STEERQUANT: STEERING QUANTIZATION ERROR WITH ACTION-GUIDED SCALING IN WORLD–ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

World–action models (WAMs) jointly generate future world states and actions through iterative denoising, using shared weights to process heterogeneous semantic streams of video, proprioceptive, and action tokens. Quantization reduces inference cost, but comparable numerical errors in different streams can have markedly different effects on final actions, making numerical accuracy alone insufficient for reliable control. We introduce **SteerQuant**, a 4-bit quantization framework for WAMs that steers errors toward computations with less influence on final actions. It maps how each stream's quantization errors affect final actions and uses this map to guide shared channel scaling. Activation scaling is further calibrated for each stream and denoising step to accommodate changes in activation ranges and action impact. This adapts quantization to different stream requirements without duplicating weights or increasing bit-widths for selected streams. To reduce the extra kernel launches and memory traffic introduced by scaling, we develop **Rudder**, a 4-bit inference engine for WAMs that fuses scaling and output compensation into low-bit kernels. Under W4A8 and W4A4, SteerQuant maintains mean LIBERO success within 0.8 percentage points of full precision, while delivering up to $2.23\times$ denoising speedup over BF16 across three WAMs with reduced peak GPU memory usage. On a real dual-arm robot, W4A8 deployment achieves a $1.35\times$ end-to-end inference speedup while maintaining average task success relative to BF16.

1 INTRODUCTION

World–action models (WAMs) build on pretrained video models to jointly model future world states and actions, using spatiotemporal priors learned from large-scale video data (Kim et al., 2026; NVIDIA, 2026; Ye et al., 2026; Yuan et al., 2026). During inference, diffusion Transformers (Peebles & Xie, 2023) repeatedly process heterogeneous semantic streams, including visual representations, proprioceptive states, and actions, across denoising steps. This repeated computation and data movement make inference costly.

Post-training quantization (PTQ) reduces these costs (Frantar et al., 2023; Li et al., 2023; Shang et al., 2023) by lowering weight and activation precision, reducing memory traffic and enabling hardware-accelerated low-precision arithmetic (e.g., INT4 and NVFP4). To preserve accuracy, existing PTQ methods typically control outlier magnitudes to reduce numerical error (Xiao et al., 2023; Ashkboos et al., 2024; Li et al., 2025; Liu et al., 2025). However, WAMs use shared weights to process heterogeneous semantic streams, so a single weight quantization choice affects multiple streams at once. The resulting errors propagate through stream interactions and subsequent denoising steps, potentially producing different final-action deviations even at similar magnitudes. Effective WAM quantization must therefore balance errors across streams according to their impact on final actions.

Achieving this balance requires identifying which streams' errors most affect final actions. We observe that comparable numerical errors across semantic streams can lead to markedly different final-action deviations and task success rates (see Sec. 3). A naive method is to calibrate weight–activation scaling independently for each stream. However, independently chosen scales generally require different quantized weight matrices, sacrificing weight sharing. Moreover, a stream whose

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.

errors strongly affect final actions at one denoising step may have less influence at another, limiting the effectiveness of fixed per-stream choices. The core challenge is therefore to adapt error trade-offs across streams as their action impact changes, while retaining one static quantized weight matrix per layer and fixed bit-widths.

To address this challenge, we propose **SteerQuant**, a 4-bit quantization framework for WAMs that steers errors toward computations with less influence on final actions. It coordinates shared channel scaling and stream-specific activation scaling across denoising steps while retaining shared weights and fixed 4-bit. To reduce the extra kernel launches and memory traffic introduced by separate scaling operations, we fuse these transformations into low-bit kernels.

Our contributions are summarized as follows:

- We reveal that heterogeneous semantic streams differ in their tolerance to quantization: comparable numerical errors can produce markedly different action deviations and task success rates. This tolerance varies across layers and denoising steps, motivating calibration guided by downstream action impact.
- We propose **SteerQuant**, which uses an Action-Impact Stream Map to guide shared channel scaling and stream-specific activation calibration across denoising steps. These two stages prioritize computations with greater action impact while retaining one static quantized weight matrix per layer and fixed bit-widths.
- We develop **Rudder**, a 4-bit inference engine for WAMs that integrates calibrated scaling and output compensation into fused kernels. By reducing extra kernel launches and intermediate memory traffic, Rudder enables efficient execution of SteerQuant while preserving the computational and memory benefits of quantization.
- We evaluate three WAMs under W4A8 and W4A4 on LIBERO and RoboLab, preserving near-BF16 LIBERO success and outperforming evaluated same-precision RoboLab baselines. We achieve up to 2.23× denoising speedup over BF16. Real dual-arm deployment delivers 1.35× end-to-end inference speedup while maintaining average task success.

2 Related Work

Post-Training Quantization. Existing PTQ methods broadly follow two directions. The first reduces numerical error through transformations (Lin et al., 2024; Shao et al., 2024; Sun et al., 2025), such as SmoothQuant's equivalent channel scaling (Xiao et al., 2023). The second incorporates action-related criteria into quantization (Xu et al., 2026). For VLA models, ActQuant combines action-guided bit allocation with weight-scale optimization (Akbari et al., 2026). For WAMs, QuantWAMs allocates weight precision using sensitivity to the joint video–action loss (Zhou et al., 2026). However, these criteria do not directly characterize how weight and activation quantization errors in coupled WAM streams propagate to final actions across denoising steps. We explore this impact to guide calibration at fixed 4-bit.

Efficient WAM Inference. Joint future-state and action prediction makes iterative WAM denoising costly. Existing acceleration methods reduce this cost by simplifying future prediction or reusing intermediate computations. For example, Fast-WAM retains video prediction during training but omits future video generation at inference (Yuan et al., 2026). DreamZero instead uses caching to reuse intermediate results and avoid repeated computation (Ye et al., 2026). These approaches reduce the amount of computation performed during inference. Our method addresses a complementary source of cost by lowering the precision of weights and activations in the retained operations. It therefore complements computation reduction and caching while preserving joint world–action denoising.

3 Preliminary and Motivation

3.1 WAM Quantization

WAMs jointly generate future world states and robot actions through iterative denoising. At each denoising step, a shared Transformer jointly processes heterogeneous semantic streams, where each stream consists of tokens encoding a particular type of information, such as video frames, robot

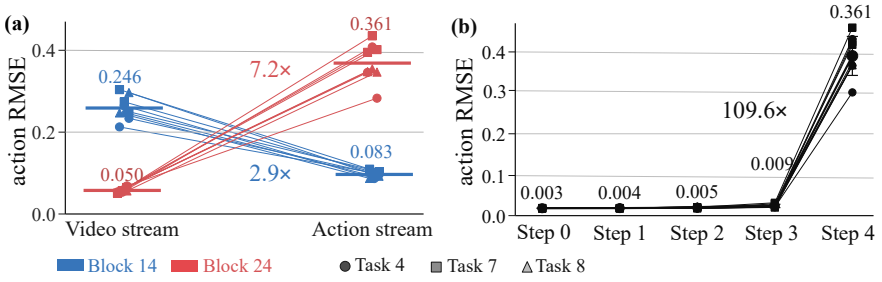


Figure 1: **Control effects of matched local errors.** **(a)** Video- and action-stream comparisons at Blocks 14 and 24 at the final denoising step, with local errors matched within each block. **(b)** Action-stream comparisons across Steps 0–4 at Block 24, with local errors matched across steps.

actions, value estimates, or proprioceptive states. Video streams provide visual context for action generation, while predicted actions drive the robot's interaction with the environment. Reducing the precision of linear operations in this shared Transformer can lower the cost of joint denoising, but also introduces errors into these coupled streams.

To characterize these errors, consider a linear operation at a fixed layer ℓ, denoising step τ, and semantic stream s, with input $\mathbf{X}$, weight $\mathbf{W}$, and full-precision output $\mathbf{Y}$. Using per-tensor quantization as an example, symmetric k-bit quantization maps $\mathbf{X}$ to its integer representation as

$$\mathbf{Q}_{\mathbf{X}} = \text{clip}\left(\text{round}\left(\frac{\mathbf{X}}{\Delta_{\mathbf{X}}}\right), -q_{\max}, q_{\max}\right), \qquad \Delta_{\mathbf{X}} = \frac{c_{\mathbf{X}}}{q_{\max}}, \tag{1}$$

where $c_{\mathbf{X}} > 0$ is the clipping threshold, $\Delta_{\mathbf{X}}$ is the quantization step size, and $q_{\max} = 2^{k-1} - 1$. Rounding and clipping are applied elementwise. The weight $\mathbf{W}$ is quantized in the same manner using its corresponding bit-width and step size, yielding

$$\mathbf{Y} = \mathbf{X}\mathbf{W} \approx \widehat{\mathbf{Y}} = \mathcal{Q}(\mathbf{X})\,\mathcal{Q}(\mathbf{W}) = \Delta_{\mathbf{X}}\Delta_{\mathbf{W}} \cdot \mathbf{Q}_{\mathbf{X}}\mathbf{Q}_{\mathbf{W}}, \tag{2}$$

where $\mathcal{Q}$ denotes quantization followed by dequantization. Thus, weight and activation quantization jointly determine the deviation of the layer output from the full-precision.

