FP=XINT: Representing Neural Networks via Low-Bit Series Basis Functions

Boyang Zhang[1,2,3], Daning Cheng[1*], Yunquan Zhang[1], Jiake Tian[2,5], Jing Li[4], Fangming Liu[2]

[1]Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[2]Pengcheng Laboratory, Shenzhen, China [3]University of Chinese Academy of Sciences, Beijing, China
[4]Harbin Institute of Technology, Shenzhen, China [5]South China University of Technology, Guangzhou, China
zhangby01@pcl.ac.cn, {chengdaning, zyq}@ict.ac.cn,
mijiake@mail.scut.edu.cn, jingli.phd@hotmail.com, fangminghk@gmail.com

Abstract

Deep neural networks are often over-parameterized, resulting in prohibitive storage and computational costs. A fundamental question is whether a complex network can be re-expressed in terms of a compact set of basis functions without sacrificing accuracy. Motivated by this perspective, we aim to approximate a dense model by decomposing it into a small number of lightweight components that capture the essential functional structure of the network. To this end, we propose a series expansion framework that rewrites a neural network as a linear combination of low-bit basis models. Within the post-training quantization setting, the full-precision model is expanded hierarchically at the tensor, layer, and model levels into a structured set of basis functions. We theoretically prove that this expansion converges exponentially to the original model. Furthermore, we design AbelianAdd and AbelianMul operations between isomorphic basis models, endowing the expansion with an Abelian group structure that naturally supports commutative and parallel computation. Experimental results across diverse architectures show that our series expansion method leverages a set of ultra-low-bit basis functions, not only preserving full-precision performance without the need for calibration data or fine-tuning, but also featuring a parallel-friendly design that enables efficient deployment.

Introduction

Deep neural networks achieve remarkable success across vision and language tasks, but their massive parameterization makes deployment on resource-constrained platforms extremely challenging. A classical mathematical principle approximates a complex function $f(x)$ is to represent it as a weighted sum of simple basis functions, $f(x) = \sum_{i=1}^{n} \alpha_i h_i(x)$, as done in existing series expansions. Such structured representations dramatically reduce complexity, exploit inherent structure, and are parallelizable. This observation raises a fundamental and unanswered question:

> Can a deep neural network itself be reconstructed by a principled expansion over a small set of basis functions, rather than stored as a dense, monolithic model?

[*]Corresponding author

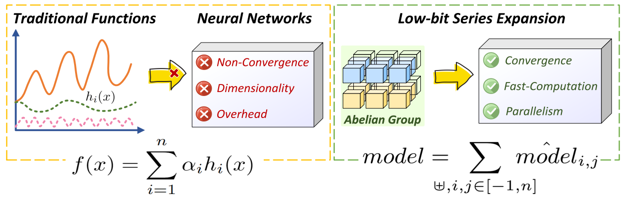

Figure 1: Series expansion methods are effective for traditional functions, but applying them to neural networks presents many challenges. Our low-bit series expansion based on Abelian groups overcomes these challenges.

Addressing this would transform our understanding and compression of neural models on heuristic parameter manipulation. However, extending classical basis expansions to modern deep networks encounters three critical obstacles highlighted in Figure 1: (i) identifying suitable basis functions amid the extreme dimensionality and nonlinearity of these models; (ii) ensuring provable convergence of the expansion back to the original network; and (iii) devising an algebraic framework that supports efficient, stable, and parallel composition of basis components.

While post-training quantization (PTQ) (Li et al. 2021; Liu et al. 2023a; Shang et al. 2024; Li et al. 2025; Zhao et al. 2025) techniques incidentally produce low-bit submodels that might serve as natural basis candidates, naïvely aggregating these components fails to meet these requirements. Such mixtures lack the expressive completeness to span the original function space. This results in additive mixtures that accumulate approximation errors rather than converge to the original network. Moreover, without a principled algebraic structure, each added submodel increases inference overhead linearly, causing severe computational inefficiency and hindering real-world deployment.

To bridge this gap, we propose the first series expansion framework explicitly designed for neural network representation. Our approach hierarchically decomposes a full-precision (FP) model into multiple low-bit basis at tensor, layer, and model levels. We theoretically prove exponential convergence of these expansions, providing strong guarantees on approximation quality. Key to our framework is the introduction of novel Abelian algebraic operations

(AbelianAdd and AbelianMul), endowing the basis models with a group structure that supports stable, commutative, and parallelizable combination. This algebraic foundation enables efficient inference by exploiting parallel hardware, overcoming the linear overhead bottleneck of naïve aggregation. Empirically, our method demonstrates strong generalization and scalability across diverse architectures, including CNNs, Transformers, LLMs. For example, our 4-bit quantization of ResNet-50 achieves full-precision accuracy (77.03%) without calibration data or retraining, highlighting the practical viability of our approach. Beyond accuracy, we design and adapt a comprehensive parallel strategy that boosts computational efficiency, paving the way for scalable deployment in real-world resource-constrained environments. Our main contributions are as follows:

1) We introduce the first representation paradigm that reconstructs trained network as a series expansion over low-bit basis functions, rethinking how networks are approximated.

2) We develop a multi-level expansion with provable convergence and define novel Abelian combination rules that enable stable and parallel integration of basis models.

3) In a PTQ setting, our approach achieves full-precision accuracy at 4-bit without calibration data or fine-tuning, demonstrating that basis expansion can turn extreme quantization into a scalable deployment mechanism.

