TTB: Test-time Baking for Efficient Rendering of Decoder-only View Synthesis Models

Chin-Yang Lin[1,2*] **Ryo Hachiuma**[1] **Jaesung Choe**[1] **Min-Hung Chen**[1]
Yu-Chiang Frank Wang[1] **Yen-Yu Lin**[2] **Wei-Chen Chiu**[2] **Yu-Lun Liu**[2] **Cheng Sun**[1]
[1]NVIDIA [2]Department of Computer Science, National Yang Ming Chiao Tung University

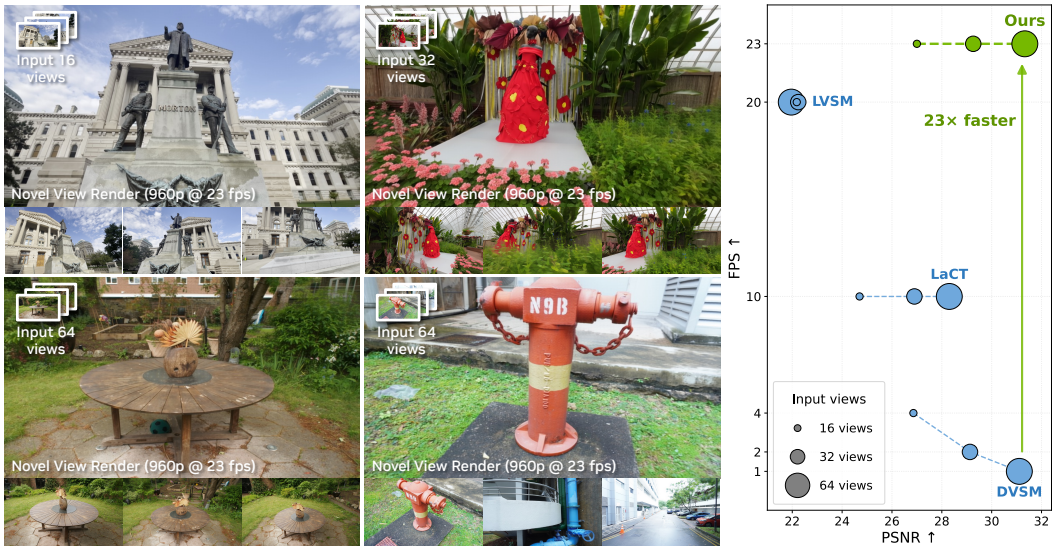

Figure 1: **T**est-**T**ime **B**aking (**TTB**) is a feed-forward, generalizable view synthesis model that bakes scene information from cross-view attention into a fast-weight MLP, achieving constant-complexity rendering with quality on par with decoder-only architectures. TTB sits on the Pareto frontier of PSNR vs. FPS, rendering up to 23× faster than cross-attention baseline at comparable quality.

Abstract

Decoder-only Large View Synthesis Models (LVSMs) with KV-cache have recently achieved state-of-the-art quality by regressing novel views from neural networks without reconstructing 3D geometry, while Test-Time Training (TTT) layers save computation but sacrifice quality. We propose **T**est-**T**ime **B**aking (**TTB**), a strategy that learns to bake global scene information from cross-view attention into a lightweight MLP, enabling fast and high-fidelity novel-view decoding. The first key is to repurpose the fast-weight MLP of TTT to learn the cross-view attention mapping explicitly from its input-output pairs, rather than replacing attention entirely, achieving the same rendering efficiency as TTT while delivering substantially higher quality. The second key addresses a modality misalignment between context prefilling and novel-view rendering: we disentangle context tokens into camera-only context rendering tokens to serve as TTB input, and context ground-truth tokens with image content to serve as the TTB mapping target, ensuring the baked MLPs are relieved of the modality generalization challenge between context and novel-view processing. Our method achieves up to **23x** rendering speedup with quality on par with the state-of-the-art across several benchmarks. Project page: `https://test-time-baking.github.io`.

*This work was conducted during an internship at NVIDIA.

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

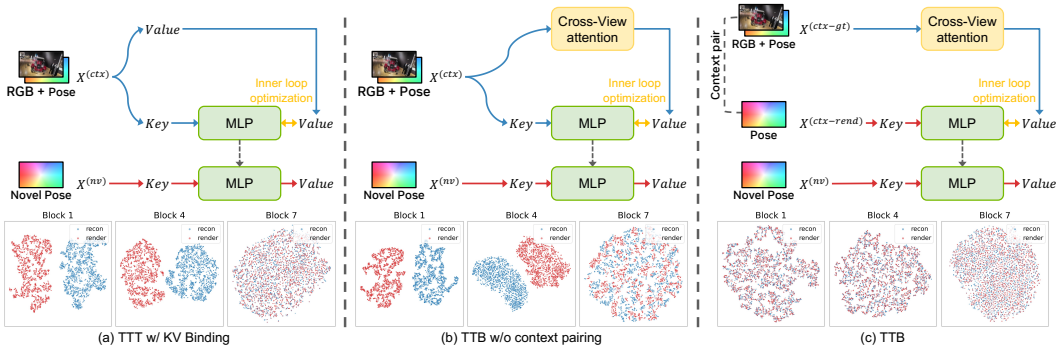

Figure 2: **Motivation.** We show three variants of the inner-loop optimization in TTT-style fast-weight MLPs. **(a)** TTT with KV binding [77] learns from raw value tokens projected from $X^{(\text{ctx})}$. **(b)** TTB without context pairing targets the cross-view attention output instead, collaborating with attention rather than replacing it. **(c)** Our full TTB additionally pairs each context view: $X^{(\text{ctx-gt})}$ carries image content and supplies the cross-view attended target, while $X^{(\text{ctx-rend})}$ is camera-only and matches the modality of $X^{(\text{nv})}$. The t-SNE plots show that only (c) aligns the modalities from the first block.

