Learnable Cloud-Guided LLM Quantization for Resource-Constrained Edge Devices

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

[1] College of Computer Science and Software Engineering, Hohai University, Nanjing, China
{241307010031,221307050007,230407040003,yingchimao}@hhu.edu.cn
[2] Huaneng Lancang River Hydropower Inc., Kunming, China
[3] College of Internet of Things, Nanjing University of Posts and Telecommunications, Nanjing, China
hexiaoming@njupt.edu.cn
[4] Center for Networked Computing, Temple University, Philadelphia, USA
jiewu@temple.edu

Abstract. Model quantization is crucial for deploying large language models (LLMs) on resource-constrained edge devices. However, in cloud-edge collaboration, edge devices (EDs) often lack the resources for on-device quantization. Moreover, existing Post-Training Quantization (PTQ) methods employ a static parameter approach, which fails to adapt to diverse local data distributions. To address these challenges, we propose a novel method, Learnable Quantization Guided by Distribution Correction (LQGDC), for generating an optimal, lightweight model that can be delivered to the ED for local inference within a cloud-edge collaborative framework. In this framework, edge devices upload a small amount of local data to the cloud as a calibration set. The cloud server then selects a suitable pre-trained model from a Model Pool and applies LQGDC to quantize the model. LQGDC introduces learnable parameters for weights, activations, and key-value (KV) cache. LQGDC employs a composite loss function that combines Mean Squared Error (MSE), cosine similarity, and Kullback-Leibler (KL) divergence to fine-tune parameters, thereby matching each device's unique data distribution. Experiments on seven datasets demonstrate that LQGDC outperforms all three current baselines in both language generation and zero-shot tasks. Specifically, when quantizing LLaMA-13B to W4A4KV4, LQGDC reduces average perplexity (PPL) by 1.92 and improves zero-shot task accuracy by 2.14% compared to the best baseline. This approach shows promise for single AI task implementation on resource-constrained EDs (e.g., complex voice command processing on smartphones, real-time visual defect detection on industrial drones, and document analysis with long-term context).

Keywords: LLMs · Model quantization · Local Inference · Cloud-Edge Collaboration · Post-Training Quantization

Published by Springer Nature Switzerland AG 2026
X. Wang et al. (Eds.): NPC 2025, LNCS 16305, pp. 423–435, 2026.
https://doi.org/10.1007/978-3-032-10459-5_33

1 Introduction

LLMs [1,2], based on the Transformer architecture [4], are central to edge intelligence but challenging to deploy on resource-constrained edge devices (EDs) due to their massive size (e.g., GPT-3 [5] has 175 billion parameters). It is crucial to deploy the capabilities of LLMs to resource-constrained EDs efficiently. This enables real-time offline defect identification in drones [3] and complex command execution on smartphones. Model quantization is frequently applied to facilitate the deployment of LLMs on EDs. While model quantization can reduce computational and memory requirements, EDs lack the resources to perform quantization locally, and cloud servers cannot access edge data for tailored model optimization. In a cloud-edge framework, a cloud server can perform complex computation tasks such as model training and quantization and subsequently deliver optimal models to EDs.

Quantization-Aware Training (QAT) and Post-Training Quantization (PTQ) are applied to quantizing LLMs. QAT preserves high accuracy but at a steep computational cost. PTQ is more practical than QAT for edge deployment due to its lower computational cost, albeit often at a slight accuracy trade-off. Studies show activation value channels contain outliers [10,11,13], reducing model accuracy. SmoothQuant [6] smoothes outliers via channel scaling. Wei et al. [7] found that outliers concentrate in asymmetric channels. OS+ uses channel displacement to eliminate asymmetric channel scaling effects and smooth outliers. However, existing static-parameter transformations struggle to smooth all channels. Furthermore, deploying LLMs for long-context tasks (e.g., document analysis or multi-round dialogue) on EDs is challenged by the high memory footprint of the Key-Value (KV) cache during autoregressive decoding. Existing methods, such as KVQuant [8] and IntactKV [9], attempt to reduce this footprint through techniques like per-channel quantization or preserving certain tokens in full precision. However, these approaches often overlook the dynamic distributions across channels and tokens, leading to significant quantization errors, especially for outliers in key and value activations.