Notations

Symbol	Meaning
$f(x)$	Target function
$h_i(x)$	Basis function
α_i	Expansion coefficient
$\sum_{\Xi_i \in \mathcal{G}}^{n}$	Abelian group $\mathcal{G}$ of operations
$model(sample)$	Network function
M	FP tensor (weights/activations)
$\widetilde{M}$	Low-bit tensor
$scale$	Quantization scale factor
Z	Zero point
$clip^-, clip^+$	Saturation clipping bounds
M_{sa}	Saturation error tensor
M_{nsy}	Non-symmetric bias (all ones)
$bias$	Scalar for asymmetric quantization
$\widetilde{M}_i$	i-th basis tensor in expansion
R_i	Residual tensor at step i
W, A	Weights and activations tensors
W_{sa}, A_{sa}	Saturation errors for W, A
$\widetilde{W}_i, \widetilde{A}_i$	i-th basis tensors for W, A
$scale_{w,i}, scale_{A,i}$	Scale factors for i-th basis
γ_{ij}	$scale_{W,i} \cdot scale_{A,j}$
$\hat{model}_{i,j}$	Basis model with scales
$\widetilde{model}_{i,j}$	Integer-parameter basis model
$\uplus$	AbelianAdd operation
$\hat{*}$	AbelianMul operation
$\mathcal{M}$	Set of isomorphic low-bit models

Target Formulation

We aim to approximate a neural network function by a series expansion of efficiently computable basis functions. Formally, classical expansions represent a function as

$$f(x) = \sum_{i=1}^{n} \alpha_i h_i(x), \tag{1}$$

where $f(x)$ is the target function and $h_i(x)$ are basis functions. Such linear combinations enable systematic approximation and naturally form an Abelian group structure (addition and multiplication), facilitating parallel computation (e.g., AllReduce or MapReduce).

For neural networks, denoting $model(sample)$ as the network output, we seek a similar expansion:

$$model(sample) = \sum_{\Xi_i \in \mathcal{G}}^{n} \hat{model}_i(sample), \tag{2}$$

where $\sum_{\Xi_i}^{n}$ operates in an Abelian group $\mathcal{G}$, replacing classical addition and multiplication, and each $\hat{model}_i$ is a computationally efficient basis model.

Traditional expansions like multidimensional Fourier series are intractable for high-dimensional inputs due to the curse of dimensionality. Thus, we redesign the series expansion to align with modern neural architectures, ensuring efficient, convergent, and parallel-friendly representation.

Low-Bit Series Expansion Algorithm

There are various strategies for selecting basis functions in neural networks. For example, in quantization techniques, low-bit models are widely used due to their computational efficiency and hardware friendliness; hence, we choose the low-bit version of the original model as the basis function. Subsequently, we employ a tensor-layer-model approach to progressively construct a series expansion framework and prove its convergence to the original model.

Tensor Low-Bit Expansion

For an original tensor $M = (m_1, m_2, \ldots, m_n)$, it can be quantized into a low-bit representation. Taking the saturated asymmetric quantization function as an example, the quantization process is expressed as:

$$\widetilde{M} = \text{clip}\left(\text{int}\left(\frac{M}{scale}\right) + Z, clip^-, clip^+\right) \tag{3}$$

where scale is the scaling factor that maps the floating-point tensor M to integers and can be precomputed based on the distribution of M. Z represents the zero point, which corresponds to asymmetric quantization and can also be precomputed from M's distribution. The $clip(\cdot)$ implements saturation by truncating values outside the range $[clip^-, clip^+]$. Different quantization parameters lead to variations in the basis functions used for quantization. Our series expansion framework adapts to these variations and provides a unified analysis. In asymmetric quantization, zero point Z acts as a bias tensor $bias \times M_{nsy}$; for symmetric quantization, this bias is zero. In saturated quantization, m_i exceeding $clip^+$

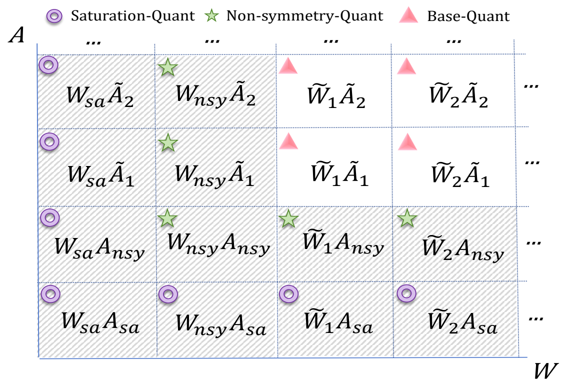

Figure 2: The expansion of tensor multiplication, where A and W are $n \times n$ tensor. The triangle represents the most basic quantization method. In addition, our series expansion framework also supports a variety of optional quantizations.

are clipped to $clip^+$, and those below $clip^-$ are clipped to $clip^-$, with the resulting error denoted as M_{sa} tensor. In contrast, non-saturated quantization omits the clipping operation. Based on the basis tensor $\widetilde{M}$ and its quantization parameters, we use Theorem 1 to construct the expansion terms and prove convergence to the original tensor M.