1 Introduction

Our research studies the task of novel-view synthesis conditioned on a set of context images with the powerful Large View Synthesis Models (LVSMs) [25]. LVSMs have recently emerged with several immediate follow-up improvements [22, 31, 58, 69, 77], achieving state-of-the-art novel-view synthesis quality across various input conditions [1, 23, 42, 64]. Their geometry-free scene representations and Transformer-based renderers are fully learning-based, making them favorable for scaling with increasing amounts of training data. The minimized 3D inductive bias also shows advantages when scenes are challenging for existing Transformer-compatible geometry-based representations (e.g., 3DGS [29]), such as surfaces with strong mirror reflections or translucent material.

We are especially interested in the decoder-only variant of LVSM [25] for its high-fidelity. DVSM [58] further introduces KV-cache to the cross-view attention layers, showing both quality and efficiency improvement. The KV-cache serves as a global scene state for novel-view queries, but the rendering time is still impractical under dense context view conditions due to the linear complexity per token when attending to the KV-cache. LaCT [77] takes a different approach by replacing cross-view attention layers with Test-Time Training (TTT) [59] layers, as illustrated in Fig. 2a. TTT performs inner-loop optimization to adapt fast-weight MLPs [49] to directly map queries to their values without the quadratic-complexity attention, which can also be interpreted as a variant of linear attention [36]. Despite achieving constant-complexity rendering speed, TTT significantly sacrifices rendering quality. A naive remedy for LaCT is to initialize from pretrained decoder-only model weights, which does improve quality, but the gap to the pretrained model remains significant.

We argue that this quality gap can be minimized by repurposing the fast-weight MLP inner-loop optimization as "baking" the information from cross-view attention, rather than replacing cross-view attention entirely as in TTT. We dub our strategy Test-Time Baking (TTB), illustrated in Fig. 2b. The key motivation is to trade context-view processing time for faster and higher-fidelity novel-view rendering, achieving the Pareto frontier of quality and speed shown in Fig. 1. Specifically, the inner-loop learning target becomes the output tokens from cross-view attention, instead of the raw value tokens used in TTT. This formulation also generalizes beyond the linear-attention interpretation of TTT [36]. In principle, the learning target of our TTB can also be tokens from arbitrarily non-linear layers or tokens enriched with additional scene information.

Furthermore, we identify a *modality misalignment* issue inherent to the context prefilling of MLP fast-weight for novel-view synthesis. In the reconstruction stage (context prefilling), the fast-weight MLPs are updated using context tokens with both camera and image information. In the rendering stage, however, the updated MLPs take novel-view tokens coming from camera-only input without any image content. This conventional pipeline poses a reconstruction-rendering input modality misalignment challenge.

Our solution to modality misalignment is simple yet effective. As illustrated in Fig. 2c, we pair each context view into *i)* context ground-truth tokens, derived from input cameras and images that produce the global scene information to be baked, and *ii)* context rendering tokens, derived from camera-only input that simulates novel-view rendering conditions. At each TTB layer, the fast-weight MLP is optimized to map context rendering tokens to their corresponding context ground-truth tokens, which carry image and cross-view attended information. The optimized TTB-MLP is then shared between context rendering tokens and the downstream novel-view rendering tokens for retrieving scene information, ensuring that their conditioning remains identical across all layers.

Interestingly, TTB draws a natural parallel with traditional implicit radiance and light field reconstruction, where MLPs are optimized to map spatial coordinates to radiance values. In TTB, the fast-weight MLP instead is optimized to map ray-patch latent coordinates to their appearance latent representations, which effectively performs few-step light-field optimization in latent space. This perspective suggests that TTB interpolates appearance latents for novel viewpoints from their ray-patch latent coordinates, motivating a convergence-oriented inner loop over one-shot summarization. In summary, our contributions are as follows:

- We propose Test-Time Baking, which retargets the inner-loop optimization of the TTT fast-weight MLP from raw value tokens to cross-view attention outputs, so that the fast-weight MLP collaborates with attention rather than replacing it.
- We identify a modality misalignment between the reconstruction and rendering stages of MLP fast-weight prefilling, and resolve it via a context pair that disentangles each context view into a ground-truth and a camera-only rendering token.
- We draw a connection between TTB's inner loop and light field optimization in latent space, which motivates a convergence-oriented training recipe (multi-step MSE) that we find particularly effective for TTB.

2 Related work

Per-Scene optimization. Early approaches to novel-view synthesis perform per-scene optimization, fitting differentiable scene representations to observed images via differentiable rendering. Coordinate-based MLPs [2, 4, 41], voxels [8, 15, 34, 43, 56, 57], and the landmark 3DGS [29] as the scene representation have been widely explored. Despite impressive quality, per-scene optimization does not scale with the increasing training scenes and degrades quickly under fewer input views.

Geometry-based feed-forward models. To overcome the scalability limitations and lengthy procedure of per-scene optimization, learning-based methods train networks to predict 3D representations from images. Early works estimated multi-plane images via cost volumes [16, 17, 40, 68, 72, 73], while NeRF-based generalizable methods [7, 37, 67, 74] extended coverage to wide-angle viewpoints at limited quality. The advent of 3DGS [29] reframed the problem as pixel-aligned GS prediction [6, 10, 24, 60, 62, 71], achieving promising results, and Large Reconstruction Models (LRMs) [21] further demonstrated Transformer scalability across various 3D representations [5, 70, 76]. One fundamental limitation of handcrafted 3D is their degradation on surfaces or materials they are not designed for, whereas the geometry-free representations we study can learn them given data. Furthermore, explicit 3D geometry lacks a compact latent representation, and efforts to manage their linearly growing output size under more input views [11, 12, 45, 63] remain a symptom of this constraint. Our TTB bakes scenes into constant-size MLPs at higher quality.

