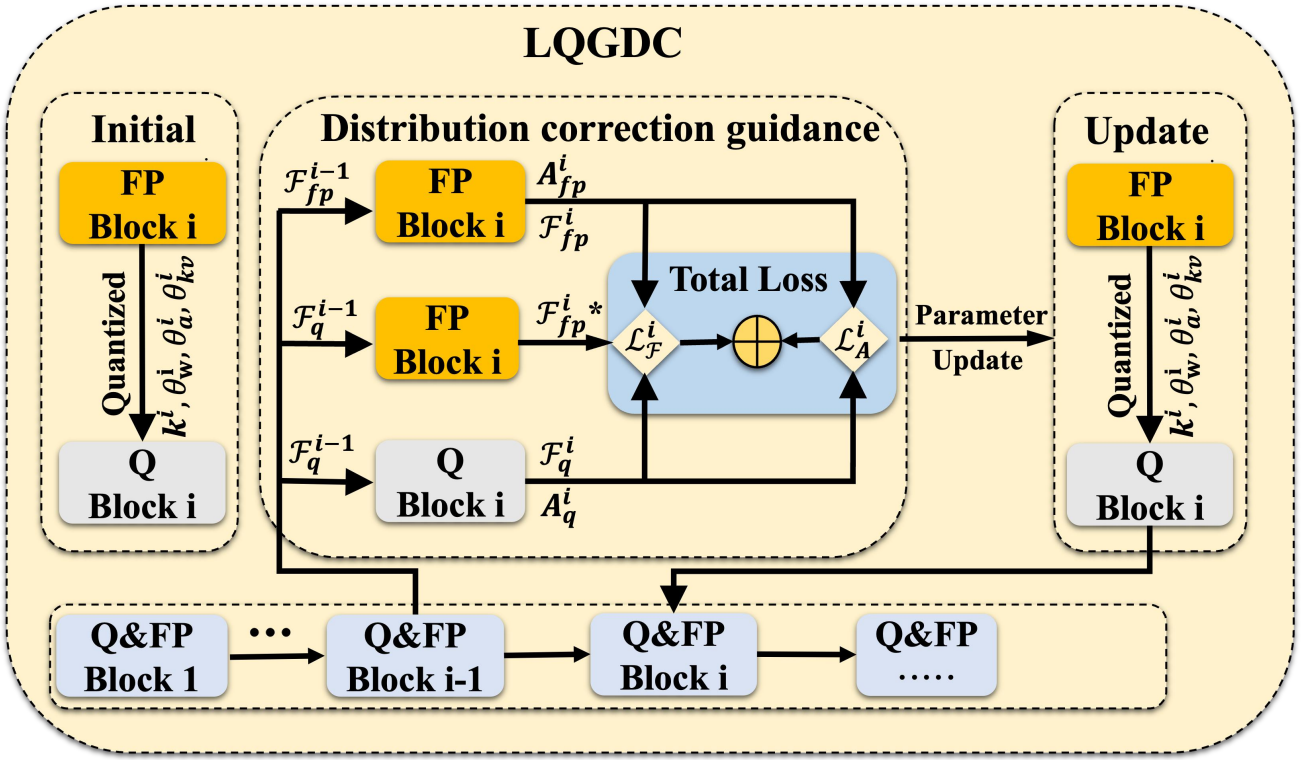
- The final optimization function:

$$k^{*}, \theta_{w}^{*}, \theta_{a}^{*}, \theta_{kv}^{*} = \min_{k^{i}, \theta_{w}^{i}, \theta_{a}^{i}, \theta_{kv}^{i}} \left(\mathcal{L}_{\mathcal{F}}^{i} + \mathcal{L}_{\mathcal{A}}^{i}\right)$$

- DCG automatically finds the optimal transformation parameters for each block.

2.4 *Distribution Correction-Guided Parameter Update*

Parameters are updated via DCG



Step1:Initialize & Transform

- **Initialize** learnable parameters
- **Apply** transformations and quantize Block *i*

Step2:Loss Calculation via DCG Module

- **Calculate Feature Loss /Attention Loss**

Setp3:Parameter Update & Progression

- **Update** parameters using the Total Loss.
- **Finalize** optimal parameters for Block *i*
- **Proceed** to Block *i* + 1 with the calibrated output and the entire process repeats

Outline

1. Introduction
2. Approach
3. Experimental Evaluation
4. Conclusion

3.1 Experimental Environment and Search Space

Experimental Environment	
Cloud Server	NVIDIA A100 GPUs with 80GB memory and 128GB RAM
Edge Devices(EDs)	*NVIDIA Jetson AGX Orin 64G* *NVIDIA Jetson Orin NX 16G*
Datasets	*WikiText-2* *C4* *ARC-e & ARC-c* *HellaSwag* *PIQA* *Winogrande*
Models	LLaMA-7B LLaMA-13B LLaMA-2-7B

Language generation

zero-shot reasoning

NVIDIA Jetson Orin NX 16G

3.2 Baseline and Metrics

Baseline:

- **SmoothQuant**
- An INT8 method that balances outliers between weights and activations.
- **OS+**
- An INT4 method using channel-wise shifts to handle asymmetry.
- **OminiQuant**
- An advanced INT4 method using block-wise MSE minimization.