To compare these deviations across computational locations, we explicitly index the same outputs as $\mathbf{Y}_{\ell,\tau,s}$ and $\widehat{\mathbf{Y}}_{\ell,\tau,s}$, and define the normalized local reconstruction error as

$$\epsilon_{\ell,\tau,s} = \frac{\left\|\widehat{\mathbf{Y}}_{\ell,\tau,s} - \mathbf{Y}_{\ell,\tau,s}\right\|_F}{\|\mathbf{Y}_{\ell,\tau,s}\|_F}. \tag{3}$$

This quantity measures the relative magnitude of local errors. However, similar local errors may lead to different deviations in final actions as they propagate through the coupled computation.

3.2 Observations

In this part, We examine how well local reconstruction error reflects the effects of quantization on final actions and task success. We inject rescaled W4A8 quantization residuals into selected layer outputs of full-precision Cosmos-Policy (Kim et al., 2026). Within each comparison, we match the normalized local reconstruction error in Eq. 3, while keeping all remaining computation in full precision. We measure standardized action RMSE relative to full-precision predictions and closed-loop task success on three LIBERO tasks, each with three initial states, yielding nine paired rollouts per condition. Appendix A provides the detailed protocol and complete results.

The more influential stream varies across blocks. As shown in Fig. 1(a), matched local errors in the video and action streams have different effects on final actions. At the final denoising step, with local errors matched within each block, video-stream errors at Block 14 produce $2.9\times$ the action RMSE of action-stream errors. At Block 24, this ordering reverses: action-stream errors produce $7.2\times$ the RMSE of video-stream errors. The difference also appears in closed-loop execution: at

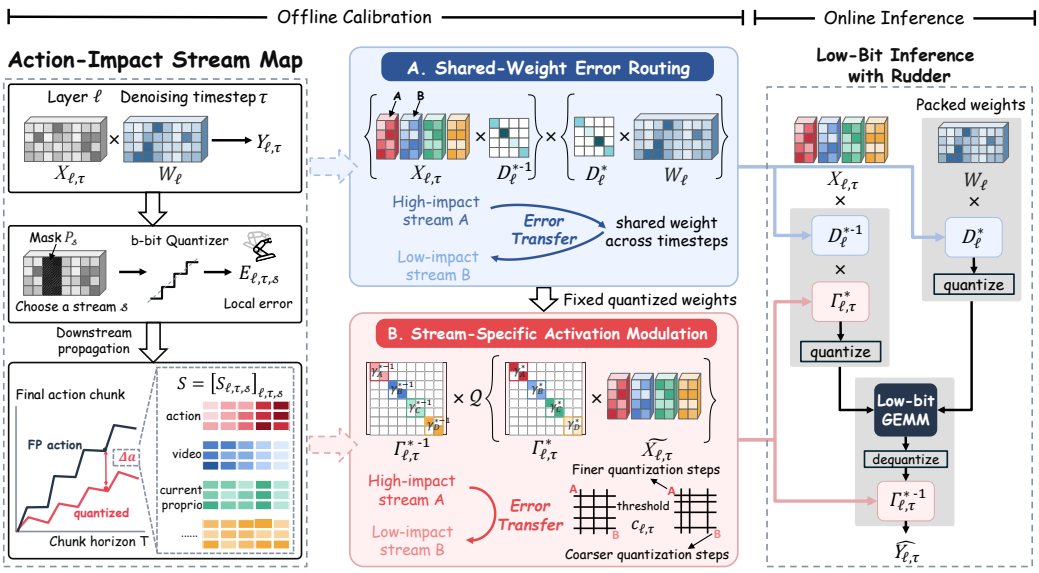


Figure 2: **Overview of SteerQuant.** $\Gamma_{\ell,\tau}$ is a diagonal matrix of token-wise stream gains. The upper-right gray box depicts offline weight preparation; the other gray boxes denote fused inference kernels.

Block 24, video-stream errors yield $9/9$ successful rollouts, compared with $4/9$ for action-stream errors. This observation shows that protection should be guided by how strongly errors in each semantic stream affect final actions.

Control impact changes across denoising steps. As shown in Fig. 1(b), control impact also varies when we hold the stream and block fixed and inject matched local errors into the action stream at Block 24 across Steps 0–4. Step 4 produces $109.6\times$ the action RMSE of Step 0. Task success is $4/9$ at Step 4, compared with $7/9$ or $8/9$ at earlier steps. This observation shows that a stream's action impact depends on when the error occurs, motivating its characterization across denoising steps as well as layers and streams.

These observations call for selective error reduction across semantic streams and denoising steps based on their impact on final actions. However, each layer shares its weights across these streams and steps, so quantization adjustments targeting one stream at one step may also affect others. The challenge is therefore to meet these differing protection needs under shared weights and fixed 4-bit.

4 Methodology

As illustrated in Figure 2, SteerQuant first constructs an *Action-Impact Stream Map* that estimates how local quantization errors affect final actions. Guided by this map, *Shared-Weight Error Routing* calibrates shared channel scaling, followed by *Stream-Specific Activation Modulation*, which refines activation quantization across streams and denoising steps while keeping the quantized weights fixed. The inference engine *Rudder* fuses scaling and output compensation into quantization and GEMM kernels for efficient execution.

4.1 Action-Impact Stream Map

To identify where quantization errors most affect final actions, we score each layer–step–stream region by its impact on the action output. Before calibrating the scaling factors, we compute the local quantization error at the target bit-widths:

$$\mathbf{E}_{\ell,\tau,s} = \widehat{\mathbf{Y}}_{\ell,\tau,s} - \mathbf{Y}_{\ell,\tau,s}. \tag{4}$$

Here, $\widehat{\mathbf{Y}}_{\ell,\tau,s}$ is the output for stream s after quantizing the weights and input activations of linear layer ℓ at step τ. The input is taken from a full-precision forward pass to exclude upstream quantization errors.

Injecting each stream's error and rerunning the remaining full-precision computation requires $\mathcal{O}(|\mathcal{D}|R)$ downstream forward evaluations for R regions and calibration set $\mathcal{D}$. Instead, we use derivatives of the full-precision model to estimate the resulting action changes. Let $\mathbf{a} \in \mathbb{R}^m$ denote the vectorized final action chunk, standardized using fixed, stabilized per-dimension standard deviations. Its Jacobian $\mathbf{J}_{\ell,\tau,s}$ with respect to $\mathrm{vec}(\mathbf{Y}_{\ell,\tau,s})$ gives the first-order action change $\mathbf{J}_{\ell,\tau,s}\,\mathrm{vec}(\mathbf{E}_{\ell,\tau,s})$. We define the action-impact score as

$$S_{\ell,\tau,s} = \left[\frac{1}{m}\mathbb{E}_{\mathcal{D}}\left[\|\mathbf{J}_{\ell,\tau,s}\,\mathrm{vec}(\mathbf{E}_{\ell,\tau,s})\|_2^2\right]\right]^{1/2}. \tag{5}$$

Here, $\mathbb{E}_{\mathcal{D}}$ averages over calibration samples. The score measures the RMS first-order action displacement; larger values indicate greater estimated impact on final actions.

Forming these Jacobians requires m coordinate vector–Jacobian products (VJPs) per sample and stores m entries for every local output element. We instead estimate the squared scores using P Rademacher projections (Hutchinson, 1989). Each projected reverse pass provides VJPs at all captured linear outputs, reusable across their stream error matrices. This yields scores for all R regions with $\mathcal{O}(|\mathcal{D}|P)$ reverse passes, without storing dense Jacobians. The map remains fixed during subsequent calibration. Appendix B.2 derives the unbiased squared-score estimator and analyzes its cost and approximation error; Appendices B.3 and B.4 examine the map's structure and stability.

4.2 Shared-Weight Error Routing

Within each layer, semantic streams and denoising steps share the same weights but differ in how their errors affect final actions. A scaling that reduces reconstruction error in one region may increase it in another. We therefore use the Action-Impact Stream Map to choose a shared channel scaling that prioritizes regions with greater action impact.

For each linear layer, a positive diagonal matrix $\mathbf{D}_\ell$ scales the weights and inversely scales the input activations:

$$\widehat{\mathbf{Y}}_{\ell,\tau,s}(\mathbf{D}_\ell) = \mathcal{Q}\big(\mathbf{X}_{\ell,\tau,s}\mathbf{D}_\ell^{-1}\big)\,\mathcal{Q}(\mathbf{D}_\ell\mathbf{W}_\ell)\,. \tag{6}$$

The two scalings cancel in full precision but change the ranges of weights and activations seen by the quantizer, allowing us to reshape quantization error at fixed bit-widths. Using the same $\mathbf{D}_\ell$ across streams and denoising steps retains one quantized weight matrix per layer.

We select this shared scaling by minimizing reconstruction error weighted by the map-derived importance $\omega^D_{\ell,\tau,s}$ (Appendix C):

$$\mathbf{D}_\ell^* = \arg\min_{\mathbf{D}_\ell} \sum_{\tau,s} \frac{\pi_\tau \omega^D_{\ell,\tau,s}}{n_s}\mathbb{E}_{\mathcal{D}}\left[\left\|\widehat{\mathbf{Y}}_{\ell,\tau,s}(\mathbf{D}_\ell) - \mathbf{Y}_{\ell,\tau,s}\right\|_F^2\right]. \tag{7}$$

Here, π_τ is the normalized calibration frequency of step τ, and n_s is the number of tokens in stream s, so that $1/n_s$ averages the loss over stream tokens. The map weights penalize errors more strongly in high-impact regions, guiding the trade-off when regions favor different scalings.

After calibration, we fold $\mathbf{D}_\ell^*$ into the weights, yielding $\widehat{\mathbf{W}}_\ell = \mathcal{Q}(\mathbf{D}_\ell^*\mathbf{W}_\ell)$. These weights are stored as packed integers with corresponding quantization scales and remain fixed during subsequent activation calibration and inference.

