BTT: QUANTIZING DIFFUSION LANGUAGE MODELS VIA BLOCK-WISE TOKEN TRANSFORMATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Low-bit activation quantization remains challenging for diffusion language models (dLLMs), whose activations vary across token positions and denoising states. Existing channel transformations reshape feature coordinates within tokens, leaving the token dimension available for complementary transformations. We propose **Block-wise Token Transformation (BTT)** for linear-input activation quantization in dLLMs. BTT applies randomized Hadamard transforms within blocks of simultaneously processed tokens before activation quantization and restores token coordinates after the linear projection. This construction preserves the linear function in exact arithmetic without quantization, while redistributing activations presented to the low-bit quantizer. The block-wise design enables efficient computation and composes naturally with fixed channel rotations. Because token transformations do not benefit all operators equally, we select them independently for each linear operator by minimizing output reconstruction loss on shared calibration inputs with fixed quantized weights, retaining the identity when no candidate improves reconstruction. Experiments show that adding BTT to QuaRot and DuQuant improves downstream accuracy under W4A4 quantization. These results demonstrate the potential of token-dimension transformations as a complement to channel transformations for low-bit dLLM quantization.

1 INTRODUCTION

Diffusion language models (dLLMs) generate text through iterative denoising, processing multiple token positions in each model evaluation (Nie et al., 2025; Ye et al., 2025). This parallel refinement offers an alternative to autoregressive decoding, but repeated denoising passes make efficient deployment important. Post-training quantization (PTQ) reduces the precision of model weights and intermediate activations without requiring full-model retraining. Joint weight–activation quantization is particularly challenging at four bits, where activation outliers and distribution changes across denoising states can substantially affect the quantized computation (Lin et al., 2026; Xu & Yang, 2025; Yan et al., 2026). Improving activation representations before quantization is therefore important for W4A4 dLLM deployment.

Many transformation-based quantization methods address this challenge along the channel dimension. SmoothQuant (Xiao et al., 2023) rescales channels to balance activation and weight quantization difficulty, while QuaRot (Ashkboos et al., 2024) uses orthogonal rotations to redistribute feature outliers. DuQuant (Lin et al., 2024a) combines block-wise rotations and permutations to improve low-bit weight–activation quantization. These methods demonstrate the value of changing the coordinates in which activations are quantized. For the input activation matrix of a linear operator, however, channels are only one available dimension. This raises a complementary question: *can transformations along the token dimension improve dLLM activation quantization beyond a fixed channel transformation?*

dLLMs provide a natural setting for investigating this question because each denoising pass processes multiple token positions together. Masked and unmasked positions can exhibit different feature patterns, which evolve as denoising proceeds. Recent dLLM quantization methods account for this structure through calibration, state-dependent processing, and modeling of errors across denoising steps (Xu & Yang, 2025; Yan et al., 2026; Huang et al., 2026). We explore an additional use of these simultaneously available activations: transforming them across token positions before quan-

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.

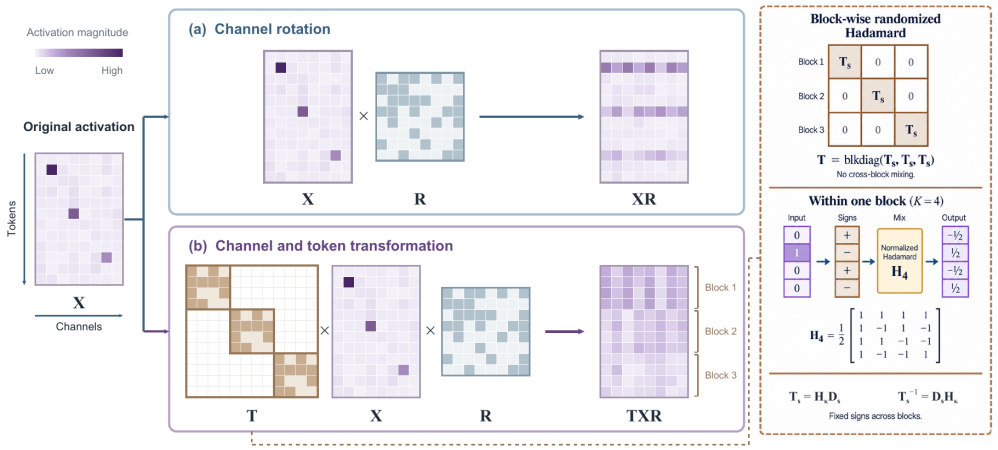


Figure 1: **Channel rotation and Block-wise Token Transformation (BTT).** Channel rotation maps activations to $\mathbf{XR}$, while BTT additionally transforms positions within each token block, yielding $\mathbf{TXR}$. The inset shows the single-block transform $\mathbf{T}_s = \mathbf{H}_K\mathbf{D}_s$, with $\mathbf{T} = \mathrm{blkdiag}(\mathbf{T}_s, \ldots, \mathbf{T}_s)$. The inverse token transform restores token coordinates after the bias-free linear projection.

tization. Because a linear projection shares its weights across tokens, a token transformation can be paired with its inverse after the projection. This allows the activation representation to change locally while preserving the unquantized linear function.

We propose **Block-wise Token Transformation (BTT)** for linear-input activation quantization in dLLMs. BTT applies orthogonal transformations within token blocks before activation quantization and restores the original token coordinates after the quantized projection, before bias addition and subsequent operations. Without quantization, the transform and its inverse cancel in exact arithmetic. With quantization, BTT changes the distribution of channel values within each transformed token and therefore the representation presented to its per-token quantizer. The resulting benefit depends on both the activation patterns within each block and how quantization errors propagate through the operator's weights. We therefore evaluate reconstructed linear outputs when selecting transformations.

BTT uses *block-diagonal randomized Hadamard transformations* to limit computational overhead. Each block transforms a contiguous group of tokens through reproducible random sign flipping followed by a fast Walsh–Hadamard transform. For a fixed block size, the arithmetic cost of the forward and inverse transformations scales linearly with sequence length. BTT also composes naturally with fixed orthogonal channel rotations: the two transformations act along different dimensions, allowing token transformations to be selected while keeping the channel-transformed weights unchanged. Figure 1 illustrates their composition.

Because different linear operators need not benefit from the same transformation, we introduce *per-operator transformation selection*. During calibration, candidate seeds for each operator are evaluated on shared activation inputs and fixed quantized weights, using masked states constructed at multiple masking ratios. Candidate outputs do not alter the forward pass used to collect calibration activations, enabling a controlled comparison of local quantization effects. We select transformations by minimizing pooled output reconstruction loss, retaining the identity when no candidate improves reconstruction. The selected seeds and enable flags remain fixed throughout inference, with no runtime search.