Metrics:

- **Perplexity (PPL) ↓**

$$PPL = exp\left(-\frac{1}{N}\sum_{i=1}^{N} logP(w_i|w_{<i})\right)$$

Predict the probability of the current word given the preceding context.

- For language generation. Measures prediction accuracy. **Lower is better.**
- **Accuracy (ACC) ↑**

$$ACC = \frac{N_C}{N_T}$$

- For zero-shot tasks. Measures reasoning ability. **Higher is better.**

3.3 Language generation

For language generation tasks (Wiki, C4) on AGX edge devices, LQGDC outperforms all baseline methods.

- W4A4KV4: Average perplexity reduced by **1.35, 1.92, and 2.88** respectively.
- W4A4 Quantization: Average perplexity reduced by **1.17, 1.63, and 2.67** respectively.

On NX edge devices, the model quantized with LQGDC achieves an average perplexity that outperforms the best baseline.

Table 1. Perplexity Comparison on Language Generation Tasks

Precision	Method	LLaMA-7B		LLaMA-13B		LLaMA-2-7B	
		WikiText2	C4	WikiText2	C4	WikiText2	C4
NVIDIA Jetson AGX Orin 64G							
FP16	–	5.68	7.31	5.09	6.69	5.47	7.14
W6A6	SmoothQuant	6.10	7.56	5.50	7.01	6.27	7.84
	OS+	6.07	7.51	5.47	6.93	6.12	7.63
	OmniQuant	6.02	7.50	5.38	6.88	5.95	7.58
	LQGDC (ours)	**5.93**	**7.38**	**5.35**	**6.80**	**5.76**	**7.35**
W4A4	SmoothQuant	46.53	56.50	79.35	96.86	98.92	92.22
	OS+	40.32	47.62	53.64	71.33	60.17	68.96
	OmniQuant	11.86	15.17	11.59	14.67	15.46	19.59
	LQGDC (ours)	**10.72**	**13.98**	**10.54**	**12.46**	**13.35**	**16.37**
W4A4KV4	OmniQuant	12.09	15.50	11.95	15.02	15.74	19.98
	LQGDC (ours)	**10.79**	**14.10**	**10.59**	**12.54**	**13.42**	**16.55**
NVIDIA Jetson Orin NX 16G							
W4A4	SmoothQuant	46.58	56.51	79.37	96.91	98.96	92.22
	OS+	40.35	47.64	53.67	71.34	60.20	68.96
	OmniQuant	11.88	15.20	11.60	14.69	15.48	19.64
	LQGDC (ours)	**10.79**	**14.02**	**10.57**	**12.48**	**13.37**	**16.38**
W4A4KV4	OmniQuant	12.11	15.54	11.98	15.04	15.78	20.04
	LQGDC (ours)	**10.86**	**14.13**	**10.63**	**12.56**	**13.45**	**16.56**

3.4 zero-shot reasoning

- **Superior Accuracy:**
- LQGDC achieves the highest average accuracy on a suite of five reasoning tasks.
- **Dominance at 4-bit Precision:**
- LQGDC achieves the highest ACC across all models and settings.
- **+2.14%** improvement over OmniQuant on AGX Orin (W4A4KV4).
- **+2.26%** improvement on NX Orin, proving its **real-world effectiveness.**
- **Enables Powerful Reasoning on the Edge:**
  Preserves complex reasoning abilities after low-bit quantization, making deployment on EDs practical and reliable.

