Not All Tasks Quantize Equally: Fisher-Guided Quantization for Visual Geometry Transformer

Yipu Zhang[1,†] **Jintao Cheng**[1,†] **Weilun Feng**[2,3,†] **Jiehao Luo**[4]
Chuanguang Yang[2] **Zhulin An**[2] **Yongjun Xu**[2,5] **Wei Zhang**[1,*]

[1]Department of Electronic and Computer Engineering, HKUST
[2]State Key Laboratory of AI Safety, Institute of Computing Technology, Chinese Academy of Sciences
[3]University of Chinese Academy of Sciences
[4]School of Electronic Science and Engineering (School of Microelectronics), South China Normal University
[5]Xiamen Institute of Data Intelligence
Code: `https://github.com/ypzhng/FGQ`

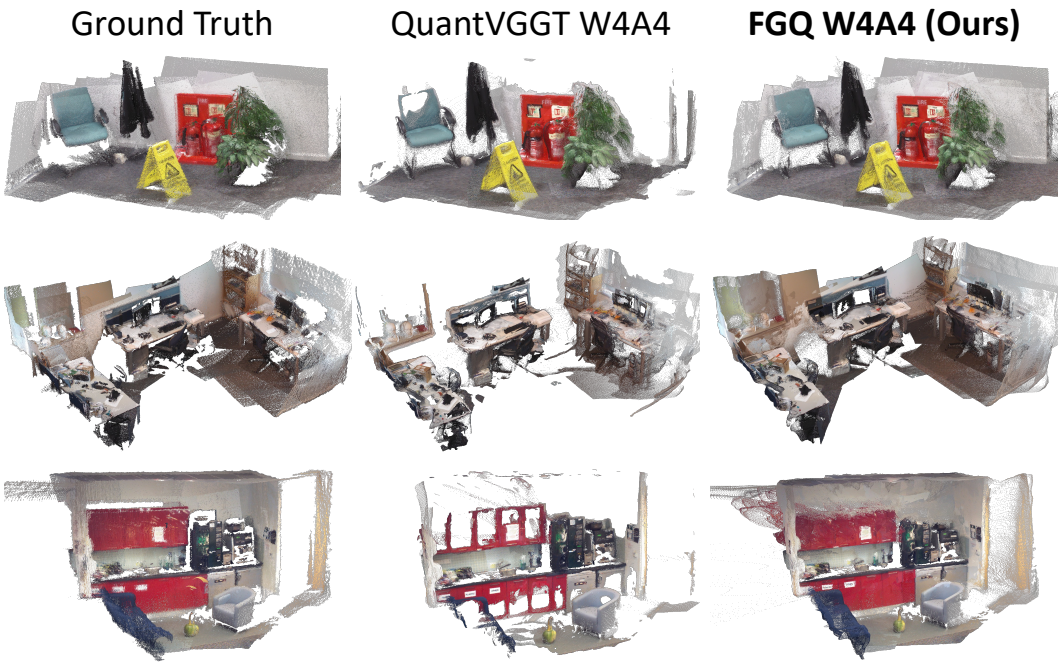


Figure 1: **FGQ** effectively preserves the point map reconstruction fidelity under 4-bit quantization, surpassing previous state-of-the-art VGGT (QuantVGGT [3]) quantization.

Abstract

Feed-forward 3D reconstruction models, represented by Visual Geometry Grounded Transformer (VGGT), jointly predict multiple visual geometry tasks

[†]Equal contribution.
[*]Corresponding author:Wei Zhang<wei.zhang@ust.hk>.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

such as depth estimation, camera pose prediction, and point map reconstruction in a single forward pass. They have been widely adopted in 3D vision applications, but their billion-scale parameters bring substantial memory and computation overhead, posing challenges for on-device deployment. Post-Training Quantization (PTQ) is an effective technique to reduce this overhead. Existing PTQ methods for feed-forward 3D models mainly focus on handling heavy-tailed activation distributions and constructing diverse calibration datasets. However, we observe that feed-forward 3D models predict multiple geometric attributes through a shared backbone, where different transformer blocks and hidden channels contribute distinctly to each task, resulting in substantially different sensitivities to quantization errors across tasks, blocks, and channels. Consequently, treating all tasks equally over-emphasizes insensitive tasks and causes significant accuracy loss on the sensitive ones. To address this issue, we propose Fisher-Guided Quantization (**FGQ**) for feed-forward 3D reconstruction models. Specifically, **FGQ** uses the diagonal Fisher information matrix to quantify the different sensitivities across tasks, blocks, and channels, and incorporates these sensitivities into the Learnable Affine Transformation during calibration to better preserve the channels and blocks most critical to each task. Extensive experiments across camera pose estimation, point map reconstruction, and depth estimation show that **FGQ** consistently outperforms state-of-the-art quantization baselines on VGGT, achieving up to 39% relative improvement under the 4-bit quantization.

1 Introduction

Feed-forward 3D reconstruction methods, represented by Visual Geometry Grounded Transformer (VGGT) [24], have achieved significant progress by jointly predicting depth, camera poses, and point maps in a single forward pass. They have been widely adopted in downstream applications such as spatial reasoning [6, 18], robotics [12, 28], and autonomous driving [2, 11]. However, their billion-scale parameters introduce prohibitive computation and memory overhead during inference, limiting real-time processing and on-device deployment [31].

A variety of methods have been proposed to address this efficiency bottleneck, including quantization [3, 15, 31], token merging [19, 21], and architectural modifications that redesign the attention or computation flow for better efficiency [22, 26].

Existing PTQ methods for feed-forward 3D models primarily address two challenges: (i) mitigating activation outliers associated with special tokens (e.g., camera and register tokens), and (ii) capturing the geometric and semantic complexity of 3D representations during calibration dataset construction. However, these approaches overlook a critical characteristic of these models: their inherent multi-task architecture. Since depth, pose, and point map prediction operate at vastly different scales and are evaluated with distinct metrics, they exhibit markedly different sensitivities to quantization-induced perturbations. As illustrated in Fig. 2, applying a uniform quantization strategy causes disproportionately severe degradation in point map reconstruction compared to camera pose and depth estimation. Beyond this task-level disparity, Fig. 3 (a)&(b) further shows that the per-block and per-channel quantization loss follows markedly different patterns across tasks. Together, these observations confirm that different transformer blocks and hidden channels contribute distinctly to each task, leading to substantially different quantization sensitivities across tasks, blocks, and channels.

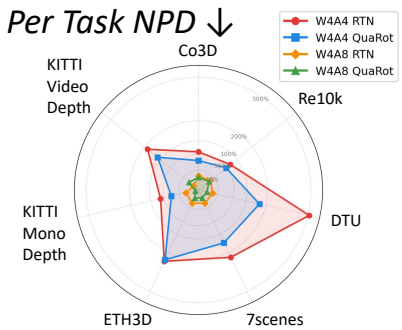


Figure 2: Normalized Performance Degradation (NPD) across various tasks (lower is better).

To this end, we propose Fisher-Guided Quantization (**FGQ**), a task-aware PTQ framework for feed-forward 3D reconstruction models. **FGQ** leverages the diagonal Fisher information matrix to jointly quantify quantization sensitivity across three granularities: tasks (depth, pose, and point map), transformer blocks, and hidden channels. These sensitivities are then incorporated

into the Learnable Affine Transformation during calibration, guiding the transformation to preserve the channels and blocks most critical to each task.

Our contributions are summarized as follows:

- We identify the multi-task sensitivity problem in PTQ for feed-forward 3D reconstruction, showing that different transformer blocks and hidden channels contribute distinctly to each task, leading to substantially different quantization sensitivities across tasks, blocks, and channels.
- We propose Fisher-Guided Quantization (**FGQ**), which uses the diagonal Fisher information matrix to quantify per-task, per-block, and per-channel sensitivity, and integrates these sensitivities into the Learnable Affine Transformation objective to guide calibration toward the most quantization-sensitive components.
- Extensive experiments on camera pose estimation, point map reconstruction, and depth estimation demonstrate that **FGQ** improves task performance by up to 39% over state-of-the-art PTQ baselines under 4-bit quantization, with the largest gains on point map reconstruction, where prior PTQ methods incur the most significant degradation.

2 Related Work

Feed-forward 3D reconstruction. Recent methods, including VGGT and π^3 [24, 25], directly predict scene geometry from images. Their learned representations support downstream rendering and scene understanding [6, 29]. Related work on 3D reasoning and generation [18, 30], further highlights the importance of connecting geometry with semantics. These applications motivate efficient 3D models. We therefore study VGGT quantization to reduce memory use while preserving reconstruction accuracy.