Our experiments show that adding BTT to QuaRot and DuQuant improves downstream accuracy under W4A4 quantization. We examine both local output reconstruction and full-model performance to assess its effects at the operator and model levels. These results support token transformations as a complement to channel transformations for low-bit dLLM quantization.

Our contributions are threefold:

- We propose **BTT** for dLLM linear-input activation quantization, pairing orthogonal transformations along the token dimension with output restoration to preserve the unquantized linear function.
- We instantiate BTT with block-diagonal randomized Hadamard transformations, enabling efficient computation and composition with fixed orthogonal channel rotations.
- We develop per-operator transformation selection using output reconstruction loss, shared-input calibration, and an identity fallback, and demonstrate downstream accuracy gains over QuaRot and DuQuant under W4A4 quantization.

2 Related Work

Post-Training Quantization for LLMs. Post-training quantization (PTQ) has become a widely studied approach for reducing the memory and computational costs of large language models without retraining. GPTQ (Frantar et al., 2023) uses approximate second-order information to perform accurate low-bit weight quantization, while AWQ (Lin et al., 2024b) identifies activation-aware salient weight channels and protects them through equivalent channel scaling. For weight-activation quantization, SmoothQuant (Xiao et al., 2023) migrates quantization difficulty from activations to weights through channel-wise rescaling, enabling efficient W8A8 inference. These methods establish strong PTQ baselines, but their treatment of activation distributions is largely based on channel-wise statistics or weight reconstruction, leaving the structure across token positions mostly unexploited.

Rotation-Based Quantization. A line of work improves low-bit quantization by transforming representations before quantization. QuaRot (Ashkboos et al., 2024) introduces function-preserving orthogonal rotations to suppress hidden-state outliers and enables 4-bit quantization of weights, activations, and KV caches. SpinQuant (Liu et al., 2025) shows that rotation choice can substantially affect quantization performance and learns rotations tailored to quantized models. DuQuant (Lin et al., 2024a) combines block-wise rotations with permutations to redistribute normal and massive activation outliers, achieving strong W4A4 performance. FreeAct (Liu et al., 2026) relaxes the conventional one-to-one correspondence between activation- and weight-side transformations and assigns different activation transformations to different token types.

BTT complements existing transformations by acting across token positions. QuaRot, SpinQuant, and DuQuant transform feature or attention-head coordinates without mixing sequence positions, including in KV-cache quantization (Ashkboos et al., 2024; Liu et al., 2025; Lin et al., 2024a). PrefixQuant (Chen et al., 2026) addresses token-wise outliers by isolating them through prefixed tokens. BTT instead applies orthogonal transformations within token blocks before quantization and restores token coordinates after linear projection. It composes with fixed channel rotations without changing the channel-transformed weights.

Quantization for Diffusion Language Models. Recent work develops quantization methods specifically for diffusion language models (dLLMs). Quantization Meets dLLMs (QDLM) (Lin et al., 2026) systematically evaluates conventional PTQ methods on dLLMs and identifies activation outliers as an obstacle to low-bit weight–activation quantization. DLLMQuant (Xu & Yang, 2025) addresses the dynamic nature of dLLM inference through temporal- and mask-aware calibration, interaction-aware activation quantization, and certainty-guided weight quantization. Quant-dLLM (Zhang et al., 2026) targets extreme low-bit weight quantization using masked calibration simulation, data-aware quantization, and adaptive mixed precision.

Other approaches explicitly account for the state and temporal structure of denoising. STaR-Quant (Yan et al., 2026) distinguishes the activation distributions of masked and unmasked tokens and compensates for quantization errors accumulated across denoising steps. FAIR-Calib (Huang et al., 2026) prioritizes quantization-sensitive token states near the commitment frontier through reweighted calibration. These studies highlight the importance of token states, masking dynamics, and denoising trajectories for dLLM quantization.

BTT investigates a complementary mechanism: applying invertible transformations across simultaneously processed tokens before activation quantization. Calibration determines the transformation and enablement of each linear operator, while these choices remain fixed across denoising steps.

BTT thus uses calibration states to select block-wise token transformations that can be combined with existing channel transformations.

3 METHOD

We propose Block-wise Token Transformation (BTT) for linear-input activation quantization in dLLMs. We first formulate an orthogonal transformation along the token dimension, then instantiate it with block-diagonal randomized Hadamard matrices. BTT composes with fixed channel rotations, while calibration determines the transformation and enablement of each linear operator.

3.1 ORTHOGONAL TOKEN TRANSFORMATIONS

At each denoising step, consider a token-shared linear map $\mathbf{Y} = \mathbf{X}\mathbf{W}$, where $\mathbf{X} \in \mathbb{R}^{L \times C_{\text{in}}}$ and $\mathbf{W} \in \mathbb{R}^{C_{\text{in}} \times C_{\text{out}}}$. Here, L denotes sequence length, and the batch dimension is omitted. Let Q_A and Q_W denote quantization followed by dequantization. We retain the baseline's dynamic per-token activation quantizer, including its range and clipping rule, and fix the quantized weights $\mathbf{W}_Q = Q_W(\mathbf{W})$. The baseline output is $\mathbf{Y}_{\text{base}} = Q_A(\mathbf{X})\mathbf{W}_Q$.

Function-preserving construction. For an orthogonal matrix $\mathbf{T} \in \mathbb{R}^{L \times L}$, we transform token coordinates before activation quantization and restore them after the linear projection:

$$\mathbf{Y}_{\text{BTT}} = \mathbf{T}^\top \left[Q_A(\mathbf{T}\mathbf{X})\mathbf{W}_Q \right]. \tag{1}$$

The token transformation acts from the left, while the shared weight matrix acts from the right. Removing quantization gives

$$\mathbf{T}^\top \left[(\mathbf{T}\mathbf{X})\mathbf{W} \right] = (\mathbf{T}^\top \mathbf{T})\mathbf{X}\mathbf{W} = \mathbf{X}\mathbf{W}. \tag{2}$$

Thus, the construction preserves the linear function in exact arithmetic without changing its weights. Each transform pair operates within one sequence around a token-shared linear projection. Bias terms, omitted in the equations, are added after restoration. The original token coordinates are recovered before subsequent operations such as RoPE, attention-score computation, nonlinear activation, or gating.

