2-BIT AUTOREGRESSIVE IMAGE GENERATION ON MOBILE: AN END-TO-END FRAMEWORK VIA COMPLEX-VALUED QAT

Anonymous authors
Paper under double-blind review

ABSTRACT

Image generation is one of the most valuable applications of generative models. As users become increasingly sensitive to data sovereignty and privacy, migrating inference from the cloud to the edge (on-device) has become an urgent necessity. Deploying generative models on mobile hardware requires low storage footprint, low inference latency, and low energy consumption. Model quantization has emerged as the predominant technical trajectory to address these challenges. Quantization-aware training (QAT) embeds quantization operators into the training process, allowing model parameters to actively adapt to low-level representation, thus exceeding the performance of post-training quantization (PTQ) in the three dimensions of storage, latency and energy consumption. This paper proposes a complex-valued QAT framework for autoregressive image generation that addresses channel pairing and layer-wise quantization sensitivity, and supports on-device deployment. We firstly introduce complex-valued quantization methods to the field of image generation. Complex-valued QAT maps weights to the set $\{\pm 1, \pm i\}$, offering superior representation capacity compared to real-valued methods under equivalent storage budgets. And we propose Weight Channel Reorder strategy, which forces complex projection points toward the coordinate axes to significantly suppress quantization error. Extensive experiments on LlamaGen and VAR show that our 2-bit quantized models approach full-precision performance (e.g., FID 2.83 vs. 2.62 on LlamaGen-XL), achieving lower FID than all compared 2-bit QAT baselines and most 6-bit PTQ methods. We further develop a GGUF export pipeline and deploy the model through `llama.cpp` with a Metal GPU backend, achieving fully offline AR image generation on iPhone. Code and demo are available at `https://anonymous.4open.science/r/AR_Quant/`.

1 INTRODUCTION

Among the diverse application domains of generative models, image generation stands out by producing directly deliverable visual content across fields such as design, entertainment, and medical imaging. By offering the shortest path from model output to practical utility, it demonstrates exceptional application value and commercial potential, garnering significant attention from both industry and end-users. However, the large-scale deployment of this technology necessitates a shift in inference from the cloud to the edge Dhar et al. (2021), particularly onto smartphone terminals. On-device inference fundamentally safeguards user privacy, mitigates risks associated with cross-border data transmission and content leakage, and enables offline deployment. Given the increasing sensitivity of users toward data sovereignty and privacy Mothukuri et al. (2021), these features have become essential prerequisites rather than optional enhancements for large-scale personalized applications.

Deploying image generation models on mobile devices necessitates a simultaneous address of three tightly coupled hardware constraints: storage footprint, inference latency, and energy consumption. Model quantization has emerged as a predominant and highly effective strategy Liu et al. (2023); Alayrac et al. (2022); Wang et al. (2023); Dong et al. (2023); Sun et al. (2023); Chen et al. (2024); Dai et al. (2023); Ge et al. (2024) to navigate this "triple constraint" by reducing parameter precision to compress model size and diminish computational overhead. Existing quantization methodologies

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

are generally categorized into two paradigms: Post-Training Quantization (PTQ) Yang et al. (2025) and Quantization-Aware Training (QAT). While PTQ requires no additional training and can achieve reasonable storage reductions, it typically operates within 4-to-8-bit precisions Wu et al. (2020); Frantar et al. (2022); Lin et al. (2024); Frantar & Alistarh (2022). Consequently, inference still relies on dense matrix multiplications, resulting in power consumption that far exceeds the capacities of mobile SoCs. In contrast, QAT integrates quantization operators directly into the training process. This enables the model to maintain generative quality at ultra-low-bit widths providing fundamental improvements across storage, latency, and energy dimensions. Although QAT incurs additional training overhead, a quantized model is trained only once to be deployed for repeated inference across millions of terminals. Thus, the cost of QAT training is entirely acceptable in practical deployment scenarios. Driven by this rationale, this paper adopts the QAT trajectory to explore ultra-low-bit quantization for on-device image generation.

QAT methods can be further divided into two categories according to different representation spaces: real-valued QAT and complex-valued QAT. Representative real-valued approaches, such as BitNet b1.58 Ma et al. (2024), constrain weights to the ternary set $\{-1, 0, +1\}$, while EfficientQAT Chen et al. (2025) employs block-wise training strategies to recover performance at low bit-widths. These methods have demonstrated effectiveness in language models, but are limited by the limited representation capacity of the real domain discrete space and still have obvious shortcomings in image generation that require extremely fine-grained features. Conversely, Complex-valued QAT maps weights onto the complex plane for quantization. A prominent example is Fairy2i Wang et al. (2025a), which quantizes weights into the set $\{\pm 1, \pm i\}$. By leveraging the richer representational of the complex domain, it preserves more nuanced weight while maintaining a 2-bit storage footprint. Compared to real-valued counterparts at equivalent bit-widths, complex quantization offers better representational ability and higher precision. Given the sensitivity of on-device deployment to storage, latency, and power consumption, the potential of complex QAT to achieve high-quality generation at 2-bit makes it a valuable technical trajectory and the methodological foundation of this work.

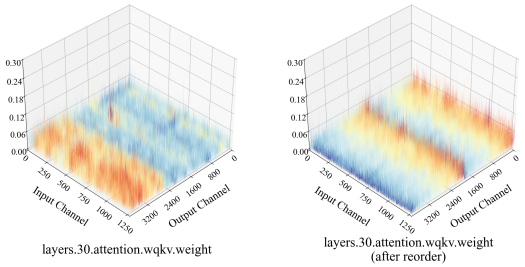


Figure 1: **Weight outliers analysis.** The attention weights of `layer 30` and its reordered of llamagen-xl are shown respectively. Red represents outliers and blue represents ordinary values.

Following this rationale, we investigate how channel pairing and selective precision preservation affect 2-bit autoregressive (AR) image generation, and address these choices through Weight Channel Reorder and sensitivity-guided layer preservation.

- **For the first time, we introduce the application of the Complex-valued QAT framework to the domain of AR image generation.**, achieving high-quality visual synthesis at 2-bit width. We propose hierarchical quantization strategy based on Hessian sensitivity analysis to protect critical layers. Thereby bridging the gap between ultra-low-bit quantization and high-quality generation.

- **We propose a key innovation: Weight Channel Reorder.** We observe that the weights of AR image models exhibit significant outliers in the output channel dimension, where certain channels possess magnitudes far exceeding others. Our reordering strategy rearranges output channels with minimal computational cost to align paired weight distributions, forcing projection points closer to the four axes and significantly reducing error. This method is simple yet elegant, seamlessly integrating with the complex quantization framework with negligible overhead.

- **We demonstrate a complete pipeline from training to on-device deployment.** We developed a GGUF export pipeline that encodes 2-bit complex-quantized models into a compact uint8 format. By integrating the exported model into `llama.cpp/ggml` and using its Metal backend for Apple GPU execution, we achieved fully offline AR image generation on an iPhone.

2 Related Work

2.1 Autoregressive models for image generation

Autoregressive (AR) models Sun et al. (2024); Tian et al. (2024); Xiong et al. (2024); Chang et al. (2022) have achieved remarkable progress in the field of visual generation, with their performance increasingly rivaling or even surpassing that of diffusion-based models. Inspired by the success of Transformers in natural language processing Vaswani et al. (2017), pioneering works such as VQ-VAE Van Den Oord et al. (2017) and VQGAN Esser et al. (2021) reformulate image generation as a sequence modeling task by quantizing images into discrete tokens. Subsequent research has further refined this paradigm: LlamaGen Sun et al. (2024) demonstrates that directly adopting vanilla LLM architectures (e.g., Llama Touvron et al. (2023)) for autoregressive pre-training on image tokens can yield superior image quality and robust generalization, thereby validating the convergence of visual generation and language modeling backbones. Concurrently, VAR Tian et al. (2024) challenges the traditional raster-scan ordering by introducing a "next-scale prediction" mechanism. This hierarchical generation process, progressing from low to high resolution, significantly enhances the model's ability to maintain global structural consistency. These models consistently exhibit prominent scaling laws Kaplan et al. (2020), where performance gains scale predictably with model parameters and data volume. However, such advancements entail prohibitive computational and storage overhead. For on-device deployment, the huge matrix multiplication and weight storage exceed the upper limit of the capabilities of mobile devices.