Efficient Feed-Forward 3D Reconstruction. Recent work targets the high compute and memory cost of feed-forward 3D reconstruction models such as VGGT. Token merging methods like FastVGGT [19] and LiteVGGT [21] fuse similar tokens by exploiting redundancy in global attention. Architectural redesigns restructure attention itself: FlashVGGT [26] approximates global attention as cross-attention to a compact set of descriptor tokens generated by spatial resampling, while AVGGT [22] converts early global layers into frame attention and subsamples K/V over patch tokens. Closest to our work, recent PTQ methods [3, 31] address two key challenges in quantizing these models: (i) mitigating activation outliers from data-independent special tokens (e.g., camera and register tokens), which yield heavy tails and extreme channel and token variance unfriendly to low-bit quantization; and (ii) dealing with calibration data that captures the geometric and semantic complexity of 3D multi-view inputs. However, prior PTQ methods apply uniform quantization across all heads and treat feed-forward 3D models as single-task models, overlooking the distinct quantization sensitivities of their depth, pose, and point map predictions.

Model Quantization. Post-training quantization (PTQ) reduces the memory and compute cost of large language models. Weight-only methods such as GPTQ [4] and AWQ [10] compress weights to low bit-widths with minimal accuracy loss. To further quantize activations and the KV cache, a recent line of work applies affine transformations to suppress outliers before quantization. QuaRot [1] uses fixed random Hadamard rotations, while SpinQuant [13] and FlatQuant [23] instead learn the transformation matrices during calibration: SpinQuant optimizes rotation matrices via Cayley optimization, and FlatQuant learns lightweight per-layer affine transformations that flatten weight and activation distributions. To better involve the sensitivity in the quantization process, we adopt the learnable affine transform based method as detailed in Sec. 3.2.3.

3 Method

3.1 Prelinminary

Visual Geometry Grounded Transformer (VGGT) [24] is a feed-forward transformer that maps a set of N input images $(I_i)_{i=1}^N$ to their 3D attributes (camera parameters, depth maps, point maps, and tracking features) in a single pass. Each image I_i is first patchified by a DINOv2 [14] encoder

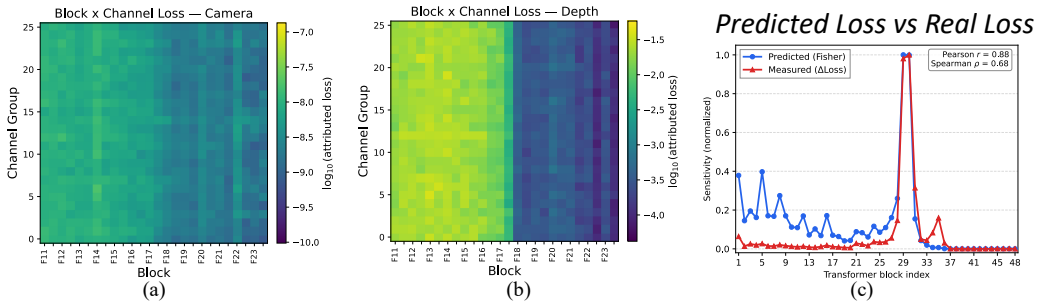


Figure 3: Fisher information accurately predicts quantization sensitivity and reveals task-specific patterns. (a) Per-block and per-channel quantization loss for camera pose estimation. (b) Per-block and per-channel quantization loss for depth estimation. The markedly quantization loss profiles between tasks motivate task-aware quantization strategies. (c) Fisher-predicted loss versus empirically measured loss across transformer blocks under W4A4 quantization, showing strong correlation ($r = 0.88$).

into K tokens $t_i \in \mathbb{R}^{K\times C}$, and the full token set is denoted $\mathbf{T} = [t_1; t_2; \ldots; t_N] \in \mathbb{R}^{N\times K\times C}$. The backbone consists of L *Alternating-Attention* (AA) blocks, each interleaving two self-attention operations:

$$\mathbf{T}_i^{(\ell+\frac{1}{2})} = \mathrm{FrameAttention}\big(\mathbf{T}_i^{(\ell)}\big), \quad i = 1, \ldots, N, \tag{1}$$

$$\mathbf{T}^{(\ell+1)} = \mathrm{GlobalAttention}\big(\mathbf{T}^{(\ell+\frac{1}{2})}\big), \tag{2}$$

where $\ell \in \{0, 1, \ldots, L-1\}$ indexes the AA block, $\mathbf{T}^{(\ell)}$ denotes the token state at the input of the ℓ-th block (with $\mathbf{T}^{(0)} = \mathbf{T}$), and $\mathbf{T}^{(\ell+\frac{1}{2})}$ denotes the intermediate state after frame attention but before global attention within the same block. Here, $\mathrm{FrameAttention}(\cdot)$ operates independently within each image's K tokens to preserve per-view structure, and $\mathrm{GlobalAttention}(\cdot)$ operates jointly over all $N \times K$ tokens to propagate information across views. The refined tokens $\mathbf{T}^{(L)}$ are then decoded by task-specific heads into the final 3D predictions.

3.2 Fisher-Guided Quantization

3.2.1 Sensitivity Varies Across Tasks

To quantify the impact of quantization on each downstream task, we define the Normalized Performance Degradation (NPD) of task k at bit-width b as

$$\mathrm{NPD}_k(b) \;=\; \frac{\big|\mathrm{Metric}_k(b) - \mathrm{Metric}_k(\mathrm{FP16})\big|}{\mathrm{Metric}_k(\mathrm{FP16})} \times 100\%. \tag{3}$$

Task-level sensitivity. As shown in Fig. 2, applying W4A4 quantization to VGGT with PTQ methods leads to highly unbalanced degradation across tasks: camera pose estimation remains relatively robust, while point map reconstruction collapses with an NPD of $200\% \sim 500\%$. This indicates that VGGT's shared backbone routes different geometric attributes through the same parameters, yet each task tolerates quantization noise to varying degrees.

Block and channel-level sensitivity. To locate the source of this asymmetry, we measure the per-block and per-channel quantization loss separately for the camera and depth tasks. As shown in Fig. 3(a) and (b), the two tasks produce clearly different loss profiles along both dimensions: blocks and channels that are critical for one task are often unimportant for the other. As a result, a uniform quantization policy is forced to compromise between tasks, and any objective that weights all tasks equally is dominated by the insensitive ones and therefore sacrifices accuracy on the sensitive ones. These observations motivate a task-aware design that allocates precision according to the sensitivity of each task at both the block and channel level.

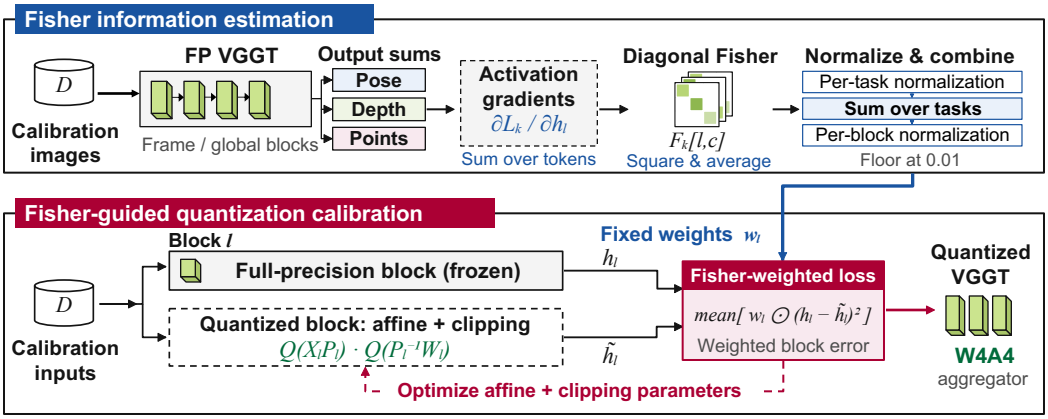


Figure 4: Overview of proposed Fisher-Guided Quantization (**FGQ**). We estimate channel-wise Fisher information from multiple geometry tasks (top) and use the resulting importance weights in the block-wise reconstruction loss to optimize affine and clipping parameters during VGGT quantization calibration (bottom).

3.2.2 Fisher Information for Sensitivity Measurement

To realize this design, we introduce a task-aware sensitivity score at the block and channel level, which is cheap to compute and can be directly plugged into the calibration objective. We derive the score from a second-order expansion of each task loss, approximate the Hessian with the Fisher information to avoid its prohibitive cost, and further reduce it to its diagonal form for efficiency.

Quantization error and the role of the Hessian. Let $\mathbf{h}_l \in \mathbb{R}^C$ denote the output of block l, where C is the channel dimension, and let $\mathcal{L}_k$ denote the loss of task k. Quantization perturbs $\mathbf{h}_l$ by $\Delta\mathbf{h}_l$, which aggregates the effect of all weight and activation rounding inside the block. A second-order Taylor expansion of $\mathcal{L}_k$ around the full-precision output gives

$$\Delta\mathcal{L}_k \;\approx\; \mathbf{g}_k^\top \Delta\mathbf{h}_l \;+\; \tfrac{1}{2}\,\Delta\mathbf{h}_l^\top \mathbf{H}_k\,\Delta\mathbf{h}_l, \tag{4}$$

where $\mathbf{g}_k$ and $\mathbf{H}_k$ are the gradient and Hessian of $\mathcal{L}_k$ with respect to $\mathbf{h}_l$. A pre-trained VGGT sits near a local optimum, and therefore $\mathbb{E}[\mathbf{g}_k] \to \mathbf{0}$ and the first-order term vanishes in expectation [9]. The expected loss change is governed by the Hessian term,

$$\mathbb{E}[\Delta\mathcal{L}_k] \;\approx\; \tfrac{1}{2}\,\Delta\mathbf{h}_l^\top \mathbf{H}_k\,\Delta\mathbf{h}_l, \tag{5}$$

which assigns a task-specific cost to each direction of perturbation: directions with large eigenvalues amplify quantization error into task loss, while directions with small eigenvalues absorb it.