Effect on quantization error. In general, $Q_A(\mathbf{T}\mathbf{X}) \neq \mathbf{T}Q_A(\mathbf{X})$. Each transformed row has its dynamic quantization parameters recomputed using the baseline rule. Define $\mathbf{E}_A(\mathbf{T}) = Q_A(\mathbf{T}\mathbf{X}) - \mathbf{T}\mathbf{X}$ and $\mathbf{E}_W = \mathbf{W}_Q - \mathbf{W}$. Then

$$\mathbf{Y}_{\text{BTT}} - \mathbf{X}\mathbf{W} = \mathbf{X}\mathbf{E}_W + \mathbf{T}^\top \mathbf{E}_A(\mathbf{T})\mathbf{W}_Q. \tag{3}$$

For fixed quantized weights, the weight-error contribution $\mathbf{X}\mathbf{E}_W$ is unchanged. The token transformation modifies the activation-error contribution, whose effect depends on projection through $\mathbf{W}_Q$ and its interaction with the weight-error contribution.

Orthogonality preserves total activation energy, $\|\mathbf{T}\mathbf{X}\|_F = \|\mathbf{X}\|_F$, while redistributing values among token rows. This changes the representation within each per-token quantization group. However, smaller absolute peaks alone do not guarantee lower error under dynamic per-token scaling. We therefore select transformations using local output reconstruction rather than activation magnitude alone, with the identity as a fallback (Section 3.4).

3.2 BLOCK-WISE RANDOMIZED HADAMARD CONSTRUCTION

To limit transformation cost, BTT partitions each sequence into contiguous, non-overlapping blocks of size $K = 2^k$, $k \geq 1$. First assume $L = MK$. Let $\mathbf{H}_K$ be the normalized Sylvester Hadamard matrix, satisfying $\mathbf{H}_K^\top = \mathbf{H}_K$ and $\mathbf{H}_K^\top \mathbf{H}_K = \mathbf{I}_K$. For a seed s, generate a Rademacher sign vector $\mathbf{d}_s \in \{-1, +1\}^K$ and define $\mathbf{D}_s = \text{diag}(\mathbf{d}_s)$. The block and sequence transformations are

$$\begin{aligned} \mathbf{T}_s &= \mathbf{H}_K \mathbf{D}_s, \\ \mathbf{T} &= \text{blkdiag}(\mathbf{T}_s, \ldots, \mathbf{T}_s) = \mathbf{I}_M \otimes \mathbf{T}_s, \end{aligned} \tag{4}$$

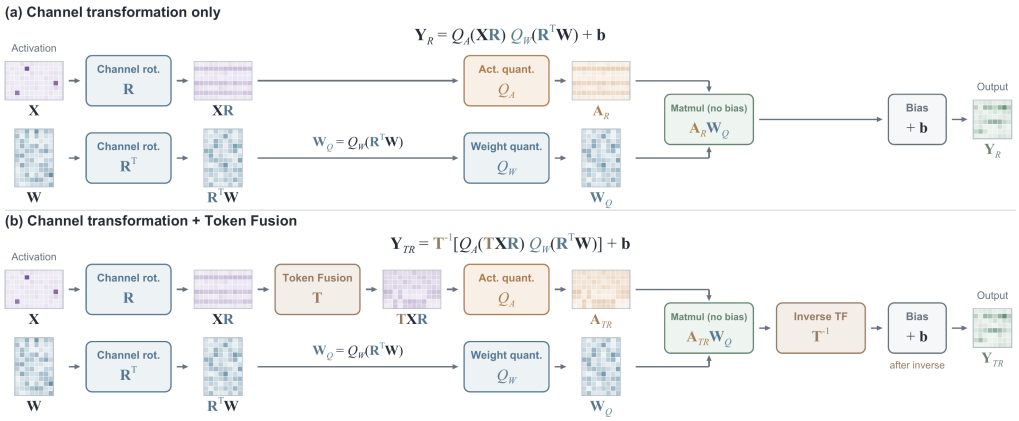


Figure 2: **Composition of BTT and channel rotation.** BTT transforms $\mathbf{XR}$ into $\mathbf{TXR}$ before activation quantization, while retaining the quantized weights $Q_W(\mathbf{R}^\top\mathbf{W})$. The inverse token transformation follows the bias-free projection, and bias is added afterward.

where $\otimes$ denotes the Kronecker product. Each output row combines tokens only within its block, and batch elements are processed separately. The same sign vector is reused across an operator's blocks and denoising steps, while different operators may select different seeds.

Both factors in $\mathbf{T}_s$ are orthogonal, so $\mathbf{T}_s^{-1} = \mathbf{T}_s^\top = \mathbf{D}_s\mathbf{H}_K$. The forward transformation applies random signs followed by a fast Walsh–Hadamard transform (FWHT); the inverse applies the FWHT followed by the same signs. Each Hadamard pass includes the $1/\sqrt{K}$ normalization. A seed and block size specify the transformation without storing a dense matrix.

For arbitrary L, we append zero rows to obtain $\widetilde{L} = K\lceil L/K\rceil$ and use $M = \widetilde{L}/K$ blocks. Padded rows participate in the forward transformation, quantized projection, and inverse transformation because they may become nonzero in transformed coordinates. We retain the first L rows only after restoration and exclude padding from calibration metrics. Padding and cropping are implicit in subsequent formulas.

The arithmetic cost of the forward and inverse transformations is

$$\mathcal{O}\Big(\widetilde{L}(C_{\text{in}} + C_{\text{out}})\log_2 K\Big)\,. \tag{5}$$

For fixed K, this cost scales linearly with sequence length. Larger blocks increase the extent and cost of token mixing, but need not improve quantization accuracy. Actual latency also depends on memory access, kernel implementation, and additional projection rows introduced by padding. Operators assigned the identity skip both transformations.

3.3 Composition with Channel Rotations

BTT can be added to a fixed channel-transformed quantization baseline. We illustrate this composition with a shared orthogonal channel rotation, as used in methods such as QuaRot (Ashkboos et al., 2024) and DuQuant (Lin et al., 2024a). Let $\mathbf{R} \in \mathbb{R}^{C_{\text{in}}\times C_{\text{in}}}$ satisfy $\mathbf{R}^\top\mathbf{R} = \mathbf{I}_{C_{\text{in}}}$ and be shared across the tokens within each block. The effective activation and weight matrices are $\mathbf{XR}$ and $\mathbf{R}^\top\mathbf{W}$.