Theorem 1. $M = M_{sa} + bias \times M_{nsy} + \sum_{i=1}^{n} scale_i \times \widetilde{M}_i$*, where* M_{nsy} *is the tensor whose all elements are 1.* $\widetilde{M}_i$ *is the tensor whose all elements are INT(X) data type and* $scale_i = 2^X \times scale_{i+1}$.

Proof. We adopt the unsaturated symmetric quantization function as a general analytical form, while other quantization methods are adapted by incorporating additional parameters based on their specific properties. First, we define the residual term: $R_1 = M_1 - scale_1 \times \widetilde{M}_1$. According to the remainder definition, all elements in R_1 have absolute values smaller than $scale_1$. Next, $scale_2$ is computed based on the distribution of R_1 to construct M_2. Similarly, the next residual term is defined as: $R_2 = M - scale_1 \times \widetilde{M}_1 - scale_2 \times \widetilde{M}_2$. By repeating this recursive process, we derive the following series expansion: $M = \sum_{i=1}^{n} scale_i \widetilde{M}_i + R_{n+1}$. The max value in R_{n+1} is smaller than $\frac{scale}{2^{Xn}}$. This procedure allows for parallel computation, and the relationship $scale_1 = 2^m \times scale_2$ holds between the scales.

For saturated symmetric quantization, the clipping operation introduces errors, which are compensated by a sparse tensor M_{sa}. Since M_{sa} can be precomputed and depends on the data distribution, it is a constant tensor. This allows the problem to be transformed into the unsaturated quantization case. In non-symmetric quantization methods, the primary distinction from symmetric quantization lies in the inclusion of a bias term during the quantization process: $m_i = bias + scale \times \widetilde{m}_i$, where $\widetilde{m}_i$ represents the quantized value of the original tensor m_i. The calculation of the bias term $bias$ depends on the type of quantization: for non-saturation quantization, $bias = \frac{m_{max} - m_{min}}{2} + m_{min}$; for saturation quantization, $bias = \frac{clip^+ - clip^-}{2} + clip^-$. Since $bias$ remains the same for elements in the tensor, we define the non-symmetry component M_{nsy}, which has the same dimensions as M. The recursive update is: $M_1 = bias \times M_{nsy} + scale_1 \times \widetilde{M}_1 + R_1$. where M_{nsy} captures the non-symmetric offset. The processing of the residual term R_i follows the same approach as in the non-saturation symmetric quantization case. Moreover, this derivation also establishes the construction process for M_{sa}, M_{nsy}, and $\widetilde{M}_i$.

Assuming $|R_1| < scale_1$ and each $scale_i$ reduces the residual ($|R_{i+1}| < |R_i|$), the residual R_n converges to zero as $n \to \infty$, meaning $\lim_{n\to\infty} R_n = 0$. This implies that the series expansion strictly converges to M.

Single-layer Low-Bit Expansion

Modern deep networks are dominated by tensor multiplication, making it the natural focus of our low-bit expansion. According to Theorem 1, both weights W and activations A can be recursively decomposed as $W = W_{sa} + bias_w \times W_{nsy} + \sum_{i=1}^{n} scale_{w,i} \widetilde{W}_i$, $A = A_{sa} + bias_A \times A_{nsy} + \sum_{i=1}^{n} scale_{A,i} \widetilde{A}_i$. We treat $bias$ as $scale_0$, $scale_{-1} = 1$, W_{nsy} as $\widetilde{W}_0$, W_{sa} as $\widetilde{W}_{-1}$, A_{nsy} as $\widetilde{A}_0$, and A_{sa} as $\widetilde{A}_{-1}$ for better description. Then we have the following Eq. 4, and we term it as tensor multiplication low-bit expansion:

$$
\begin{aligned}
WA = (W_{sa} + bias_w \times W_{nsy} + \sum_{i=1}^{n} scale_{w,i} \times \widetilde{W}_i) \times \\
(A_{sa} + bias_A \times A_{nsy} + \sum_{i=1}^{n} scale_{A,i} \times \widetilde{A}_i) \quad (4) \\
= \sum_{i,j \in [-1,n]} scale_{W,i} scale_{A,j} \widetilde{W}_i \widetilde{A}_j = \sum_{i,j \in [-1,n]} \gamma_{ij} \widetilde{W}_i \widetilde{A}_j .
\end{aligned}
$$

We define $\gamma_{ij} \triangleq scale_{W,i}\, scale_{A,j}$. The above result corresponds to the general form of Theorem 1. For different quantization basis functions, the series expansion offers flexible options by adjusting W_{sa}, A_{sa} and $bias$. Figure 2 illustrates the optimization strategies and available options for low-bit tensor multiplication.

For a single layer of the neural network $Layer(W, A)$, we construct the low-bit expansion of the layer as follows: 1) Expanding the original tensor multiplication. 2) Choosing one term of low-bit tensor multiplication, $\gamma_{ij} W_i A_j$, as a kernel, build the layer $Layer(W_i, A_j)$. Then we have single-layer low-bit expansion as follows:

$$
Layer(W, A) = \sum_{i,j \in [-1,n]} \gamma_{ij} Layer(W_i, A_j). \qquad (5)
$$

We use $\widetilde{layer}_{i,j}$ to indicate $Layer(W_i, A_j)$, and use $\hat{Layer}_{i,j}$ to indicate $\gamma_{ij} Layer(W_i, A_j)$.

Similarly, due to the properties of tensor multiplication and the convergence of single-layer low-bit expansion, we can easily obtain the convergence of single-layer expansion.