Geometry-free feed-forward models. An alternative paradigm sidesteps the limitations of geometry-based models by representing scenes as latent representations and rendering via neural networks. Early methods [13, 46, 47, 53, 54, 55] were constrained by limited network capacity, while Large View Synthesis Models (LVSMs) [25] broke this barrier by leveraging powerful Transformers. Encoder-decoder and decoder-only LVSM variants are explored. The encoder-decoder LVSM [25] encodes context views into scene tokens and decodes novel views conditioned on the encoded scene, with subsequent works refining scene encoding strategies [22, 31, 69] and extending to generative [1] and uncalibrated camera [23, 42, 64, 78] settings. The decoder-only LVSM [25], on the other hand, uses a single network for both context prefilling and novel-view rendering. Despite higher fidelity, its original form inefficiently processes context views independently for each novel view. DVSM [58] addresses this by prefilling context views into a KV-cache shared across all novel views, achieving

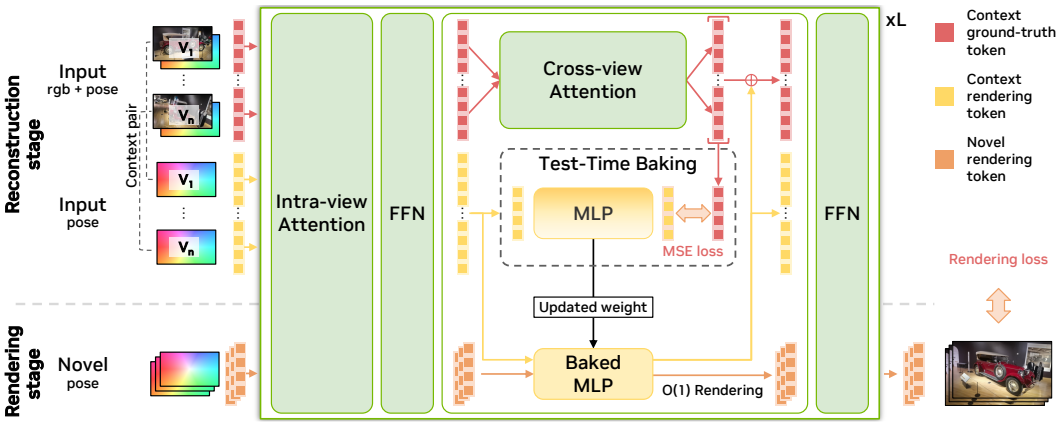

Figure 3: **Pipeline.** Our model stacks L identical blocks of Intra-view Attention, FFN, Test-Time Baking, and FFN. In the reconstruction stage, each context view is duplicated into a context pair, where the context ground-truth token $X^{(\text{ctx-gt})}$ carries RGB and pose while the context rendering token $X^{(\text{ctx-rend})}$ is pose-only and matches the modality of the novel rendering token $X^{(\text{nv})}$. At each TTB layer, the inner-loop *update* bakes the fast-weight MLP via MSE loss to map $X^{(\text{ctx-rend})}$ toward the cross-view attention output of $X^{(\text{ctx-gt})}$. The baked MLP is *applied* to serve three token streams: it propagates both $X^{(\text{ctx-rend})}$ and $X^{(\text{nv})}$ to the next layer, and provides feedback to $X^{(\text{ctx-gt})}$ as an auxiliary signal. In the novel-view rendering stage, each token retrieves global scene information in $O(1)$ from the baked MLP, instead of $O(N)$ via cross-attention. The rendering loss is backpropagated through the inner-loop baking process to train the model end-to-end.

state-of-the-art quality under both sparse and dense view settings. However, the linearly growing KV-cache size makes rendering speed impractical for larger numbers of input views. LaCT [77] replaces cross-view attention with Test-Time Training layers to achieve constant rendering complexity, but at a significant quality cost compared to DVSM. Our Test-Time Baking achieves the same constant rendering complexity while substantially closing the quality gap left by LaCT.

Test-time Training layers. TTT [59] has recently emerged as a popular alternative to attention for its linear time and constant space complexity. We refer interested readers to [36] for an in-depth review and analysis. We are particularly interested in the variant presented in LaCT [77], which has demonstrated success in accelerating various vision and 3D tasks [9, 14, 18, 26, 63], albeit at the cost of quality relative to their attention-based counterparts. Our TTB departs from this paradigm in two ways: rather than replacing attention, TTB learns directly from attention outputs; and rather than performing mere token-to-token information exchange, TTB explicitly learns to map camera-ray tokens to their corresponding image tokens, yielding improved novel-view synthesis quality.

3 Method

We start by formally introducing our task and the general pipeline in Sec. 3.1. We detail the decoder-only architecture we build on in Sec. 3.2. Finally, we present Test-Time Baking in Sec. 3.3.

3.1 Overview

Given a set of V context images with known poses and intrinsics $\{\mathbf{I}_i \in \mathbb{R}^{H\times W\times 3}, \mathbf{P}_i \in \mathbb{SE}(3), \mathbf{K}_i \in \mathbb{R}^{3\times 3}\}_{i=1}^{V}$, our goal is to synthesize novel views for an arbitrary camera query $(\mathbf{P}', \mathbf{K}')$. We focus on reconstruction rather than generation, assuming novel views can be directly regressed from the context views. Approaches to this end generally decompose into two stages:

$$S = \text{Recon}(\mathbf{I}, \mathbf{P}, \mathbf{K}), \quad \bar{\mathbf{I}} = \text{Rend}(\mathbf{P}', \mathbf{K}', S), \tag{1}$$

where $\text{Recon}(\cdot)$ processes context views into a method-specific scene representation S, and $\text{Rend}(\cdot)$ synthesizes the novel view $\bar{\mathbf{I}}$ conditioned on S. Recent feed-forward models instantiate $\text{Recon}(\cdot)$ as a Transformer. The Large Reconstruction Model (LRM) [21] family employs a human-defined 3D and renderer, while the Large View Synthesis Model (LVSM) [25] series defines S as a latent state and

implements $\mathrm{Rend}(\cdot)$ as a neural network. As both $\mathrm{Recon}(\cdot)$ and $\mathrm{Rend}(\cdot)$ are differentiable, models are trained end-to-end against ground-truth novel views $\mathbf{I}'$ by:

$$\mathcal{J} = \mathrm{MSE}(\bar{\mathbf{I}}, \mathbf{I}') + \lambda \cdot \mathrm{Percep}(\bar{\mathbf{I}}, \mathbf{I}'), \tag{2}$$

where Percep is the neural perceptual loss [27] and λ balances the photometric objectives.