Fisher information as a tractable surrogate. While Eq. 5 formalizes the appropriate notion of block-wise sensitivity, direct evaluation of $\mathbf{H}_k$ is computationally prohibitive in our setting. Second-order automatic differentiation incurs substantial runtime overhead per block and per task, and explicit storage of $\mathbf{H}_k$ requires $\mathcal{O}(C^2)$ memory per block. To avoid these costs, we use a Fisher-based surrogate for the Hessian. This substitution is common in the pruning and quantization literature [8, 27] and is motivated by the following standard identity.

For our calibration setting, x denotes a set of input images. The variable y denotes the target signal used to compute the task loss $\mathcal{L}_k$. We write $q_k(x, y) = q_k(x)q_k(y \mid x)$ for the task-specific calibration distribution. For readability, the proposition below omits the task subscript and writes q instead of q_k.

Proposition 3.1 (Conditional Hessian–Fisher identity)**.** *Let z be a differentiable argument of a normalized conditional model $p_z(y \mid x)$, where z may denote model parameters or an intermediate activation treated as a local input to the remaining network. Define the negative log-likelihood loss $\ell(z; x, y) = -\log p_z(y \mid x)$. Assume standard regularity conditions that allow differentiation under the integral. If, at $z^\star$, the data conditional distribution is matched by the model, i.e., $q(y \mid x) = p_{z^\star}(y \mid x)$ for $q(x)$-almost every x, then*

$$\mathbb{E}_{x\sim q(x),\, y\sim q(y|x)}\left[\nabla_z^2 \ell(z^\star; x, y)\right] = \mathbb{E}_{x\sim q(x),\, y\sim q(y|x)}\left[\nabla_z \ell(z^\star; x, y) \nabla_z \ell(z^\star; x, y)^\top\right]. \tag{6}$$

The matrix on the right-hand side is the Fisher information matrix with respect to z.

We give a self-contained proof of Proposition 3.1, which is an instance of the second Bartlett identity, in Appendix A. The proposition shows that, under a correctly specified likelihood and population expectation, the expected Hessian of the negative log-likelihood equals the Fisher information matrix. In practice, we use this identity as a motivation for replacing the block Hessian by an empirical Fisher estimated from first-order gradients at a well-trained checkpoint.

Instantiating this surrogate at the output of block l gives

$$\mathbb{E}[\Delta\mathcal{L}_k] \;\approx\; \tfrac{1}{2}\,\Delta\mathbf{h}_l^\top \mathbf{F}_k^{h_l} \Delta\mathbf{h}_l, \qquad \left[\mathbf{F}_k^{h_l}\right]_{c,c'} = \mathbb{E}_{(x,y)\sim\mathcal{D}_{\mathrm{cal}}}\left[\frac{\partial\mathcal{L}_k(x,y)}{\partial h_{l,c}}\,\frac{\partial\mathcal{L}_k(x,y)}{\partial h_{l,c'}}\right]. \tag{7}$$

Here $\mathcal{D}_{\mathrm{cal}}$ is a small calibration set. Since $\mathbf{F}_k^{h_l}$ only uses first-order gradients, it can be estimated with one backward pass per task. Moreover, evaluating the Fisher at the block output is convenient because $\Delta\mathbf{h}_l$ already contains the total perturbation induced by quantizing block l. Thus a single block-level Fisher term captures the joint sensitivity of all quantized operations inside the block. As shown in Fig. 3(c), this Fisher-based score is also empirically predictive of the measured quantization loss, with Pearson correlation $r = 0.88$.

Diagonal approximation. Although the Fisher surrogate avoids second-order derivatives, the full matrix $\mathbf{F}_k^{h_l}$ is still a $C \times C$ matrix for each block and task. This is costly to store and its off-diagonal entries can be hard to estimate from a small calibration set. We therefore use the diagonal empirical Fisher:

$$F_k[l,c] = \mathbb{E}_{(x,y)\sim\mathcal{D}_{\mathrm{cal}}}\left[\left(\frac{\partial\mathcal{L}_k(x,y)}{\partial h_{l,c}(x)}\right)^2\right] \approx \frac{1}{N}\sum_{n=1}^{N}\left(\frac{\partial\mathcal{L}_k(x_n,y_n)}{\partial h_{l,c}(x_n)}\right)^2. \tag{8}$$

This reduces storage from $\mathcal{O}(C^2)$ to $\mathcal{O}(C)$. The quantity $F_k[l,c]$ is the second moment of the per-sample gradient on channel c. Under the population Fisher identity, where the expected score is zero, it also equals the gradient variance. Keeping only the diagonal discards cross-channel terms and treats channel perturbations as locally independent. This approximation is sufficient for our goal, since we use the score to rank channel importance rather than to recover the full curvature matrix.

With this approximation, Eq. 7 becomes

$$\mathbb{E}[\Delta\mathcal{L}_k] \approx \frac{1}{2}\mathbb{E}_{x\sim\mathcal{D}_{\mathrm{cal}}}\left[\sum_{c=1}^{C} F_k[l,c]\left(\Delta h_{l,c}(x)\right)^2\right]. \tag{9}$$

Thus, the loss increase is approximated by a channel-wise weighted reconstruction error at the block output. Each $F_k[l,c]$ measures how sensitive task k is to squared error on channel c of block l. Compared with a uniform reconstruction loss, this form assigns larger weights to channels that are more important for the task. We use these task-aware channel weights in the learnable affine transformation described next.

3.2.3 Integrating Fisher Information into Learnable Affine Transformations

Building on the per-channel Fisher score $F_k[l,c]$ from Section 3.2.2, we use task sensitivity to guide the calibration of learnable affine transformations. Let $l \in \{1,\dots,L\}$ index the calibrated transformer blocks, $c \in \{1,\dots,C\}$ index hidden channels, and $k \in \{1,\dots,K\}$ index VGGT task heads.

Block-wise calibration objective. Learnable affine transformation methods [13, 23] insert an invertible matrix $\mathbf{P}_l$ into a linear layer and fold its inverse into the weight:

$$\mathbf{Y}_l = \left(\mathbf{X}_l\mathbf{P}_l\right)\left(\mathbf{P}_l^{-1}\mathbf{W}_l\right) = \mathbf{X}_l\mathbf{W}_l. \tag{10}$$

Quantization is then applied to the transformed activation $\mathbf{X}_l\mathbf{P}_l$ and transformed weight $\mathbf{P}_l^{-1}\mathbf{W}_l$, whose distributions are easier to quantize while preserving the full-precision computation before quantization.

For block l, standard calibration optimizes the learnable parameters $\boldsymbol{\theta}_l$ by matching the quantized block output $\tilde{\mathbf{h}}_l$ to the full-precision output $\mathbf{h}_l$:

$$\mathcal{L}^l_{\text{uni}}(\boldsymbol{\theta}_l) = \mathbb{E}_x \left[\frac{1}{|\mathcal{T}_l|C} \sum_{t \in \mathcal{T}_l} \sum_{c=1}^{C} \big(h_{l,t,c}(x) - \tilde{h}_{l,t,c}(x; \boldsymbol{\theta}_l)\big)^2 \right], \tag{11}$$

where $\mathcal{T}_l$ denotes the tokens of block l. This uniform loss weights all channels equally, although errors in different channels can have very different effects on downstream task predictions.

Task-aware Fisher aggregation. VGGT predicts multiple tasks, and each task head gives a Fisher score $F_k[l, c]$ for channel c in block l. Since raw gradient magnitudes can differ across task heads, we first normalize each task by its mean Fisher value:

$$\overline{F}_k = \frac{1}{LC} \sum_{l'=1}^{L} \sum_{c'=1}^{C} F_k[l', c']. \tag{12}$$

We then combine the normalized scores with equal task weights:

$$s[l, c] = \sum_{k=1}^{K} \frac{F_k[l, c]}{\overline{F}_k}. \tag{13}$$

Here, L is the number of calibrated blocks and C is the hidden channel dimension. This normalization prevents a task with larger raw gradients from dominating the calibration weights.

Per-block normalization. Because calibration is performed block by block, we normalize the combined score within each block:

$$w_l[c] = \frac{s[l, c]}{\frac{1}{C} \sum_{c'=1}^{C} s[l, c']}. \tag{14}$$

Thus, each block has mean weight 1, so the loss scale is comparable across blocks. We also apply a small floor

$$w_l[c] \leftarrow \max\big(w_l[c], 0.01\big), \tag{15}$$

which prevents low-Fisher channels from being completely ignored.