Model Low-Bit Expansion

A deep model consists of multiple layers. When expanded into t basis models, all layers whose computational kernels

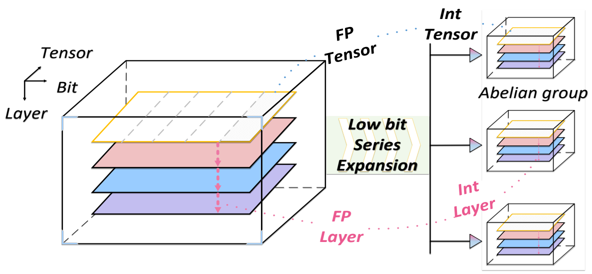

Figure 3: Our series expansion at different levels and specific operation details. Finally, the FP model is expanded into the sum of multiple INT models.

involve multiplications are decomposed, while parameter-free layers are simply duplicated in each basis model. Thus, the expansion of the model follows 3 steps: 1) expand all tensor multiplications layer by layer; 2) duplicate the remaining non-parametric layers and average their outputs; 3) construct $\widehat{model}_{i,j}$, where each $\widehat{model}_{i,j}$ selects the term $\widehat{Layer}_{i,j}$ from the low-order expansion of its corresponding original layer and aggregates them across all layers.

Establishing the Abelian Group. For the full model, we focus on designing operations on basis models that allow efficient parallelization across multiple expanded models. Conventional multiply–add chains enforce strict sequential dependencies, which prevent such fusion. To overcome this, we introduce new operations so that the set of basis models forms an Abelian group. This structure mirrors the commutative reduction primitives (e.g., AllReduce) commonly used in distributed training. First, we generalize addition to neural networks by defining the operation AbelianAdd ($\uplus$).

Definition 1. $\uplus$*: In neural networks, the output of each layer is multiplied by the scale, added, and broadcast to the input of the next layer.*

This operation applies to current neural networks because they are composed of layers with parameters and outputs, and AbelianAdd builds an Abelian group. We denote repeated application of $\uplus$ as $\sum_{\uplus,i=0}^{n}$. Let $\mathcal{M}$ denote the set of all isomorphic low-bit base models $\widetilde{model}_{i,j}$, sharing the same architecture but differing in quantized parameters. For any $\forall m_1, m_2 \in \mathcal{M}$, we define:

$$m_1 \uplus m_2 = Model(\mathbf{W}_1 + \mathbf{W}_2, \mathbf{A}_1 + \mathbf{A}_2), \tag{6}$$

where W_1, W_2 and A_1, A_2 are the corresponding weights and activations of m_1 and m_2. Then, the set $(\mathcal{M}, \uplus)$ forms an Abelian group.

Furthermore, since each parameter in layer j can be written as $W = scale_j \times \widetilde{W}$ with an integer-valued matrix $\widetilde{W}$ (and similarly for activations), we can introduce a complementary operation based on scaling—termed AbelianMul—to handle these multiplicative factors explicitly. Next, we define the AbelianMul ($\hat{*}$) as following:

Definition 2. *For a scaling vector* $\boldsymbol{U} = (u_1, u_2, ..., u_k)$, $\boldsymbol{U}\hat{*}model = model(u_1 \times W_1, u_2 \times W_2, ..., u_k \times W_k)$.

In the low-bit expansion of the model, the vector U corresponds to the scale factors. The operations $(\uplus, \hat{*})$ equip $\mathcal{M}$ with an Abelian group structure, enabling efficient construction of basis models without altering the model architecture.

Model Low-Bit Expansion. For multi-layer networks, the basis functions h_i in Eq. 1 are instantiated through the operations AbelianAdd and AbelianMul, which enable the expansion to bypass the curse of dimensionality. Because the weights and activations in each layer can be expressed as integer multiples of their corresponding scaling factors, the resulting low-order expansion of any deep model naturally conforms to Theorem 2.

Theorem 2. *For any locally continuous deep learning model* ***model****, whose core computation kernel is matrix multiplication (or can be equivalently expressed as multiplication), with weights* $\mathbf{W}$ *and activations* $\mathbf{A}$*, we obtain the following expansion:*

$$\begin{aligned} \boldsymbol{model} &= \sum_{\uplus,i,j\in[-1,n]} \widehat{model}_{i,j} \\ &= \sum_{\uplus,i,j\in[-1,n]} scale_{i,j}\hat{*}\widetilde{model}_{i,j}, \end{aligned} \tag{7}$$

where $\widehat{model}_{i,j}$ denotes the scaled version of the low-bit model, and $\widetilde{model}_{i,j}$ denotes the model with integer parameters for each layer.

Proof. Based on the Eq. 6, it follows that

$$\begin{aligned} &\sum_{\uplus,i,j\in[-1,n]} \widehat{model}_{i,j} = \sum_{\uplus,i,j\in[-1,n]} Model(\mathbf{W}_i, \mathbf{A}_j, sample) \\ &= Model\left(\sum_{i\in[-1,n]} \mathbf{W}_i, \sum_{j\in[-1,n]} \mathbf{A}_j, sample\right) \end{aligned} \tag{8}$$

By Theorem 1, the partial sums $\sum_{i\in[-1,n]} \mathbf{W}_i$ and $\sum_{j\in[-1,n]} \mathbf{A}_j$ converge exponentially to $\mathbf{W}$ and $\mathbf{A}$, respectively. Consequently, by local continuity of the model, we obtain $model = \sum_{\uplus,i,j\in[-1,n]} model_{i,j}$.