BTT acts from the left and channel rotation from the right, yielding $\mathbf{TXR}$. Without quantization,

$$\mathbf{T}^\top\left[(\mathbf{TXR})(\mathbf{R}^\top\mathbf{W})\right] = \mathbf{XW}. \tag{6}$$

With quantization, the computation becomes

$$\mathbf{Y}_{\text{joint}} = \mathbf{T}^\top\left[Q_A(\mathbf{TXR})Q_W(\mathbf{R}^\top\mathbf{W})\right]. \tag{7}$$

Channel rotation changes feature coordinates within tokens, while BTT forms linear combinations across positions within token blocks. During BTT selection, channel parameters and quantized

weights remain fixed; only dynamic activation quantization parameters are recomputed for each candidate. Every candidate is therefore evaluated against the same channel-transformed baseline. Figure 2 illustrates the computation.

3.4 Per-Operator Transformation Selection

The benefit of BTT depends on an operator's activation patterns and weights. We therefore select a transformation independently for each linear operator by minimizing local output reconstruction error.

Candidates and shared-input evaluation. For operator l, consider $\mathcal{T}_l = \{\mathbf{I}, \mathbf{T}_{l,1}, \ldots, \mathbf{T}_{l,S}\}$. The randomized candidates use the configured block size and different seeds; the identity disables BTT. Calibration sequences are split into disjoint selection and validation subsets before constructing states at multiple masking ratios. All states from one sequence remain in the same subset. We denote the resulting state sets by $\mathcal{J}_{\text{sel}}$ and $\mathcal{J}_{\text{val}}$.

Each operator input $\mathbf{X}_{l,j}$ is collected from a fixed calibration pass with BTT disabled. All candidates share this input and the quantized weights $\mathbf{W}_{Q,l} = Q_W(\mathbf{W}_l)$. Candidate outputs are evaluated locally without altering the collecting forward pass. When channel transformations are enabled, the inputs and weights below denote the baseline's fixed effective representations. Define

$$\begin{aligned} \mathbf{Y}_{l,j}^{\text{ref}} &= \mathbf{X}_{l,j}\mathbf{W}_l, \\ \mathbf{Y}_{l,j}^{\text{base}} &= Q_A(\mathbf{X}_{l,j})\mathbf{W}_{Q,l}, \\ \mathbf{Y}_{l,j}^{Q}(\mathbf{T}) &= \mathbf{T}^{\top}\left[Q_A(\mathbf{T}\mathbf{X}_{l,j})\mathbf{W}_{Q,l}\right]. \end{aligned} \tag{8}$$

The reference is the full-precision local linear output on the shared input. Using the same input and quantized weights for every candidate isolates their local effects from changes in upstream activations. Dynamic activation-quantization parameters are recomputed for each candidate. Errors are measured after token restoration and exclude padded rows.

Reconstruction objective. For a state set $\mathcal{J}$, define the pooled normalized loss

$$\mathcal{L}_l^{\text{quant}}(\mathbf{T}; \mathcal{J}) = \frac{\sum_{j\in\mathcal{J}} \left\| \mathbf{Y}_{l,j}^{Q}(\mathbf{T}) - \mathbf{Y}_{l,j}^{\text{ref}} \right\|_F^2}{\sum_{j\in\mathcal{J}} \left\| \mathbf{Y}_{l,j}^{\text{ref}} \right\|_F^2 + \epsilon}, \tag{9}$$

where $\epsilon > 0$ ensures numerical stability. We select

$$\widetilde{\mathbf{T}}_l \in \arg\min_{\mathbf{T}\in\mathcal{T}_l} \mathcal{L}_l^{\text{quant}}(\mathbf{T}; \mathcal{J}_{\text{sel}}), \tag{10}$$

preferring the identity in a tie. Including the identity in the candidate set allows BTT to remain disabled for operators whose searched transformations do not strictly reduce reconstruction loss. Define the loss reduction relative to the identity as

$$\Delta_l^{\text{quant}}(\mathbf{T}; \mathcal{J}) = \mathcal{L}_l^{\text{quant}}(\mathbf{I}; \mathcal{J}) - \mathcal{L}_l^{\text{quant}}(\mathbf{T}; \mathcal{J}). \tag{11}$$

A positive value indicates lower local reconstruction error than the baseline without BTT.

4 Experiments

We evaluate BTT on code generation and mathematical reasoning to assess its effect on quantized dLLMs. The experiments cover two model families, W4A4 and W8A8 quantization, and three PTQ baselines. We use output reconstruction loss to select BTT candidates and compare two projection scopes. Since experiment groups differ in operator enablement and inference settings, we report gains relative to the corresponding baseline for each group.

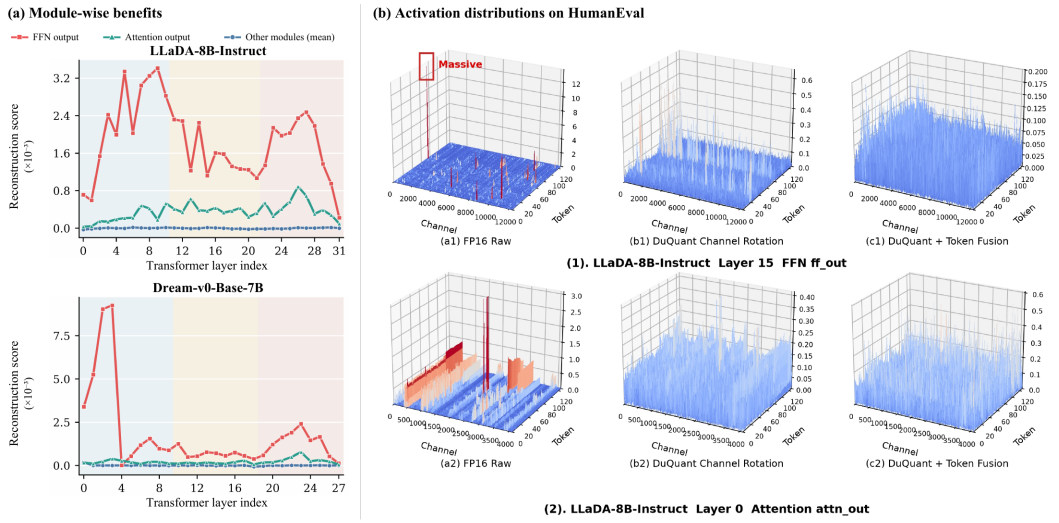


Figure 3: **Module-wise benefits and activation redistribution of BTT.** (a) Reconstruction scores across transformer layers for LLaDA-8B-Instruct (top) and Dream-v0-Base-7B (bottom), using the best of 16 candidate seeds per linear module. "Other modules" denotes the mean over the five remaining module types. (b) Activation magnitudes on HumanEval for the layer-15 FFN and layer-0 attention output projections of LLaDA-8B-Instruct, comparing FP16, DuQuant, and DuQuant with BTT. Vertical scales differ across the activation plots.