Fisher-guided calibration loss. **FGQ** replaces the uniform calibration loss with a Fisher-weighted objective:

$$\mathcal{L}^l_{\text{FGQ}}(\boldsymbol{\theta}_l) = \mathbb{E}_x \left[\frac{1}{|\mathcal{T}_l|C} \sum_{t \in \mathcal{T}_l} \sum_{c=1}^{C} w_l[c] \big(h_{l,t,c}(x) - \tilde{h}_{l,t,c}(x; \boldsymbol{\theta}_l)\big)^2 \right]. \tag{16}$$

This objective allocates more calibration capacity to channels whose errors are more likely to affect VGGT task outputs, while still preserving all channels through the floor term.

4 Evaluation

4.1 Experimental Setup

Evaluation settings. We primarily evaluate **FGQ** on the open-source VGGT-1.2B model [24]. We consider the main 3D vision tasks supported by VGGT and evaluate each task on standard benchmarks: (i) **camera pose estimation** on Co3Dv2 [16] and Re10K [32]; (ii) **point map reconstruction** on 7-Scenes [20], DTU [7], and ETH3D [17]; and (iii) **depth estimation** on KITTI [5] for both monocular depth and video depth settings.

For the learnable affine quantization scheme, we implement FGQ on top of FlatQuant [23]. FlatQuant is used as a strong and public quantization backbone, while FGQ elaborates the calibration objective. Detailed quantization and calibration settings are provided in Appendix B. We report results under both **W4A8** and **W4A4** settings to evaluate FGQ under different quantization regimes. To further test generalization, we also extend **FGQ** to the π^3 model [25]. The result can be found in Appendix D.

Table 1: Camera pose estimation results for VGGT on Co3Dv2 [16] and Re10K [32]. AUC metrics measure the area under the cumulative error curve at different thresholds (higher is better). Best results in **bold**.

Method	Bitwidth (W/A)	Co3Dv2				Re10K			
		AUC@30↑	AUC@15↑	AUC@5↑	AUC@3↑	AUC@30↑	AUC@15↑	AUC@5↑	AUC@3↑
FP16	16/16	0.9731	0.9462	0.8462	0.7630	0.8667	0.7818	0.5803	0.4688
RTN	4/8	0.9619	0.9238	0.7822	0.6691	0.8483	0.7514	0.5318	0.4158
QuaRot	4/8	0.9672	0.9343	0.8098	0.7032	0.8480	0.7537	0.5307	0.4038
FlatQuant	4/8	0.9674	0.9351	0.8051	0.7191	0.8481	0.7543	0.5262	0.3958
QuantVGGT	4/8	0.9684	0.9363	0.8127	0.7117	0.8516	0.7587	0.5379	0.4125
FGQ	**4/8**	**0.9686**	**0.9371**	**0.8185**	**0.7230**	**0.8566**	**0.7651**	**0.5485**	**0.4318**
RTN	4/4	0.6547	0.3950	0.0474	0.0123	0.4334	0.2520	0.0590	0.0224
QuaRot	4/4	0.8169	0.6471	0.2252	0.0904	0.5722	0.3952	0.1381	0.0663
FlatQuant	4/4	0.9295	0.8648	0.6405	0.4691	0.7240	0.5849	0.3233	0.2169
QuantVGGT	4/4	0.9387	0.8840	0.6973	0.5419	0.7703	0.6437	0.3845	0.2710
FGQ	**4/4**	**0.9425**	**0.8887**	**0.7025**	**0.5575**	**0.7704**	**0.6443**	**0.3898**	**0.2756**

Table 2: Point map reconstruction results for VGGT on 7-Scenes [20] and ETH3D [17]. Acc measures accuracy (higher is better), Comp measures completeness (lower is better), and N.C. measures normal consistency (higher is better). Best results in **bold**.

Method	Bitwidth (W/A)	7-Scenes						ETH3D					
		Acc.↓		Comp.↓		N.C.↑		Acc.↓		Comp.↓		N.C.↑	
		mean	med	mean	med	mean	med	mean	med	mean	med	mean	med
FP16	16/16	0.044	0.024	0.056	0.033	0.733	0.846	0.263	0.167	0.288	0.167	0.846	0.947
RTN	4/8	0.046	0.027	0.060	0.037	0.729	0.839	0.271	0.183	0.299	0.174	0.840	0.938
QuaRot	4/8	0.047	0.027	0.056	0.033	**0.736**	**0.848**	0.262	0.174	0.272	0.161	0.846	0.951
FlatQuant	4/8	0.045	0.025	**0.055**	**0.031**	0.732	0.845	0.262	0.169	0.287	0.168	0.845	0.943
QuantVGGT	4/8	0.044	0.025	0.056	0.033	0.729	0.840	0.269	0.185	0.292	0.170	0.842	0.942
FGQ	**4/8**	**0.043**	**0.024**	**0.055**	**0.031**	**0.736**	**0.848**	**0.260**	**0.165**	**0.270**	**0.158**	**0.848**	**0.949**
RTN	4/4	0.146	0.103	0.134	0.086	0.600	0.655	0.944	0.843	1.339	0.920	0.603	0.656
QuaRot	4/4	0.107	0.079	0.165	0.128	0.663	0.749	0.917	0.781	1.151	0.757	0.667	0.753
FlatQuant	4/4	0.056	0.036	0.070	0.047	0.717	0.824	0.291	0.193	0.297	0.173	0.833	0.935
QuantVGGT	4/4	0.053	0.032	0.085	0.059	0.719	0.828	0.312	0.206	0.305	0.180	0.832	0.937
FGQ	**4/4**	**0.048**	**0.031**	**0.059**	**0.036**	**0.723**	**0.832**	**0.275**	**0.177**	**0.279**	**0.164**	**0.834**	**0.940**

Baselines. We compare **FGQ** against representative PTQ methods: (i) **RTN**: round-to-nearest quantization without calibration, serving as a lower bound; (ii) **QuaRot** [1]: fixed Hadamard rotation for activation smoothing; (iii) **FlatQuant** [23]: learnable affine transformations with uniform reconstruction loss, which serves as our base method; (iv) **QuantVGGT** [3]: a recent VGGT-specific PTQ method that addresses activation outliers from special tokens. For fair comparison, all methods use the same calibration set and quantization configuration.

4.2 Main Results

Camera pose estimation. Table 1 evaluates camera pose estimation on Co3Dv2 and Re10K. Under W4A8, most methods remain close to FP16, indicating that this task is relatively robust to moderate activation quantization. In this setting, **FGQ** still gives the best overall results on Re10K and competitive results on Co3Dv2. In the more challenging W4A4 setting, where activation quantization introduces a larger performance drop, **FGQ** consistently performs on par with or better than existing quantization baselines across datasets. Compared with the dense reconstruction results below, the improvement on pose estimation is modest, suggesting that this global prediction task is less sensitive to the Fisher-guided channel selection than dense geometric outputs.

Point map reconstruction. Table 2 and 3 report point map reconstruction results on 7-Scenes, ETH3D, and DTU. **FGQ** shows a clearer advantage on this task. Point maps are dense geometric outputs, so activation quantization errors affect many spatial locations and can directly degrade surface accuracy, completeness, and normal consistency. Under W4A8, **FGQ** closely matches FP16 and obtains the strongest overall performance. Under W4A4, the advantage becomes more pronounced. On 7-Scenes, **FGQ** reduces the mean completeness error from 0.085 to 0.059 compared with QuantVGGT, while also improving mean accuracy from 0.053 to 0.048. On ETH3D, **FGQ**

Table 4: Depth estimation results on KITTI (monocular and video depth). Lower is better for AbsRel, SqRel, and RMSE; higher is better for $\delta < 1.25$. Best results in **bold**.

Method	Bitwidth (W/A)	KITTI (Mono)				KITTI (Video)			
		AbsRel↓	SqRel↓	RMSE↓	$\delta < 1.25$↑	AbsRel↓	SqRel↓	RMSE↓	$\delta < 1.25$↑
FP16	16/16	0.092	0.459	3.902	0.936	0.058	0.373	3.618	0.961
RTN	4/8	0.093	0.453	3.903	0.936	0.058	0.348	3.500	0.964
QuaRot	4/8	0.087	0.443	3.897	0.943	0.059	0.364	3.642	0.963
FlatQuant	4/8	0.087	0.410	3.623	0.946	0.052	0.314	3.350	0.968
QuantVGGT	4/8	0.091	0.462	3.931	0.936	0.054	0.339	3.450	0.967
FGQ	**4/8**	**0.086**	**0.396**	**3.566**	**0.947**	**0.051**	**0.292**	**3.240**	**0.970**
RTN	4/4	0.148	0.728	5.159	0.823	0.164	0.985	5.615	0.748
QuaRot	4/4	0.117	0.487	4.083	0.903	0.126	0.713	4.896	0.838
FlatQuant	4/4	0.084	0.399	3.650	0.948	0.062	0.339	3.488	0.960
QuantVGGT	4/4	0.088	0.446	3.842	0.938	0.058	0.336	3.486	0.963
FGQ	**4/4**	**0.079**	**0.389**	**3.642**	**0.951**	**0.056**	**0.313**	**3.384**	**0.965**

improves mean accuracy from 0.312 to 0.275 and mean completeness from 0.305 to 0.279. On DTU, **FGQ** also gives the best completeness and normal consistency among W4A4 methods. These results indicate that **FGQ** better preserves the features needed for dense 3D geometry. Qualitative comparisons are shown in Figure 1 and Appendix G. On 7-Scenes, **FGQ** (W4A4) preserves finer structural details than QuantVGGT (W4A4), while on DTU the QuantVGGT (W4A4) reconstruction appears visibly blurry and **FGQ** (W4A4) remains sharp.