To address these challenges, we propose a cloud-edge collaborative framework that offloads quantization to the cloud, guided by a small amount of edge data. We propose a novel method, the Learnable Quantization Guided by Distribution Correction (LQGDC) algorithm, in this framework. Unlike conventional methods that use static parameters, LQGDC introduces learnable parameters for equivalence transformations on weights, activations, and the KV cache. It optimizes these parameters via a composite loss function combining Mean Squared Error (MSE), cosine similarity, and Kullback-Leibler (KL) divergence to align quantized outputs with full-precision versions in value, direction, and attention distribution. Within this framework, the cloud generates an optimized, low-bit SubNet for a target ED by applying LQGDC to quantize a selected LLM.

The main contributions of this paper are as follows:

- **Framework Design.** We propose a novel cloud-edge collaboration framework where the cloud performs complex quantization guided by a small amount of edge data, effectively bridging the information gap between them.

- **Algorithmic Innovation.** We design an algorithm, LQGDC, that uses learnable equivalence transformations and a distribution-correction-guided loss function. LQGDC enables dynamic adaptation to data distributions and generates high-fidelity, personalized low-bit models.
- **Effectiveness.** LQGDC outperforms all baseline methods. For instance, at W4A4KV4 precision, it reduces the average perplexity of LLaMA-2-7B by 2.88 on language tasks and boosts the accuracy of LLaMA-13B by up to 2.26% on zero-shot tasks, while also achieving a 12.50% additional memory saving over standard 4-bit methods.

2 System Model

The system model comprises a Cloud Server and EDs. The Cloud Server maintains a Model Pool and executes the LQGDC algorithm. EDs with finite resources provide local calibration data and run the final optimized model locally. The workflow of this framework is divided into the following three phases:

Phase 1: Model Selection and Data Upload. The process begins at the EDs, which request a suitable base model (e.g., LLaMA family [12]) from the cloud based on their specific application needs. Subsequently, the ED prepares and uploads a small, representative **calibration dataset** derived from its local data as guidance for quantization. To protect user privacy, sensitive information will be scrubbed from these samples locally before upload.

Phase 2: Cloud-Side Quantization and Optimization. Upon receiving the request and data, the Cloud Server initiates the core quantization process. First, the server loads the specified model, runs it once to cache the original, full-precision output of each block, denoted as $\mathcal{F}_{fp}^{i}$, and initializes all learnable quantization parameters. The model is then quantized and optimized block-by-block in a sequential manner, where the output of the previous quantized block serves as the input to the next, thereby mitigating error across layers.

1. **Learnable Transformation**. LQGDC applies the learnable equivalent transformations to the weights, activations, and KV cache of the current block to adjust their numerical ranges, preparing them for low-bit quantization.
2. **Loss Calculation and Optimization**. We update the parameters block by block through minimizing a composite loss function $\mathcal{L} = \mathcal{L}_{\mathcal{F}}^{i} + \mathcal{L}_{A}^{i}$. The quantized output $\mathcal{F}_{q}^{i}$, the original full-precision output $\mathcal{F}_{fp}^{i}$, and the output $\mathcal{F}_{fp}^{i}*$ from the previous quantized input are used to calculate feature loss $\mathcal{L}_{\mathcal{F}}^{i}$, which ensures consistency in value and direction and corrects accumulated errors. The attention loss $\mathcal{L}_{A}^{i}$ is calculated by comparing the attention distribution A_{fp}^{i} and A_{q}^{i} to restore the accuracy of the attention calculation.