3.2 Decoder-only View Synthesis Model

Our method builds on the Decoder-only View Synthesis Model (DVSM) [58], which uses a shared Transformer decoder for both $\mathrm{Recon}(\cdot)$ and $\mathrm{Rend}(\cdot)$, and represents the scene S as KV caches. Specifically, the model consists of L repeated blocks, each comprising a sequence of layers:

$$\mathbf{X}_\ell = (\mathrm{FFN}_{\ell,2} \circ \mathrm{CrossAttn}_\ell \circ \mathrm{FFN}_{\ell,1} \circ \mathrm{IntraAttn}_\ell)(\mathbf{X}_{\ell-1}), \tag{3}$$

where $\mathrm{IntraAttn}(\cdot)$ and $\mathrm{CrossAttn}(\cdot)$ perform intra-view and cross-view attention, respectively, and $\mathrm{FFN}(\cdot)$ denotes a fully connected layer. Residual connections [19] and QK-normalization [20] are applied for training stability. During reconstruction, context-view tokens are prefilled into a KV-cache at each $\mathrm{CrossAttn}(\cdot)$. During rendering, novel-view tokens query the cached KV to retrieve global scene information from each $\mathrm{CrossAttn}(\cdot)$ layer.

Regarding input tokenization, camera intrinsics and poses are first converted into Plücker ray maps [44] for linear patch embedding. Context-view and novel-view tokens are initialized as:

$$\mathbf{X}_0^{(\mathrm{ctx})} = \mathrm{LayerNorm}(\mathrm{PE}^{(\mathrm{ray})}(\mathbf{P}, \mathbf{K}) + \mathrm{PE}^{(\mathrm{rgb})}(\mathbf{I})), \quad \mathbf{X}_0^{(\mathrm{nv})} = \mathrm{LayerNorm}(\mathrm{PE}^{(\mathrm{ray})}(\mathbf{P}', \mathbf{K}')), \tag{4}$$

where $\mathrm{PE}^{(\mathrm{ray})}$ and $\mathrm{PE}^{(\mathrm{rgb})}$ are linear patch projections for the Plücker ray maps and context images, respectively, Here, $\mathbf{X}_0^{(\mathrm{ctx})} \in \mathbb{R}^{V \times \frac{H}{p} \times \frac{W}{p} \times D}$ denotes tokens from V context views with patch size p used for KV-cache prefilling, and $\mathbf{X}_0^{(\mathrm{nv})} \in \mathbb{R}^{U \times \frac{H}{p} \times \frac{W}{p} \times D}$ denotes tokens for U novel-view queries, each processed independently. Finally, a shallow model head regresses pixel values as:

$$\bar{\mathbf{I}} = (\mathrm{PixShuf} \circ \mathrm{Sigmoid} \circ \mathrm{Linear} \circ \mathrm{LayerNorm})(\mathbf{X}_L^{(\mathrm{nv})}), \tag{5}$$

where $\mathrm{PixShuf}(\cdot)$ denotes pixel shuffling [51] and $\mathbf{X}_L^{(\mathrm{nv})}$ is the last-layer tokens out of Eq.3.

3.3 Test-time Baking

We present our technical contribution to improve the rendering-time efficiency of DVSM [58] while retaining its high rendering quality. The issue of DVSM is that its KV-cache size grows linearly with the number of context tokens, making it slow and impractical for novel-view queries under high-resolution or dense input context views. A naive strategy is to replace each $\mathrm{CrossAttn}(\cdot)$ in Eq. 3 with a Test-Time Training (TTT) [59] layer, achieving constant space and linear time complexity as demonstrated in LaCT [77]. However, we observe significant quality degradation even when finetuning the TTT variant from DVSM pretrained weights. This motivates us to explore a better solution that improves upon TTT. A full model pipeline is illustrated in Fig. 3.

3.3.1 Repurposing TTT: collaborating instead of replacing attention

Our motivation is to trade reconstruction (prefilling) computation, which only needs to be performed once per scene, for improved rendering-time speed and quality. The resulting solution is simple yet effective. Rather than entirely replacing cross-view attention with TTT fast-weight MLP as in the LaCT [77], we repurpose TTT to explicitly learn from the input-output pairs out of cross-view attention. Consequently, we no longer need to cache the KV; only the updated MLP fast weights are stored. The space and time complexity of novel-view rendering is therefore identical to that of TTT. We call our strategy Test-Time Baking (TTB), which can be viewed as a generalized form of TTT.

Formally, the fast-weight update rules of TTT and TTB are:

$$\theta^{(\mathrm{TTT})} = \theta - \eta \nabla_\theta \mathcal{L}(f_\theta(\mathbf{W}_k \mathbf{X}), \mathbf{W}_v \mathbf{X}), \tag{6a}$$

$$\theta^{(\mathrm{TTB})} = \theta - \eta \nabla_\theta \mathcal{L}(f_\theta(\mathbf{W}_k \mathbf{X}), g(\mathbf{X}^*)), \tag{6b}$$

respectively, where θ is the fast weights of the MLP $f(\cdot)$, η and $\mathcal{L}$ are the inner-loop learning rate and objective, $\mathbf{W}_k$ and $\mathbf{W}_v$ are linear projection matrices, and $\mathbf{X}$ is the input tokens. The difference

between TTT and TTB lies in the mapping target: $\mathbf{W}_v\mathbf{X}$ and $g(\mathbf{X}^*)$. In principle, we can employ an arbitrary function g or tokens $\mathbf{X}^*$ with additional modality for the mapping target $g(\mathbf{X}^*)$. Note that TTB falls back to TTT when $\mathbf{X}^*{=}\mathbf{X}$ and g is a linear projection. In our case, we instantiate $g(\cdot)$ as the $\mathrm{CrossAttn}(\cdot)$ from DVSM, which shifts TTB beyond the linear-attention interpretation [36] of TTT. Although this incurs additional reconstruction-time cost, once prefilled, rendering remains identical to TTT with substantially higher quality thanks to the enriched global information from $g(\cdot)$.