4.1 Experimental Setup

Models and benchmarks. We evaluate LLaDA-8B-Instruct and Dream-v0-Base-7B on HumanEval and MBPP for code generation, and GSM8K and MATH for mathematical reasoning. We report pass@1 on HumanEval and MBPP, strict accuracy on GSM8K, and verification accuracy computed with `math_verify` on MATH. Each evaluation uses the full test set specified by its benchmark configuration. All scores are percentages, and differences are reported in percentage points.

Quantization and inference settings. We implement model decoding using Fast-dLLM (Wu et al., 2026). The caching configurations used in each experiment group are specified below and alongside the corresponding tables. We evaluate DuQuant and QuaRot at W4A4, and DuQuant and SmoothQuant at W8A8, where W and A denote weight and activation precision. QuaRot uses GPTQ for weight quantization. Its Dream experiments retain 16-bit keys and values (W4A4K16V16). Dream DuQuant uses prefix caching, whereas Dream QuaRot uses no caching. We include FP16 reference scores and compare each BTT configuration with its corresponding quantized baseline without BTT. Protocol differences are stated alongside the result tables.

BTT configurations. We select candidate transformations by minimizing pooled normalized linear-output reconstruction loss. We evaluate BTT on FFN output projections alone (FFN), and on both FFN and attention output projections (FFN+Attn). These names specify the operators eligible for BTT; identity fallback, when used, may leave individual operators disabled. The DuQuant runs use identity fallback, while the reported QuaRot runs use a variant without it. The QuaRot runs therefore differ from the fallback-enabled procedure described in Section 3.4.

4.2 Main Results

Tables 1 and 2 summarize the downstream results. Applying BTT to FFN output projections improves the average score in every evaluated quantizer–model–precision group. Under W4A4, the average gains range from 1.50 to 3.13 percentage points across DuQuant and QuaRot, with Hu-

Table 1: **W4A4 results with DuQuant and QuaRot.** HE denotes HumanEval. GSM8K uses strict accuracy and MATH uses `math_verify`. All scores are percentages; higher is better. Bold denotes the highest score among the quantized baseline and the two BTT configurations within each method–model group. Avg. Δ is the mean of the four score differences from the corresponding quantizer–model baseline, in percentage points. Upward arrows indicate gains and downward arrows indicate declines; dashes denote reference rows.

	LLaDA-8B-Instruct					Dream-v0-Base-7B				
Method	**HE**	**MBPP**	**GSM8K**	**MATH**	**Avg. Δ**	**HE**	**MBPP**	**GSM8K**	**MATH**	**Avg. Δ**
FP16 reference	43.29	39.40	77.56	32.72	–	49.39	55.20	71.80	34.02	–
DuQuant										
Baseline	34.15	34.40	74.75	30.20	–	37.80	43.80	61.56	25.18	–
+ BTT: FFN	**39.02**	**37.20**	76.35	**30.54**	↑ 2.40	**41.46**	**49.40**	**63.68**	**26.30**	↑ 3.13
+ BTT: FFN+Attn	36.59	34.80	**76.65**	30.08	↑ 1.16	32.93	47.80	63.38	26.14	↑ 0.48
QuaRot (GPTQ)										
Baseline	32.93	33.80	**75.66**	**29.96**	–	31.10	41.60	56.25	22.40	–
+ BTT: FFN	**39.02**	**34.40**	75.28	29.66	↑ 1.50	**37.80**	**41.80**	**58.45**	22.42	↑ 2.28
+ BTT: FFN+Attn	35.37	33.20	75.51	29.94	↑ 0.42	34.76	**41.80**	55.88	**23.10**	↑ 1.05

Table 2: **W8A8 results with DuQuant and SmoothQuant.** HE denotes HumanEval. GSM8K uses strict accuracy and MATH uses `math_verify`. All scores are percentages; higher is better. Bold denotes the highest score among the quantized baseline and the two BTT configurations within each method–model group. Avg. Δ is the mean of the four score differences from the corresponding quantizer–model baseline, in percentage points. Upward arrows indicate gains and downward arrows indicate declines; dashes denote reference rows.

	LLaDA-8B-Instruct					Dream-v0-Base-7B				
Method	**HE**	**MBPP**	**GSM8K**	**MATH**	**Avg. Δ**	**HE**	**MBPP**	**GSM8K**	**MATH**	**Avg. Δ**
FP16 reference	43.29	39.40	77.56	32.72	–	49.39	55.20	71.80	34.02	–
DuQuant										
Baseline	42.07	36.20	**78.09**	32.66	–	50.00	**56.40**	**72.93**	**34.08**	–
+ BTT: FFN	42.68	38.20	77.94	32.48	↑ 0.57	**56.10**	54.60	72.78	33.84	↑ 0.98
+ BTT: FFN+Attn	**43.90**	**39.00**	77.03	**33.06**	↑ 0.99	51.83	54.60	72.78	33.82	↓ 0.10
SmoothQuant										
Baseline	42.68	36.60	**79.30**	32.88	–	51.22	53.60	72.33	33.54	–
+ BTT: FFN	42.68	**39.80**	78.01	**33.46**	↑ 0.62	**51.83**	**54.80**	73.09	**34.14**	↑ 0.79
+ BTT: FFN+Attn	**43.29**	38.20	78.17	32.28	↑ 0.12	49.39	52.20	**73.46**	33.78	↓ 0.47

manEval improvements of up to 6.70 points. These results support the use of token transformations alongside existing channel rotations for low-bit dLLM quantization.

The average gains are smaller at W8A8, ranging from 0.57 to 0.98 points for FFN-only BTT across DuQuant and SmoothQuant. The improvements with SmoothQuant also extend the observed benefits beyond rotation-based baselines. However, the effect varies across benchmarks, and a positive average gain can coexist with declines on individual tasks. All gains are measured against the corresponding baseline under the settings reported in each table.

Extending BTT to attention output projections does not consistently improve performance. FFN-only BTT achieves the higher average score in seven of the eight evaluated groups, while FFN+Attn performs better for LLaDA with DuQuant at W8A8. This pattern motivates selective application of BTT and further analysis of its effects across projection types. The results describe the observed accuracy changes; small differences are not interpreted as statistically established improvements.