Phase 3: Model Distribution and Local Inference. The optimization cycle described above is repeated for every block in the model until all blocks have been individually calibrated. Finally, these optimized blocks are assembled into

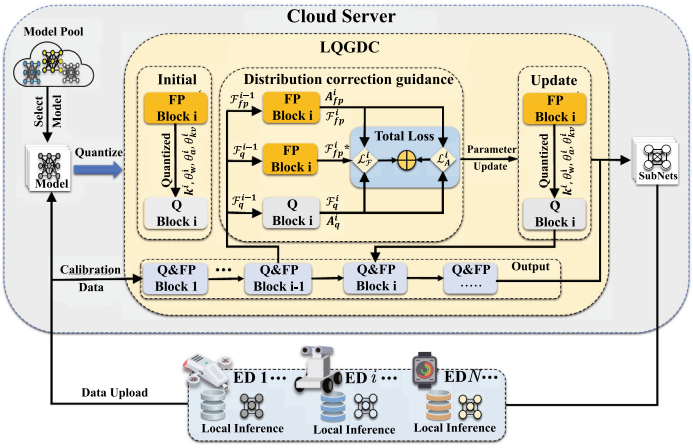

Fig. 1. Workflow of the framework with LQGDC Quantization

a final **Quantized SubNet**. After verification, this model is distributed back to EDs. Due to its significantly smaller size, the model can be run efficiently for local inference, ensuring low resource consumption, offline capability, and user data privacy (Fig. 1).

Through this workflow, our framework successfully leverages the cloud's computational power to solve the edge's challenges, providing a practical and effective solution for personalized, high-performance AI applications on the edge.

3 Methodology

LQGDC utilizes a series of learnable transformations to mitigate accuracy loss during quantization. Our approach consists of five core modules: Learnable Weight-Activation Smoothing (LWAS) for mitigating outliers in weights and activations, Learnable Weight-Activation Clipping (LWAC) for determining layer-wise dynamic quantization ranges, Learnable KV Cache Transformation and Smoothing (LKVTS) for KV cache quantization, Historical KV Quantization (HKVQ) for efficient long-context handling, and Distribution Correction Guidance (DCG) for optimizing all parameters.

3.1 Learnable Weights and Activation Value Equivalent Transformations

This subsection presents the LWAS and LWAC modules. Quantizing activations in LLMs is difficult due to the presence of significant outliers. To address this, the LWAS module introduces a learnable, per-channel smoothing factor k, which shifts the quantization difficulty from the activations to the weights through an equivalence transformation, which is given by

$$Y = (X \cdot \mathrm{diag}(k)^{-1})(\mathrm{diag}(k) \cdot W) + B = \hat{X} \cdot \hat{W} + B, \tag{1}$$

where the input activations X and weights W are transformed into $\hat{X}$ and $\hat{W}$, respectively, and B is the bias term. This leads to more balanced numerical ranges that are less affected by quantization errors caused by outliers by adjusting the values of learnable parameters k. Even after the smoothing process, some residual outliers may persist in the weights and activations. The LWAC module introduces learnable clipping thresholds $\theta_w = \{\alpha_w, \beta_w\}$ for weights and $\theta_a = \{\alpha_a, \beta_a\}$ for activations. In weight quantization, we clip the balanced weights $\hat{W}$ and then map them to the target integer range $[0, 2^N - 1]$, which is given by

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

where $[\cdot]$ is the rounding operation and $clamp(\cdot, 0, 2^N - 1)$ clips the result to the target integer $[0, 2^N - 1]$. The scaling factor s_x and zero-point z are dynamically adjusted by clipping parameters $\alpha_w, \beta_w \in [0, 1]$, which are given by

$$s_x = \frac{\alpha_w max(\hat{W}) - \beta_w min(\hat{W})}{2^N - 1}, z = -[\frac{\beta_w min(\hat{W})}{s_x}], \theta_w = \{\alpha_w, \beta_w\}. \tag{3}$$

The learnable clipping parameters, $\{\theta_a, \theta_w\}$, are limited by the sigmoid function. Their role is to dynamically adjust the quantization range of the weights, allowing for full adaptation to their distribution while preventing invalid outliers from consuming the limited bit-width representation.