Depth estimation. Table 4 reports depth estimation results on KITTI for both monocular and video inputs. **FGQ** consistently outperforms the quantization baselines in the W4A8 setting, achieving the best results across all metrics on both evaluation protocols. The gains are especially clear on error metrics such as AbsRel, SqRel, and RMSE, showing that Fisher-guided calibration better preserves the dense depth structure under activation quantization. In the more aggressive W4A4 setting, **FGQ** remains the strongest overall method. It achieves the best AbsRel, SqRel, and threshold accuracy on KITTI Mono, and the best overall error metrics on KITTI Video among the W4A4 methods. These results are consistent with the point map reconstruction experiments: **FGQ** provides larger benefits on dense geometric tasks than on global pose estimation, since dense outputs are more sensitive to spatially accumulated quantization errors.

Table 3: Point map reconstruction for VGGT on DTU [7]. Best results in **bold**.

Method	Bitwidth (W/A)	DTU					
		Acc.↓		Comp.↓		N.C.↑	
		Mean	Med	Mean	Med	Mean	Med
FP16	16	1.308	0.761	1.929	1.015	0.665	0.750
RTN	4/8	1.363	0.786	1.904	0.976	0.669	0.753
QuaRot	4/8	1.401	0.815	1.899	0.978	0.668	0.754
FlatQuant	4/8	1.429	0.840	1.910	0.977	0.665	0.752
QuantVGGT	4/8	1.292	0.752	1.944	1.007	0.667	0.753
FGQ	**4/8**	**1.287**	**0.742**	**1.881**	**0.965**	**0.675**	**0.762**
RTN	4/4	7.987	5.648	4.355	2.593	0.657	0.731
QuaRot	4/4	3.470	2.107	2.145	1.077	0.665	0.750
FlatQuant	4/4	1.478	0.829	1.895	0.982	0.667	0.754
QuantVGGT	4/4	1.488	0.837	1.933	1.001	**0.669**	0.756
FGQ	**4/4**	**1.420**	**0.801**	**1.879**	**0.965**	**0.669**	**0.757**

4.3 Ablation study

We study the effect of different task losses in the empirical Fisher estimation. All variants use the same quantization setting, calibration data, optimization steps, and evaluation protocol. The plain baseline removes Fisher weighting and uses an unweighted calibration objective. Co3Dv2 mainly evaluates camera pose accuracy, while DTU evaluates point-map reconstruction. As shown in Table 5, Fisher weighting improves both benchmarks. On Co3Dv2, adding point and depth Fisher improves AUC@15 from 0.8825 to 0.8875, while further adding camera Fisher gives the largest gain and reaches 0.8952. On DTU, adding point Fisher to the camera-depth Fisher gives the best Acc. mean, reducing it from 1.458 to 1.446. This shows that the Fisher term is most useful when it includes the task aligned with the evaluation metric. At the same time, the gains from multi-task Fisher suggest that different task losses provide complementary channel sensitivity signals.

Table 5: Ablation study on the Fisher Information design.

Fisher Info.	Co3Dv2 AUC@15 ↑	Fisher Info.	DTU Acc. mean ↓
plain (w/o Fisher)	0.8825	plain (w/o Fisher)	1.537
point	0.8868	camera	1.465
point + depth	0.8875	camera + depth	1.458
point + depth + camera	**0.8952**	camera + depth + point	**1.446**

4.4 Efficiency Analysis

We evaluate the runtime efficiency of our W4A4 quantized model on a single NVIDIA A100 40G GPU. To measure the benefit of real low-bit execution, we use fake quantization as the baseline and report speedups from the real quantized implementation. Our implementation builds on the Triton kernel from FlatQuant [23] and tailors it to our design. We decompose the latency into parameter loading and compute kernel time, where the compute kernel includes the affine transformation, quantization, and quantized linear computation. For parameter loading, RTN and our method achieve similar speedups, $3.41\times$ and $3.42\times$, since both benefit from 4-bit weight and activation storage. For the compute kernel, RTN reaches $6.01\times$, while our method reaches $4.67\times$. The gap comes from the extra affine transformation used by our method. At the full block level, RTN obtains $2.01\times$ speedup and our method still achieves $1.81\times$ speedup. These results show that our method keeps most of the real quantization efficiency while adding only moderate runtime overhead for the accuracy-preserving affine transformation.

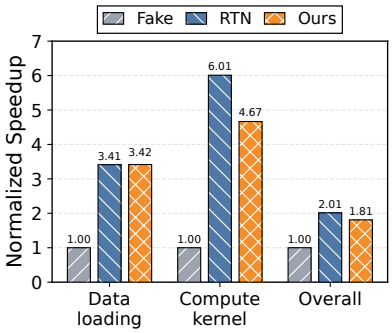


Figure 5: Normalized Speedup over the same W4A4 quantization settings.

5 Conclusion

We presented Fisher-Guided Quantization (**FGQ**), a post-training quantization method for feed-forward 3D reconstruction models. Our main observation is that different transformer blocks and hidden channels in the model contribute differently to each task. As a result, uniformly treating all outputs during calibration can therefore preserve insensitive tasks while harming the tasks that are more sensitive to quantization noise. **FGQ** addresses this issue by using diagonal Fisher information to estimate task-aware block and channel sensitivities, and by incorporating these sensitivities into the learnable affine transformation during calibration. Extensive experiments show that **FGQ** consistently outperforms existing PTQ baselines under 4-bit quantization, with the largest gains appearing on point map reconstruction, where prior PTQ methods suffer the most severe degradation.

Acknowledgments and Disclosure of Funding

This work is supported by AI Chip Center for Emerging Smart Systems (ACCESS), Hong Kong SAR and Collaborative Research Fund (UGC CRF) C5032-23G.

References

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

[2] Jintao Cheng, Kang Zeng, Zhuoxu Huang, Xiaoyu Tang, Jin Wu, Chengxi Zhang, Xieyuanli Chen, and Rui Fan. Mf-mos: A motion-focused model for moving object segmentation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12499–12505. IEEE, 2024.

[3] Weilun Feng, Haotong Qin, Mingqiang Wu, Chuanguang Yang, Yuqi Li, Xiangqi Li, Zhulin An, Libo Huang, Yulun Zhang, Michele Magno, et al. Quantized visual geometry grounded transformer. *arXiv preprint arXiv:2509.21302*, 2025.

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

[5] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The international journal of robotics research*, 32(11):1231–1237, 2013.

[6] Wenbo Hu, Jingli Lin, Yilin Long, Yunlong Ran, Lihan Jiang, Yifan Wang, Chenming Zhu, Runsen Xu, Tai Wang, and Jiangmiao Pang. g^2vlm: Geometry grounded vision language model with unified 3d reconstruction and spatial reasoning. *arXiv preprint arXiv:2511.21688*, 2025.

[7] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 406–413, 2014.

[8] Woosuk Kwon, Sehoon Kim, Michael W Mahoney, Joseph Hassoun, Kurt Keutzer, and Amir Gholami. A fast post-training pruning framework for transformers. *Advances in Neural Information Processing Systems*, 35:24101–24116, 2022.

[9] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

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

[11] Junhong Lin, Kangli Wang, Shunzhou Wang, Songlin Fan, Ge Li, and Wei Gao. Vgd: Visual geometry gaussian splatting for feed-forward surround-view driving reconstruction. *arXiv preprint arXiv:2510.19578*, 2025.

[12] Yuanzhe Liu, Jingyuan Zhu, Yuchen Mo, Gen Li, Xu Cao, Jin Jin, Yifan Shen, Zhengyuan Li, Tianjiao Yu, Wenzhen Yuan, et al. Palm: Progress-aware policy learning via affordance reasoning for long-horizon robotic manipulation. *arXiv preprint arXiv:2601.07060*, 2026.

[13] 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. *arXiv preprint arXiv:2405.16406*, 2024.

[14] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[15] Sicheng Pan, Chen Tang, Shuzhao Xie, Ke Yang, Weixiang Zhang, Jiawei Li, Bin Chen, Shu-Tao Xia, and Zhi Wang. Tail-aware post-training quantization for 3d geometry models. *arXiv preprint arXiv:2602.01741*, 2026.