More analysis. We empirically validate that swapping the inner-loop target from $\mathbf{W}_v\mathbf{X}$ to $\mathrm{CrossAttn}(\mathbf{X})$ broadens the *effective view field* of the baked fast-weight MLP. At TTB block 7 with $V{=}16$, we measure how much each context image contributes to the TTB output at the novel-view rendering tokens via input-gradient saliency, sorted and averaged across output positions (Fig. 4). The $\mathbf{W}_v\mathbf{X}$ variant peaks sharply at top-1 then decays, while $\mathrm{CrossAttn}(\mathbf{X})$ flattens the peak and sustains a heavier mid-rank tail. This matches our design intent: $\mathrm{CrossAttn}(\mathbf{X})$ already aggregates across all V context views, so each memorized pair inherits a wider receptive field.

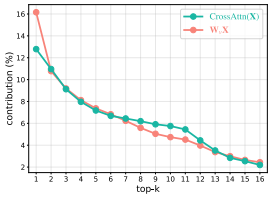

Figure 4: **Per-rank context contribution.**

3.3.2 Modality misalignment and context pair

The baked MLPs via Eq. 6b receive inputs of different modalities between the reconstruction and rendering stages. As defined in Eq. 4, context tokens $\mathbf{X}_0^{(\text{ctx})}$ used for prefilling contain image content, whereas novel-view tokens $\mathbf{X}_0^{(\text{nv})}$ are with camera-only information. This *modality misalignment* poses an unnecessary generalization challenge to the baked MLP.

We address this by disentangling input context tokens, $\mathbf{X}_0^{(\text{ctx})}$ from Eq.4, into a paired representation:

$$\mathbf{X}_0^{(\text{ctx-gt})} = \text{LayerNorm}(\text{PE}^{(\text{ray})}(\mathbf{P},\mathbf{K}) + \text{PE}^{(\text{rgb})}(\mathbf{I})),\ \mathbf{X}_0^{(\text{ctx-rend})} = \text{LayerNorm}(\text{PE}^{(\text{ray})}(\mathbf{P},\mathbf{K})), \quad (7)$$

where $\mathbf{X}_0^{(\text{ctx-rend})}$ are the *context rendering tokens* with camera-only input, and $\mathbf{X}_0^{(\text{ctx-gt})}$ are the *context ground-truth tokens* with additional RGB input. We use $\mathbf{X}^{(\text{ctx-gt})}$ to serve TTB mapping target while using $\mathbf{X}^{(\text{ctx-rend})}$ to serve TTB input. Specifically, at each TTB layer, the fast-weight MLP is baked to learn the mapping $\mathbf{X}^{(\text{ctx-rend})} \mapsto \mathrm{CrossAttn}(\mathbf{X}^{(\text{ctx-gt})})$ via the update rule in Eq. 6b. The baked MLP is then shared across both context rendering $\mathbf{X}^{(\text{ctx-rend})}$ and the downstream novel-view rendering $\mathbf{X}^{(\text{nv})}$ tokens to produce feature for the subsequent layers. As visualized in Fig. 2, our context pairing ensures distribution alignment from the early layers, which facilitates feature interpolation. Conversely, the distribution only aligned in the later layers without context pairing due to the modality misalignment between reconstruction and rendering stages. Despite doubling the number of tokens during reconstruction, rendering-stage efficiency remains the same, which aligns with our motivation to trade reconstruction time for rendering-time quality.

3.3.3 TTB layer details

The design of our TTB layer is inspired by the TTT layer in LaCT [77]. We remove the asymmetric QK projection and instead use a shared linear projection to produce queries and keys. We adopt SwiGLU-MLP [50] as the fast-weight MLP architecture, and apply an additional linear layer to predict per-token learning rates. For inner-loop optimization, we employ Muon [28] optimizer with weight normalization [48] for a predefined T steps. We find that using MSE as the inner-loop objective for our TTB yields better results than the unnormalized dot-product loss used in LaCT [77]'s TTT. We also find that learning the residual to the mapping target is easier to bake than learning the target directly. Specifically, the final TTB update rule is:

$$\theta_{t+1} = \theta_t - \text{Muon}\left(\nabla_{\theta_t} \left\| f_{\theta_t}\left(\mathbf{W}_k\mathbf{X}^{(\text{ctx-rend})}\right) - \mathbf{W}_v\left(\mathrm{CrossAttn}(\mathbf{X}^{(\text{ctx-gt})}) - \mathbf{X}^{(\text{ctx-rend})}\right)\right\|_2\right), \quad (8)$$

where $\mathrm{Muon}(\cdot)$ performs Muon gradient orthogonalization [28], $\mathbf{W}_k$ and $\mathbf{W}_v$ are linear projection matrices for the TTB input and mapping target. The final baked weights θ_T are then applied to update both the context rendering and novel-view rendering tokens and provide feedback to the context ground-truth tokens (the output arrows from the baked MLP in Fig. 3).

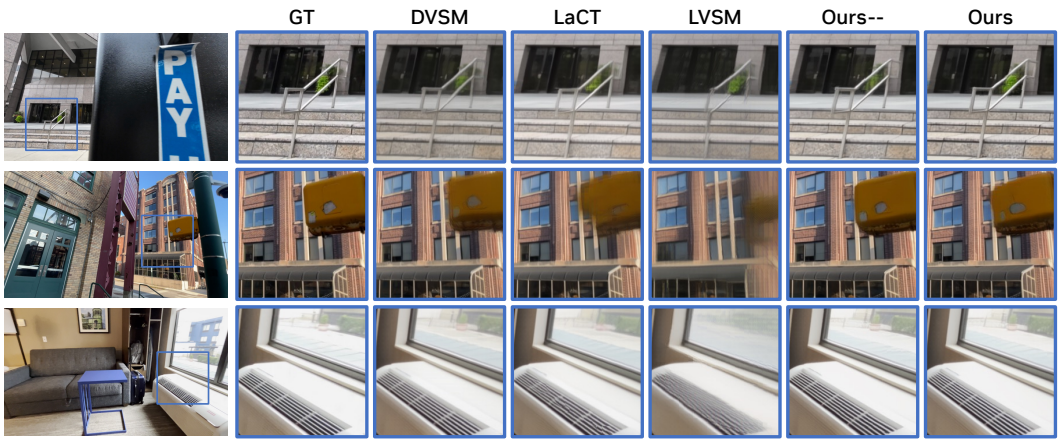

Figure 5: **Qualitative comparison on DL3DV.** TTB preserves fine details on par with DVSM [58], while LaCT [77] and LVSM [25] produce visible blur.