2.2 Quantization aware training

Mainstream model quantization methodologies are primarily categorized into Post-Training Quantization (PTQ) Yang et al. (2025) and Quantization-Aware Training (QAT). Although PTQ offers the advantage of being training-free, its performance typically collapses in ultra-low-bit regimes due to the drastic increase in rounding noise. In contrast, QAT incorporates well designed quantization operators into the training loop, enabling the model to actively adapt to perturbations induced by discretization. Among early seminal works, LSQ Esser et al. (2019) significantly improved convergence stability at low bit-widths by treating the quantization step size as a learnable parameter. In the era of Large Language Models (LLMs), BitNet b1.58 Ma et al. (2024) introduced a transformative paradigm that implements 1.58-bit (ternary $\{+1, 0, -1\}$) weights from an architectural level, while EfficientQAT Chen et al. (2025) achieves performance recovery at low bit-widths with minimal computational overhead through block-wise training. Furthermore, ParetoQ Liu et al. (2025) investigates the Pareto optimality of non-uniform inter layer bit allocation, iFairy Wang et al. (2025b) and Fairy2i Wang et al. (2025a) leverage complex-valued spaces to enhance the representational capacity of discrete sets. For visual field, Q-ViT Li et al. (2022) alleviates the problem of low-bit quantization accuracy degradation of the visual Transformer through layer-by-layer quantization perception optimization. Based on the above previous work, we decided to introduce complex quantization into the autoregressive image generation model to solve the problem of insufficient representation of real numbers.

3 Method

Our method builds upon the complex-valued QAT paradigm and extends it into a complete training-to-deployment framework for autoregressive image generation, as illustrated in Figure 2. The framework comprises three stages. First, we adapt the complex-valued QAT pipeline — originally designed for language models — to AR visual generation, making specific adjustments for the unique demands of long-sequence visual synthesis. Second, we propose weight channel reorder, a simple yet effective strategy that suppresses quantization error in the complex plane with negligible overhead. Third, we develop an end-to-end deployment pipeline that exports the 2-bit quantized model to GGUF format and runs it in `llama.cpp/ggml`, using the Metal backend to execute inference on Apple GPUs.

3.1 Complex-valued quantization

To perform complex-valued quantization on autoregressive image generation model, the real-valued weights must first be mapped into the complex domain. Consider a standard linear layer defined by $\mathbf{y} = \mathbf{W}\mathbf{x}$, where $\mathbf{x} \in \mathbb{R}^m$, $\mathbf{y} \in \mathbb{R}^n$, and $\mathbf{W} \in \mathbb{R}^{n \times m}$. Vectors are divided into pairs by dimension to

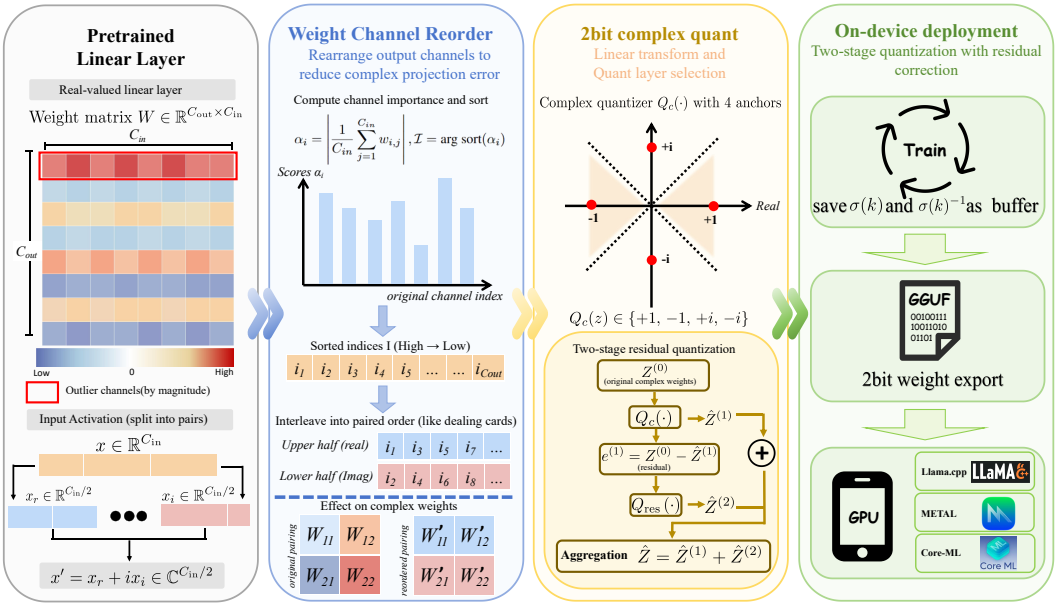


Figure 2: **Overview of the End-to-End 2-Bit Framework.** The complete process covers cross-domain method migration, accuracy optimization, model training to on-device inference deployment: (1) We propose weight channel reorder, aimed at mitigating potential quantization errors in subsequent steps. (2) We introduced the complex quantization method into the autoregressive image generation model and made targeted adaptations. (3) We export the trained model in 2bits and deploy it on mobile device.

represent real and imaginary components, which correspond to complex vectors $\mathbf{x}' = \mathbf{x}_r + i\mathbf{x}_i \in \mathbb{C}^{\frac{m}{2}}$ and $\mathbf{y}' = \mathbf{y}_r + i\mathbf{y}_i \in \mathbb{C}^{\frac{n}{2}}$. Concurrently, the weight matrix $\mathbf{W}$ is decomposed into four sub-blocks $\mathbf{W}_{uv} \in \mathbb{R}^{\frac{n}{2} \times \frac{m}{2}}$. A pair of complex matrices $\mathbf{A}, \mathbf{B} \in \mathbb{C}^{\frac{n}{2} \times \frac{m}{2}}$ for lossless linear transformation is shown below:

$$\begin{aligned} \mathbf{A} = \mathbf{A_r} + i\mathbf{A_i} &= \frac{1}{2}(\mathbf{W}_{11} + \mathbf{W}_{22}) + \frac{i}{2}(\mathbf{W}_{21} - \mathbf{W}_{12}) \\ \mathbf{B} = \mathbf{B_r} + i\mathbf{B_i} &= \frac{1}{2}(\mathbf{W}_{11} - \mathbf{W}_{22}) + \frac{i}{2}(\mathbf{W}_{21} + \mathbf{W}_{12}) \end{aligned} \tag{1}$$

Full derivation can be found in the appendix A. After mapping the real weights to the complex plane, complex quantization directly replaces the complex weights with unit values $\{\pm 1, \pm i\}$ on the nearest axis.

Hierarchical quantization strategy. To preserve generation quality under 2-bit quantization, we retain a small set of sensitive layers in full precision. We analyze the pretrained full-precision model using teacher-forced autoregressive token negative log-likelihood (NLL). Let $\boldsymbol{\theta}$ denote the model parameters, $\ell_n(\boldsymbol{\theta})$ the loss for calibration example n, and $\mathcal{L}(\boldsymbol{\theta}) = N^{-1}\sum_{n=1}^{N} \ell_n(\boldsymbol{\theta})$ the mean calibration loss. For layer l, let $\mathbf{W}_l$ denote its weight matrix and define the vectorized quantization perturbation as $\boldsymbol{\delta}_l = \mathrm{vec}(Q(\mathbf{W}_l) - \mathbf{W}_l)$. If $\boldsymbol{\theta}^{(l)}$ differs from $\boldsymbol{\theta}$ only by quantizing layer l, a second-order Taylor expansion gives

$$\Delta\mathcal{L}_l = \mathcal{L}(\boldsymbol{\theta}^{(l)}) - \mathcal{L}(\boldsymbol{\theta}) \approx \mathbf{g}_l^\top \boldsymbol{\delta}_l + \frac{1}{2}\boldsymbol{\delta}_l^\top \mathbf{H}_l \boldsymbol{\delta}_l, \tag{2}$$

where $\mathbf{g}_l$ and $\mathbf{H}_l$ are the gradient and Hessian of $\mathcal{L}$ with respect to the vectorized weights of layer l, evaluated at the pretrained parameters. This local expansion motivates examining both gradient and curvature information when assessing layer sensitivity.