3.2 Learnable Scale and Offset Quantization for KV Cache

This subsection describes the LKVTS and HKVQ modules. When deploying LLMs on EDs for single AI tasks, there's demand for long-text reasoning. This necessitates the quantization of the KV cache to conserve memory usage on EDs. Outliers present across various channels and tokens can introduce significant errors in KV cache quantization, thereby impairing the precision of attention computation. To mitigate the degradation in model precision caused by outliers within KV cache channels, the LKVTS module introduces two learnable parameters for an equivalent transformation: a channel-wise translation parameter σ and a scaling parameter γ. These parameters independently center the numerical distribution of each channel to achieve symmetry and scale the channel's numerical range, making quantization easier. This channel-wise balancing is given by

$$Y_{kv} = \underbrace{\{(Y_{kv} - \sigma)/\gamma\}}_{\widetilde{Y}_{kv}} \cdot \gamma + \sigma, \quad \theta_{kv} = \{\sigma, \gamma\}. \tag{4}$$

In this equivalent transformation, we utilize learnable translation and scaling parameters $\theta_{kv} = \{\sigma, \gamma\}$ to significantly reduce the numerical disparity between channels, making the transformed distribution $(Y_{kv} - \sigma)/\gamma$ easier to quantize.

To handle outliers in KV cache tokens, the quantization range for each token is adjusted independently. For the channel-smoothed matrix $\widetilde{Y}_{kv}$, its median M

is used for centering, which creates a more symmetric distribution and reduces outlier errors. The scaling factor s_{kv} then maps the maximum absolute deviation to the target bit-width N. This dynamic token quantization process, which yields the quantized matrix $\widetilde{Y}_q$, is given by

$$\begin{aligned} \widetilde{Y}_q &= clamp\left(\left[\frac{\widetilde{Y}_{kv} - M}{s_{kv}}\right], 0, 2^N - 1\right), \\ s_{kv} &= \frac{\max(|\widetilde{Y}_{kv} - M|)}{2^N - 1}, \quad M = \text{Median}(\widetilde{Y}_{kv}). \end{aligned} \tag{5}$$

This approach enhances robustness by computing the scaling factor s_{kv} per token, allowing adaptation to each token's unique numerical distribution. This reduces the difficulty of KV cache quantization.

Traditional KV cache quantization saves memory by converting all KV pairs to low precision, but this affects attention accuracy by introducing errors when quantizing the most recent tokens. We propose the HKVQ module, illustrated in Fig. 2, that resolves this issue by quantizing the historical KV cache to a low-precision format (e.g., INT4) while maintaining the current KV pairs in full precision (e.g., FP16). For attention computation, the dequantized historical cache is concatenated with the full-precision current KV data. This approach enhances robustness by computing a per-token scaling factor s_{kv}, which adapts to each token's unique numerical distribution, thereby simplifying the quantization process.

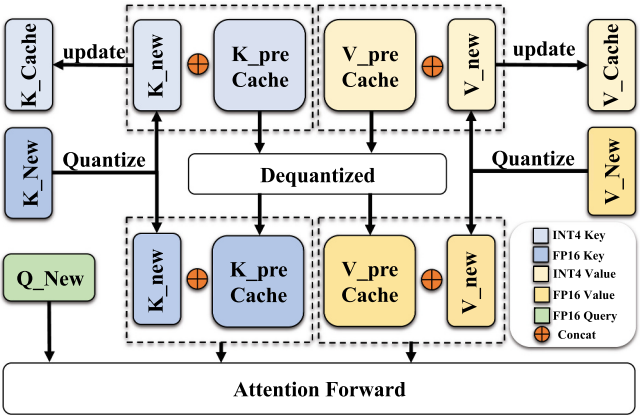

Fig. 2. Historical KV Cache Structure Diagram

This method operates in two distinct phases. During the initial prefill phase, all KV pairs are kept in full precision for maximum accuracy. Subsequently, in the decoding phase, only the historical KV cache is quantized, while the newly