Table 1: **Quantitative comparison on DL3DV.**

		16 input views					32 input views					64 input views				
Method	Venue	PSNR↑	SSIM↑	LPIPS↓	Time↓	FPS↑	PSNR↑	SSIM↑	LPIPS↓	Time↓	FPS↑	PSNR↑	SSIM↑	LPIPS↓	Time↓	FPS↑
3DGS [29]	ToG'23	21.20	0.708	0.264	>600s	>50	23.60	0.779	0.213	>600s	>50	26.43	0.854	0.167	>600s	>50
LongLRM [12]	ICCV'25	22.66	0.740	0.292	0.4s	>50	24.10	0.783	0.254	1s	>50	24.77	0.804	0.239	3s	>50
LongLRM++ [11]	arxiv'25	24.40	0.795	0.231	1.6s	14	26.43	0.846	0.180	4.7s	14	27.30	0.869	0.161	16s	14
tttLRM [63]	CVPR'26	23.60	0.784	0.255	3.6s	-	25.07	0.822	0.215	7.2s	-	25.95	0.844	0.195	15s	-
LVSM [25]	ICLR'25	22.18	0.694	0.351	3.8s	20	22.24	0.691	0.352	15s	20	21.96	0.678	0.364	59s	20
LaCT [77]	ICLR'26	24.70	0.793	0.224	1.9s	10	26.90	0.837	0.185	3.7s	10	28.30	0.857	0.169	7.3s	10
DVSM [58]	arXiv'26	26.86	0.852	0.170	4.4s	4	29.13	0.893	0.129	15s	2	31.11	0.920	0.104	60s	1
Ours--		26.26	0.833	0.191	1.9s	23	28.12	0.867	0.158	3.7s	23	29.29	0.882	0.144	7.5s	23
Ours		27.00	0.849	0.177	5.4s	23	28.92	0.882	0.145	18s	23	30.20	0.897	0.131	65s	23
Ours (duplicate)		-	-	-	-	-	29.26	0.890	0.136	18s	23	31.33	0.919	0.108	65s	23

Table 2: **Two input views on Re10k at 512×512.** Results are denoted as PSNR↑ / SSIM↑ / LPIPS↓.

LVSM [25]	Decoder-only LVSM [25]	Efficient-LVSM [22]	DVSM [58]	Ours--	Ours
28.55 / 0.894 / 0.173	29.53 / 0.904 / 0.141	29.86 / 0.905 / 0.147	31.17 / 0.924 / 0.129	31.00 / 0.921 / 0.130	**31.28 / 0.926 / 0.127**

3.3.4 Fast-weight local-duplicate inference

The baked fast weights have a fixed capacity, causing quality to plateau as the number of context views V grows. We address this at inference time by clustering the context views into K spatially-local groups via k-means on per-view camera poses, with $K = V/B$ for a nominal budget of B views per group. Each group is reconstructed independently by the unmodified model, so that cross-view attention and the inner-loop update only involve the views of that group, yielding per-group, per-layer fast weights $\{\{\theta_{k,\ell}\}_{\ell=1}^{L}\}_{k=1}^{K}$. At rendering time, each novel view is routed to the group with the nearest centroid and queries only the corresponding copy. Rendering thus remains $O(1)$ per view and each copy only has to absorb about B views, while the stored state grows linearly with K.

4 Experiments

4.1 Implementation details

We use 768 latent channels with 12 decoder blocks at patch size 8, replacing each cross-view attention with a TTB layer. We initialize our models from the corresponding DVSM [58] checkpoint. We use AdamW [38] with cosine annealing and perceptual loss weight $\lambda = 0.2$. Models are trained on DL3DV-10K [35] dataset with 64 A100 GPUs. We first train at 340p for 100K iterations with learning rate 4×10^{-4}, then finetune at 480p, 720p, and 960p for 10K iterations each with learning rate 1×10^{-5}. We randomly sample either 16 or 32 context views per scene with the same number of target views. Ablations use a smaller setup on 8 A100 GPUs at patch size 16 truncated to 720p, with efficiency metrics still reported at patch size 8 for consistency. For evaluation, we use the 140 held-out DL3DV testing scenes and follow previous view sampling [12].

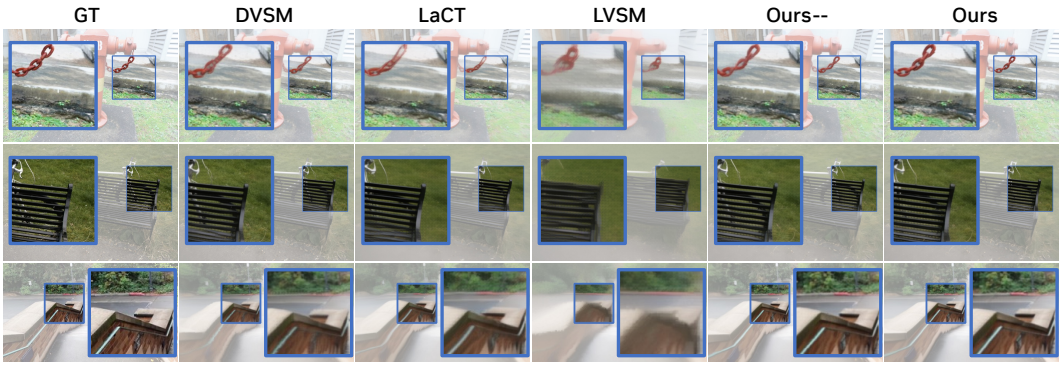

Figure 6: **Qualitative comparison on in-the-wild datasets.** TTB generalizes to out-of-domain scenes and preserves fine details consistent with the in-domain results.