[16] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10901–10911, 2021.

[17] Thomas Schops, Johannes L Schonberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3260–3269, 2017.

[18] Yifan Shen, Yuanzhe Liu, Jingyuan Zhu, Xu Cao, Xiaofeng Zhang, Yixiao He, Wenming Ye, James Rehg, and Ismini Lourentzou. Fine-grained preference optimization improves spatial reasoning in vlms. *Advances in Neural Information Processing Systems*, 38:17929–17960, 2026.

[19] You Shen, Zhipeng Zhang, Yansong Qu, Xiawu Zheng, Jiayi Ji, Shengchuan Zhang, and Liujuan Cao. Fastvggt: Training-free acceleration of visual geometry transformer. *arXiv preprint arXiv:2509.02560*, 2025.

[20] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2930–2937, 2013.

[21] Zhijian Shu, Cheng Lin, Tao Xie, Wei Yin, Ben Li, Zhiyuan Pu, Weize Li, Yao Yao, Xun Cao, Xiaoyang Guo, et al. Litevggt: Boosting vanilla vggt via geometry-aware cached token merging. *arXiv preprint arXiv:2512.04939*, 2025.

[22] Xianbing Sun, Zhikai Zhu, Zhengyu Lou, Bo Yang, Jinyang Tang, Liqing Zhang, He Wang, and Jianfu Zhang. Avggt: Rethinking global attention for accelerating vggt. *arXiv preprint arXiv:2512.02541*, 2025.

[23] Yuxuan Sun, Ruikang Liu, Haoli Bai, Han Bao, Kang Zhao, Yuening Li, Jiaxin Hu, Xianzhi Yu, Lu Hou, Chun Yuan, et al. Flatquant: Flatness matters for llm quantization. In *International Conference on Machine Learning*, pages 57587–57613. PMLR, 2025.

[24] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5294–5306, 2025.

[25] Yifan Wang, Jianjun Zhou, Haoyi Zhu, Wenzheng Chang, Yang Zhou, Zizun Li, Junyi Chen, Jiangmiao Pang, Chunhua Shen, and Tong He. π^3: Permutation-equivariant visual geometry learning. *arXiv preprint arXiv:2507.13347*, 2025.

[26] Zipeng Wang and Dan Xu. Flashvggt: Efficient and scalable visual geometry transformers with compressed descriptor attention. *arXiv preprint arXiv:2512.01540*, 2025.

[27] Huanrui Yang, Yafeng Huang, Zhen Dong, Denis A Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Yuan Du, Kurt Keutzer, and Shanghang Zhang. Fisher-aware quantization for detr detectors with critical-category objectives. *arXiv preprint arXiv:2407.03442*, 2024.

[28] Sizhe Yang, Linning Xu, Hao Li, Juncheng Mu, Jia Zeng, Dahua Lin, and Jiangmiao Pang. Robo3r: Enhancing robotic manipulation with accurate feed-forward 3d reconstruction. *arXiv preprint arXiv:2602.10101*, 2026.

[29] Tianjiao Yu, Xinzhuo Li, Yifan Shen, Yuanzhe Liu, and Ismini Lourentzou. Core3d: Collaborative reasoning as a foundation for 3d intelligence. *arXiv preprint arXiv:2512.12768*, 2025.

[30] Tianjiao Yu, Xinzhuo Li, Yifan Shen, Onkar Susladkar, Yuanzhe Liu, Xiaona Zhou, and Ismini Lourentzou. Elsa3d: Elastic semantic anchoring for unified 3d understanding and generation. *arXiv preprint arXiv:2607.06565*, 2026.

[31] Yipu Zhang, Jintao Cheng, Xingyu Liu, Zeyu Li, Carol Jingyi Li, Jin Wu, Lin Jiang, Yuan Xie, Jiang Xu, and Wei Zhang. Versaq-3d: A reconfigurable accelerator enabling feed-forward and generalizable 3d reconstruction via versatile quantization. *arXiv preprint arXiv:2601.20317*, 2026.

[32] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*, 2018.

A Proof of Proposition 3.1

We prove the identity for a continuous label space. The same argument applies to discrete labels by replacing integrals with sums. Fix an input x. Since $p_z(y \mid x)$ is a normalized conditional distribution,

$$\int p_z(y \mid x)\, dy = 1. \tag{17}$$

Under the stated regularity conditions, we may differentiate Eq. (17) with respect to z. Differentiating once gives

$$0 = \int \nabla_z p_z(y \mid x)\, dy = \int p_z(y \mid x) \nabla_z \log p_z(y \mid x)\, dy. \tag{18}$$

Thus the expected score is zero:

$$\mathbb{E}_{y \sim p_z(\cdot|x)}[\nabla_z \log p_z(y \mid x)] = 0. \tag{19}$$

Differentiating Eq. (18) once more gives

$$\begin{aligned} 0 &= \int \nabla_z \left[p_z(y \mid x) \nabla_z \log p_z(y \mid x) \right] dy \\ &= \int p_z(y \mid x) \left[\nabla_z^2 \log p_z(y \mid x) + \nabla_z \log p_z(y \mid x) \nabla_z \log p_z(y \mid x)^\top \right] dy. \end{aligned} \tag{20}$$

Equivalently,

$$\mathbb{E}_{y \sim p_z(\cdot|x)} \left[-\nabla_z^2 \log p_z(y \mid x) \right] = \mathbb{E}_{y \sim p_z(\cdot|x)} \left[\nabla_z \log p_z(y \mid x) \nabla_z \log p_z(y \mid x)^\top \right]. \tag{21}$$

Now define the negative log-likelihood loss

$$\ell(z; x, y) = -\log p_z(y \mid x). \tag{22}$$

Then

$$\nabla_z^2 \ell(z; x, y) = -\nabla_z^2 \log p_z(y \mid x), \qquad \nabla_z \ell(z; x, y) = -\nabla_z \log p_z(y \mid x). \tag{23}$$

Substituting these relations into Eq. (21) yields

$$\mathbb{E}_{y \sim p_z(\cdot|x)} \left[\nabla_z^2 \ell(z; x, y) \right] = \mathbb{E}_{y \sim p_z(\cdot|x)} \left[\nabla_z \ell(z; x, y) \nabla_z \ell(z; x, y)^\top \right]. \tag{24}$$

Evaluating Eq. (24) at $z = z^\star$ and using the assumption $q(y \mid x) = p_{z^\star}(y \mid x)$, we obtain

$$\mathbb{E}_{y \sim q(\cdot|x)} \left[\nabla_z^2 \ell(z^\star; x, y) \right] = \mathbb{E}_{y \sim q(\cdot|x)} \left[\nabla_z \ell(z^\star; x, y) \nabla_z \ell(z^\star; x, y)^\top \right]. \tag{25}$$

Finally, taking expectation over $x \sim q(x)$ gives

$$\mathbb{E}_{x \sim q(x),\, y \sim q(y|x)} \left[\nabla_z^2 \ell(z^\star; x, y) \right] = \mathbb{E}_{x \sim q(x),\, y \sim q(y|x)} \left[\nabla_z \ell(z^\star; x, y) \nabla_z \ell(z^\star; x, y)^\top \right], \tag{26}$$

which proves Proposition 3.1. □

B Quantization and Calibration Settings

Implementation based on FlatQuant. FGQ is implemented on top of FlatQuant [23]. We use the same learnable affine transformation modules, learnable clipping parameters, and quantizer placement as FlatQuant. All pretrained model parameters are frozen before calibration. Calibration is performed block by block, and only the calibration parameters in the current block are optimized. Specifically, the trainable parameters are the FlatQuant transformation parameters, including the transformation matrices and diagonal scales, together with the learnable weight-clipping and activation-clipping factors. The original linear weights and biases, normalization layers, patch embedding/DINOv2 backbone, prediction heads, and all non-current blocks are kept frozen. This setup isolates the effect of replacing the standard FlatQuant reconstruction loss with the proposed Fisher-guided calibration loss.

Quantization setting. For the default W4A4 setting, both weights and activations are quantized to 4 bits. Weight and activation quantization are symmetric. Weights are quantized per output channel, while activations are quantized per token without grouping. Learnable weight clipping and learnable activation clipping are enabled. Quantization is applied to the attention projections, including Q, K, V, and output projections, as well as the MLP linear layers in the VGGT aggregator frame and global blocks. The DINOv2/patch embedding module, camera/depth/point/track heads, normalization parameters, and special tokens are not quantized.

Tasks and metrics. VGGT jointly predicts three tasks. For **depth estimation**, we report absolute relative error (AbsRel) and δ_1 accuracy (percentage of pixels with relative error < 1.25). For **camera pose estimation**, we report the Area Under the Curve (AUC) of pose accuracy across multiple thresholds. For **point map reconstruction**, we report Accuracy (Acc.), Completeness (Comp.), and Normal Consistency (N.C.), with both mean and median values.