We use two statistics as empirical proxies for ranking layers. For calibration example n, let $\mathbf{g}_{l,n}$ and $\mathbf{H}_{l,n}$ denote the gradient and Hessian of ℓ_n with respect to the same vectorized layer weights. We define

$$\mathcal{S}_l^{(F)} = \frac{1}{N}\sum_{n=1}^{N} \|\mathbf{g}_{l,n}\|_2^2, \quad \mathcal{S}_l^{(H)} = \frac{1}{N}\sum_{n=1}^{N} \mathbb{E}_{\mathbf{v}}\left[\left(\mathbf{v}^\top \mathbf{H}_{l,n}\mathbf{v}\right)^2\right], \mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{3}$$

Here, $\mathcal{S}_l^{(F)}$ is the trace of the empirical Fisher matrix, and $\mathcal{S}_l^{(H)}$ measures the second moment of directional curvature under Gaussian probes. We estimate the latter using Hessian-vector products without explicitly constructing the Hessian. Both scores are averaged over calibration examples without normalization by layer parameter count. Their utility as ranking proxies is evaluated against measured quantization-induced NLL increases.

Figure 3 shows normalized $\mathcal{S}_l^{(F)}$ and $\mathcal{S}_l^{(H)}$ for ten highly sensitive layers in VAR-d20. To assess whether these indicators predict the loss increase caused by quantization, we independently quantize each of 103 candidate linear layers using the same WCR and two-stage complex quantization procedure, with all other parameters in full precision and no retraining. We compute the Fisher and squared-HVP sensitivity scores using 20 ImageNet validation images. On a separate, non-overlapping set of 128 validation images, we measure the NLL increase caused by quantizing each layer independently; the resulting increases have Spearman correlations of 0.796 and 0.840 with the Fisher and squared-HVP scores, respectively. Quantizing `word_embed` and `head` causes the two largest NLL increases (0.651 and 0.277, respectively). Together with the component ablations, these results motivate retaining both layers in full precision while quantizing the remaining eligible linear layers to 2 bits.

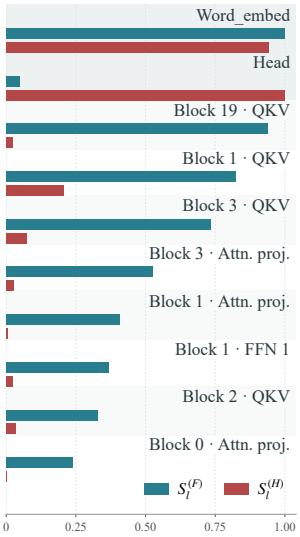


Figure 3: **Layer sensitivity in VAR-d20,** measured by $S_l^{(F)}$ (teal) and $S_l^{(H)}$ (red).

3.2 Weight Channel Reorder

After the complex mapping of real-valued weights via the linear transformation, we obtain complex weights in the form of $A_r + iA_i$ and $B_r + iB_i$. The complex plane is partitioned into four distinct decision regions by the boundaries defined by $y_{\text{im}} = \pm y_{\text{re}}$. These boundaries intersect at the origin to form an "X-shape," and each region is quantized with a point on the axis, $\{\pm 1, \pm i\}$. Specifically, the quantization decision rule is defined as follows:

$$\mathcal{Q}(z) = i^k, \quad \text{where } \arg(z) \in \left[\frac{(2k-1)\pi}{4}, \frac{(2k+1)\pi}{4}\right), \; z \in \mathbb{C} \tag{4}$$

for $k \in \{0, 1, 2, 3\}$. Complex-valued quantization maps points in the complex plane to unit points on either the real or imaginary axis, as shown in equation 4. However, when the numerical distribution of the four sub-matrices $\mathbf{W}_{uv}$ is uneven, $\mathbf{A}$ and $\mathbf{B}$ are further away from the complex plane axis, which will bring higher quantization errors and thus reduce quantization accuracy. This motivates choosing channel pairings that improve axis alignment, which we evaluate alongside their effect on generation quality. Through an analysis of the weight distributions in autoregressive image generation models, we observe that weights within the same output channel often exhibit consistent significance Su et al. (2025); Liang et al. (2025) as shown in Figure 1. To ensure that the projected points in the complex plane lie closer to the axes, it is desirable for the sub-blocks $\mathbf{W}_{1i}$ and $\mathbf{W}_{2i}$ to be as similar as possible. To this end, we propose Weight Channel Reorder:

$$\alpha_i = \left| \frac{1}{C_{in}} \sum_{j=1}^{C_{in}} w_{i,j} \right|, \quad \mathcal{I} = \arg\,\text{sort}_{i \in \{1,\ldots,C_{out}\}}(\alpha_i) \tag{5}$$

We define α_i as the absolute value of the signed mean of output channel i, yielding a ranking score that is sensitive to within-channel sign cancellation. Let $\mathcal{I}$ denote the index sequence sorted in descending order of these scores, where $\mathcal{I}_k$ represents the original row index of the channel ranked k-th.

$$\sigma(k) = \begin{cases} \mathcal{I}_{2k}, & \text{if } 0 \le k < \frac{N}{2} \\ \mathcal{I}_{2(k-\frac{N}{2})+1}, & \text{if } \frac{N}{2} \le k < N \end{cases} \tag{6}$$

The permutation function $\sigma(k)$ is defined in Equation equation 6. Specifically, we divide the weight matrix into two blocks by half of the output channel. The sorted channels are then alternately assigned to the upper and lower blocks in a sequential manner, a process analogous to dealing a deck of cards. After quantization, we will restore the reordered channels to their original indices: $\mathbf{W} \leftarrow \mathbf{W}_{reordered}[\sigma^{-1}(k), :]$, to process the input features normally.

3.3 On-device deployment pipeline

Lifecycle Management of Channel Rearrangement Buffers. Strict consistency of the permutation index $\sigma(k)$ and its inverse $\sigma(k)^{-1}$ must be maintained across the training, exportation, and inference stages to ensure the integrity of the 2-bit complex quantization. During the training stage, we implement a lazy construction and persistent caching mechanism; specifically, for a linear layer with weights $\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, we first compute a ranking score α_i for each output channel i as Equation equation 5. Based on the sorted indices $\mathcal{I}$, the permutation $\sigma(k)$ is constructed by interleaving the channels into upper and lower blocks as shown in Equation equation 6. This permutation and its inverse $\sigma(k)^{-1}$ are registered as persistent buffers (`persistent=True`) during the initial forward pass, enabling the training process to follow the transformation $\mathbf{y} = [\text{ComplexQuant}(\mathbf{W}[\sigma(k), :])][\sigma(k)^{-1}, :] \cdot \mathbf{x}$. To prevent numerical drift between BF16 and FP32 during the exportation stage, $\sigma(k)$ is re-evaluated under the target precision, and the weights are stored in the GGUF format using a rearranged layout complex quantization, with $\sigma(k)^{-1}$ embedded as a standalone `I32` tensor. Finally, during the inference stage, the system executes a one-time "reorder $\rightarrow$ quantize $\rightarrow$ inverse-reorder" pipeline in the first step and caches the resulting quantized weights directly into `weight.data`. This permanent transformation ensures that all subsequent forward passes utilize the optimized weight format with zero additional computational overhead, facilitating efficient deployment on mobile devices.

Weight export. Each complex-valued weight element is encoded with 2 bits (for real-valued weight is 1 bit, because two real-valued weights correspond to one complex-valued weight) after quantization according to the real/imaginary dominant component and symbol, and is packed into 1 byte for every 4 lines. Residual quantization is also stored in 1 bit, ultimately achieving 2 bit compression.

Metal GPU backend. The quantized Transformer backbone runs in `llama.cpp/ggml`. For iPhone deployment, the runtime dispatches quantized matrix operations to the Apple GPU through its Metal backend. The Metal compute kernels unpack the GGUF-encoded weights and execute autoregressive token generation directly on the GPU, avoiding repeated CPU–GPU data transfers. Since the quantized real-valued weights contain only $\{\pm 1, 0\}$, the kernels can replace general multiplications with sign changes or zero-value skips where applicable. The full-precision VQ-VAE decoder is executed after token generation as a separate decoding stage.