generated KV pair remains at full precision and is combined with the dequantized history. This approach prevents the accumulation of errors and significantly reduces memory usage for long texts, offering an efficient and high-precision solution for LLMs on EDs. The LKVTS and HKVQ modules enable the LQGDC to address both the numerical challenges of KV cache quantization and the memory constraints, contributing to the model compression achieved by our method.

3.3 Distribution Correction-Guided Equivalence Transformation Parameter Update

This subsection presents the DCG module. In the quantization process, the optimal learnable parameters, $(k, \theta_w, \theta_{kv}, \theta_a)$, are determined by a unified optimization process aimed at distribution correction. This process employs a block-wise strategy utilizing a small amount of calibration data, with the ultimate goal of minimizing the output discrepancy between the original and the quantized model, which is given by

$$\arg \min_{k,\theta_w,\theta_a,\theta_{kv}} \|WX - Q_w(W; k; \theta_w, \theta_{kv}) Q_a(X; k; \theta_a, \theta_{kv})\|, \tag{6}$$

where $Q(\cdot)$ represents the quantization function. The terms $Q_w(W; k; \theta_w, \theta_{kv})$ and $Q_a(X; k; \theta_a, \theta_{kv})$ denote the quantized values of the new weights and activations resulting from the prior equivalent transformation. This formula ascertains the optimal parameter combination for the model by minimizing the discrepancy between the original and quantized outputs.

For the output of a Transformer block, it is essential to not only align the numerical values but also to preserve the semantic direction. Therefore, we designed a loss function composed of four terms, which is given by

$$\begin{aligned} \mathcal{L}_{\mathcal{F}}^{i} = & \|\mathcal{F}_{fp}^{i} - \mathcal{F}_{q}^{i}\|_2^2 - \log\left(\frac{\mathcal{F}_{q}^{i} \cdot \mathcal{F}_{fp}^{i}}{\|\mathcal{F}_{q}^{i}\| \|\mathcal{F}_{fp}^{i}\|}\right) \\ & + \|\mathcal{F}_{fp}^{i} * - \mathcal{F}_{q}^{i}\|_2^2 - \log\left(\frac{\mathcal{F}_{q}^{i} \cdot \mathcal{F}_{fp}^{i}*}{\|\mathcal{F}_{q}^{i}\| \|\mathcal{F}_{fp}^{i} * \|}\right), \end{aligned} \tag{7}$$

where $\mathcal{F}_{q}^{i}$ is the quantized output of the i-th Transformer block, $\mathcal{F}_{fp}^{i}$ is the full-precision output of the i-th Transformer block, $\mathcal{F}_{fp}^{i}*$ is the full-precision output of the i-th Transformer block derived from the quantized output of the $(i-1)$-th Transformer block, and $\|\cdot\|_2$ is the Euclidean norm of a vector.

The first and second terms use MSE and Cosine Similarity, respectively, to ensure the quantized output $\mathcal{F}_{q}^{i}$ approximates the original full-precision output $\mathcal{F}_{fp}^{i}$ in both value and direction. The third and fourth terms aim to mitigate inter-layer error accumulation. We introduce a special baseline $\mathcal{F}_{fp}^{i}*$, which represents the ideal full-precision output that the current block should produce after receiving the quantized output from the previous block. By minimizing the difference between $\mathcal{F}_{fq}^{i}*$ and $\mathcal{F}_{q}^{i}$, we ensure the model maintains semantic coherence across multiple stacked layers.