Based on Theorem 2, the inference process can be expanded into the pattern shown in Figure 3. In this pattern, all $\widetilde{model}$ are low computation resource models whose main computation kernel is low-bit integer or sparse. Each layer computes the quantized activation and tensor multiplication independently, which is shown in the following section, and all reduce the output of each $\widetilde{model}$.

The Complexity of Series Expansion

While Theorem 2 establishes the equivalence of the expansion to the original model, the resulting formulation also enables further reductions in computational complexity when considering parallel execution and model performance.

The Weight Expansion Upper Bound. When both W and A are expanded into t terms, naive expansion requires t^2 low-bit matrix multiplications (Figure 2). However, for a well-trained model with loss ℓ, the approximate effect of quantization error on weights can be linearized:

$$\ell(model(W)) - \ell(model(W + \text{error})) = \frac{\partial \ell}{\partial W} \cdot \text{error} \tag{9}$$

Method	**Bits (W/A)**	**ResNet-18**	**ResNet-34**	**ResNet-50**	**ResNet-101**	**RegNetX-600MF**	**Inception-V3**
Full-precision		71.01	73.30	76.63	77.30	73.52	77.40
				4-bit Weights / 4-bit Activations			
ACIQ [2019]	4/4	67.00	69.10	73.80	-	-	60.40
AdaQuant [2020]	4/4	67.40	70.30	73.70	74.40	68.20	72.60
PD-Quant [2023]	4/4	69.30	-	75.09	-	70.95	-
Pack-PTQ [2025]	4/4	68.74	-	74.74	-	70.96	-
Series_Ex (ours)	4/4	**70.37**	**72.75**	**77.03**	**76.60**	**71.80**	**76.09**
				2-bit Weights / 4-bit Activations			
LAPQ [2019]	2/4	0.18	0.14	0.17	-	0.12	-
CL-Calib [2024]	2/4	65.14	-	70.92	-	64.50	-
Pack-PTQ [2025]	2/4	61.29	-	63.47	-	59.88	-
Series_Ex (ours)	2/4	**70.26**	**71.95**	**74.23**	**75.10**	**70.04**	**74.27**
				2-bit Weights / 2-bit Activations			
ACIQ [2019]	2/2	0.12	0.20	0.11	0.21	-	0.11
AdaQuant [2020]	2/2	0.11	-	0.12	0.14	-	0.13
BRECQ [2021]	2/2	42.54	-	29.01	-	3.62	-
PD-Quant [2023]	2/2	53.08	-	56.98	-	55.13	-
CL-Calib [2024]	2/2	54.45	-	58.30	-	56.39	-
Series_Ex (ours)	2/2	**59.14**	**63.58**	**62.13**	**61.64**	**59.60**	**49.87**

Table 1: Comparison of post-training quantization methods under different bit-width basis settings. Results are averaged over multiple rounds. Our results have significant advantages in extremely low-bit quantization.

As the gradient $\partial\ell/\partial W$ approaches zero after convergence, the loss becomes insensitive to small perturbations in W. Ablations confirm that expanding W beyond 2–3 terms yields negligible accuracy gains. Therefore, only activations require multiple terms, reducing the complexity from $O(t^2)$ to $O(t)$, i.e., $model = \sum_{i=1}^{k}\sum_{j=1}^{t} \text{scale}_{i,j} \widetilde{\hat{*}model}_{i,j}, k \ll t$, instead of a full t^2 combination.

The Computation Complexity of M_{nsy} Multiplication. Note that the M_{nsy} is the tensor whose all elements are one, implying that it is a rank-one matrix. Specifically, $M_{nsy} = \mathbf{1}\mathbf{1}^T$, where $\mathbf{1} = (1, 1, ..., 1)$ is an all-ones vector. So, for many matrix multiplication $MM_{nsy} = M\mathbf{1}^T\mathbf{1} = (M\mathbf{1}^T)\mathbf{1}$. The computation complexity of the latter process is $O(n^2)$.

The Parallelization of Computing $\widetilde{M_i}$. In the proof of Theorem 1, the series expansion algorithm first uses a non-saturated quantization process to produce $\widetilde{M_1}$. During the computation of $\widetilde{M_i}, i > 1$, the maximum element in the corresponding R_i is set to $scale_{i-1}$. This hierarchical scaling, $scale_i = 2^X scale_{i+1}$, allows each term in the expansion to be computed independently, enabling parallel execution. Furthermore, based on the general form of non-saturating symmetric quantization, it can be deduced that the (i, j) element in $\widetilde{M_k}$ is computed as $\widetilde{M_k}(i, j) = INTX(M(i, j)/scale_k) - 2^X INTX(M(i, j)/scale_{k-1})$. By introducing M_{nsy} and M_{sa}, other quantization schemes can be transformed into the non-saturating symmetric form.