Calibration data. We use Co3Dv2 [16] as the calibration data source. For calibration, we sample 64 clips, with 4 frames per clip. The batch size is 2 clips, corresponding to 8 frames per mini-batch. The default random seed is 42. The sampled calibration clips are loaded once and then reused for all blocks and all calibration epochs. The implementation samples directly from the raw Co3Dv2 directory by first sampling a category, then a sequence, and then 4 images from the sequence image folder. The current sampler does not use annotation-based quality filters. Frames are randomly sampled from the available images in each sequence, and no explicit temporal stride or temporal ordering constraint is imposed.

Optimization hyperparameters. Calibration uses AdamW. Unless overridden by parameter groups, AdamW uses the PyTorch default hyperparameters: $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, and weight decay 0.01. No gradient clipping is used. Mixed precision is enabled by default using CUDA bfloat16 autocast, without a gradient scaler. The base learning rate for the FlatQuant transformation matrices and diagonal scales is 10^{-2}. The learning rate for the learnable weight-clipping and activation-clipping factors is 10 times larger, i.e., 10^{-1} in the default W4A4 setting.

The learning rate is decayed with a cosine schedule. The scheduler is stepped once per optimizer step. For each block, the number of calibration epochs is 15. With 64 clips and a batch size of 2 clips, this gives 32 mini-batches per epoch and 480 optimizer steps per block. VGGT contains 48 calibrated aggregator blocks, so the full W4A4 calibration uses $48 \times 480 = 23040$ optimizer steps. The cosine schedule uses

$$T_{\max} = \texttt{epochs} \times (\texttt{nsamples}//\texttt{cali_bsz}),$$

with minimum learning rate $\eta_{\min} = 10^{-5}$ for the default setting.

Model mode during calibration. The model is kept in evaluation mode during calibration. Thus, dropout and stochastic-depth layers are disabled, and batch-normalization layers, if present, use evaluation behavior. Gradients are enabled only for the current block's calibration parameters. For each block, the full-precision output of that block is first used as the reconstruction target, and the quantized block output is then optimized to match this target.

Fisher information estimation. For FGQ, the Fisher information is estimated once before calibration and then kept fixed throughout calibration. We use the same Co3Dv2 sampling configuration as calibration: 64 clips, 4 frames per clip, image size 518, and random seed 42. The Fisher-estimation clips and calibration clips are generated independently by the same deterministic sampling procedure, rather than being explicitly saved and reused by identity.

Our implementation uses a diagonal activation Fisher approximation rather than a parameter Fisher. Hooks are registered on the outputs of the VGGT aggregator frame and global blocks. For VGGT, this gives 48 calibrated blocks. We estimate Fisher information for three output tasks: camera, depth, and world points, corresponding to `pose_enc`, `depth`, and `world_points`. For each task, we use the sum of the task output as the scalar objective and accumulate squared gradients with respect to the hooked block activations. For task k, block ℓ, and channel c, the raw Fisher estimate is

$$F_{k,\ell,c} = \frac{1}{N} \sum_{n=1}^{N} \sum_{t} \left(\frac{\partial \mathcal{L}_k}{\partial h_{\ell,t,c}^{(n)}} \right)^2,$$

Table 6: Calibration overhead and end-task performance comparison. Numbers in red indicate the change of FGQ relative to the Baseline PTQ (FlatQuant).

Method	Calibration Overhead		Performance	
	GPU memory (GB)	GPU time (hours)	AUC@15 (Co3Dv2)↑	DTU Acc. mean↓
FP16	–	–	0.9462	1.308
Baseline PTQ (FlatQuant)	3.54	0.87	0.8825	1.537
FGQ (Ours)	**3.62**$_{+0.08}$	**0.92**$_{+0.05}$	**0.8952**$_{+0.0127}$	**1.446**$_{-0.091}$

where $h^{(n)}_{\ell,t,c}$ is the activation of channel c at token t in block ℓ for sample n, and $\mathcal{L}_k$ is the scalar objective for task k. The Fisher tensor therefore has shape $3 \times 48 \times 1024$, corresponding to tasks, blocks, and channels. Forward passes for Fisher estimation use CUDA bfloat16 autocast, while gradients are cast to float32 before accumulation. The saved Fisher tensor is stored in float32.

Before being used in FGQ calibration, the raw Fisher tensor is normalized. First, each task Fisher is divided by its global mean. The normalized task Fishers are then summed with equal weight. Finally, the resulting per-block Fisher weights are normalized so that each block has mean weight 1, and the minimum weight is clamped to 0.01. No clipping or damping is applied to the raw Fisher values before this loading-time normalization.

FGQ calibration loss. FlatQuant uses an unweighted block-output reconstruction loss,

$$\mathcal{L}_{\text{FlatQuant}} = \text{MSE}\left(\mathbf{y}^q_\ell, \mathbf{y}^{\text{fp}}_\ell\right),$$

where $\mathbf{y}^q_\ell$ and $\mathbf{y}^{\text{fp}}_\ell$ denote the quantized and full-precision outputs of the current block ℓ, respectively. FGQ replaces this loss with a Fisher-weighted reconstruction loss,

$$\mathcal{L}_{\text{FGQ}} = \frac{1}{|\mathcal{T}|C} \sum_{t\in\mathcal{T}} \sum_{c=1}^{C} w_{\ell,c} \left(y^q_{\ell,t,c} - y^{\text{fp}}_{\ell,t,c}\right)^2,$$

where $w_{\ell,c}$ is the fixed Fisher-derived weight for channel c in block ℓ, $\mathcal{T}$ is the set of tokens, and C is the channel dimension. Calibration proceeds block by block in the order

$$\texttt{frame_0}, \texttt{global_0}, \ldots, \texttt{frame_23}, \texttt{global_23}.$$

Before backpropagation, the scalar loss is divided by its detached value. This normalization preserves the gradient direction while stabilizing the gradient scale.

C Calibration Overhead

All calibration experiments are conducted on a single NVIDIA A100 40GB GPU. In a representative W4A4 run, FlatQuant takes 52.53 minutes for calibration, while FGQ takes 55.54 minutes. The corresponding peak GPU memory usage is 3.54 GB for FlatQuant and 3.62 GB for FGQ. Compared with FlatQuant, FGQ increases the calibration time by 3.01 minutes, corresponding to a relative overhead of approximately 5.73%, and increases the peak GPU memory usage by only 0.08 GB, corresponding to approximately 2.26%. Because Fisher estimation uses only 64 clips with 4 frames per clip and is performed once, its cost remains small compared with model training. The detailed calibration memory usage and computation time are given in Table 6.

D FGQ Evaluation Result on π^3

We further extend our method to π^3 [25], the most recent feed-forward 3D reconstruction model. Table 7 summarizes FGQ's performance on π^3.

On camera pose estimation, RTN collapses under 4/4 quantization, with AUC@3 on Co3Dv2 dropping from 0.5140 (FP16) to 0.0246, showing that π^3 is highly sensitive to quantization error. FlatQuant recovers a large portion of this gap but still suffers noticeable degradation at tight thresholds such as AUC@5 and AUC@3. FGQ consistently improves over FlatQuant on every threshold of both

Table 7: π^3 [25] evaluation results. Top: camera pose estimation on Co3Dv2 [16] and Re10K [32] (AUC, higher is better). Bottom: point map reconstruction on DTU [7] and ETH3D [17] (Acc and Comp lower is better; N.C. higher is better). Best results in **bold**.

Method	Bitwidth (W/A)	Co3Dv2				Re10K			
		AUC@30↑	AUC@15↑	AUC@5↑	AUC@3↑	AUC@30↑	AUC@15↑	AUC@5↑	AUC@3↑
FP16	16/16	0.9441	0.8882	0.6811	0.5140	0.8907	0.8212	0.6511	0.5523
RTN	4/4	0.7684	0.5491	0.0954	0.0246	0.5662	0.4040	0.1538	0.0740
FlatQuant	4/4	0.9119	0.8240	0.5022	0.2686	0.8298	0.7317	0.5204	0.4142
FGQ	**4/4**	**0.9394**	**0.8795**	**0.6711**	**0.5032**	**0.8436**	**0.7502**	**0.5517**	**0.4461**

Method	Bitwidth (W/A)	DTU						ETH3D					
		Acc.↓		Comp.↓		N.C.↑		Acc.↓		Comp.↓		N.C.↑	
		mean	med	mean	med	mean	med	mean	med	mean	med	mean	med
FP16	16/16	1.138	0.606	1.926	0.624	0.662	0.747	0.188	0.123	0.211	0.128	0.884	0.971
RTN	4/4	7.744	5.814	19.290	17.076	0.585	0.630	1.353	1.263	2.030	1.647	0.589	0.628
FlatQuant	4/4	1.396	0.745	1.917	0.622	0.662	0.746	0.197	0.134	0.217	0.145	0.870	0.959
FGQ	**4/4**	**1.358**	**0.726**	**1.878**	**0.599**	**0.663**	**0.747**	**0.193**	**0.130**	**0.215**	**0.131**	**0.874**	**0.961**

Co3Dv2 and Re10K, with the largest gains observed at tighter thresholds. For example, on Co3Dv2 FGQ raises AUC@3 from 0.2686 (FlatQuant) to 0.5032, approaching the FP16 result of 0.5140.