4.3 Stream-Specific Activation Modulation

As discussed in Section 3.2, the relative action impact of semantic streams changes across denoising steps. A shared channel scaling provides one compromise across all steps, but cannot follow these changing priorities. We therefore adjust activation quantization while keeping the weights fixed.

At each layer and step, a shared clipping threshold establishes a common reference range and base quantization scale. Stream gains adjust each stream's effective range relative to this reference, allowing different streams to receive greater protection at different steps. We bound both the threshold and gains to limit extreme range choices that can cause excessive clipping or coarse quantization.

For the transformed activation $\widetilde{\mathbf{X}}_{\ell,\tau,s} = \mathbf{X}_{\ell,\tau,s}(\mathbf{D}_\ell^*)^{-1}$, the modulated output is

$$\widehat{\mathbf{Y}}_{\ell,\tau,s}(\gamma_{\ell,\tau,s}, c_{\ell,\tau}) = \gamma_{\ell,\tau,s}^{-1}\,\mathcal{Q}\Big(\gamma_{\ell,\tau,s}\widetilde{\mathbf{X}}_{\ell,\tau,s}; c_{\ell,\tau}\Big)\,\widehat{\mathbf{W}}_\ell. \tag{8}$$

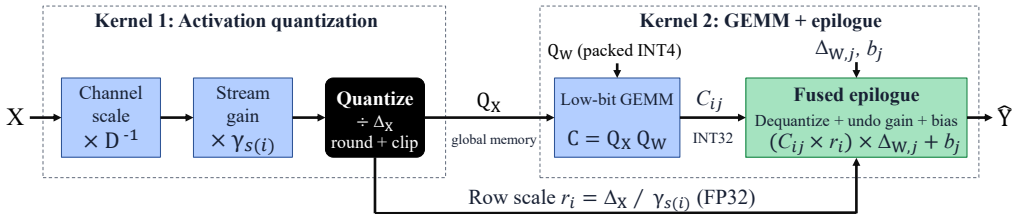


Figure 3: **Rudder's fused execution.** The first kernel combines channel scaling, stream gains, and activation quantization. The second performs low-bit GEMM and applies dequantization, inverse-gain compensation, and bias before writing the output. Only integer activations and row scales pass between kernels. Dashed boxes indicate kernel boundaries.

The inverse gain compensates for the input scaling at the output. Using map-derived weights $\omega^{\gamma}_{\ell,\tau,s}$, we jointly calibrate the gain vector and shared threshold:

$$
\begin{aligned}
(\boldsymbol{\gamma}^{*}_{\ell,\tau}, c^{*}_{\ell,\tau}) = \arg\min_{\boldsymbol{\gamma},\, c} \sum_{s} \frac{\omega^{\gamma}_{\ell,\tau,s}}{n_s} \mathbb{E}_{\mathcal{D}} \left[\left\| \widehat{\mathbf{Y}}_{\ell,\tau,s}(\gamma_s, c) - \widetilde{\mathbf{X}}_{\ell,\tau,s} \widehat{\mathbf{W}}_{\ell} \right\|_F^2 \right] \\
\text{s.t.} \quad c_{\min} \le c \le c_{\max}, \qquad \gamma_{\min} \le \gamma_s \le \gamma_{\max} \quad \forall s, \\
\sum_{s} n_s \log \gamma_s = 0.
\end{aligned}
\tag{9}
$$

All streams share the same positive bounds. When these constraints require a trade-off across streams, the map weights favor lower reconstruction errors in those with greater impact on final actions. Appendix C details the weight construction and effective quantization ranges. The calibrated gains and thresholds are stored in a layer–step lookup table, preserving uniform bit-widths without online optimization.

4.4 Rudder

Executing channel scaling, stream gains, and output compensation separately adds kernel launches and intermediate memory traffic at every denoising step. Rudder, our 4-bit inference engine for WAMs, reduces this overhead through two fused kernels while preserving stream-specific quantization (Figure 3). Below, layer and step indices are omitted, and all scaling parameters denote their calibrated values.

The first kernel applies channel scaling $\mathbf{D}^{-1}$ and stream gains during activation quantization. It writes only integer activations $\mathbf{Q_X}$ and row scales $r_i = \Delta_{\mathbf{X}}/\gamma_{s(i)}$, where $s(i)$ identifies the stream of row i and $\Delta_{\mathbf{X}} = c/q_{\max}$. The row scale incorporates inverse-gain compensation, eliminating a separate compensation operation.

The second kernel reuses the packed INT4 weights $\mathbf{Q_W}$ across streams and steps. Using CUTLASS (NVIDIA), it accumulates $\mathbf{C} = \mathbf{Q_X Q_W}$ in INT32 and produces the output in its epilogue:

$$
\widehat{Y}_{ij} = \mathrm{cast}_{\mathrm{FP16/BF16}}\left[\left(\mathrm{FP32}(C_{ij}) r_i\right) \Delta_{\mathbf{W},j} + b_j \right]. \tag{10}
$$

Here, $\Delta_{\mathbf{W},j}$ is the weight scale for output channel j, and b_j is the optional bias. This fusion avoids writing scaled floating-point activations and INT32 accumulation results to global memory, retaining only $\mathbf{Q_X}$ and $\mathbf{r}$ between kernels.

5 Experiments

5.1 Setup

Models and benchmarks. We evaluate Cosmos-Policy (Kim et al., 2026) and FastWAMJoint (Yuan et al., 2026) on LIBERO (Liu et al., 2023), and Cosmos-Edge (NVIDIA, 2026) on RoboLab (Yang et al., 2026).

Table 1: **Closed-loop task success rates (%) on LIBERO and RoboLab.** For LIBERO, Avg. denotes the mean across the four suites.

		LIBERO										RoboLab
		Cosmos-Policy					FastWAMJoint					Cosmos-Edge
Precision	Method	Spatial ↑	Object ↑	Goal ↑	Long ↑	Avg. ↑	Spatial ↑	Object ↑	Goal ↑	Long ↑	Avg. ↑	SR ↑
FP	Original	98.3	100.0	97.9	98.0	98.5	99.2	99.4	98.6	98.4	98.9	21.83
W4A8	SmoothQuant	98.0	99.7	98.4	95.6	97.9	96.8	98.4	97.2	96.2	97.2	13.67
	PTQ4DiT	98.8	99.6	96.1	97.7	98.1	97.8	98.6	98.0	96.0	97.6	13.67
	Q-DiT	97.5	99.9	96.8	96.7	97.7	97.4	98.4	97.8	96.6	97.6	12.17
	SteerQuant	98.3	99.9	98.1	98.0	**98.6**	97.8	99.4	98.2	96.8	**98.1**	**14.67**
W4A4	QuaRot	98.2	99.2	96.1	98.1	97.9	99.0	99.6	98.2	97.8	**98.7**	12.25
	Atom	96.2	98.7	96.6	97.3	97.2	97.8	97.4	98.8	98.6	98.2	13.50
	SVDQuant	98.4	99.7	96.7	97.7	98.1	98.2	98.4	98.6	97.6	98.2	16.00
	SteerQuant	98.3	99.7	97.7	97.5	**98.3**	98.8	99.8	98.4	96.6	98.4	**17.33**

Baselines and quantization settings. We evaluate W4A8 and W4A4 quantization, where W and A denote the bit-widths of weights and activations, respectively. For W4A8, we compare against SmoothQuant (Xiao et al., 2023), PTQ4DiT (Wu et al., 2024), and Q-DiT (Chen et al., 2025). For W4A4, we compare against QuaRot (Ashkboos et al., 2024), Atom (Zhao et al., 2024), and SVDQuant (Li et al., 2025). The original full-precision (FP) model serves as a reference. Within each model, all methods use the same checkpoint and inference settings.

Evaluation metrics. We assess control performance using closed-loop success rate (SR), supplemented by standardized action RMSE relative to FP in ablation studies. We assess efficiency using denoising latency and speedup relative to FP, peak allocated GPU memory, and packed weight storage. The shared quantization scope and additional details are provided in Appendix D.

5.2 Task Success

Table 1 compares task success at FP, W4A8, and W4A4. We compare each quantized model with its own FP reference and with baselines at the same nominal precision.

Results on LIBERO. As shown in Table 1, SteerQuant largely preserves the FP performance of both models. Under W4A8, its average SRs reach 98.6% on Cosmos-Policy and 98.1% on FastWAMJoint, exceeding the strongest baseline by 0.5 percentage points (pp) on each model. Under W4A4, the average SR of 98.3% on Cosmos-Policy surpasses the strongest baseline by 0.2 pp, while the 98.4% result on FastWAMJoint falls 0.3 pp below QuaRot. Across both models and precision settings, the average SR decreases by at most 0.8 pp relative to FP, supporting the effectiveness of SteerQuant in maintaining closed-loop task performance under low-bit quantization.

Results on RoboLab. On Cosmos-Edge, SteerQuant outperforms all evaluated quantization baselines at both precision settings (Table 1). Under W4A8, it reaches 14.67%, surpassing SmoothQuant and PTQ4DiT by 1.00 pp. Under W4A4, it attains 17.33%, outperforming the strongest baseline, SVDQuant, by 1.33 pp. These results show that SteerQuant retains more task performance than the evaluated same-precision baselines on RoboLab, although a gap to FP remains.