4 Experiments

4.1 Experimental Settings

Models and Dataset. All our experiments are conducted on two prominent autoregressive image generation models: LlamaGen Sun et al. (2024) and VAR Tian et al. (2024). These models represent the next-image-token and next-scale autoregressive paradigms. To evaluate the robustness of our quantization framework across different model sizes, we employ two scales for each model. Since we mainly target class to image tasks, we use ImageNet-1k Deng et al. (2009) as training data.

Baseline. We compare against a diverse set of representative quantization approaches, covering both QAT and PTQ paradigms. For QAT baselines, we adopt EfficientQAT, BitNet b1.58, and Fairy2i. The former two represent real-valued ultra-low-bit QAT methods, while Fairy2i provides a direct comparison with existing complex-valued quantization. For PTQ baselines, we include a set of widely used and competitive methods: SmoothQuant Xiao et al. (2023), RTN, OS+ Wei et al. (2023), OmniQuant Shao et al. (2023), QuaRot Ashkboos et al. (2024) and SVDQuant Li et al. (2024). Among them, SVDQuant is specially designed for visual models. They represent different design strategies, including outlier suppression, learnable weight reconstruction, and low-rank approximation.

Metrics. We evaluate the quality and diversity of generated samples using a suite of widely adopted metrics, including Fréchet Inception Distance (FID) Heusel et al. (2017), Inception Score (IS) Salimans et al. (2016), spatial FID (sFID) Esser et al. (2021), as well as precision. All metrics are computed following the protocol provided by OpenAI Dhariwal & Nichol (2021) to ensure fair comparison with prior work.

Training details. The models are initialized from the officially released checkpoints and fine-tuned for 25 epochs under each experimental configuration including baselines. We adopt a two-stage learning rate schedule: the learning rate is set to 1×10^{-5} for the first 20 epochs and then reduced to

Table 1: **ImageNet 256×256 class-conditional evaluation.** Main results of LlamaGen series under different quantization settings. Each setting generates 50,000 images for calculating metrics.

Bit Width	Methods	LlamaGen-L				LlamaGen-XL			
		FID↓	sFID↓	IS↑	Precision↑	FID↓	sFID↓	IS↑	Precision↑
FP	-	3.07	6.17	258.41	0.83	2.62	5.60	246.58	0.81
PTQ (6Bits)	SmoothQuant	4.78	6.80	163.81	0.77	3.53	5.90	177.88	0.76
	OS+	4.80	8.93	267.29	0.83	3.80	8.41	261.67	0.81
	OmniQuant	4.19	8.54	213.22	0.78	3.36	7.92	248.32	0.78
	QuaRot	**3.69**	9.18	244.09	0.83	3.26	8.00	280.87	0.83
	SVDQuant	3.90	9.47	251.87	0.83	3.22	7.56	283.86	0.83
QAT (2Bits)	EfficientQAT	5.02	9.52	208.38	0.79	3.11	6.34	286.27	0.83
	BitNet b1.58	4.95	8.52	191.50	0.80	3.09	7.02	249.86	0.82
	Fairy2i	4.23	9.26	230.48	0.82	<u>3.06</u>	6.76	270.71	0.82
	Ours	<u>3.71</u>	6.62	218.43	0.81	**2.83**	6.28	268.46	0.82

Table 2: **ImageNet 256×256 class-conditional evaluation.** Main results of VAR series under different quantization settings. Each setting generates 50,000 images for calculating metrics.

Bit Width	Methods	VAR-d16				VAR-d20			
		FID↓	sFID↓	IS↑	Precision↑	FID↓	sFID↓	IS↑	Precision↑
FP	-	3.37	7.91	277.42	0.85	2.69	7.45	305.29	0.84
PTQ (6Bits)	RTN	14.91	19.40	139.68	0.62	11.65	12.75	149.47	0.64
	OS+	4.66	12.27	216.07	0.77	3.80	8.94	232.14	0.76
	OmniQuant	**3.95**	8.30	316.12	0.87	**3.01**	7.46	330.59	0.84
	QuaRot	4.76	9.75	212.35	0.77	3.34	9.68	244.33	0.77
	SVDQuant	6.10	10.40	191.58	0.71	4.81	10.77	230.20	0.74
QAT (2Bits)	EfficientQAT	69.24	38.12	26.08	0.36	60.78	21.51	28.38	0.37
	BitNet b1.58	6.41	8.45	175.04	0.82	4.69	10.22	206.34	0.82
	Fairy2i	5.53	10.01	194.11	0.83	4.72	10.92	213.25	0.78
	Ours	<u>4.37</u>	8.54	231.44	0.85	<u>3.28</u>	8.03	250.19	0.83

1×10^{-6} for the remaining 5 epochs. All other hyperparameters follow the original official settings. For LlamaGen, we quantize all linear layers except for the output layer. For VAR, all linear layers are quantized with the exception of the word_embed and head layers.

4.2 Main results

We conducted extensive experiments on the ImageNet 256×256 class-conditional image generation task. Across all four models, our 2-bit method achieves the lowest FID among QAT baselines, including Fairy2i. Notably, LlamaGen-XL attains an FID of 2.83, only approximately 8% higher than its full-precision counterpart (2.62). Compared with Fairy2i, our method consistently reduces FID across both LlamaGen and VAR, with improvements of 0.23 on LlamaGen-XL and 1.44 on VAR-d20. Similar improvements are observed on LlamaGen-L and VAR-d16, demonstrating consistent gains across model scales. Notably, EfficientQAT exhibits unusually high FID on VAR despite achieving a validation loss of 6.068, comparable to our 6.089, suggesting degraded generation quality rather than training divergence, potentially due to a mismatch between grouped affine quantization and the VAR architecture. Moreover, despite using only 2 bits, our complex-valued quantization remains competitive with 6-bit PTQ methods, surpassing all of them on LlamaGen-XL and most on VAR-d20. These results demonstrate the effectiveness of complex-valued quantization across two distinct image generation paradigms.

4.3 Storage and Inference Efficiency

To evaluate storage and inference efficiency on mobile devices, we export the quantization-aware-trained VAR-d20 and LlamaGen-XL checkpoints from PyTorch to GGUF and integrate them with `llama.cpp/ggml`. The export retains the components required to run each generator outside the PyTorch environment while storing the quantized Transformer weights compactly. For VAR-d20, the model size decreases from 2293.76 to 186.82 MiB, a 12.3× reduction, as shown in Figure 4. LlamaGen-XL decreases from approximately 2959.36 MB to 304 MB, a 9.7× reduction. Energy consumption is estimated from the battery drain over 100 images, with idle power consumption subtracted.

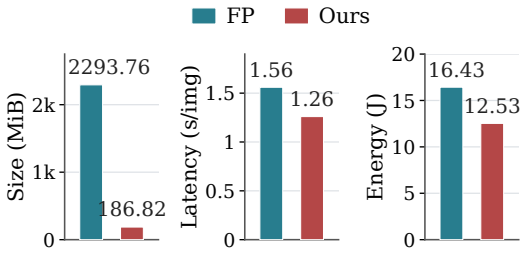


Figure 4: **Metal inference efficiency of VAR-d20.** Model size, generation latency, and energy consumption of the full-precision and 2-bit models.

During inference, the mobile runtime follows each model's reference PyTorch generation procedure. It uses the exported class embeddings, token embeddings, and image-grid metadata together with the corresponding CFG, temperature, and top-k/top-p sampling settings. Each model retains its own latent-code generation schedule, and the generated codes are passed to the existing VQ decoder to reconstruct the image. To match weight-only quantization-aware training, activations remain in BF16 rather than being quantized at inference time. On an iPhone 17 Pro Max equipped with an A19 Pro chip, `llama.cpp/ggml` coordinates autoregressive generation and dispatches the quantized Transformer computations to the Apple GPU through Metal. Under this end-to-end setup, VAR-d20 generation time decreases from 1.56 to 1.26 s/image with 2-bit quantization, corresponding to a 1.24× speedup, while energy consumption decreases from 16.43 to 12.53 J/image. An on-device result is shown in Figure 5, with additional samples in Appendix B.

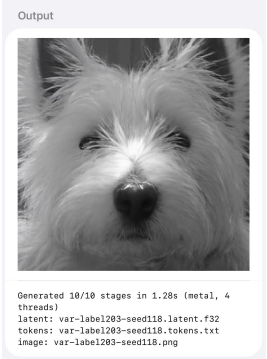


Figure 5: **iOS deployment** of class label 203.