4.3 Understanding the Effect of BTT

Layer- and projection-wise benefits. Figure 3(a) reports reconstruction scores across transformer layers for LLaDA-8B-Instruct and Dream-v0-Base-7B. The score measures the reduction in local output reconstruction loss relative to the baseline without BTT; higher values indicate greater im-

Table 3: **Projection scope and transformation controls.** HumanEval and MBPP pass@1 (%) under DuQuant W4A4. The FFN, attention, FFN+Attn, and all-linear rows apply BTT with channel rotation enabled. Bold indicates the best result among the four BTT scopes for each model and task.

	LLaDA		Dream	
Configuration	HumanEval	MBPP	HumanEval	MBPP
Projection scope and controls				
No rotation, no BTT	1.83	0.20	0.00	0.00
No rotation, with BTT	1.22	0.00	0.00	0.00
Rotation only (no BTT)	34.15	34.40	37.80	43.80
FFN output	**39.02**	**37.20**	41.46	**49.40**
Attention output	35.37	34.60	**43.90**	44.20
FFN + attention output	36.59	34.80	32.93	47.80
All linear modules	35.36	34.80	37.20	44.40

provement. Each linear module uses its best candidate among 16 seeds, and "Other modules" averages the five remaining projection types. The comparison illustrates how the benefit of BTT varies across layers, projection types, and models. This variation motivates selecting transformations separately for each operator and retaining the identity when no candidate improves reconstruction. The plotted scores characterize the searched candidates rather than the downstream gains of the final enabled configuration.

Activation redistribution. Figure 3(b) visualizes activation magnitudes on HumanEval for the layer-15 FFN output projection and layer-0 attention output projection of LLaDA-8B-Instruct. The FP16 activations contain localized peaks and channel-aligned structures. Comparing DuQuant with DuQuant plus BTT illustrates how the additional token transformation redistributes activations across positions within each block. The panels use different vertical scales, so their visual appearance alone does not quantify peak suppression or reconstruction improvement. Together with the reconstruction scores, these visualizations illustrate the effect of BTT on activation representations, without implying that every activation maximum decreases or every downstream task improves.

4.4 Ablation Studies

Projection scope and transformation controls. Table 3 examines where BTT should be applied and how it interacts with channel rotation. Among the four BTT scopes, FFN-only BTT achieves the highest scores on both tasks for LLaDA. On Dream, attention-only BTT performs best on HumanEval, while FFN-only BTT performs best on MBPP. Extending BTT to both output projections or all linear modules does not consistently improve accuracy. These results favor selective application of BTT and show that the preferred projection scope can depend on the model and task. The no-rotation controls yield near-zero scores, and adding BTT does not recover performance in these runs, suggesting that token transformations complement channel rotation in this setting.

5 Conclusion

We introduced Block-wise Token Transformation (BTT) for low-bit activation quantization of diffusion language models. BTT applies randomized Hadamard transformations within token blocks and restores token coordinates after linear projection, preserving the unquantized linear function while composing with channel rotations. Per-operator selection based on reconstruction loss and an identity fallback adapts the transformation to local quantization behavior. Experiments demonstrate W4A4 accuracy gains when BTT is combined with QuaRot and DuQuant, while ablations reveal that benefits vary across operators, models, and tasks. These findings support token transformations as a complementary direction for dLLM quantization. Future work includes reducing transformation overhead and improving the alignment between local reconstruction objectives and downstream performance.

AI USE STATEMENT

We used generative AI tools, including ChatGPT and Codex, to assist with manuscript drafting and editing, literature lookup, methodological discussion, interpretation of experimental summaries, and LaTeX preparation. We also used image-generation tools to create figure drafts and explore layouts. The authors take full responsibility for the final manuscript, including all AI-assisted text, figures, technical claims, and reported results.

REPRODUCIBILITY STATEMENT

Section 3 describes the BTT formulation, the randomized Hadamard construction, its composition with channel rotations, and reconstruction-based per-operator transformation selection. Section 4.1 specifies the evaluated models, benchmarks, quantization settings, and BTT configurations. Protocol differences are documented alongside the result tables. Appendix A reports additional experiments on seed selection and block size, while Appendix .3 describes the hardware, workload, timing procedure, and repeated measurements used to assess inference overhead.

REFERENCES

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. In *Advances in Neural Information Processing Systems*, volume 37, pp. 100213–100240, 2024. doi: 10.52202/079017-3180. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/b5b939436789f76f08b9d0da5e81af7c-Abstract-Conference.html.

Mengzhao Chen, Yi Liu, Jiahao Wang, Yi Bin, Wenqi Shao, and Ping Luo. PrefixQuant: Eliminating outliers by prefixed tokens for large language models quantization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2026. doi: 10.1109/TPAMI.2026.3711802. URL https://doi.org/10.1109/TPAMI.2026.3711802.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=tcbBPnfwxS.

Haoyu Huang, Linlin Yang, Sheng Xu, Boyu Liu, Guodong Guo, Zhongqian Fu, Hang Zhou, and Baochang Zhang. FAIR-Calib: Frontier-aware instability-reweighted calibration for post-training quantization of diffusion large language models. *arXiv preprint arXiv:2606.06547*, 2026. URL https://arxiv.org/abs/2606.06547.

Haokun Lin, Haobo Xu, Yichen Wu, Jingzhi Cui, Yingtao Zhang, Linzhan Mou, Linqi Song, Zhenan Sun, and Ying Wei. DuQuant: Distributing outliers via dual transformation makes stronger quantized LLMs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 87766–87800, 2024a. doi: 10.52202/079017-2786. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/9febda1c8344cc5f2d51713964864e93-Abstract-Conference.html.

Haokun Lin, Haobo Xu, Yichen Wu, Ziyu Guo, Renrui Zhang, Zhichao Lu, Ying Wei, Qingfu Zhang, and Zhenan Sun. Quantization meets dLLMs: A systematic study of post-training quantization for diffusion LLMs. *Machine Intelligence Research*, 2026. doi: 10.1007/s11633-025-1624-x. URL https://link.springer.com/article/10.1007/s11633-025-1624-x.

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. In *Proceedings of Machine Learning and Systems*, volume 6, 2024b. URL https://proceedings.mlsys.org/paper_files/paper/2024/hash/42a452cbafa9dd64e9ba4aa95cc1ef21-Abstract-Conference.html.