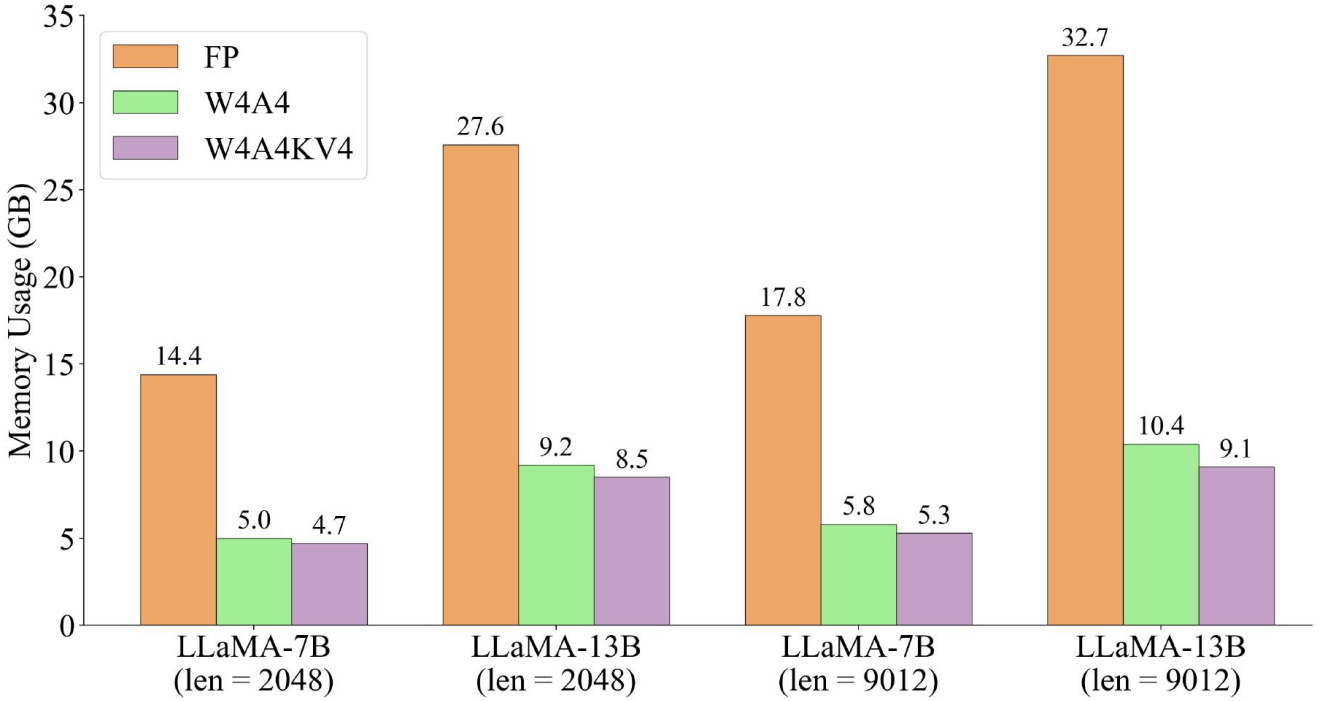
Table 2. Zero-shot Task Accuracy Comparison (%)

Model	Precision	Method	PiQA	ARCe	ARCc	HellaSwag	Winogrande	Avg.
NVIDIA Jetson AGX Orin 64G								
LLaMA-13B	FP16	–	79.10	59.93	44.57	76.20	70.19	65.99
	W6A6	SmoothQuant	77.83	56.36	42.54	75.27	68.71	64.14
		OS+	78.29	56.90	43.09	75.09	69.22	64.36
		OmniQuant	**78.40**	57.28	42.91	**75.82**	68.27	64.54
		LQGDC (ours)	78.02	**57.75**	**42.91**	75.36	**69.22**	**64.65**
	W4A4	SmoothQuant	58.24	35.17	28.34	44.56	49.82	43.22
		OS+	63.00	40.32	30.38	53.61	51.54	47.77
		OmniQuant	69.69	47.34	33.10	58.96	55.80	52.98
		LQGDC (ours)	**71.16**	**47.62**	**34.17**	**63.85**	**57.40**	**54.84**
	W4A4KV4	OmniQuant	69.07	47.10	32.79	58.48	55.22	52.53
		LQGDC (ours)	**71.09**	**47.31**	**34.04**	**63.62**	**57.31**	**54.67**
NVIDIA Jetson Orin NX 16G								
LLaMA-13B	W4A4	SmoothQuant	57.54	34.40	27.37	43.62	48.89	42.36
		OS+	61.94	39.64	29.87	52.71	50.68	46.97
		OmniQuant	68.78	46.73	32.67	58.19	55.08	52.29
		LQGDC (ours)	**70.10**	**46.91**	**33.66**	**62.90**	**56.52**	**54.02**
	W4A4KV4	OmniQuant	68.09	46.31	31.94	57.85	54.41	51.63
		LQGDC (ours)	**70.17**	**46.48**	**33.39**	**62.90**	**56.49**	**53.89**

3.5 long-context inference

Test compression efficiency of LQGDC-quantized models on AGX 64G With batch size = 1 and context length = 9012:

- LLaMA-13B (W4A4KV4): Inference memory footprint reduced by **12.50%** vs. W4A4.
- Compression ratio reaches **72.17%** vs. original FP16 precision model.
- This makes deploying LLMs for **long-context inference** feasible on resource-constrained EDs.



3.6 Ablation studies

Table 3. Ablation Study of LQGDC on LLaMA-13B

Configuration	Average Accuracy (%)	Δ
LQGDC	54.67	–
- LWAS	50.02	4.65
- LWAC	53.35	1.32
- LKVTS	49.73	4.94
- HKVQ	51.96	2.71
- DCG	**41.74**	**12.93**

- DCG is the most critical component
- Handling Outliers is Crucial

smoothing activations (-LWAS, -4.65%) and KV cache (-LKVTS, -4.94%)

Outline

1. Introduction
2. Approach
3. Experimental Evaluation
4. Conclusion

Conclusion

- **Learnable Transformations and long-text reasoning optimizations:**

✓ **LWAS**、**LWAC**、**LKVTS introduce** learnable parameters for weights, activations, and KV cache.**HKVQ** preserves critical attention accuracy while significantly compressing memory usage.

- **Intelligent, Guided Optimization**

✓ **DCG** module uses a multi-objective loss function (MSE, Cosine Similarity, KL Divergence) to find the optimal parameters for each block.

LQGDC successfully maintains model precision at low bit-widths, **significantly enhancing the capability for complex, long-text inference on resource-constrained edge devices.**

Thank you!