5 Ablation Study

This section presents our results on channel reorder, layer-wise sensitivity, and zero-shot capability. More extensive ablation experiments are provided in the Appendix C.

5.1 WCR and Quantized Layer Selection

Table 3: **Ablation study on WCR and quantized layer selection.** Quant ratio denotes the proportion of linear-layer parameters quantized at the specified bit width.

Fairy2i	WCR	skip `word_embed`	skip `head`	skip `ada_lin`	Bits	Quant ratio	FID↓	sFID↓	IS↑	Precision↑
✓	✗	✗	✗	✗	2	100.000%	4.72	10.92	213.25	0.78
✓	✓	✗	✗	✗	2	100.000%	3.96	8.27	239.91	0.85
✓	✗	✓	✓	✗	2	99.118%	3.54	8.96	252.04	0.82
✓	✓	✓	✗	✗	2	99.992%	3.35	8.46	258.47	0.83
✓	✓	✓	✓	✗	2	99.118%	3.28	8.03	250.19	0.83
✓	✓	✓	✓	✓	2	65.778%	3.16	8.80	257.30	0.82
✗	✗	✗	✗	✗	4	100.000%	4.65	9.97	226.20	0.84

Table 3 jointly evaluates WCR and hierarchical layer selection on VAR-d20. Without WCR, preserving `word_embed` and head reduces FID from 4.72 to 3.54, while adding WCR under the same preservation setting further reduces it to 3.28, showing that both components contribute to generation quality. Adding WCR to the fully quantized 2-bit model reduces FID from 4.72 to 3.96 and sFID from 10.92 to 8.27, confirming its effectiveness in suppressing quantization error. Notably, this 2-bit

configuration outperforms the RTN 4-bit baseline FID 4.65, despite using only half the bit width. Keeping the sensitive `word_embed` and `head` layers in full precision further reduces FID to 3.35 and 3.28, consistent with the sensitivity analysis in Section 3.1. Although additionally excluding `ada_lin` achieves an FID of 3.16, the quantization ratio drops sharply from 99.118% to 65.778%, while sFID degrades to 8.80. We therefore retain full precision only for `word_embed` and `head`.

5.2 WCR Score Ablation

Table 4 compares different channel-scoring functions for constructing the WCR permutation. Random reordering provides only a limited improvement over the vanilla layout, whereas magnitude-aware scores substantially improve axis alignment. Both mean absolute magnitude and RMS reduce Axis NMSE to approximately 0.135 and raise the near-axis rate above 51%. Our score achieves the lowest Axis NMSE of 0.1324 and the highest near-axis rate of 51.87%. Unlike mean absolute magnitude and RMS, which measure only channel energy, our score takes the absolute value after signed aggregation, thereby preserving information about within-channel sign cancellation and directional bias. Table 4 supports improved axis alignment under this scoring rule, while Table 3 separately demonstrates the generation-quality benefit of WCR.

Table 4: **Ablation of WCR channel-scoring functions.** Lower Axis NMSE and a higher near-axis rate indicate better alignment with the complex-plane axes.

Metrics	Vanilla	Random	$mean(\|W_c\|)$	RMS	Ours
Axis NMSE↓	0.2293	∼0.21	0.1347	0.1351	0.1324
Near-axis rate↑	48.94%	∼49.59%	51.51%	51.49%	51.87%

5.3 Zero-shot Capability

As illustrated in Figure 6, we evaluate the zero-shot performance of our method on the VAR-d20 model using 2-bit complex-valued quantization. Our approach successfully preserves the inherent zero-shot capabilities of the VAR framework across various tasks, including in-painting, out-painting, and class-conditional editing. In contrast to BitNet b1.58, which struggles to maintain structural coherence, our method effectively captures contextual dependencies between image tokens.

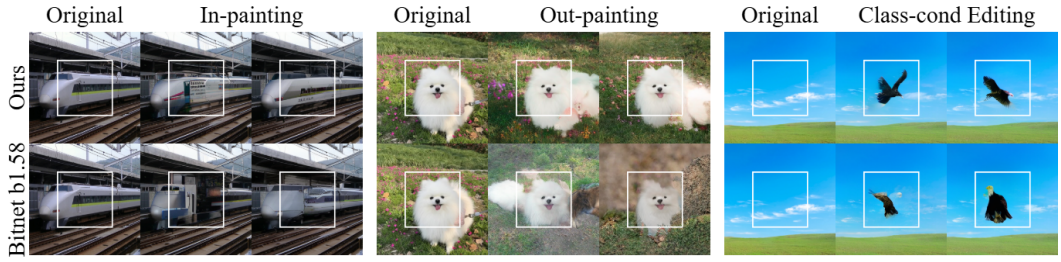


Figure 6: **Zero-shot tasks.** Three different tasks were compared with BitNet b1.58 on VAR-d20. The leftmost column of each set of pictures is the original image, and the inside and outside of the white box are the regenerated tiles.

6 Conclusion

We presented a complex-valued QAT framework for AR image generation that addresses channel pairing and layer-wise precision allocation. WCR selects channel pairings using weight statistics, while sensitivity analysis and quantization interventions motivate preserving a small set of layers at full precision. Experiments on LlamaGen and VAR show lower FID than the compared ultra-low-bit QAT baselines. On VAR-d20, component ablations demonstrate improvements from each component and a combined FID of 3.28 with 99.118% of linear-layer parameters quantized. On LlamaGen-XL, the framework achieves FID 2.83 compared with 2.62 at full precision. GGUF export and the llama.cpp/ggml Metal backend enable fully offline iPhone inference, connecting the quantization results to a working mobile deployment.

AI USE STATEMENT

We only use AI tools for language polishing. Specifically, we used a large language model (LLM) to assist in refining the wording, improving readability, correcting grammatical issues, and enhancing the clarity of several sections of the paper. The AI tools were not involved in the conception or design of the research methodology, benchmark construction, experimental design, data collection, annotation, implementation, evaluation, result analysis, or scientific interpretation. All research decisions, experiments, analyses, and conclusions were independently conducted and verified by the human authors. The authors take full responsibility for the content, correctness, and originality of the paper.

REFERENCES

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736, 2022.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. *Advances in Neural Information Processing Systems*, 37:100213–100240, 2024.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11315–11325, 2022.

Mengzhao Chen, Wenqi Shao, Peng Xu, Jiahao Wang, Peng Gao, Kaipeng Zhang, and Ping Luo. Efficientqat: Efficient quantization-aware training for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10081–10100, 2025.

Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 24185–24198, 2024.

Xiaoliang Dai, Ji Hou, Chih-Yao Ma, Sam Tsai, Jialiang Wang, Rui Wang, Peizhao Zhang, Simon Vandenhende, Xiaofang Wang, Abhimanyu Dubey, et al. Emu: Enhancing image generation models using photogenic needles in a haystack. *arXiv preprint arXiv:2309.15807*, 2023.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Sauptik Dhar, Junyao Guo, Jiayi Liu, Samarth Tripathi, Unmesh Kurup, and Mohak Shah. A survey of on-device machine learning: An algorithms and learning theory perspective. *ACM Transactions on Internet of Things*, 2(3):1–49, 2021.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Runpei Dong, Chunrui Han, Yuang Peng, Zekun Qi, Zheng Ge, Jinrong Yang, Liang Zhao, Jianjian Sun, Hongyu Zhou, Haoran Wei, et al. Dreamllm: Synergistic multimodal comprehension and creation. *arXiv preprint arXiv:2309.11499*, 2023.

Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12873–12883, 2021.

Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. *arXiv preprint arXiv:1902.08153*, 2019.

Elias Frantar and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. *Advances in Neural Information Processing Systems*, 35:4475–4488, 2022.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Yuying Ge, Sijie Zhao, Jinguo Zhu, Yixiao Ge, Kun Yi, Lin Song, Chen Li, Xiaohan Ding, and Ying Shan. Seed-x: Multimodal models with unified multi-granularity comprehension and generation. *arXiv preprint arXiv:2404.14396*, 2024.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Muyang Li, Yujun Lin, Zhekai Zhang, Tianle Cai, Xiuyu Li, Junxian Guo, Enze Xie, Chenlin Meng, Jun-Yan Zhu, and Song Han. Svdquant: Absorbing outliers by low-rank components for 4-bit diffusion models. *arXiv preprint arXiv:2411.05007*, 2024.