Experiments

Experiment Details

We conduct series expansion quantization experiments on various models on Imagenet (Deng et al. 2009) and NLP tasks (Rajpurkar et al. 2016; Williams, Nangia, and Bowman 2017), including ResNet (He et al. 2016), RegNet (Radosavovic et al. 2020), etc. We assess the performance of large language models (LLMs) on the MMLU (Hendrycks et al. 2020) benchmark to evaluate the scalability and generality of our method. Our method sets hyperparameters consistently on all models and quantizes channel by channel. The basis function selects the class of integer quantization functions. We determine the value of $clip^{+/-}$ during saturation quantization to minimize the impact of M_{sa}. In non-saturated quantization, we use the expected quantization noise in the Laplace distribution as the clipping function. We quantify weights and activations separately. Then the activations of all base models are broadcast and quantized. For all PTQ experiments, we set the first and last layer quantization to 8-bits. Our code is based on pytorch. The code will be made public. All experiments are conducted on a single NVIDIA A800 GPU.

Comparison to State-of-the-arts

CNN (ImageNet). We compare our method with state-of-the-art PTQ algorithms (Banner et al. 2018; Cai et al. 2020; Li et al. 2021; Nahshan et al. 2021; Wei et al. 2022; Liu et al. 2023a; Shang et al. 2024; Li et al. 2025; Zhao et al. 2025) under various low-bit basis function settings. Our method consistently improves performance for each low-bit basis function configuration. As shown in Table 1, at W4A4 our approach matches full-precision accuracy, and even surpasses it on ResNet-50, demonstrating that the integer-valued basis functions can effectively approximate floating-point computations. In the more challenging W2A2 setting—ultra low-bit regime where most PTQ methods collapse—our method remains stable, achieving over 60% ac-

Models	Method	Bits (W/A)	Accuracy	Model Size	Training Data	Runtime	FT	Calibration
ResNet-18 (FP:71.01)	DSQ	4/4	69.56	5.81M	1.2M	100h	w/	-
	QDrop	4/4	69.17	5.81M	**0**	0.43h	w/	1024
	PD-Quant	4/4	69.3	5.81M	**0**	1.11h	w/	1024
	Series_Ex(ours)	4/4	**70.37**	5.81M*	**0**	**0.39h**	**w/o**	**0**
	Series_Ex(ours)	2/Mix(2/4/8)	69.01	**3.01M***	**0**	2.5h	**w/o**	**0**
MobileNetV2 (FP:72.49)	DSQ	4/4	64.8	2.26M	1.2M	192h	w/	-
	Series_Ex(ours)	4/4	**71.1**	2.26M*	**0**	**1.94h**	**w/o**	**0**
	Series_Ex(ours)	2/Mix(2/4/8)	65.21	**1.18M***	**0**	5.5h	**w/o**	**0**

Table 2: Comparison of our algorithm with different quantization methods and accuracy time comparison using mixed precision quantization. Our algorithm does not require fine-tuning (FT) and calibration sets.

curacy even on deep architectures.

Model	ResNet-18				
Bits	W3A3	W2A4	W4A2	W8A8	W32A32
AdaQuant	60.09	0.11	-	-	71.01
QDrop	65.56	64.66	57.56	-	71.06
Series_Ex(ours)	**68.8**	**70.26**	**60.57**	**71.00**	71.01
Quant-Time	5.334s	3.236s	4.456s	0.063s	-

Table 3: The performance and time of weights and activations under different low-bit basis functions.

Table 2 compares our series expansion to PTQ and QAT (Gong et al. 2019) without parallelism. Our approach consistently achieves higher accuracy without the need for calibration data or fine-tuning. For instance, on ResNet-18, the model is decomposed into three 4-bit basis models, with activations expanded at inference as per Eq. 7. This expansion reduces storage to just 37.5% of the full-precision model. Even with the extra activation expansion, inference remains faster than real-time quantization methods, as activation quantization parameters are computed once. Table 3 further shows that our framework flexibly adapts to different bit-widths basis. Specifically, INT8 basis functions match FP accuracy, while INT4 and even INT2 deliver up to 40–60% speedup with minimal degradation. These results confirm the series expansion framework enables practical, accurate ultra-low-bit inference including 2-bit.

Method	SQuAD1.1(F1)	MNLI(Acc mm)
Full Prec.	88.42	84.57
AdaQuant	5.17	-
BRECQ	68.58	31.91
QDrop	75.97	69.19
Series_Ex(ours)	**79.30**	**72.31**

Table 4: Performance on NLP tasks compared with existing methods on W4A4 basis.

Transformers (SQuAD and MNLI). Table 4 demonstrates the effectiveness of our series expansion algorithm beyond vision, applied to NLP tasks with BERT. Compared to existing methods, our algorithm does not require additional data, whereas the QDROP method necessitates extra data examples. The series expansion algorithm outperforms existing methods and achieves significant performance improvements. This highlights the versatility and strong generalizability of our approach.

Method	MMLU				
	Hums.	STEM	Social	Other	Avg.
LLaMA3-8B	**59.0**	55.3	76.0	**71.5**	**64.8**
Normal	56.8	52.9	73.6	69.4	62.5
GPTQ	53.3	57.7	74.2	69.0	63.0
QLoRA	49.3	50.3	65.8	64.2	56.7
Series_Ex(ours)	58.9	**55.6**	**76.1**	71.3	64.7
LLaMA2-7B	43.1	36.4	51.6	52.3	45.7
GPTQ	41.3	34.8	48.7	50.6	43.8
LLM-QAT	–	–	–	–	42.7
Series_Ex(ours)	42.8	**36.8**	**51.9**	**52.3**	**45.7**
Qwen2.5-3B	56.6	**61.5**	76.8	70.3	**65.3**
Series_Ex(ours)	**56.6**	61.4	**76.8**	**70.3**	65.2

Table 5: Performance comparison of different 4-bit quantization methods on MMLU across various model sizes.