Table 3: **Quantitative comparison on in-the-wild datasets.**

		MipNerf360 [3]			Free [66]			Hike [39]			Efficiency	
Method	Size	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	Time↓	FPS↑
Input: 64 views × 518,400 pixels per view (960p)												
LVSM	165M	20.07	0.459	0.559	20.27	0.517	0.444	18.60	0.433	0.502	59s	20
LaCT	379M	23.01	0.557	0.435	24.61	0.680	0.265	22.39	0.648	0.305	7.3s	10
DVSM	163M	24.76	0.660	0.337	25.57	0.756	0.205	23.10	0.697	0.262	60s	1
Ours--	244M	23.93	0.597	0.397	25.28	0.708	0.253	22.53	0.660	0.300	7.5s	23
Ours	271M	24.49	0.624	0.375	26.05	0.744	0.225	23.77	0.707	0.259	65s	23
Ours (duplicate)	271M	25.05	0.667	0.341	26.82	0.800	0.187	25.37	0.782	0.195	65s	23
Input: 64 views × 512 pixels at image longer side												
LagerNVS	1052M	20.98	0.481	0.503	19.50	0.486	0.396	18.54	0.436	0.415	2.3s	7
DVSM	163M	24.04	0.607	0.422	24.72	0.717	0.268	23.39	0.705	0.264	5.0s	12
Ours--	244M	23.18	0.552	0.478	24.27	0.665	0.323	22.41	0.646	0.295	1.8s	67
Ours	271M	23.62	0.568	0.463	24.92	0.692	0.299	23.52	0.687	0.262	6.3s	67
Ours (duplicate)	271M	24.03	0.602	0.434	25.45	0.733	0.267	24.96	0.760	0.206	6.3s	67

TTB layer. We use a single-headed fast-weight SwiGLU-MLP [50] with a hidden dimension of 3072, a $4\times$ expansion of the model dimension. The fast weight is updated with the Muon optimizer [28] and we run $T=3$ inner-loop update steps. The per-token learning rate is predicted by an additional linear head with base learning rate of 0.01. The inner-loop objective maps context-rendering tokens to the cross-view attention output of context-ground-truth tokens with the MSE loss. Fast-weight local-duplicate inference (Sec. 3.3.4) is turned off except stated otherwise. When enabled, we set B=16. For some experiments we also report a variant denoted TTB-- or Ours--, which replaces the cross-view attention target with a linear projection $W_v X^{(\text{ctx-gt})}$ and thus avoids the quadratic-complexity cross-view attention at reconstruction, making reconstruction time linear in V.

4.2 Comparisons

DL3DV. Tab. 1 and Fig. 5 report results on DL3DV with 16, 32, and 64 input views at 960p. TTB substantially closes the quality gap between DVSM [58] and LaCT [77] while rendering at a constant 23 FPS, up to $23\times$ faster than DVSM and $2.3\times$ faster than LaCT. Our model is built on DVSM with 12 blocks while LaCT uses 24 blocks (model size in Tab. 3). The Ours-- without cross-view attention is comparable to LaCT quality with less parameters and faster rendering. With local-duplicate inference (Sec. 3.3.4), Ours (duplicate) further closes the residual gap at large V and approaches DVSM quality at the same rendering speed. Fig. 5 confirms these trends qualitatively, where TTB preserves thin and high-frequency structures such as handrails, window grilles, and small text that LaCT and LVSM [25] render with visible blur.

In-the-wild. We further evaluate the generalization ability on out-of-domain datasets with 16 and 64 context views in Tab. 3. We keep the aspect ratio of the context images and resize to the resolutions preferred by the models. The synthesized novel views are resized to match ground-truth source resolutions for evaluation. Comparisons to LaCT [77] and DVSM [58] are consistent with the in-domain evaluation—our quality is often comparable to DVSM but at a much faster rendering; we use more context prefilling time than LaCT but gaining quality with $2.3\times$ faster rendering. LagerNVS [61] is a very recent work which builds upon VGGT [65] and trains on multiple datasets.

Table 4: **Ablation study from TTT to TTB on DL3DV.** (light =worst, dark =best).

Method	Quality (V=32)			Recon-time↓			Rend-fps↑		
	PSNR↑	SSIM↑	LPIPS↓	V=16	V=32	V=64	V=16	V=32	V=64
(a) = DVSM [58]	24.68	0.776	0.220	1.300	4.907	18.396	12.1	6.8	3.7
(b) = (a) + TTT [77]	23.12	0.704	0.286	0.439	0.862	1.706	49.5	49.1	49.1
TTB (Ours):									g
(c) = (b) + context pairing	23.78	0.728	0.264	0.626	1.237	2.458	50.8	50.8	50.7
(d) = (b) + cross-view attn.	24.13	0.742	0.252	1.267	5.178	17.745	50.4	50.4	50.5
(e) = (c) + MSE loss, T=3	24.22	0.747	0.248	0.994	1.939	3.828	51.0	50.6	50.6
(f) = (e) + cross-view attn.	24.84	0.768	0.230	1.998	6.251	20.996	50.9	50.4	50.5

Table 5: **Inner-loop optimization ablation.**

Framework	T	Loss	PSNR↑	SSIM↑	LPIPS↓
TTT [77]	1	dot-prod	23.12	0.704	0.286
TTT [77]	3	MSE	23.39	0.715	0.277
TTB--	1	dot-prod	23.78	0.728	0.264
TTB--	2	MSE	24.16	0.744	0.251
TTB--	3	MSE	24.22	0.747	0.248
TTB--	5	MSE	24.29	0.751	0.246

Table 6: **TTB layer design ablation.**

Feedback	Residual	PSNR↑	SSIM↑	LPIPS↓
✗	✗	24.32	0.748	0.243
✗	✓	24.75	0.764	0.233
✓	✗	24.76	0.764	0.231
✓	✓	24.84	0.768	0.230

We use our intermediate checkpoints at 480p to better align with LagerNVS's input resolution. Our quality outperforms LagerNVS by a large margin with 3.1–9.5× faster rendering.

Re10K. We additionally compare on Re10K with two low-resolution context views. This is not our main target scenario as the rendering time gain from cross-view attention becomes marginal. We finetune from the Re10K checkpoint of DVSM [58] with DINO injection. Results are in Tab. 2. Our TTB can reproduce DVSM cross-attention quality in this setup. See supplementary for full results.