5.3 Inference Efficiency and Memory Footprint

Table 2 reports denoising latency, speedup relative to FP, peak GPU memory, and weight storage. All measurements use an NVIDIA RTX 4090 GPU, and all quantized configurations use low-bit kernels. Measurement details are provided in Appendix D.

Denoising latency and speedup. SteerQuant achieves the lowest denoising latency among the evaluated methods at both precision settings on all three models. Under W4A4, it delivers speedups of $2.229\times$, $1.859\times$, and $1.690\times$ over FP on Cosmos-Policy, FastWAMJoint, and Cosmos-Edge, respectively. Compared with QuaRot, the fastest W4A4 baseline on each model, it further reduces latency by 4.3%, 12.5%, and 8.5%, respectively. Under W4A8, it also achieves lower latency than SmoothQuant on all three models.

Table 2: **Inference efficiency and memory footprint across models.** Lat.: denoising latency (ms); Spd.: speedup relative to FP; Peak: peak allocated GPU memory (GB); Storage: packed weight storage (GiB). PTQ4DiT and Q-DiT are omitted because their official implementations do not provide low-bit inference kernels for the evaluated settings.

		Cosmos-Policy				FastWAMJoint				Cosmos-Edge			
Precision	Method	Lat. ↓	Spd. ↑	Peak ↓	Storage ↓	Lat. ↓	Spd. ↑	Peak ↓	Storage ↓	Lat. ↓	Spd. ↑	Peak ↓	Storage ↓
FP	Original	261.460	1.000×	4.241	3.644	360.450	1.000×	12.116	11.230	881.637	1.000×	8.492	6.276
W4A8	SmoothQuant	134.967	1.937×	1.607	1.191	263.368	1.369×	3.363	2.983	570.893	1.544×	6.405	4.314
	SteerQuant	133.455	1.959×	1.642	1.191	260.977	1.381×	3.363	2.983	563.858	1.564×	6.397	4.312
W4A4	QuaRot	122.657	2.132×	1.604	1.189	221.647	1.626×	3.363	2.983	569.943	1.547×	6.396	4.312
	Atom	192.997	1.355×	1.679	1.258	309.021	1.166×	3.510	3.188	766.027	1.151×	6.463	4.361
	SVDQuant	131.093	1.994×	1.755	1.321	348.767	1.033×	3.756	3.371	581.278	1.517×	6.503	4.411
	SteerQuant	117.322	2.229×	1.645	1.194	193.916	1.859×	3.370	2.983	521.589	1.690×	6.398	4.314

Table 3: **Contribution of individual components and action guidance.** Standardized action RMSE is averaged over W4A8 and W4A4 on Cosmos-Policy, using 1,800 observations from 600 trajectories across 40 tasks. Paired, unadjusted 95% CIs are computed by resampling whole trajectories within each task–seed stratum and refer to $\Delta\mathrm{RMSE} = \mathrm{RMSE}_{\mathrm{SteerQuant}} - \mathrm{RMSE}_{\mathrm{row}}$. Negative values favor SteerQuant.

Variant	Routing	Modulation	Weighting	RMSE ↓	95% CI of ΔRMSE
Base	×	×	Uniform	0.1581	$[-0.0430, -0.0357]$
Routing only	✓	×	Map-derived	0.1224	$[-0.0044, -0.0026]$
Modulation only	×	✓	Map-derived	0.1529	$[-0.0368, -0.0312]$
Both, uniform	✓	✓	Uniform	0.1416	$[-0.0262, -0.0052]$
SteerQuant	✓	✓	Map-derived	**0.1189**	–

Memory and storage footprint. Peak allocated GPU memory includes resident weights, activations, and temporary buffers, whereas packed weight storage covers stored model weights only. Table 2 reports these quantities in GB and GiB, respectively. Across models and precision settings, SteerQuant reduces peak GPU memory by 24.7–72.2% and packed weight storage by 31.3–73.4% relative to FP.

5.4 Ablation Studies

Contribution of individual components and action guidance. Table 3 evaluates Shared-Weight Error Routing, Stream-Specific Activation Modulation, and weighting derived from the Action-Impact Stream Map. Shared-Weight Error Routing alone achieves lower action RMSE than Stream-Specific Activation Modulation alone, and combining both components further improves action fidelity. The full method reduces RMSE by 24.79% relative to the base quantizer. Replacing map-derived weighting with uniform weighting increases RMSE, supporting the contribution of guidance from the Action-Impact Stream Map.

Quantization error analysis. Figure 4 (left) compares local output errors under uniform weighting and weighting derived from the Action-Impact Stream Map. Within each layer–step unit, we rank streams by their reference action-impact scores estimated on the calibration set. The top and bottom three streams form the high- and low-impact groups, respectively, with group assignments fixed across variants. In 935 of the 1,120 units (83.48%), map-derived weighting reduces error in the high-impact group while increasing it in the low-impact group. This pattern supports preferential error reduction in streams with greater estimated impact on final actions, while allowing larger errors in lower-impact streams.

5.5 Performance on Real-world Tasks

We evaluate FastWAMJoint fine-tuned on three real-world tasks using the AgileX Cobot Magic dual-arm platform and an NVIDIA RTX 4090 GPU (24 GB). Both methods use the same checkpoint and initial conditions, with 12 trials per task. Figure 4 (right) shows representative executions. As shown

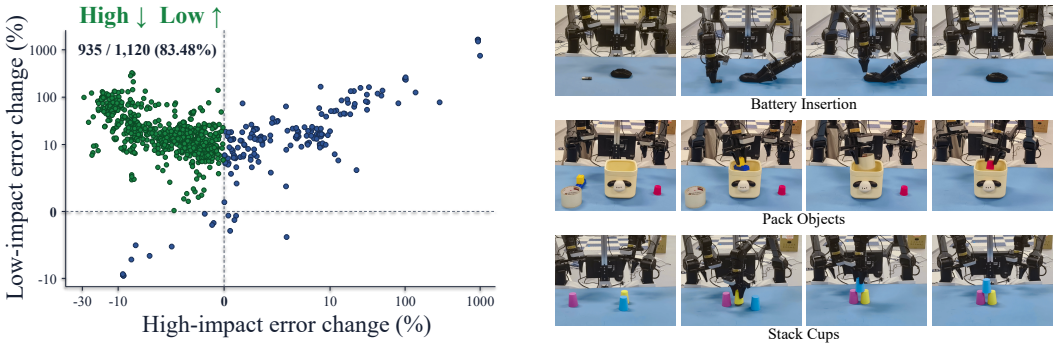


Figure 4: **Action-guided quantization error analysis and real-world execution. Left:** Cosmos-Policy W4A8 on 1,800 observations. Errors are token-normalized squared Frobenius differences from FP outputs under matched FP inputs, averaged over observations and streams within each group. Axes show percentage changes relative to uniform weighting on symmetric logarithmic scales. Green points indicate lower high-impact and higher low-impact error. **Right:** Representative SteerQuant executions on the AgileX Cobot Magic dual-arm platform. Rows show battery insertion, object packing, and cup stacking, with time progressing from left to right.

Table 4: **Real-world task performance and inference efficiency.** Task scores are reported in percent and averaged equally across tasks. Scoring details are provided in Appendix D. We report the median end-to-end inference latency and speedup, measured from sending a policy request to receiving the predicted actions. Robot motion execution time is excluded.

Method	Battery Insertion	Pack Objects	Stack Cups	Avg. score (%) ↑	Latency (ms) ↓	Speedup ↑
FP (BF16)	83.33	69.44	75.00	75.93	462.79	1.00×
SteerQuant (W4A8)	75.00	88.89	83.33	82.41	341.97	1.35×

in Table 4, W4A8 SteerQuant achieves 82.41% average task score versus 75.93% for BF16, with a 1.35× end-to-end inference speedup. Although battery insertion score decreases, the overall results show faster inference with higher average task success.

6 Limitations and Future Work

Future work can broaden the evaluation and deployment of SteerQuant in several directions. First, our efficiency evaluation is conducted on an NVIDIA RTX 4090. Extending Rudder to other GPU architectures and resource-constrained edge devices would broaden its deployment opportunities. Second, our evaluation covers three WAMs. Future work could explore additional model architectures and scales to further assess the applicability of action-guided quantization. Third, our real-world experiments focus on three manipulation tasks using a dual-arm robot. Expanding evaluation to more tasks and robot platforms would provide a broader picture of SteerQuant's practical benefits.

7 Conclusion

We presented SteerQuant, a post-training quantization framework that guides error allocation across heterogeneous semantic streams according to their influence on final actions. Its action-guided scaling coordinates shared weights and stream-specific activation ranges across denoising steps, preserving one static quantized weight matrix per layer at fixed bit-widths. The Rudder inference engine integrates these transformations into fused low-bit kernels. Across three WAMs, SteerQuant preserves near-BF16 success on LIBERO, outperforms the evaluated same-precision baselines on RoboLab, and achieves up to 2.23× denoising speedup with reduced GPU memory usage. Real-world W4A8 deployment delivers a 1.35× end-to-end inference speedup while maintaining average task success. These findings support downstream action impact as a useful guide for efficient quantization of coupled world–action computation.

AI USE STATEMENT

We used generative AI tools to polish the language and improve the clarity and readability of the manuscript. All AI-assisted revisions were reviewed and verified by the authors, who take full responsibility for the content, claims, and artifacts presented in this work.

ETHICS STATEMENT

This work studies post-training quantization of pretrained world–action models. We use LIBERO and DROID data for calibration and evaluate on LIBERO and RoboLab, in accordance with their respective terms of use. Real-world experiments involve three manipulation tasks on the AgileX Cobot Magic dual-arm platform. Quantization can alter policy behavior and task performance, and improved average success rates do not guarantee safe execution in individual cases. Our evaluations do not establish safety for deployment around humans or in safety-critical settings.