For point map reconstruction, FGQ outperforms FlatQuant on all metrics across DTU and ETH3D, and its results closely track the FP16 baseline (e.g., DTU Acc. mean 1.358 vs. FP16's 1.138, ETH3D N.C. mean 0.874 vs. FP16's 0.884). Overall, these results demonstrate that FGQ preserves π^3's 3D geometric reasoning quality under aggressive 4-bit quantization while incurring minimal calibration overhead.

E Limitations

FGQ has several limitations that we leave for future work. First, our experiments focus on standard INT4 weight-activation quantization. We have not yet studied how Fisher-guided calibration interacts with newer low-precision formats, such as MXFP4 or other floating-point 4-bit variants. Since these formats have different dynamic ranges and rounding behavior, they may benefit from different calibration schedules or Fisher weighting strategies.

Second, FGQ is evaluated with a fixed calibration protocol. We use the same calibration data scale and preprocessing across experiments to isolate the effect of Fisher-guided weighting. We do not attempt to optimize the calibration set construction, such as the category balance, scene diversity, or view distribution. Studying how to choose the most informative calibration samples for feed-forward 3D reconstruction is an interesting direction, but is orthogonal to the calibration objective proposed in this work.

F Impact Statement

This work advances the practical deployment of feed-forward 3D reconstruction models by substantially reducing their memory and computational overhead through post-training quantization, enabling on-device 3D perception for applications such as augmented and virtual reality, robotics, autonomous driving, and assistive technologies. By lowering hardware requirements, FGQ helps democratize access to advanced 3D vision capabilities for researchers and developers with limited resources, while also reducing the energy consumption and carbon footprint associated with running billion-parameter models.

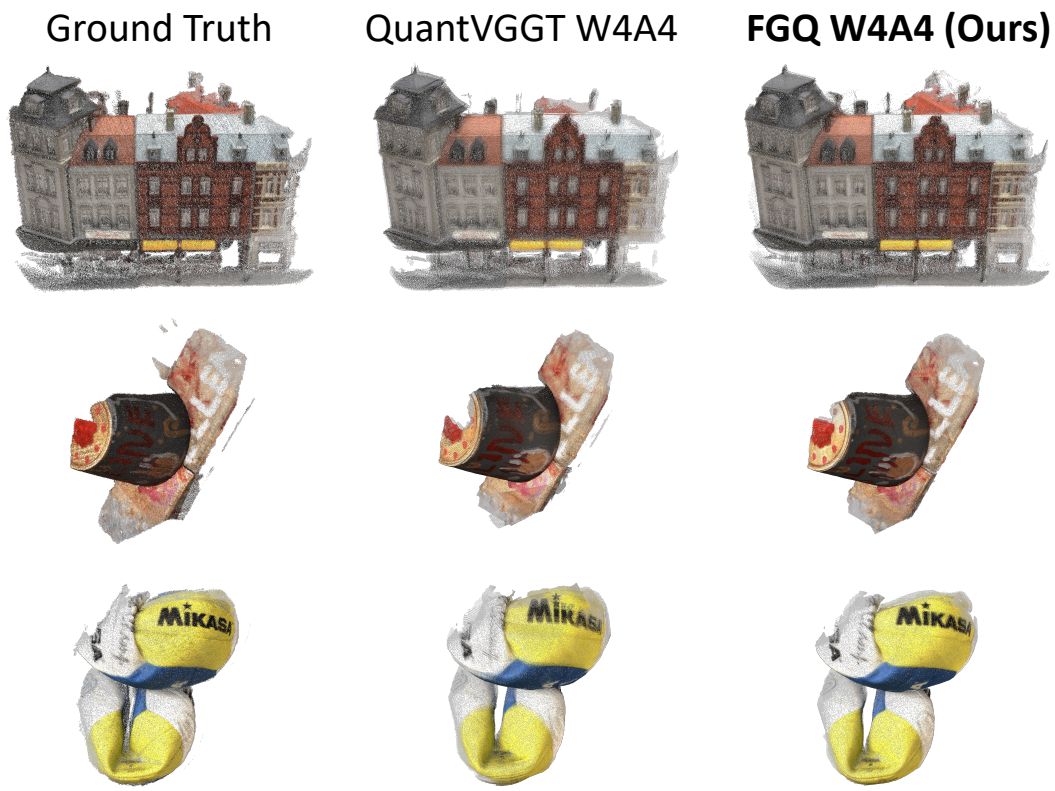


Figure 6: Visual comparison results for VGGT on DTU [7] for Ground Truth, QuantVGGT [3] W4A4, and FGQ W4A4.

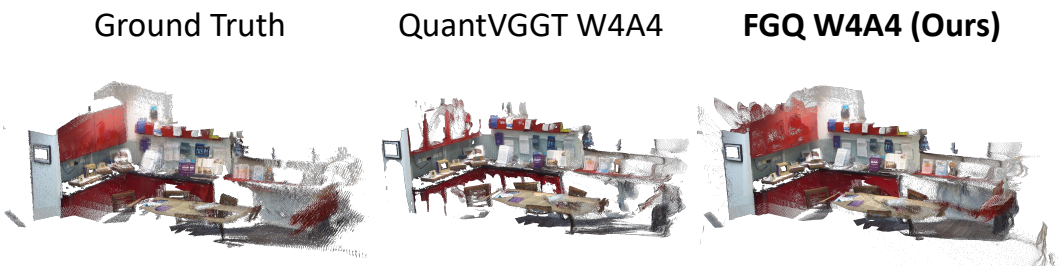


Figure 7: Visual comparison results for VGGT on 7-Scenes [20] for Ground Truth, QuantVGGT [3] W4A4, and FGQ W4A4.

G More Visualization Results

Here we exhibit more visualization result for VGGT [24] on DTU [7] (Fig. 6) and 7-Scenes [20] (Fig. 7).

Moreover, we provide the visualization result for π^3 [25] on 7-Scenes [20] in Fig. 8 to show FGQ's improvement over RTN.

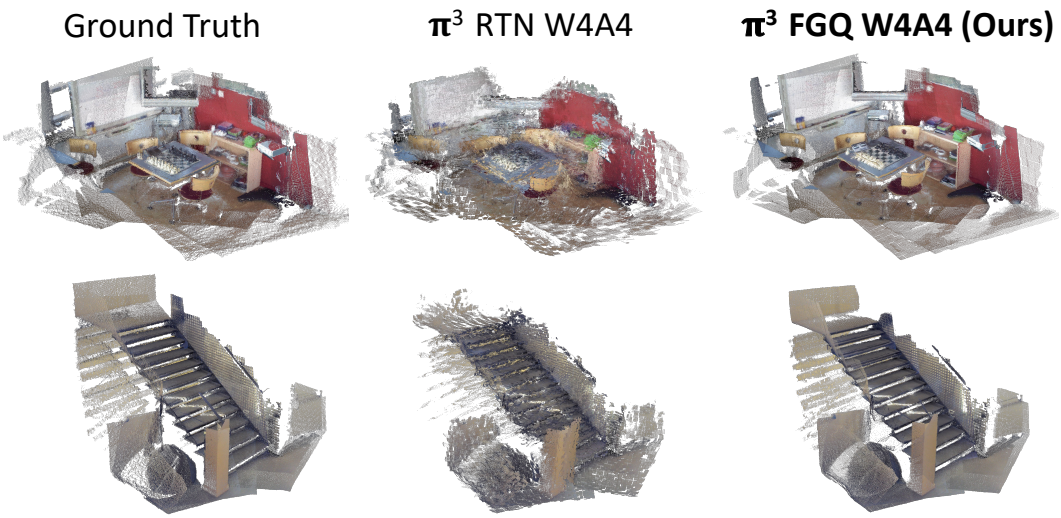


Figure 8: Visual comparison results for π^3 on 7-Scenes [20] for Ground Truth, RTN W4A4, and FGQ W4A4.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: This paper presents **FGQ**, a task-sensitivity aware PTQ method for feed forward 3D reconstruction. The claims in the abstract and introduction are well aligned with our key observation and our proposed solution to effectively address the identified problem.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of our work in Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.

- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: We provide the theoretical proof of the adopted proposition in Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All the details are provided in Sec. 4 and Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed

instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We offer the code in supplemental material, and the code will be publicly available after the paper acceptance.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide the details in Sec. 4 and Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We are following the standard evaluation method adopted by mainstream feed-forward 3D reconstructions, where the error bar has not been used.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We introduce the related information in Sec. 4.4 and Appendix C.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our research follows the NeurIPS Code of Ethics.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We fully discuss the potential social impacts in Appendix F.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: This paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All datasets used are from publicly available sources. They are released under open-access or research licenses, which are respected and cited in our paper.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: This paper does not release new assets.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: This paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLM is used only for polishing and grammar checking.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.