Int2	Latency(s)	IPS	Speedup	Memory(MB)	Savings	Acc.	Ratio
Orgin.	1.071	93.37	0%	97.80	0%	71.01	0
GPTQ	0.557	181.13	+94%	9.80	-89%	Fail	×
BRECQ	0.601	166.20	+78%	8.80	-91%	42.54	↓28.47
Series_Ex(ours)	0.560	178.57	+92%	15.60	-84%	59.14	↓11.87

Table 6: Comparison with the PTQ method at 2 bit-width in ResNet-50. IPS represents (image per second).

Large Language Models (MMLU). As shown in Table 5, although our work primarily focuses on small models for specific tasks, we still explore its application in LLMs (Touvron et al. 2023; Yang et al. 2024). Following the quantization settings of W4A16, our approach remains effective in LLMs. On MMLU, our method achieves accuracy nearly equivalent to the original model and significantly outperforms existing method (Frantar et al. 2023; Dettmers et al. 2023; Liu et al. 2023b) while maintaining comparable performance, demonstrating the efficiency and superiority of series expansion.

Practical Deployment at Extreme Bit-widths Basis. To demonstrate the practical viability of our approach, we com-

Model	**ResNet-18**		**ResNet-50**		**LLaMA2-7B**
Bit	Int 2	Int 4	Int 2	Int 4	Int 4
Num_Expansion	2W 3A	2W 2A	2W 4A	2W 4A	2W 0A
Memory (MB)	12.4 (↓46%)	23.2 (-)	29.8 (↓41.8%)	51.2 (-)	7427.2 (-)
Latency (ms)	0.2 (↓50%)	0.29 (↓27.5%)	0.55 (↓48.1%)	0.74 (↓30%)	12.7 (↓31.7%)
BOPS (G)	22(↓80.7%)	76 (↓33%)	37 (↓85.1%)	144 (↓41.7%)	1876 (↓96.2%)

Table 7: Computational efficiency and the number of expansion terms under different bit-widths. Num_Expansion: aWbA for a weight and b activation expansions counts. BOPS: Bit Operations. Blue numbers indicate gains over the 8-bit basis model.

pare it against existing PTQ baselines under an aggressive 2-bit quantization setup (Table 6). While prior methods such as GPTQ and BRECQ achieve substantial speedup and memory savings, they either completely fail or suffer significant accuracy degradation, rendering them unsuitable for real-world use. In contrast, our method delivers comparable deployment efficiency—achieving over 90% speedup and 84% memory reduction—while maintaining a notably higher accuracy (59.14%), underscoring its robustness and usability in real-time, resource-constrained applications.

Ablation

Ablation of W_{sa}, W_{nsy}. Figure 4a shows the effect of saturation (W_{sa}) and asymmetric (W_{nsy}) basis functions. Disabling clipping ($W_{sa} = 0$) barely affects accuracy after series expansion on ResNet-50 and ResNet-18, and introducing asymmetric quantization ($bias \times W_{nsy} \neq 0$) yields results almost identical to full precision. These parameters are data-dependent and can be precomputed, confirming that our framework is compatible with various quantization schemes.

Ablation of the Expansion. As illustrated in Figure 4b, increasing the number of expansion terms gradually narrows the activation gap between the quantized and full-precision ResNet-50 models, leading to improved accuracy that quickly saturates. Four expansion terms are sufficient to match full-precision performance; beyond five, the residual error continues to shrink but yields negligible accuracy gains while incurring significant computational overhead. In practice, we stop expansion once the maximum activation discrepancy drops below 10^{-4}. For large Transformer-based models, we find that 2–3 weight expansions and at most 0–3 activation expansions are usually sufficient. Since weight quantization errors are much smaller, weights are typically expanded no more than twice, while activations dominate the expansion. This observation is consistent with our theoretical analysis.

Discussion

Series Expansion $\neq$ Naive Ensemble. Our method differs fundamentally from naively combining multiple INT models. Unlike simple linear ensembles of multiple INT models, which merely average predictions and cannot reconstruct the original model, our approach is grounded in a structured series expansion. This expansion decomposes the full-precision model into a sequence of low-bit basis components with theoretical guarantees. Existing ensemble strategies applied to INT models bring no accuracy gain and sometimes even reduce performance, highlighting the necessity of our expansion framework.

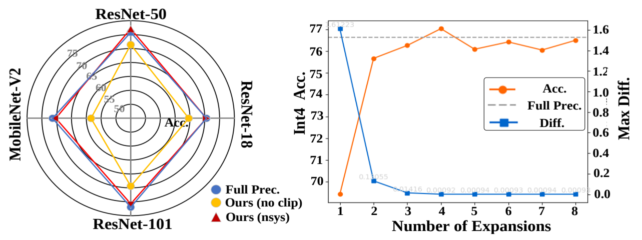

Figure 4: The left sub-figure(a) shows our experiments on saturation and asymmetric quantization. The right sub-figure(b) shows the changes in loss and accuracy as the number of expansions increases.