REPRODUCIBILITY STATEMENT

Sections 4 and 5 describe the proposed quantization method, the Rudder inference engine, and the experimental setup. Appendix A details the controlled intervention experiments; Appendices B and C provide the action-impact estimation procedure, supporting analysis, and calibration formulations; and Appendix D specifies the quantization scope, calibration data, and evaluation protocols. We plan to release the calibration and evaluation code, together with the Rudder inference kernels, upon publication.

REFERENCES

Arash Akbari, Arman Akbari, Masih Eskandar, Qitao Tan, Yixiao Chen, Jingwu Luo, Bertha Pangaribuan, Liyun Zhang, Jennifer Dy, Geng Yuan, Xue Lin, Gaowen Liu, Stratis Ioannidis, and Yanzhi Wang. ActQuant: Sub-4-bit action-guided quantization for vision-language-action models. *arXiv preprint arXiv:2605.24011*, 2026.

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. Curran Associates, Inc., 2024.

Lei Chen, Yuan Meng, Chen Tang, Xinzhu Ma, Jingyan Jiang, Xin Wang, Zhi Wang, and Wenwu Zhu. Q-DiT: Accurate post-training quantization for diffusion transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 28306–28315, 2025.

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.

Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. DROID: A large-scale in-the-wild robot manipulation dataset. In *Proceedings of Robotics: Science and Systems*, 2024.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163*, 2026.

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 component for 4-bit

diffusion models. In *International Conference on Learning Representations*, volume 2025, pp. 97831–97858, 2025.

Xiuyu Li, Yijiang Liu, Long Lian, Huanrui Yang, Zhen Dong, Daniel Kang, Shanghang Zhang, and Kurt Keutzer. Q-diffusion: Quantizing diffusion models. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 17489–17499, 2023.

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, pp. 87–100, 2024.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 44776–44791. Curran Associates, Inc., 2023.

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*, volume 2025, pp. 92009–92032, 2025.

NVIDIA. CUTLASS: Cuda templates for linear algebra subroutines. `https://github.com/NVIDIA/cutlass`. Accessed September 2026.

NVIDIA. Cosmos 3: Omnimodal world models for physical AI. *arXiv preprint arXiv:2606.02800*, 2026.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182, 2023.

Yuzhang Shang, Zhihang Yuan, Bin Xie, Bingzhe Wu, and Yan Yan. Post-training quantization on diffusion models. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1972–1981, 2023.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Gao Peng, Yu Qiao, and Ping Luo. Omniquant: Omnidirectionally calibrated quantization for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 45472–45496, 2024.

Yuxuan Sun, Ruikang Liu, Haoli Bai, Han Bao, Kang Zhao, Yuening Li, Jiaxin Hu, Xianzhi Yu, Lu Hou, Chun Yuan, Xin Jiang, Wulong Liu, and Jun Yao. FlatQuant: Flatness matters for LLM quantization. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 57587–57613. PMLR, 2025.

Junyi Wu, Haoxuan Wang, Yuzhang Shang, Mubarak Shah, and Yan Yan. PTQ4DiT: Post-training quantization for diffusion transformers. In *Advances in Neural Information Processing Systems*, volume 37, pp. 62732–62755. Curran Associates, Inc., 2024.

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.

Yuhao Xu, Yantai Yang, Zhenyang Fan, Yufan Liu, Yuming Li, Bing Li, and Zhipeng Zhang. QVLA: Not all channels are equal in vision-language-action model's quantization. In *International Conference on Learning Representations*, 2026.

Xuning Yang, Rishit Dagli, Alex Zook, Hugo Hadfield, Ankit Goyal, Stan Birchfield, Fabio Ramos, and Jonathan Tremblay. RoboLab: A high-fidelity simulation benchmark for analysis of task generalist policies. In *Proceedings of Robotics: Science and Systems*, 2026.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-WAM: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026.

Yilong Zhao, Chien-Yu Lin, Kan Zhu, Zihao Ye, Lequn Chen, Size Zheng, Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, and Baris Kasikci. Atom: Low-bit quantization for efficient and accurate LLM serving. In *Proceedings of Machine Learning and Systems*, volume 6, pp. 196–209, 2024.

Jiacheng Zhou, Jinfan Lv, Ruixuan Li, Longtai Zhang, Yan Wang, Wenqiang Zhang, and Lizhe Qi. QuantWAMs: Calibrating at the right granularity for world action models. *arXiv preprint arXiv:2607.28405*, 2026.

A Experimental Details for Preliminary Analysis

A.1 Experimental Setup

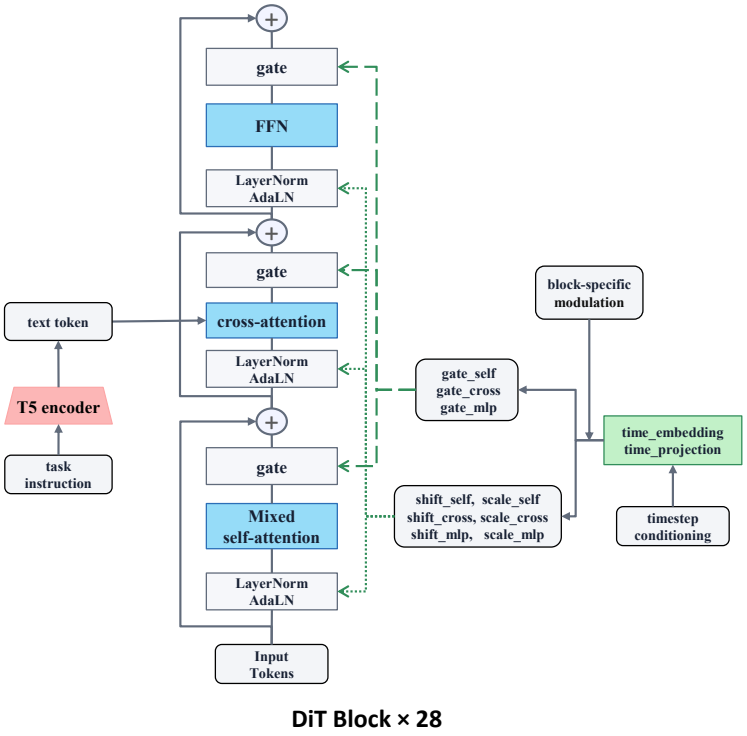


Figure 5: **DiT backbone of the evaluated Cosmos-Policy checkpoint.** The Cosmos-Policy checkpoint inherits the 28-block Cosmos-Predict2-2B Video2World diffusion Transformer without architectural modifications (Kim et al., 2026). Each block applies AdaLN-modulated mixed self-attention, T5-conditioned cross-attention, and a feedforward network. The three blue modules contain ten Linear layers; each residual intervention targets one Linear output.

We evaluated the full-precision Cosmos-Policy checkpoint on LIBERO-10 Tasks 4, 7, and 8. These tasks require placing two mugs on separate plates, placing two packaged objects in a basket, and placing two moka pots on a stove, respectively. We used three default initial states per task, yielding

nine paired rollouts per condition. All evaluations used policy seed 195, deterministic environment resets with seed 0, five denoising steps, and action chunks of shape 16×7.

Figure 5 summarizes the Transformer block used by the evaluated checkpoint. Its mixed self-attention, cross-attention, and feedforward modules contain ten Linear layers. Each intervention targets the output of a single Linear layer; we identify its location by the Transformer block and focus on block depth rather than comparisons among Linear families. The model contains 28 Transformer blocks, indexed from 1, whereas denoising steps are indexed from 0. The stream–depth experiment intervened at Step 4 on the video or action stream at Blocks 14 and 24. The step-wise experiment fixed Block 24 and the action stream while varying the intervention over Steps 0–4.

A.2 Matched Quantization-Residual Intervention

For each policy query, we first run an FP pass and record the input and output $\mathbf{Y}^{\mathrm{FP}}_{\ell,\tau}$ of the target Linear layer ℓ at denoising step τ. Using the same FP input, we quantize the layer's weights and input activations with W4A8 round-to-nearest quantization and compute the resulting output $\mathbf{Y}^{\mathrm{W4A8}}_{\ell,\tau}$. This produces the local quantization residual

$$\mathbf{r}_{\ell,\tau} = \mathbf{Y}^{\mathrm{W4A8}}_{\ell,\tau} - \mathbf{Y}^{\mathrm{FP}}_{\ell,\tau}. \tag{11}$$

Let P_s extract the output rows corresponding to stream s, so that $\mathbf{Y}^{\mathrm{FP}}_{\ell,\tau,s} = P_s(\mathbf{Y}^{\mathrm{FP}}_{\ell,\tau})$. At the injection site, we retain all unselected streams at their FP values and scale the selected residual independently for every policy query:

$$\begin{aligned} \widetilde{\mathbf{Y}}_{\ell,\tau,s} &= \mathbf{Y}^{\mathrm{FP}}_{\ell,\tau,s} + \alpha_{\ell,\tau,s} P_s(\mathbf{r}_{\ell,\tau}), \\ \alpha_{\ell,\tau,s} &= \epsilon_{\ell,\tau,s} \frac{\left\| \mathbf{Y}^{\mathrm{FP}}_{\ell,\tau,s} \right\|_F}{\left\| P_s(\mathbf{r}_{\ell,\tau}) \right\|_F}. \end{aligned} \tag{12}$$

The remaining downstream computation is performed in full precision. Here, $\epsilon_{\ell,\tau,s}$ is the target relative local error. Consequently,

$$\frac{\left\| \widetilde{\mathbf{Y}}_{\ell,\tau,s} - \mathbf{Y}^{\mathrm{FP}}_{\ell,\tau,s} \right\|_F}{\left\| \mathbf{Y}^{\mathrm{FP}}_{\ell,\tau,s} \right\|_F} \approx \epsilon_{\ell,\tau,s}.$$

We used $\epsilon_{\ell,\tau,s} = 0.25$ at Block 14 and $\epsilon_{\ell,\tau,s} = 0.50$ at Block 24. For the step-wise comparison, $\epsilon_{\ell,\tau,s} = 0.50$ was fixed across all five steps.