Yanjing Li, Sheng Xu, Baochang Zhang, Xianbin Cao, Peng Gao, and Guodong Guo. Q-vit: Accurate and fully quantized low-bit vision transformer. *Advances in neural information processing systems*, 35:34451–34463, 2022.

Yesheng Liang, Haisheng Chen, Zihan Zhang, Song Han, and Zhijian Liu. Paroquant: Pairwise rotation quantization for efficient reasoning llm inference. *arXiv preprint arXiv:2511.10645*, 2025.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of machine learning and systems*, 6: 87–100, 2024.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

Zechun Liu, Changsheng Zhao, Hanxian Huang, Sijia Chen, Jing Zhang, Jiawei Zhao, Scott Roy, Lisa Jin, Yunyang Xiong, Yangyang Shi, et al. Paretoq: Improving scaling laws in extremely low-bit llm quantization. *arXiv preprint arXiv:2502.02631*, 2025.

Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. The era of 1-bit llms: All large language models are in 1.58 bits. *arXiv preprint arXiv:2402.17764*, 2024.

Viraaji Mothukuri, Reza M Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. A survey on security and privacy of federated learning. *Future Generation Computer Systems*, 115:619–640, 2021.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. Omniquant: Omnidirectionally calibrated quantization for large language models. *arXiv preprint arXiv:2308.13137*, 2023.

Zunhai Su, Zhe Chen, Wang Shen, Hanyu Wei, Linge Li, Huangqi Yu, and Kehong Yuan. Rotatekv: Accurate and robust 2-bit kv cache quantization for llms via outlier-aware adaptive rotations. *arXiv preprint arXiv:2501.16383*, 2025.

Peize Sun, Yi Jiang, Shoufa Chen, Shilong Zhang, Bingyue Peng, Ping Luo, and Zehuan Yuan. Autoregressive model beats diffusion: Llama for scalable image generation. *ArXiv*, abs/2406.06525, 2024. URL `https://api.semanticscholar.org/CorpusID:270371603`.

Quan Sun, Qiying Yu, Yufeng Cui, Fan Zhang, Xiaosong Zhang, Yueze Wang, Hongcheng Gao, Jingjing Liu, Tiejun Huang, and Xinlong Wang. Emu: Generative pretraining in multimodality. *arXiv preprint arXiv:2307.05222*, 2023.

Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *Advances in neural information processing systems*, 37:84839–84865, 2024.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Feiyu Wang, Xinyu Tan, Bokai Huang, Yihao Zhang, Guoan Wang, Peizhuang Cong, and Tong Yang. Fairy2i: Training complex llms from real llms with all parameters in $\{\pm 1, \pm i\}$. *arXiv preprint arXiv:2512.02901*, 2025a.

Feiyu Wang, Guoan Wang, Yihao Zhang, Shengfan Wang, Weitao Li, Bokai Huang, Shimao Chen, Zihan Jiang, Rui Xu, and Tong Yang. iFairy: the First 2-bit Complex LLM with All Parameters in $\{\pm 1, \pm i\}$. *arXiv preprint arXiv:2508.05571*, 2025b.

Wenhai Wang, Zhe Chen, Xiaokang Chen, Jiannan Wu, Xizhou Zhu, Gang Zeng, Ping Luo, Tong Lu, Jie Zhou, Yu Qiao, et al. Visionllm: Large language model is also an open-ended decoder for vision-centric tasks. *Advances in Neural Information Processing Systems*, 36:61501–61513, 2023.

Xiuying Wei, Yunchen Zhang, Yuhang Li, Xiangguo Zhang, Ruihao Gong, Jinyang Guo, and Xianglong Liu. Outlier suppression+: Accurate quantization of large language models by equivalent and effective shifting and scaling. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 1648–1665, 2023.

Di Wu, Qi Tang, Yongle Zhao, Ming Zhang, Ying Fu, and Debing Zhang. Easyquant: Post-training quantization via scale optimization. *arXiv preprint arXiv:2006.16669*, 2020.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*, pp. 38087–38099. PMLR, 2023.

Jing Xiong, Gongye Liu, Lun Huang, Chengyue Wu, Taiqiang Wu, Yao Mu, Yuan Yao, Hui Shen, Zhongwei Wan, Jinfa Huang, et al. Autoregressive models in vision: A survey. *arXiv preprint arXiv:2411.05902*, 2024.

Jiawei Yang, Zhongbo Li, Zeqin Feng, and Yongqiang Xie. A survey on neural network quantization. In *Proceedings of the 2025 6th International Conference on Computer Information and Big Data Applications*, CIBDA '25, pp. 384–394, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400713163. doi: 10.1145/3746709.3746773. URL https://doi.org/10.1145/3746709.3746773.

A LOSSLESS REAL-COMPLEX TRANSFORMATION DERIVATION

Here we will prove in detail the weighted real-complex linear transformation in Section 3.1:

Theorem.

Let $\mathbf{W} \in \mathbb{R}^{n\times m}$ be a real-valued matrix partitioned as

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}_{11} & \mathbf{W}_{12} \\ \mathbf{W}_{21} & \mathbf{W}_{22} \end{bmatrix} \tag{7}$$

where each block is of size $\frac{n}{2} \times \frac{m}{2}$. For any $\mathbf{x} = [\mathbf{x}_r; \mathbf{x}_i] \in \mathbb{R}^m$, define the complex vector $\mathbf{x}' = \mathbf{x}_r + i\mathbf{x}_i \in \mathbb{C}^{\frac{m}{2}}$, and similarly $\mathbf{y}' = \mathbf{y}_r + i\mathbf{y}_i \in \mathbb{C}^{\frac{n}{2}}$ with $\mathbf{y} = [\mathbf{y}_r; \mathbf{y}_i]$.

Then there exist unique matrices $\mathbf{A}, \mathbf{B} \in \mathbb{C}^{\frac{n}{2}\times\frac{m}{2}}$ such that

$$\mathbf{y}' = \mathbf{A}\mathbf{x}' + \mathbf{B}\overline{\mathbf{x}'} \tag{8}$$

and this transformation is lossless, i.e., equivalent to $\mathbf{y} = \mathbf{W}\mathbf{x}$.

Proof.

We break down the complex equation $\mathbf{y}' = \mathbf{A}\mathbf{x}' + \mathbf{B}\overline{\mathbf{x}'}$ back to the real number representation. Let $\mathbf{A} = \mathbf{A_r} + i\mathbf{A_i}$, $\mathbf{B} = \mathbf{B_r} + i\mathbf{B_i}$. Now $\mathbf{A}\mathbf{x}'$ and $\mathbf{B}\overline{\mathbf{x}'}$ can be transformed to:

$$\mathbf{A}\mathbf{x}' = (\mathbf{A}_r + i\mathbf{A}_i)(\mathbf{x}_r + i\mathbf{x}_i) = (\mathbf{A}_r\mathbf{x}_r - \mathbf{A}_i\mathbf{x}_i) + i(\mathbf{A}_i\mathbf{x}_r + \mathbf{A}_r\mathbf{x}_i) \tag{9}$$

$$\mathbf{B}\overline{\mathbf{x}'} = (\mathbf{B}_r + i\mathbf{B}_i)(\mathbf{x}_r - i\mathbf{x}_i) = (\mathbf{B}_r\mathbf{x}_r + \mathbf{B}_i\mathbf{x}_i) + i(\mathbf{B}_i\mathbf{x}_r - \mathbf{B}_r\mathbf{x}_i) \tag{10}$$

Add the above two parts together to get the real part $\mathbf{y_r}$ and the imaginary part $\mathbf{y_i}$ of $\mathbf{y}'$:

$$\mathbf{y}_r = (\mathbf{A}_r + \mathbf{B}_r)\mathbf{x}_r + (\mathbf{B}_i - \mathbf{A}_i)\mathbf{x}_i \tag{11}$$

$$\mathbf{y}_i = (\mathbf{A}_i + \mathbf{B}_i)\mathbf{x}_r + (\mathbf{A}_r - \mathbf{B}_r)\mathbf{x}_i \tag{12}$$