Efficiency and Limitations. For activations, quantization parameters are computed online only once, and subsequent expansion terms reuse these parameters offline, eliminating repeated calibration. The weights are stored as low-bit integers, and activations are discarded immediately after use. Although multiple lightweight basis models are stored, their small size and limited number keep total memory well below full precision. This basis expansion approach enables efficient and accurate ultra-low-bit (e.g., INT2) deployment.

Table 7 reports the efficiency of our series-expansion framework under different bit configurations, including memory usage, latency, and bit-operation counts. For a fair comparison, all latencies were measured on same hardware (batch size=1, 100 tokens). Relative to practical 8-bit deployment, inference efficiency is significantly improved. When quantized to 4 bits, memory footprint are similar to those of the 8-bit model, yet both latency and compute demand are clearly reduced. At the more aggressive 2-bit setting, expansion allows accuracy to be preserved while nearly halving latency and cutting BOPS by 80–96%.

Conclusion

This paper proposes a deep model series expansion framework that replaces the computationally intensive original model with multiple efficient basis function models. We apply this framework to PTQ, expanding the full-precision model into an integer-based version and prove its convergence. Theoretical and experimental results show that the algorithm improves the parallel capability of the model and guarantees the performance of the low-bit model without the need for calibration sets and fine-tuning.

Ethical Statement

This work adheres to the AAAI Code of Ethics and has no potential ethical implications.

Acknowledgments

This work was supported by the National Key Research and Development Program of China under Grant 2024YFB4505602, the National Natural Science Foundation of China under Grant 62372432 and the Key Technologies Program of Taihang Laboratory in Shanxi Province, China under Grant THYF-JSZX-24010300.

This work was also supported in part by the Major Key Project of PCL under Grant PCL2024A06 and PCL2025A10, and in part by the Shenzhen Science and Technology Program under Grant RCJC20231211085918010. This work was also supported in part by NSFC (62476070).

References

Banner, R.; Nahshan, Y.; Hoffer, E.; and Soudry, D. 2018. ACIQ: Analytical Clipping for Integer Quantization of neural networks. *CoRR*, abs/1810.05723.

Cai, Y.; Yao, Z.; Dong, Z.; Gholami, A.; Mahoney, M. W.; and Keutzer, K. 2020. Zeroq: A novel zero shot quantization framework. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 13169–13178.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.

Dettmers, T.; Pagnoni, A.; Holtzman, A.; and Zettlemoyer, L. 2023. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36: 10088–10115.

Frantar, E.; Ashkboos, S.; Hoefler, T.; and Alistarh, D. 2023. GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers. In *The Eleventh International Conference on Learning Representations*.

Gong, R.; Liu, X.; Jiang, S.; Li, T.; Hu, P.; Lin, J.; Yu, F.; and Yan, J. 2019. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, 4852–4861.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hendrycks, D.; Burns, C.; Basart, S.; Zou, A.; Mazeika, M.; Song, D.; and Steinhardt, J. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Li, C.; Jiang, R.; Song, Z.; Yu, P.; Zhang, Y.; and Guo, Y. 2025. Pack-PTQ: Advancing Post-training Quantization of Neural Networks by Pack-wise Reconstruction. *ArXiv*, abs/2505.00259.

Li, Y.; Gong, R.; Tan, X.; Yang, Y.; Hu, P.; Zhang, Q.; Yu, F.; Wang, W.; and Gu, S. 2021. Brecq: Pushing the limit of post-training quantization by block reconstruction. *arXiv preprint arXiv:2102.05426*.

Liu, J.; Niu, L.; Yuan, Z.; Yang, D.; Wang, X.; and Liu, W. 2023a. Pd-quant: Post-training quantization based on prediction difference metric. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 24427–24437.

Liu, Z.; Oguz, B.; Zhao, C.; Chang, E.; Stock, P.; Mehdad, Y.; Shi, Y.; Krishnamoorthi, R.; and Chandra, V. 2023b. Llm-qat: Data-free quantization aware training for large language models. *arXiv preprint arXiv:2305.17888*.

Nahshan, Y.; Chmiel, B.; Baskin, C.; Zheltonozhskii, E.; Banner, R.; Bronstein, A. M.; and Mendelson, A. 2021. Loss aware post-training quantization. *Machine Learning*, 110(11-12): 3245–3262.

Radosavovic, I.; Kosaraju, R. P.; Girshick, R.; He, K.; and Dollár, P. 2020. Designing network design spaces. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10428–10436.

Rajpurkar, P.; Zhang, J.; Lopyrev, K.; and Liang, P. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Shang, Y.; Liu, G.; Kompella, R. R.; and Yan, Y. 2024. Enhancing Post-training Quantization Calibration through Contrastive Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 15921–15930.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Wei, X.; Gong, R.; Li, Y.; Liu, X.; and Yu, F. 2022. Qdrop: Randomly dropping quantization for extremely low-bit post-training quantization. *arXiv preprint arXiv:2203.05740*.

Williams, A.; Nangia, N.; and Bowman, S. R. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Yang, A.; Yang, B.; Hui, B.; Zheng, B.; Yu, B.; Zhou, C.; Li, C.; Li, C.; and et al., D. L. 2024. Qwen2 Technical Report. *arXiv preprint arXiv:2407.10671*.

Zhao, K.; Zhuang, Z.; Zhang, M.; Guo, C.; Shu, Y.; and Yang, B. 2025. Enhancing Diversity for Data-free Quantization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, 20969–20978. Computer Vision Foundation / IEEE.