The FP and intervened action chunks were predicted from the same observation and sampler seed. For rollout r and policy query q, the standardized action RMSE is

$$e_{r,q} = \sqrt{\frac{1}{16 \times 7} \sum_{k=1}^{16} \sum_{d=1}^{7} \left(\frac{\widetilde{a}_{r,q,k,d} - a^{\mathrm{FP}}_{r,q,k,d}}{\sigma_d + 10^{-6}} \right)^2}, \tag{13}$$

where σ_d is the action standard deviation from the LIBERO training statistics. We first average $e_{r,q}$ over all policy queries in an episode, and then report the mean and sample standard deviation over the nine paired episodes. The intervened actions, rather than the auxiliary FP predictions, are executed in the closed-loop environment.

A.3 Additional Statistical Details

Table 5 provides the complete numerical results underlying Fig. 1, including the realized local errors, condition-level action RMSE summaries, and closed-loop success counts.

For the five repeated step conditions, a Friedman test detected a difference in action deviation ($\chi^2(4) = 30.22$, $P = 4.41 \times 10^{-6}$), indicating that matched local errors did not produce equal action deviations across denoising steps. The mean Step 4–Step 0 difference in action RMSE was 0.3578, with a task-aware paired-bootstrap 95% confidence interval of 0.3214–0.3987. This interval excludes zero, supporting a larger action deviation from intervention at Step 4 than at Step 0 under matched local errors.

Table 5: **Complete results of the controlled intervention experiments.** Action RMSE is reported as mean ± s.d. over nine task–initial-state rollouts. Success rate is reported as the number of successful rollouts out of nine. Block indices are one-based and step indices are zero-based.

Setting	Block	Step	Stream	Target error	Observed error	Action RMSE	Success Rate
FP	–	–	–	–	–	–	$7/9$
Stream × depth	14	4	Video	0.25	0.250007	0.245637 ± 0.029194	$5/9$
Stream × depth	14	4	Action	0.25	0.250016	0.083329 ± 0.007876	$9/9$
Stream × depth	24	4	Video	0.50	0.499968	0.050042 ± 0.005754	$9/9$
Stream × depth	24	4	Action	0.50	0.500003	0.361126 ± 0.045779	$4/9$
Step	24	0	Action	0.50	0.499986	0.003295 ± 0.000331	$7/9$
Step	24	1	Action	0.50	0.500002	0.003606 ± 0.000432	$8/9$
Step	24	2	Action	0.50	0.500001	0.004670 ± 0.001770	$7/9$
Step	24	3	Action	0.50	0.500004	0.008696 ± 0.003911	$8/9$
Step	24	4	Action	0.50	0.500003	0.361126 ± 0.045779	$4/9$

Table 6: Calibration setup for constructing the Cosmos-Policy Action-Impact Stream Map.

Property	Configuration
Calibration observations	320 LIBERO observations
Transformer blocks	28
Analyzed Linear families per block	10
Stream-aligned / text-context families	8 / 2
Temporal semantic streams	9
Denoising steps	5, indexed 0–4
Regions per observation	10,360
Weight quantization	Symmetric per-output-channel W4
Activation quantization	Symmetric static per-tensor A8
Action chunk	16×7
Rademacher projections	16
Denoising seed	195
Projection seed	20260809

B Action-Impact Stream Map: Construction and Analysis

B.1 Calibration Setup of Cosmos-Policy

We construct a separate Action-Impact Stream Map for each target weight–activation bit-width configuration. The construction details and the structure and stability analyses presented in this appendix focus on the W4A8 map for Cosmos-Policy, computed using 320 LIBERO calibration observations. Table 6 summarizes the setup.

Region definition. Each region is indexed by a Linear layer ℓ, a denoising step τ, and a semantic stream s. The index ℓ identifies both the Transformer block and the Linear family within that block. Each of the 28 blocks contains ten analyzed Linear layers: the query, key, value, and output projections of self-attention and cross-attention, and two MLP projections. Eight of these Linear families operate on nine temporal semantic streams: blank control, current proprioception, current wrist image, current primary image, action, future proprioception, future wrist image, future primary image, and value. For these families, the stream mask P_s selects the output rows corresponding to stream s. The remaining two families, the cross-attention key and value projections, operate on global text context. Each is treated as a single region with $P_s = I$. Across five denoising steps, this yields $28 \times 5 \times (8 \times 9 + 2) = 10{,}360$ regions per observation.

Map estimation. We estimate the action response to the joint weight–activation quantization error using 16 Rademacher projections and vector–Jacobian products along the full-precision trajectory. For each calibration observation, all map-estimation passes use the same conditioning and diffusion random draws.

B.2 Efficient Estimation of the Action-Impact Stream Map

Estimation target. For a fixed observation x and region (ℓ, τ, s), let

$$\mathbf{v} = \mathrm{vec}(\mathbf{E}_{\ell,\tau,s}(x)), \qquad \mathbf{J} = \mathbf{J}_{\ell,\tau,s}(x),$$

where $\mathbf{E}_{\ell,\tau,s}$ is the local joint weight–activation quantization error defined in Section 4.1, and $\mathbf{J}_{\ell,\tau,s}$ is the Jacobian of the standardized final action chunk with respect to $\mathrm{vec}(\mathbf{Y}_{\ell,\tau,s})$. The first-order action response is $\mathbf{Jv}$. We estimate its mean squared magnitude,

$$e_{x,\ell,\tau,s} = \frac{1}{m}\|\mathbf{Jv}\|_2^2, \tag{14}$$

where $m = Td_a$, T is the action-chunk horizon, and d_a is the action dimension.

Random-projection estimator. Let $\mathbf{z}_1, \ldots, \mathbf{z}_P \in \{-1, +1\}^m$ be independent random vectors with independent Rademacher entries. We estimate Eq. 14 by

$$\widehat{e}_{x,\ell,\tau,s} = \frac{1}{Pm}\sum_{p=1}^{P}\left\langle \mathbf{J}^\top \mathbf{z}_p, \mathbf{v}\right\rangle^2. \tag{15}$$

Each term can be computed using a vector–Jacobian product (VJP) because

$$\left\langle \mathbf{J}^\top \mathbf{z}_p, \mathbf{v}\right\rangle = \mathbf{z}_p^\top \mathbf{Jv}. \tag{16}$$

Since $\mathbb{E}[\mathbf{z}_p\mathbf{z}_p^\top] = \mathbf{I}$,

$$\begin{aligned}\mathbb{E}_{\mathbf{z}}\left[\widehat{e}_{x,\ell,\tau,s}\right] &= \frac{1}{m}\mathbf{v}^\top\mathbf{J}^\top\mathbb{E}\left[\mathbf{z}_p\mathbf{z}_p^\top\right]\mathbf{Jv} \\ &= \frac{1}{m}\|\mathbf{Jv}\|_2^2.\end{aligned} \tag{17}$$

Thus, the estimator is unbiased for the mean-squared first-order action response.

We aggregate these estimates over the calibration set $\mathcal{D}$, with $|\mathcal{D}| = N_{\mathrm{cal}}$, to obtain

$$\widehat{S}_{\ell,\tau,s} = \left[\frac{1}{N_{\mathrm{cal}}}\sum_{x\in\mathcal{D}}\widehat{e}_{x,\ell,\tau,s}\right]^{1/2}. \tag{18}$$

For a fixed calibration set, unbiasedness applies to $\widehat{S}^2_{\ell,\tau,s}$, not generally to $\widehat{S}_{\ell,\tau,s}$.

VJP reuse and computational cost. For each observation and projection, one reverse pass provides VJPs at all captured Linear outputs along the full-precision trajectory. At each output, we extract the VJP rows corresponding to stream s, vectorize them, and contract the result with $\mathrm{vec}(\mathbf{E}_{\ell,\tau,s})$ using Eq. 16. For the text-context regions, the full output VJP is used. Thus, the same reverse pass is reused across all regions.

Directly evaluating exact action displacements over R regions requires $\mathcal{O}(N_{\mathrm{cal}}R)$ downstream forward evaluations. In contrast, the proposed estimator uses $P = 16$ projections and requires $\mathcal{O}(N_{\mathrm{cal}}P)$ reverse passes. Error construction and contractions still scale with the number of regions, but require neither separate downstream replays nor an explicit Jacobian.

Local linearization error. The estimator above targets the first-order response. To characterize its difference from the exact displacement, write

$$\mathbf{y} = \mathrm{vec}(\mathbf{Y}_{\ell,\tau,s}(x)).$$

For this fixed observation and region, let $f(\mathbf{u})$ denote the standardized final action chunk obtained by replacing the selected stream's output with $\mathrm{unvec}(\mathbf{u})$, holding all unselected streams at their FP values at the injection site, and running the remaining computation in full precision. Conditioning and sampling randomness are held fixed. Then $\mathbf{J} = Df(\mathbf{y})$.