Moreover, to restore the attention distribution of the quantized model, an attention relative entropy loss function is utilized to mitigate computational errors in the attention layer arising from quantization, which is denoted as

$$\mathcal{L}_A^i = D_{KL}(A_q^i \| A_{fp}^i) + D_{KL}(A_{fp}^i \| A_q^i), \tag{8}$$

where D_{KL} is the KL divergence used to measure the difference between two distributions, A_q^i represents the quantized attention output in the i-th Transformer block, and A_{fp}^i signifies the full-precision attention output in the i-th Transformer block. This symmetric loss function has two terms that work in opposite directions. The first term measures the error from the quantized distribution to the original one, while the second term measures the error in the reverse direction. By optimizing both simultaneously, the function bidirectionally corrects the quantized distribution, pulling it closer to the full-precision version for a more accurate alignment. By combining two loss functions, the final optimization objective function is formed, which is given by

$$k^*, \theta_w^*, \theta_a^*, \theta_{kv}^* = \min_{k^i, \theta_w^i, \theta_a^i, \theta_{kv}^i} (\mathcal{L}_F^i + \mathcal{L}_A^i). \tag{9}$$

The value of the loss function will approach zero infinitely when the distribution of the quantized output matches the full-precision output. Thereby, the loss function can effectively guide the quantization process and update the learnable equivalent transformation parameters of all blocks.

4 Performance Evaluation

4.1 Experimental Setup

Experiments were conducted on a server with an NVIDIA A100 80G GPU and deployed on NVIDIA Jetson AGX Orin 64G and Orin NX 16G edge devices. We evaluate model performance using Perplexity (PPL) [21] on WikiText-2 [14] and C4 [15] datasets for language generation, and Accuracy (ACC) on five commonsense reasoning tasks (ARC-e & ARC-c [16], HellaSwag [17], PIQA [18], Winogrande [19]) for zero-shot reasoning. We compare our method, LQGDC, against three mainstream PTQ baselines: SmoothQuant [6], an INT8 method that balances activation outliers; OS+ [7], an INT4 method that uses offline channel-wise shifts; and OmniQuant [20], an INT4 method that minimizes block-wise MSE. A calibration set of 128 randomly selected 2048-token segments from WikiText2 is used to optimize the learnable parameters.

4.2 Comparative Analysis of Performance

LQGDC significantly outperforms all baselines across the evaluated models, precisions, and tasks. This superior performance stems from LQGDC adapting to data distributions, a capability that static PTQ methods lack.

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**

In language generation tasks (Table 1), LQGDC not only achieves optimal language generation performance with higher bit-width quantization, but it also performs exceptionally well in the challenging scenarios of low-bit-width quantization. At W4A4 precision, where methods like SmoothQuant and OS+ suffer catastrophic performance collapse, LQGDC excels. For instance, on the LLaMA-13B model, LQGDC achieves a PPL of 10.54 on the diverse C4 dataset, a substantial improvement over the 14.67 scored by OmniQuant. This can be attributed to our distribution correction guidance, which allows the model to adapt to varied data distributions where other methods falter.

Moreover, this advantage is maintained even when quantization is applied to the KV cache (W4A4KV4). The PPL of the LLaMA-2-7B model can be reduced by an average of 2.88 for language generation tasks, outperforming the best baseline OmniQuant quantization method. LQGDC's specialized learnable quantization strategy for the KV cache not only reduces memory consumption in long-text scenarios but also effectively maintains model accuracy. Crucially, its consistent performance on resource-constrained NVIDIA Jetson Orin NX 16G demonstrates the feasibility of deploying LQGDC on EDs.

LQGDC's superiority extends to complex zero-shot reasoning (Table 2). At W4A4KV4 precision, LQGDC demonstrates exceptional performance on zero-shot reasoning tasks. On the LLaMA-13B model, LQGDC achieves an average accuracy of 54.67%, representing a notable 2.14% improvement over OmniQuant. This demonstrates that LQGDC can significantly reduce the memory footprint for long-context tasks without compromising the model's ability to reason. This is because LQGDC effectively reduces quantization error and improves quantized inference precision through a process of learnable transformation and distribution correction-guided quantization. LQGDC maintains a relatively high accuracy for zero-shot tasks on resource-constrained NVIDIA Jetson Orin NX 16G, where LQGDC achieves an even greater improvement of 2.26% over OmniQuant. This demonstrates the feasibility of LQGDC for deployment on EDs.