Xiaohao Liu, Xiaobo Xia, Manyi Zhang, Ji-Fu Li, Xianzhi Yu, Fei Shen, Xiu Su, See-Kiong Ng, and Tat-Seng Chua. FreeAct: Freeing activations for LLM quantization. *arXiv preprint arXiv:2603.01776*, 2026. URL https://arxiv.org/abs/2603.01776.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. SpinQuant: LLM quantization with learned rotations. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=ogO6DGE6FZ.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-1689. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/48b383b24230e0e6e649d9c98dae4d8c-Abstract-Conference.html.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dLLM: Training-free acceleration of diffusion LLM by enabling KV cache and parallel decoding. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=3Z3Is6hnOT.

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 *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 38087–38099. PMLR, 2023. URL https://proceedings.mlr.press/v202/xiao23c.html.

Chen Xu and Dawei Yang. DLLMQuant: Quantizing diffusion-based large language models. *arXiv preprint arXiv:2508.14090*, 2025. URL https://arxiv.org/abs/2508.14090.

Xin Yan, Aqiang Wang, Zhenglin Wan, Xingrui Yu, and Ivor Tsang. STaR-Quant: State-time consistent post-training quantization for diffusion large language models. *arXiv preprint arXiv:2606.04945*, 2026. URL https://arxiv.org/abs/2606.04945.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b: Diffusion large language models. *arXiv preprint arXiv:2508.15487*, 2025. URL https://arxiv.org/abs/2508.15487.

Tianao Zhang, Zhiteng Li, Xianglong Yan, Haotong Qin, Yong Guo, and Yulun Zhang. Quant-dLLM: Post-training extreme low-bit quantization for diffusion large language models. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/805da7ef883245cb35e012cc179a5f6f-Abstract-Conference.html.

Table 4: **Held-out benefits of seed selection.** Δ denotes the additional output-NMSE reduction of the selected seed relative to the mean of 16 candidates. Values and 95% confidence intervals are in units of 10^{-6}; higher is better. Intervals use paired module-level percentile bootstrap. Win rate is the percentage of modules with positive improvement. "All" includes the five remaining linear projection types.

Model	Projections	Modules	Δ	95% CI	Win (%)
LLaDA	All	224	4.87	$[3.60, 6.23]$	73.7
	FFN out.	32	9.65	$[4.39, 15.10]$	71.9
	Attn out.	32	6.82	$[3.87, 9.89]$	84.4
Dream	All	196	7.78	$[4.54, 11.90]$	73.0
	FFN out.	28	27.40	$[9.06, 51.80]$	75.0
	Attn out.	28	6.04	$[3.32, 9.03]$	82.1

Table 5: **Sensitivity to BTT block size under DuQuant W4A4.** HumanEval and MBPP pass@1 (%). The baseline corresponds to this block-size sweep. Bold denotes the highest score in each column.

	LLaDA		**Dream**	
Configuration	HumanEval	MBPP	HumanEval	MBPP
Rotation only (no BTT)	34.15	34.40	37.80	43.80
$K = 32$	38.23	35.60	39.63	48.00
$K = 64$	39.02	37.20	41.46	49.40
$K = 128$	**40.70**	**37.50**	**45.12**	**51.60**

A Appendix

.1 Generalization of Seed Selection

We evaluate whether seed selection improves local reconstruction on inputs beyond those used for selection. For each module, we select a seed from 16 candidates using WikiText2 samples 0–63 and evaluate it on disjoint samples 64–127. The reference is the mean held-out performance of the same 16 candidates.

Table 4 reports the additional output-NMSE reduction achieved by the selected seed relative to this reference. The mean improvement across all modules is 4.87×10^{-6} for LLaDA and 7.78×10^{-6} for Dream, with positive improvements in 73.7% and 73.0% of modules, respectively. All reported module-level bootstrap intervals lie above zero. These results indicate improved held-out local reconstruction on average within the calibration distribution; they do not establish the statistical significance of downstream accuracy gains.

.2 Sensitivity to Block Size

Table 5 compares BTT block sizes $K \in \{32, 64, 128\}$. Gains are measured against the baseline reported in this sweep. Among the tested sizes, $K = 128$ achieves the highest scores on both tasks for both models. LLaDA's scores increase across the three block sizes, whereas Dream exhibits a non-monotonic trend: $K = 64$ performs worse than $K = 32$ on both tasks.

These results favor $K = 128$ within this sweep, but do not establish that larger blocks always improve accuracy. The sweep measures downstream accuracy and does not determine the best accuracy–latency trade-off.

.3 Inference Efficiency

We measure the inference overhead of BTT on an NVIDIA H800 80GB GPU using the first 32 HumanEval prompts. Each run uses batch size 1, a generation length of 512, exactly 16 model forward passes per prompt, and no caching. We perform four warm-up prompts and five paired

Table 6: **Inference overhead of BTT under DuQuant W4A4.** Latency is reported as mean ± sample standard deviation over five repetitions. Throughput counts fixed generation positions rather than decoded tokens after EOS truncation.

Model	Method	Latency (s/prompt) ↓	Positions/s ↑
LLaDA-8B-Instruct	DuQuant W4A4	1.548 ± 0.024	330.8
	DuQuant W4A4 + BTT	1.617 ± 0.023	316.7
Dream-v0-Base-7B	DuQuant W4A4	0.950 ± 0.012	538.9
	DuQuant W4A4 + BTT	1.008 ± 0.011	507.8

repetitions, running both conditions sequentially on the same GPU with alternating execution order. Timing excludes model loading, calibration, warm-up, and scoring.

Both conditions use the same DuQuant W4A4 configuration. BTT is applied to FFN output projections with $K = 64$ using a CUDA FWHT implementation. The quantized backend uses INT8 GEMM to compute with W4A4 codes. Table 6 reports generation latency and throughput measured in generated positions per second. The mean paired latency overhead of BTT is 4.44% for LLaDA and 6.12% for Dream.

.4 Additional Activation Visualizations

Figures 4 and 5 present additional activation visualizations for LLaDA-8B-Instruct and Dream-v0-Base-7B, respectively, using examples from GSM8K, HumanEval, MATH, and MBPP. For each example, we compare activations before transformation, after the DuQuant channel transformation, and after combining DuQuant with BTT. Both FFN and attention output projections are included, with the corresponding layer indices indicated in the figures.

These examples illustrate how BTT redistributes activation magnitudes across the token dimension after channel transformation. The resulting patterns vary across models, modules, and inputs; a reduction in the maximum activation is not observed in every case. The plots use independent vertical scales and should be interpreted together with their numerical axis ranges. These visualizations provide qualitative examples rather than dataset-wide statistics. For Dream-v0-Base-7B, attention-side BTT is included for diagnostic purposes only.