If the Jacobian Df is L_J-Lipschitz in the induced Euclidean operator norm along the segment from $\mathbf{y}$ to $\mathbf{y} + \mathbf{v}$, then

$$\|f(\mathbf{y}+\mathbf{v}) - f(\mathbf{y}) - \mathbf{Jv}\|_2 \le \frac{L_J}{2}\|\mathbf{v}\|_2^2. \tag{19}$$

Under this smoothness assumption, the linearization error is bounded by a term quadratic in the local quantization error magnitude.

Table 7: Agreement with the 320-observation reference map as calibration size increases. Entries report medians and 2.5th–97.5th percentile intervals over 200 subsets at each size. Observations are sampled without replacement within each subset.

Observations	Spearman correlation	Top-10% overlap
5	0.898 [0.859, 0.958]	0.957 [0.914, 0.965]
10	0.927 [0.864, 0.969]	0.966 [0.929, 0.973]
20	0.947 [0.874, 0.981]	0.973 [0.954, 0.979]
40	0.960 [0.911, 0.987]	0.980 [0.972, 0.984]
80	0.977 [0.938, 0.993]	0.985 [0.981, 0.988]
160	0.993 [0.970, 0.999]	0.989 [0.986, 0.992]

B.3 Structure of the Action-Impact Stream Map

We analyze the Cosmos-Policy action-impact stream map constructed from 320 LIBERO observations under the W4A8 reference configuration. For a group of regions $\mathcal{G}$, we aggregate the estimated squared first-order action responses as

$$\widehat{S}_{\mathcal{G}} = \left[\frac{1}{N_{\mathrm{cal}}|\mathcal{G}|} \sum_{x \in \mathcal{D}} \sum_{i \in \mathcal{G}} \widehat{e}_{x,i} \right]^{1/2}, \tag{20}$$

where i indexes a region (ℓ, τ, s). This quantity summarizes the action-impact scores of individual regions; it does not estimate the combined action displacement from simultaneously perturbing all regions in $\mathcal{G}$.

Marginal action impact. Figure 6 summarizes the aggregated action-impact scores across denoising steps, Transformer blocks, semantic streams, and Linear families. The final step has an aggregated score $47.2\times$ that of Step 3, while Blocks 19–28 have an aggregated score $6.0\times$ that of Blocks 1–9. The action stream has the largest aggregated score among the temporal semantic streams, and the two MLP projections have the largest Linear-family scores. These results show substantial differences in the estimated first-order action impact of local quantization errors across regions.

Temporal and depth-dependent semantic structure. Figure 7 shows that the stream with the largest aggregated action-impact score changes with depth and denoising step. After aggregation over denoising steps and the eight stream-aligned Linear families, this stream varies across earlier blocks but is consistently the action stream from Block 17 onward. After aggregation over blocks and the same eight Linear families, current proprioception has the largest score at Steps 0–2, future wrist image at Step 3, and action at Step 4.

Figure 8 complements these summaries by showing the joint depth–step distribution of action-impact scores for each of the nine temporal streams on shared scales. Together, these figures reveal variation in the estimated action impact of quantization errors across Linear layers, denoising steps, and semantic streams, motivating calibration that prioritizes regions with greater impact on final actions.

B.4 Stability of the Estimated Map

We assess the stability of regional action-impact rankings under changes in calibration-sample selection, with the reference quantization configuration and map-estimation settings held fixed.

Split-half agreement. We randomly divide the 320 observations into two disjoint subsets of 160 and estimate a map from each using Eq. 18. Across 200 partitions, the median Spearman correlation over all 10,360 regions is 0.967 (2.5th–97.5th percentiles: 0.953–0.974). The median top-10% overlap is 0.979 (2.5th–97.5th percentiles: 0.974–0.984), where overlap is the intersection size of the two top-ranked sets, each containing 1,036 regions, divided by 1,036. Median correlations computed separately within each denoising step range from 0.890 to 0.999. Thus, agreement remains high both across the full map and within individual denoising steps.

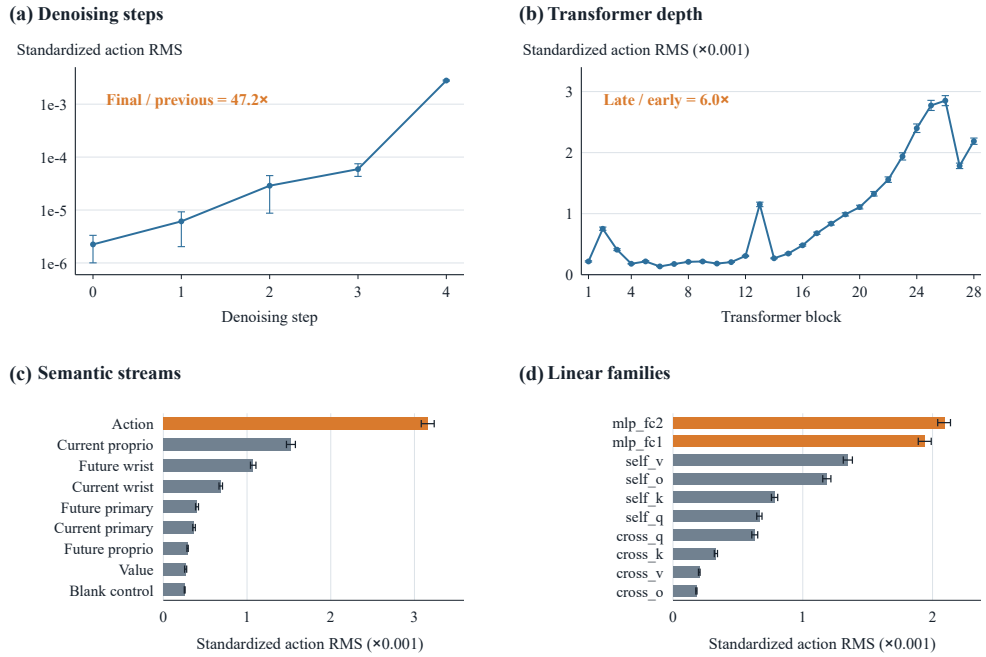


Figure 6: **Marginal action impact across four dimensions.** (a) Denoising steps. (b) Transformer blocks. (c) Temporal semantic streams. (d) Linear families. Values are aggregated action-impact scores computed over 320 observations and the regions in each group using Eq. 20. Panel (c) includes only the eight stream-aligned Linear families. Error bars show 95% observation-bootstrap intervals from 10,000 resamples. The final/previous ratio compares Steps 4 and 3; the late/early ratio compares Blocks 19–28 with Blocks 1–9.

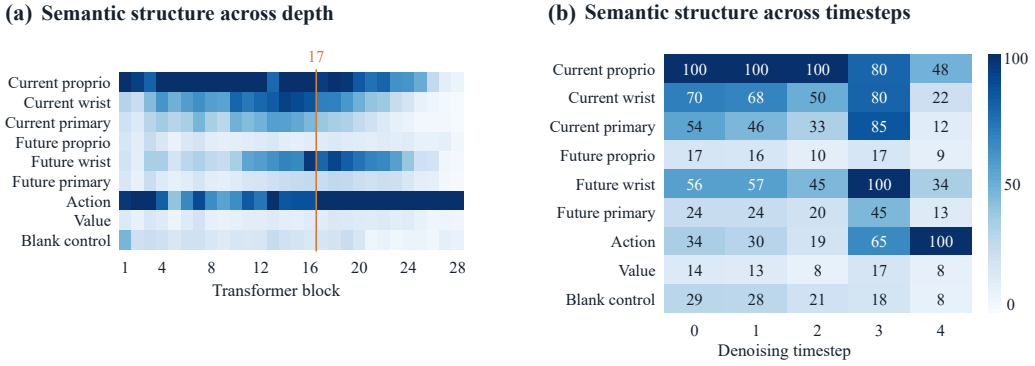


Figure 7: **Relative action impact of semantic streams.** (a) Stream scores within each Transformer block, aggregated over denoising steps and the eight stream-aligned Linear families. (b) Stream scores within each denoising step, aggregated over blocks and the eight stream-aligned Linear families. Both panels aggregate over 320 observations using Eq. 20. Each column is divided by its maximum and multiplied by 100, so colors indicate relative action-impact scores within a column. The marker at Block 17 identifies the first block from which the action stream consistently has the largest block-wise aggregated score.