In order to make the above complex number operation equivalent to the real number operation $\mathbf{y} = \mathbf{W}\mathbf{x}$, that is:

$$\begin{bmatrix} \mathbf{y}_r \\ \mathbf{y}_i \end{bmatrix} = \begin{bmatrix} \mathbf{W}_{11} & \mathbf{W}_{12} \\ \mathbf{W}_{21} & \mathbf{W}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x}_r \\ \mathbf{x}_i \end{bmatrix} \tag{13}$$

By equalizing the corresponding coefficients, we get four equations about $\mathbf{A}_r, \mathbf{A}_i, \mathbf{B}_r, \mathbf{B}_i$:

$$\mathbf{W}_{11} = \mathbf{A}_r + \mathbf{B}_r, \quad \mathbf{W}_{12} = \mathbf{B}_i - \mathbf{A}_i \tag{14}$$

$$\mathbf{W}_{21} = \mathbf{A}_i + \mathbf{B}_i, \quad \mathbf{W}_{22} = \mathbf{A}_r - \mathbf{B}_r \tag{15}$$

Now we solve the set of equations (14) (15) and we can get:

$$\mathbf{A}_r = \frac{1}{2}(\mathbf{W}_{11} + \mathbf{W}_{22}), \quad \mathbf{B}_r = \frac{1}{2}(\mathbf{W}_{11} - \mathbf{W}_{22}) \tag{16}$$

$$\mathbf{A}_i = \frac{1}{2}(\mathbf{W}_{21} - \mathbf{W}_{12}), \quad \mathbf{B}_i = \frac{1}{2}(\mathbf{W}_{21} + \mathbf{W}_{12}) \tag{17}$$

Finally combined into the complex form:

$$\mathbf{A} = \frac{1}{2}(\mathbf{W}_{11} + \mathbf{W}_{22}) + \frac{i}{2}(\mathbf{W}_{21} - \mathbf{W}_{12}) \tag{18}$$

$$\mathbf{B} = \frac{1}{2}(\mathbf{W}_{11} - \mathbf{W}_{22}) + \frac{i}{2}(\mathbf{W}_{21} + \mathbf{W}_{12}) \tag{19}$$

Based on the above derivation, we convert the real linear layer to the complex linear layer losslessly.

B MORE ON-DEVICE SAMPLES

Following the experimental setup described in Section 4.3, we deployed the 2-bit complex-quantized LlamaGen-XL and VAR-d20 on an iPhone.

Figure 7 showcases the autoregressive sampling results generated locally on an iPhone. The visual quality remains virtually indistinguishable from the full-precision baseline. Despite the substantial scale of VAR-d20 (600M parameters) and the requirement of generating 680 tokens across 10 stages, the system achieves an average generation time of 1.26 seconds per image, which is acceptable for local execution. Figure 8 demonstrates that the inference quality of our quantized LlamaGen-XL (775M parameters) on mobile devices is perfectly consistent with that of high-end GPUs.

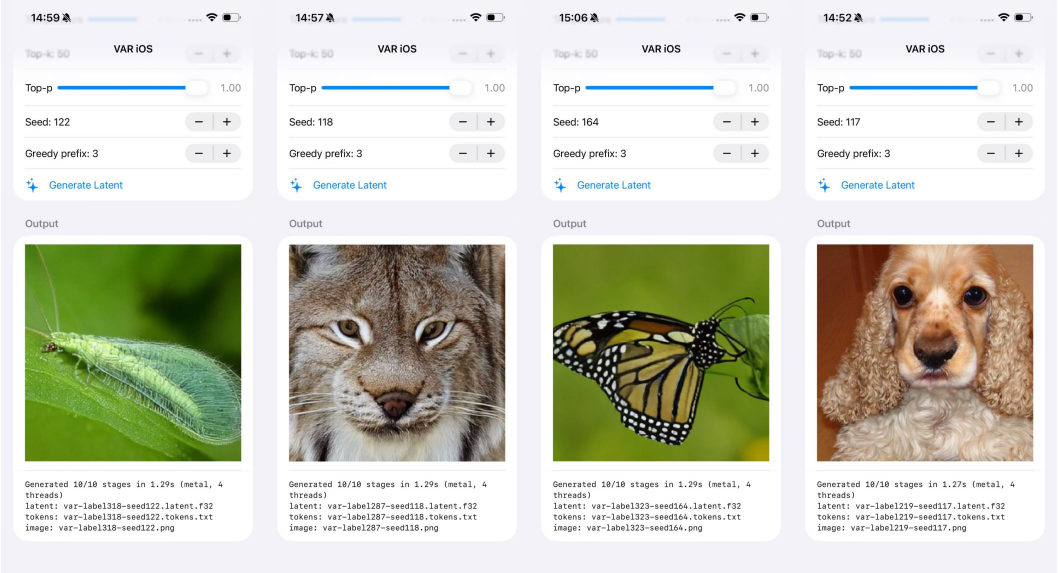


Figure 7: **VAR samples on iPhone.** From left to right are image samples with a size of 256*256 generated under random seeds with class labels [318,287,323,219].

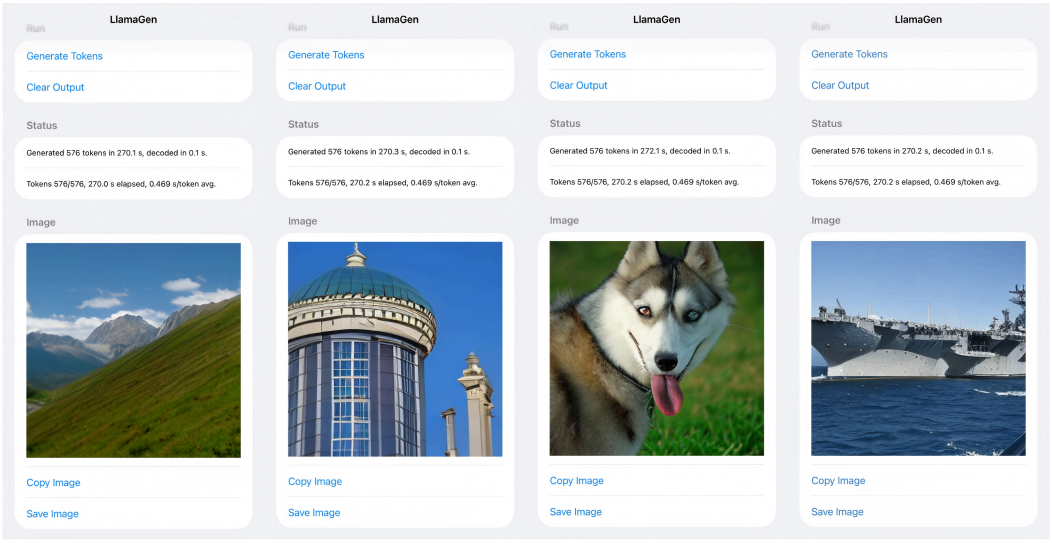


Figure 8: **LlamaGen samples on iPhone.** From left to right are image samples with a size of 384*384 generated under random seeds with class labels [979,538,250,403].

C MORE ABLATIONS

All experiments were conducted on 8 Nvidia H800s. All results for LlamaGen and VAR-d16 were obtained with a classifier-free guidance (CFG) scale of 2.0, VAR-d20 utilizes a CFG scale of 1.5.

C.1 TRAIN LOSS COMPARISON

As illustrated in Figure 9, our complex quantization method demonstrates rapid convergence within 20k training steps and achieves lower training loss than BitNet b1.58. Specifically, our approach significantly outperforms BitNet b1.58 in terms of loss, which translates to superior FID performance. Interestingly, while EfficientQAT achieves a lower loss than our method on both model scales, it is surpassed by our approach in final FID scores. Furthermore, the FID of EfficientQAT continues to decrease throughout the training process, indicating that the model does not suffer from overfitting. Given that the loss function of LlamaGen is designed for next-token prediction, we hypothesize that EfficientQAT may primarily capture local correlations between adjacent tokens rather than fully comprehending global image information. These findings suggest that complex quantization not only captures the global semantic integrity of images but also introduces effective regularization, thereby enhancing the robustness of the generated images.

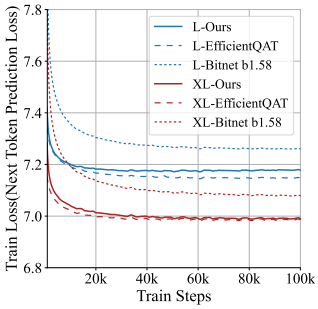


Figure 9: **Train loss comparison.** Here we show the data of the first 100k training steps.