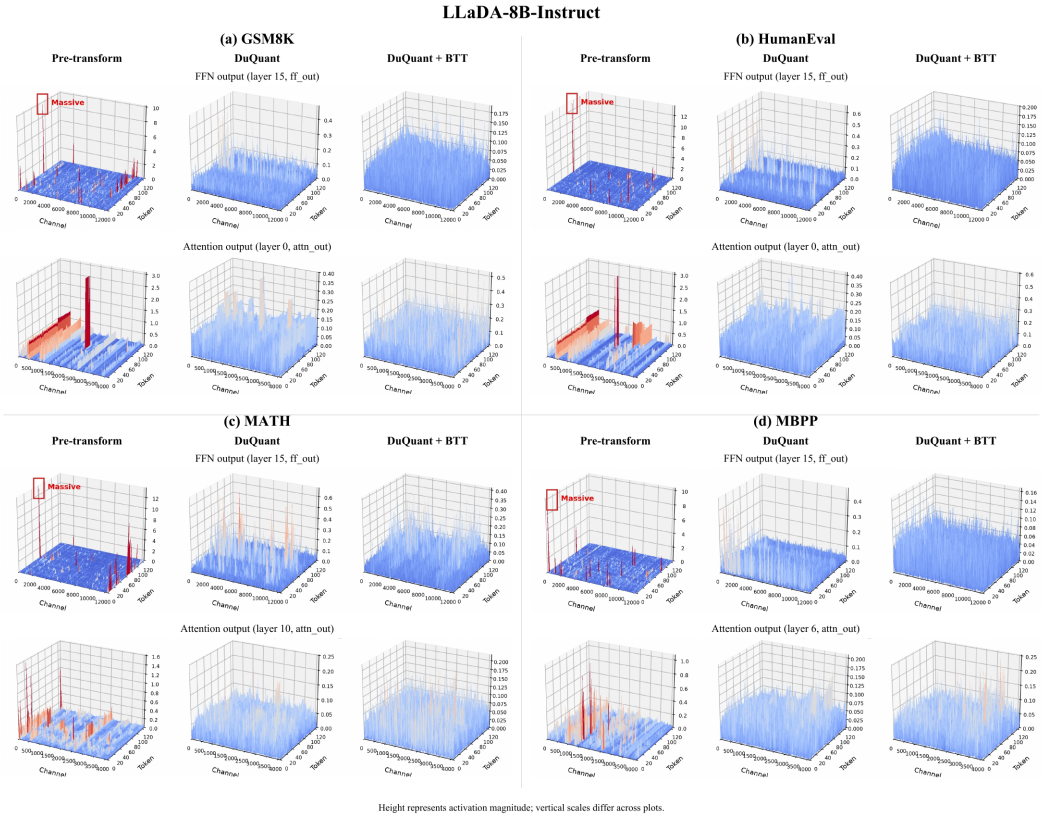


Figure 4: **Activation distributions of LLaDA-8B-Instruct.** Panels (a)–(d) show examples from GSM8K, HumanEval, MATH, and MBPP, respectively. Within each panel, the top and bottom rows show the FFN and attention output projections. Columns compare pre-transform activations, DuQuant, and DuQuant with BTT. Layer indices are indicated above each row. Heights represent activation magnitudes; vertical scales differ across plots.

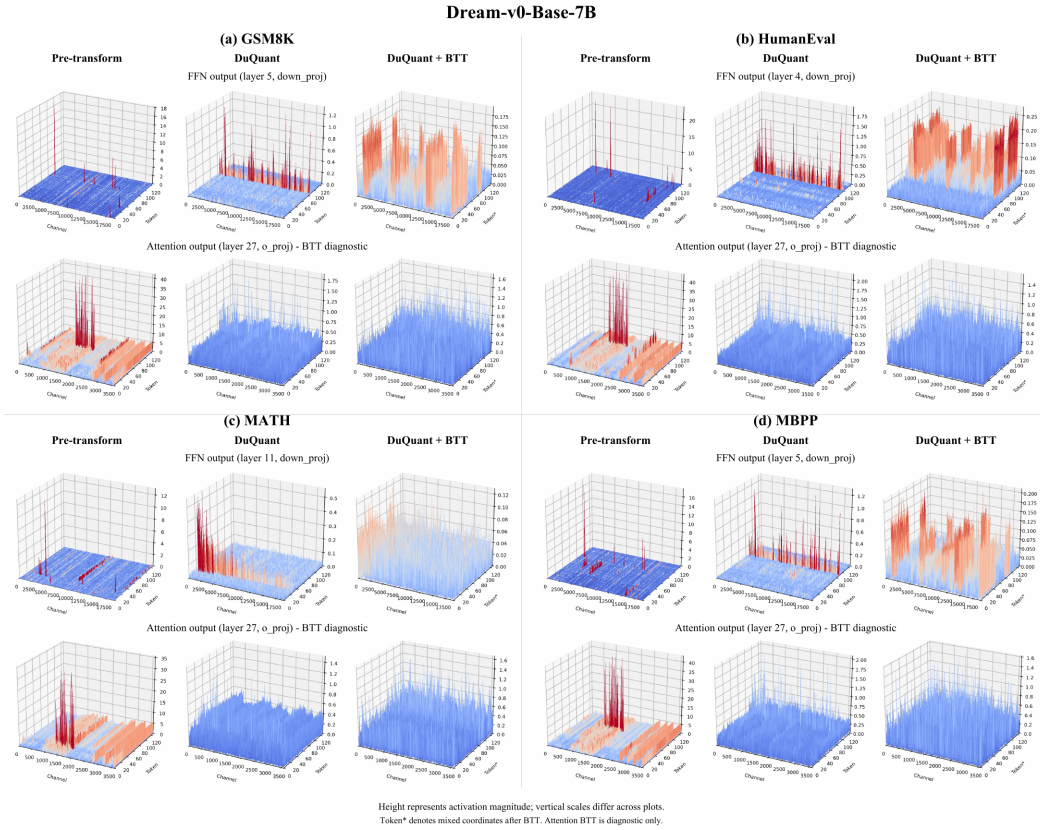


Figure 5: **Activation distributions of Dream-v0-Base-7B.** Panels (a)–(d) show examples from GSM8K, HumanEval, MATH, and MBPP, respectively. Within each panel, the top and bottom rows show the FFN and attention output projections. Columns compare pre-transform activations, DuQuant, and DuQuant with BTT. Attention-side BTT is shown for diagnostic purposes only. Token* denotes mixed coordinates after BTT. Vertical scales differ across plots, while colors are shared within each three-plot comparison.