Calibration-size convergence. For each sample size in Table 7, we draw 200 subsets, sampling observations without replacement within each subset, and compare their estimated maps with the 320-observation reference. We use the same Spearman correlation and top-10% overlap metrics as in the split-half analysis. Median agreement increases with sample size, and even small subsets recover most of the highest-impact regions in the reference map. Because the subsets are contained

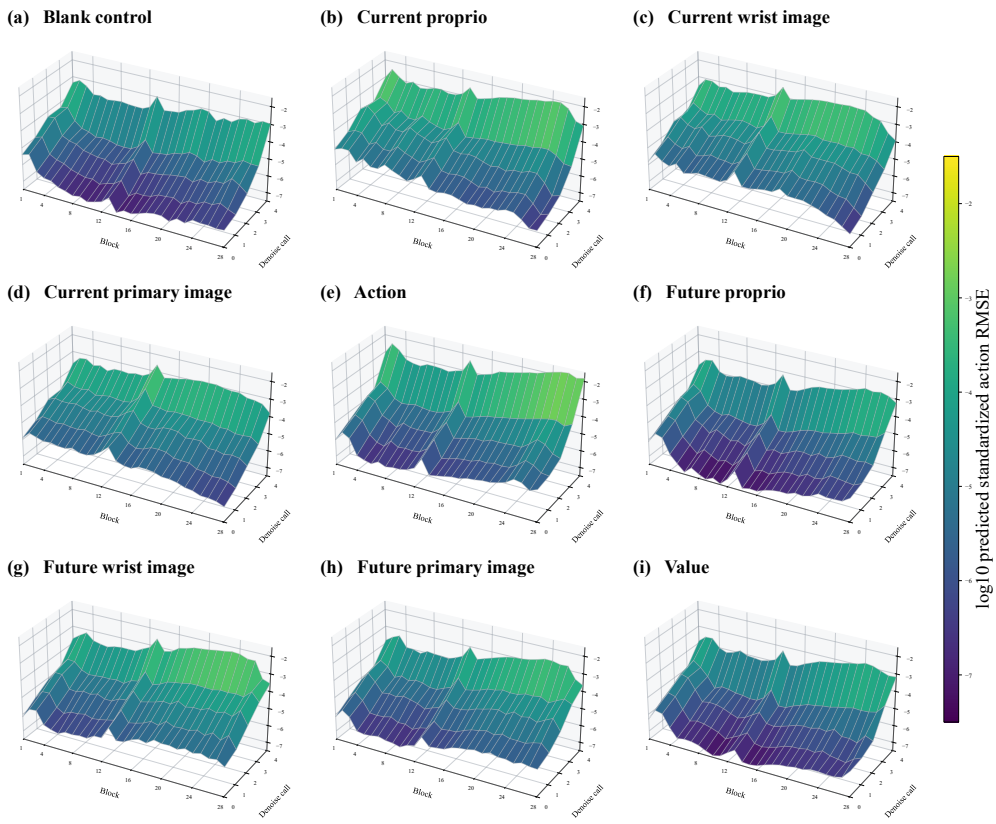


Figure 8: **Action-impact surfaces for nine temporal semantic streams.** Each panel shows the $\log_{10}$ of the aggregated action-impact score across 28 Transformer blocks and five denoising steps. At each block–step pair, scores are aggregated over the eight stream-aligned Linear families and 320 observations using Eq. 20. Surface height and color encode the same quantity. All panels share the same vertical and color scales.

in the reference pool, this analysis measures convergence to the full-pool estimate, complementing the disjoint split-half comparison.

Together, these results show stable regional action-impact rankings across calibration subsets, supporting the use of a fixed map to guide subsequent calibration.

C Calibration Weights and Activation Constraints

C.1 From Action-Impact Scores to Calibration Weights

Let $\widehat{S}_{\ell,\tau,s}$ denote the estimated action-impact score. We convert its square into a positive calibration weight before normalization:

$$u_{\ell,\tau,s} = \left(\widehat{S}^2_{\ell,\tau,s} + \eta\right)^{\rho}, \qquad \eta > 0, \quad 0 < \rho \leq 1. \tag{21}$$

The constant η ensures positive weights, while $\rho < 1$ reduces the dominance of large scores.

Since $\mathbf{D}_\ell$ is shared across denoising steps and streams, its weights are normalized jointly over both:

$$\omega^D_{\ell,\tau,s} = \frac{u_{\ell,\tau,s}}{\sum_{\tau',j} \pi_{\tau'} u_{\ell,\tau',j}}, \qquad \sum_{\tau,s} \pi_\tau \omega^D_{\ell,\tau,s} = 1. \tag{22}$$

Here, π_τ is the normalized calibration frequency of step τ, with $\sum_\tau \pi_\tau = 1$.

Activation calibration operates separately at each layer and step, so its weights are normalized over streams:

$$\omega^{\gamma}_{\ell,\tau,s} = \frac{u_{\ell,\tau,s}}{\sum_j u_{\ell,\tau,j}}, \qquad \sum_s \omega^{\gamma}_{\ell,\tau,s} = 1. \tag{23}$$

The estimated map and both sets of weights remain fixed throughout calibration.

C.2 Effective Activation Ranges and Constraints

For a fixed layer and denoising step, inverse-gain compensation gives stream s an effective clipping threshold c/γ_s and quantization step size $c/(q_{\max}\gamma_s)$. Here, $q_{\max}$ corresponds to the activation bit-width. Increasing γ_s therefore reduces rounding intervals while narrowing the unclipped range.

All streams share the same gain bounds, which restrict their effective thresholds to

$$\frac{c}{\gamma_s} \in \left[\frac{c}{\gamma_{\max}}, \frac{c}{\gamma_{\min}}\right]. \tag{24}$$

The normalization $\sum_s n_s \log \gamma_s = 0$ implies

$$\prod_s \left(\frac{c}{\gamma_s}\right)^{n_s/\sum_j n_j} = c. \tag{25}$$

Thus, c is the token-weighted geometric mean of the effective thresholds. Together with the bounds on c and γ_s, this restricts the combinations of quantization ranges available across streams. When no feasible configuration attains every stream's individually preferred threshold, the map-weighted objective guides the trade-off among them.

The reconstruction target $\widetilde{\mathbf{X}}_{\ell,\tau,s}\widehat{\mathbf{W}}_{\ell}$ uses unquantized activations and fixed quantized weights, isolating the additional error from activation quantization. After calibration, all streams use the shared base step size $c_{\ell,\tau}/q_{\max}$ before inverse-gain compensation, which produces their stream-specific effective step sizes.

D Additional Experimental Details

D.1 Quantization Settings

Quantization scope. We apply the same quantization scope to Cosmos-Policy, FastWAMJoint, and Cosmos-Edge. We quantize the query, key, value, and output projections of both self-attention and cross-attention, together with the FFN up- and down-projection layers. Their weights use 4-bit precision, while their input activations use 8-bit or 4-bit precision under W4A8 or W4A4, respectively. Normalization, RoPE, Softmax, GELU, the KV cache, and residual additions remain in BF16 (Figure 9).

Quantization format. For SteerQuant, weights are quantized symmetrically per output channel. Activations are quantized symmetrically using an offline-calibrated base scale shared across streams at each layer and denoising step. Stream gains and inverse-gain compensation produce stream-specific effective scales.

D.2 Calibration and Evaluation

Calibration data. We sample 320 observations from the LIBERO training set for Cosmos-Policy, 100 from the LIBERO training set for FastWAMJoint, and 512 from the DROID (Khazatsky et al., 2024) training set for Cosmos-Edge. Within each model, all methods use the same calibration set.

Evaluation data. Cosmos-Policy is evaluated over 3,000 rollouts, covering 40 tasks with 25 trials per task under each of three random seeds. FastWAMJoint and Cosmos-Edge are evaluated over 2,000 and 1,200 rollouts, respectively. The calibration and evaluation data are non-overlapping.

Evaluation protocol. All methods use matched initial states and random seeds. We report the mean closed-loop success rate across tasks. For ablations, standardized action RMSE is computed

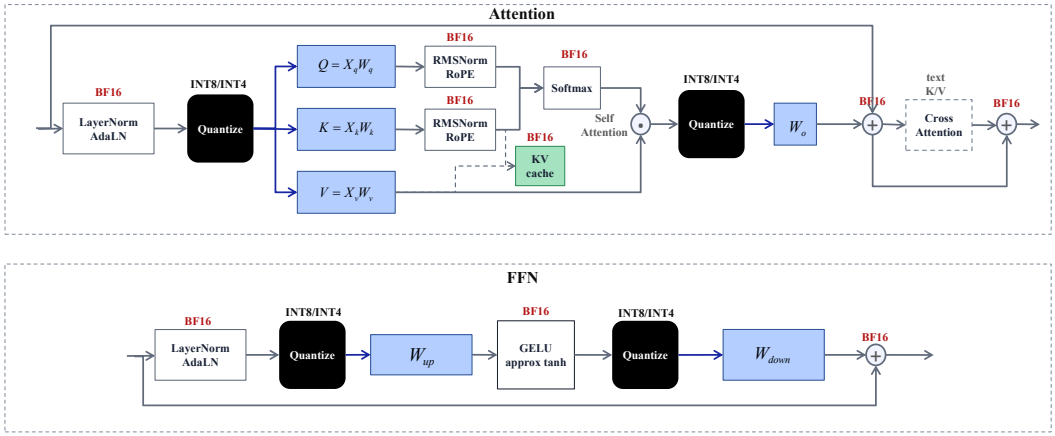


Figure 9: **Transformer linear-layer quantization scope.** The same quantization scope is used for Cosmos-Policy, FastWAMJoint, and Cosmos-Edge.

relative to full-precision predictions using fixed training-set statistics, with matched observations and sampling randomness.

Efficiency measurements. All three models follow the same general benchmarking procedure on an NVIDIA RTX 4090 GPU. We measure DiT replay latency using FP-captured inputs, CUDA Graphs, and CUDA events, with warm-up runs before timing. For Cosmos-Policy, we use five warm-up sequences and 20 timed sequences and report the median. Speedup is computed relative to each model's BF16 reference.

Real-world evaluation. We evaluate FastWAMJoint fine-tuned on the real-world tasks, using the same checkpoint for BF16 and W4A8 SteerQuant. Each method is evaluated over 12 trials per task under matched initial conditions. Pack Objects receives a score of $k/3$, where k is the number of successfully handled objects; the other tasks use binary success scores. We report each task's mean score over 12 trials as a percentage.