D OPERATOR EFFICIENT IMPLEMENTATION AND TRAINING ACCELERATION

To minimize the additional computational overhead introduced by Quantization-Aware Training (QAT), we implemented a high-performance quantization operator based on the Triton programming language. Our implementation is deeply optimized for modern GPU architectures, focusing on the following three key aspects:

D.1 FULL-PROCESS OPERATOR FUSION

In the original PyTorch implementation, operations such as complex-domain decomposition, residual projection, and weight reconstruction require multiple calls to basic operators. This leads to frequent intermediate tensor allocations and redundant global memory (HBM) read/write operations. To address this, we encapsulate the entire quantization logic into a single fused kernel. By retaining intermediate variables—such as $\mathbf{A}$, $\mathbf{B}$, and their residuals—within on-chip SRAM or registers, our approach reduces the number of memory accesses from several rounds to a single read and a single write. This achieves the theoretical lower bound for memory I/O and significantly mitigates latency in memory-bound scenarios. Consequently, the computational overhead of the quantization operation becomes nearly negligible within the overall training iteration.

D.2 HARDWARE-AWARE MEMORY OPTIMIZATION

To enhance memory access efficiency, we leverage Triton's block pointer technology (`tl.make_block_ptr`) to precisely control data movement, fully exploiting the Tensor Memory Accelerator (TMA) units introduced in the Hopper architecture for asynchronous memory I/O. For the $n/2 \times m/2$ tiling layout, our implementation ensures strict memory alignment and vectorized loading, thus maximizing global memory bandwidth utilization. Furthermore, during the calculation of scaling factors, we employ an optimized reduction tree structure, which significantly boosts computational throughput while maintaining rigorous numerical stability.

D.3 PERSISTENT SCHEDULING AND AUTOTUNING

To accommodate neural network layers of varying scales, we employ a persistent kernel design pattern. By implementing a custom grid scheduling strategy, thread blocks remain active on the

Streaming Multiprocessors (SMs) and process tasks in a loop, effectively minimizing the overhead of frequent kernel launches during large-scale parallel training. Furthermore, we integrate an autotuning mechanism that dynamically identifies the optimal block size and parallelism configuration for diverse hardware environments. This ensures that our operators exhibit superior performance portability and scalability across multiple generations of GPU architectures.

E COMPLEX QUANTIZATION ALGORITHM

Algorithm 1: Complex-Valued Residual Quantization with Channel Reordering

Input: Pre-trained model $\mathcal{M}$, bit-width $b = 2$, training set $\mathcal{D}$.
Output: Quantized model $\mathcal{M}_{quant}$ with complex-valued residual layers.

Phase 1: Global Layer Transformation

for *each linear layer* $L \in \mathcal{M}$ *with weight* $W \in \mathbb{R}^{n \times m}$ **do**

- **Step 1.1: Weight Channel Reorder (Section 3.2)**
- Calculate ranking score α_i for each channel and sort from high to low using Eq. (5)
- Reorder weight channels to align distributions: $W_{reordered} \leftarrow W[\sigma(k), :]$, using Eq. (6)
- **Step 1.2: Real-to-Complex Layer Mapping (Section 3.1)**
- Partition $W_{reordered}$ into four sub-blocks: $W_{11}, W_{12}, W_{21}, W_{22}$
- Construct complex weight components A, B and input x' according to Eq. (1)
- Initialize complex-valued layer $L_{complex}$ to replace L
- **Phase 2: Two-Stage Residual Quantization (Section 3.1)**
- **Step 2.1: Primary Complex Quantization**
- Find optimal scaling factors s_A, s_B for complex components
- $\hat{A}^{(1)} = \text{Dequant}(\mathcal{Q}(A), s_A)$ `// Q is X-shape quantization`
- $\hat{B}^{(1)} = \text{Dequant}(\mathcal{Q}(B), s_B)$
- **Step 2.2: Residual Refinement**
- Compute residual errors: $e_A = A - \hat{A}^{(1)}$ and $e_B = B - \hat{B}^{(1)}$
- Quantize residuals: $\hat{A}^{(2)} = \mathcal{Q}_{res}(e_A)$ and $\hat{B}^{(2)} = \mathcal{Q}_{res}(e_B)$
- Assemble final quantized weights: $\hat{A} = \hat{A}^{(1)} + \hat{A}^{(2)}$, $\hat{B} = \hat{B}^{(1)} + \hat{B}^{(2)}$
- **Phase3: Channel Order Restoration**
- Map complex weights back to real weights, using Eq. (14)(15) Restore row order to align with original model: $\hat{W} \leftarrow \hat{W}_{reordered}[\sigma(k)^{-1}, :]$

end

Phase 4: Quantization-Aware Training (QAT)

while *not converged* **do**

- Sample batch $(x, y) \in \mathcal{D}$
- Perform forward pass: $y' = \hat{A}x' + \hat{B}\overline{x'}$
- Compute loss $\mathcal{L}(y', y)$
- Update full-precision complex weights using **Straight-Through Estimator (STE)**

end

return $\mathcal{M}_{quant}$

F GENERATED SAMPLES

Here we show some image generation results on LlamaGen and VAR after 2-bit complex quantization.



Figure 10: **More LlamaGen generation samples.** From left to right are image samples with a size of 384*384 generated under random seeds with class labels [207,291,387,417,483,972,973,975].



Figure 11: **More VAR generation samples.** From left to right are image samples with a size of 256*256 generated under random seeds with class labels [207,291,387,417,483,812,928,972].

G FISHER DIAGONAL TRACE

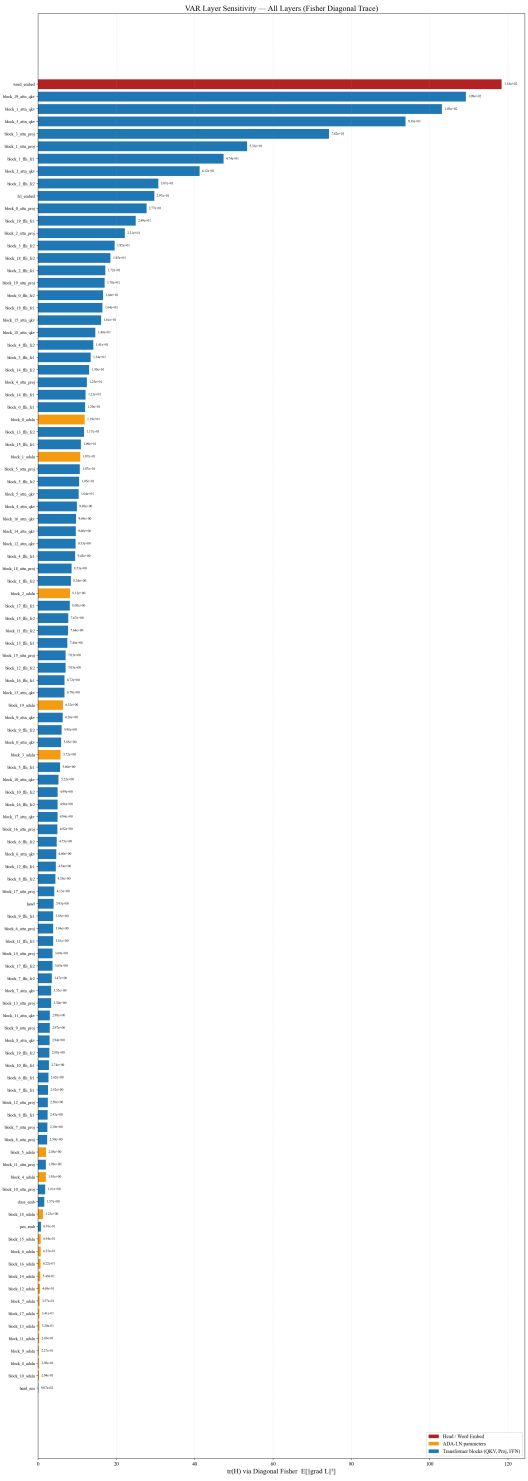


Figure 12: **Full layer-wise $\mathcal{S}_l^{(F)}$ analysis.** Despite containing only 42K parameters (0.007% of the model), the `word_embed` layer yields the highest Fisher trace, indicating extreme per-parameter sensitivity.