4.3 Inference Memory and Ablation Experiment

The substantial memory compression from our method is shown in Fig. 3, which compares the memory consumption of our LQGDC (W4A4KV4) method against FP16 and standard W4A4 quantization across varying sequence lengths. As shown by the curves in Fig. 3, while all quantization methods reduce memory compared to FP16, our method demonstrates increasing advantages as sequence length grows, particularly beyond 2048 tokens. The W4A4KV4 quantization is particularly effective for long contexts; for LLaMA-13B with a sequence length of 9012, memory usage is reduced to just 9.1 GB. This represents a 72.17% compression compared to FP16 and a 12.50% savings over the standard W4A4 method. These memory savings, achieved with minimal performance degradation as evidenced, validate LQGDC as a highly efficient solution for edge deployment.

An ablation study on LLaMA-13B (Table 3) evaluates the contributions of LQGDC's components. Results show DCG is the most critical component. The severe performance degradation when removing DCG (−12.93%) underscores its role as the central coordination mechanism; without this global objective, local parameter optimizations fail to maintain semantic coherence across layers. Additionally, removing LWAS and LKVTS leads to significant declines of 4.65% and 4.94%, confirming their importance in mitigating outliers in activations and KV cache that consume substantial quantization bandwidth and impair information retention. Excluding HKVQ results in a 2.71% drop, highlighting its role in balancing memory efficiency and attention accuracy during autoregressive decoding. Removing the LWAC module also causes a 1.32% decline, indicating that adaptive range adjustment helps minimize clipping error compared to standard min-max quantization.

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

Model	Precision	Method	PiQA	ARCe	ARCc	HellaSwag	Winogrande	Avg.
NVIDIA Jetson AGX Orin 64G								
	FP16	–	79.10	59.93	44.57	76.20	70.19	65.99
LLaMA-13B	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**

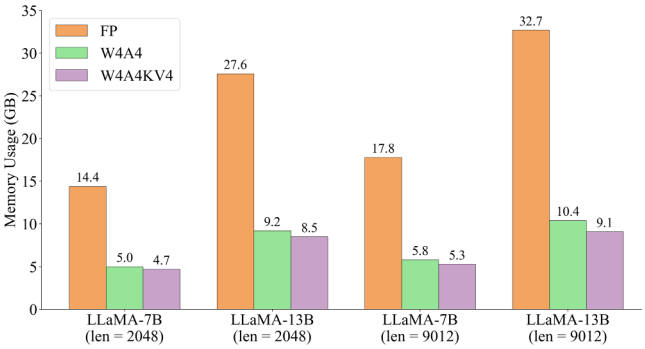

Fig. 3. Inference Memory Consumption of LQGDC at Various Text Lengths

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**

5 Conclusion

To address challenges in quantizing models with internal outliers, adapting to diverse data distributions on EDs, and performing on-device quantization, we propose a quantization method called Learnable Quantization Guided by Distribution Correction (LQGDC) in a cloud-edge collaboration framework. LQGDC utilizes learnable parameters to dynamically smooth the model's outliers, making them easier to quantize. LQGDC employs a weight-activation smoothing factor to migrate outliers from activations to weights and uses the clipping parameters to optimize the quantization range, reducing truncation errors. For the KV cache, LQGDC combines channel-wise smoothing with dynamic token quantization to improve memory efficiency and attention accuracy. The entire process is guided by a multi-objective loss function (MSE, cosine similarity, and KL divergence) that updates these learnable parameters block by block. Experiments on seven datasets demonstrate that LQGDC outperforms all three current baselines in both language generation and zero-shot tasks. Specifically, when quantizing LLaMA-13B to W4A4KV4, LQGDC reduces average perplexity by 1.92 and improves zero-shot task accuracy by 2.14% compared to the best baseline. These improvements indicate that LQGDC is a promising approach for significantly enhancing the performance of quantized models while reducing on-device computational resource requirements.

Acknowledgment. This work was supported in part by the Key Research and Development Program of China under Grant 2022YFC3005401; in part by the Technology Talent and Platform Program of Yunnan Province under Grant 202405AK340002; in part by the Technology Project of Huaneng Group under Grant HNKJ20-H46 and Grant HNKJ24H167; and in part by the High Performance Computing Platform, Hohai University.

References

1. Zhang, S., Roller, S., Goyal, N., et al.: OPT: Open Pre-trained Transformer Language Models. arXiv preprint arXiv:2205.01068 (2022)
2. Le Scao, T., Fan, A., Akiki, C., et al.: BLOOM: A 176B-Parameter Open-Access Multilingual Language Model. arXiv preprint arXiv:2211.05100 (2022)
3. Abas, K., Obraczka, K., Miller, L.: Solar-powered, wireless smart camera network: an IoT solution for outdoor video monitoring. Comput. Commun. **118**, 217–233 (2018)
4. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
5. Brown, T., Mann, B., Ryder, N., et al.: Language models are few-shot learners. In: Advances in Neural Information Processing Systems, vol. 33, pp. 1877–1901 (2020)
6. 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)
7. Wei, X., Zhang, Y., Li, Y., 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)
8. Hooper, C., Kim, S., et al.: KVQuant: towards 10 million context length LLM inference with KV cache quantization. In: Advances in Neural Information Processing Systems, vol. 37, pp. 1270–1303 (2024)
9. Liu, R., et al.: IntactKV: improving large language model quantization by keeping pivot tokens intact. In: ACL, pp. 7716–7741 (2024)
10. Luo, Z., Kulmizev, A., Mao, X.: Positional artefacts propagate through masked language model embeddings. In: ACL-IJCNLP, vol. 1, pp. 5312–5327 (2021)
11. Bondarenko, Y., Nagel, M., Blankevoort, T.: Understanding and overcoming the challenges of efficient transformer quantization. In: EMNLP, pp. 7947–7969 (2021)
12. Touvron, H., et al.: Llama: open and efficient foundation language models. arXiv preprint arXiv:2302.13971 (2023)
13. Dettmers, T., Lewis, M., Belkada, Y., Zettlemoyer, L.: LLM.int8(): 8-bit matrix multiplication for transformers at scale. arXiv preprint arXiv:2208.07339 (2022)
14. Merity, S., Xiong, C., Bradbury, J., Socher, R.: Pointer sentinel mixture models. In: ICLR (2017)
15. Colin, R., et al.: Exploring the limits of transfer learning with a unified text-to-text transformer. Mach. Learn. Res. **21** (2020)
16. Clark, P., et al.: Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. arXiv preprint arXiv:1803.05457 (2018)
17. Zellers, R., Holtzman, A., Bisk, Y., et al.: HellaSwag: can a machine really finish your sentence? In: ACL, pp. 4791–4800 (2020)
18. Bisk, Y., Zellers, R., Le Bras, R., et al.: PIQA: reasoning about physical commonsense in natural language. In: AAAI, pp. 7432–7439 (2020)
19. Sakaguchi, K., et al.: WinoGrande: an adversarial winograd schema challenge at scale. In: AAAI, pp. 8732–8740 (2020)
20. Shao, W., Chen, M., Zhang, Z., et al.: OmniQuant: omnidirectionally calibrated quantization for large language models. In: ICLR (2024)
21. Li, S., Ning, X., Wang, L., et al.: Evaluating Quantized Large Language Models. arXiv preprint arXiv:2402.18158 (2024)