4.3 Ablation studies

From TTT to TTB. Tab. 4 decomposes the gap between a vanilla LaCT-style TTT replacement [77] and our full TTB into three additive components: context pairing, switching to MSE with T=3 inner steps (yielding TTB--), and replacing the linear-projection target with cross-view attention output. Each contributes meaningfully, and stacking all three closes the quality gap to DVSM [58] while retaining TTT's $\sim$50 FPS rendering. The reconstruction cost grows quadratically with V due to the in-loop cross-view attention, consistent with our reconstruction time for rendering quality trade-off.

Inner-loop optimization. Tab. 5 ablates the inner-loop loss and number of update steps T. Switching from LaCT's unnormalized dot-product loss [77] to MSE consistently improves quality for both TTT and TTB--, with TTB-- benefiting more ($+0.44$ vs $+0.27$ PSNR). Increasing T brings monotonic but diminishing gains, leading us to adopt T=3 with MSE as a sweet spot between quality and reconstruction cost. This asymmetric response aligns with our light field view—TTB's inner loop fits a target function and benefits from convergence, while TTT-KVB is closer to a one-shot linear write.

TTB layer design. Tab. 6 ablates two design choices in our TTB layer. Residual learning reformulates the inner-loop target from the absolute cross-view attention output to its residual against $X^{(\text{ctx-rend})}$, reducing the magnitude of the regression target and easing the few-step inner-loop optimization. Feedback additionally applies the baked MLP to update the context ground-truth tokens, so that all three token streams propagate through identical operators in subsequent layers. Both designs contribute roughly equally in isolation ($+0.43$ and $+0.44$ PSNR) and combine to $+0.52$.

Limitations. TTB is constrained by the fixed capacity of its fast-weight MLP, causing quality to saturate at large V; the local-duplicate routing that mitigates this is a heuristic, and learning it during training is left for future work. Its inner-loop cross-view attention also makes reconstruction quadratic in V, a drawback for single-shot scenarios. Finally, TTB is built on reconstruction rather than generation, interpolating within context views without extrapolating to unobserved viewpoints.

5 Conclusion

We present Test-Time Baking (TTB), which retargets the TTT inner-loop optimization from raw value tokens to cross-view attention outputs, turning the fast-weight MLP from a replacement for attention into its collaborator. Combined with a context pair that resolves the reconstruction-rendering modality misalignment, TTB bakes DVSM-level scene information into a fixed-size MLP while retaining TTT-level rendering complexity. Across DL3DV, Re10K, and in-the-wild benchmarks, TTB matches DVSM [58] quality at a constant 23 FPS and consistently surpasses LaCT [77]. We hope this baking perspective offers a new way to integrate attention and TTT beyond view synthesis.

Acknowledgments and Disclosure of Funding

This work was supported by NVIDIA Taiwan AI Research & Development Center (TRDC). This research was funded by the National Science and Technology Council, Taiwan, under Grants NSTC 112-2222-E-A49-004-MY2, 113-2628-E-A49-023-, 115-2628-E-A49-024-, and 111-2628-E-A49-018-MY4. Yu-Lun Liu acknowledges the Yushan Young Fellow Program by the MOE in Taiwan.

References

[1] Mohammad Asim, Christopher Wewer, and Jan Eric Lenssen. Scenetok: A compressed, diffusable token space for 3d scenes. In *CVPR*, 2026.

[2] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV*, 2021.

[3] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, 2022.

[4] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. In *ICCV*, 2023.

[5] Eric Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J. Guibas, Jonathan Tremblay, S. Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3d generative adversarial networks. In *CVPR*, 2021.

[6] David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. Pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *CVPR*, 2024.

[7] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *ICCV*, 2021.

[8] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *ECCV*, 2022.

[9] Xingyu Chen, Yue Chen, Yuliang Xiu, Andreas Geiger, and Anpei Chen. TTT3R: 3d reconstruction as test-time training. In *ICLR*, 2025.

[10] Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. In *ECCV*, 2024.

[11] Ziwen Chen, Hao Tan, Peng Wang, Zexiang Xu, and Fuxin Li. Long-lrm++: Preserving fine details in feed-forward wide-coverage reconstruction. *arXiv preprint arXiv:2512.10267*, 2025.

[12] Ziwen Chen, Hao Tan, Kai Zhang, Sai Bi, Fujun Luan, Yicong Hong, Fuxin Li, and Zexiang Xu. Long-lrm: Long-sequence large reconstruction model for wide-coverage gaussian splats. In *ICCV*, 2025.

[13] Yilun Du, Cameron Smith, Ayush Tewari, and Vincent Sitzmann. Learning to render novel views from wide-baseline stereo pairs. In *CVPR*, 2023.

[14] Sven Elflein, Ruilong Li, Sérgio Agostinho, Zan Gojcic, Laura Leal-Taixé, Qunjie Zhou, and Aljosa Osep. Vgg-t^3: Offline feed-forward 3d reconstruction at scale. In *CVPR*, 2026.

[15] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022.

[16] Xiaodong Gu, Zhiwen Fan, Siyu Zhu, Zuozhuo Dai, Feitong Tan, and Ping Tan. Cascade cost volume for high-resolution multi-view stereo and stereo matching. In *CVPR*, 2019.

[17] Xiaoyang Guo, Kai Yang, Wukui Yang, Xiaogang Wang, and Hongsheng Li. Group-wise correlation stereo network. In *CVPR*, 2019.

[18] Dongchen Han, Yining Li, Tianyu Li, Zixuan Cao, Ziming Wang, Jun Song, Yu Cheng, Bo Zheng, and Gao Huang. Vit^3: Unlocking test-time training in vision. In *CVPR*, 2026.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[20] Alex Henry, Prudhvi Raj Dachapally, Shubham Shantaram Pawar, and Yuxuan Chen. Query-key normalization for transformers. In *EMNLP*, 2020.

[21] Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. LRM: large reconstruction model for single image to 3d. In *ICLR*, 2024.

[22] Xiaosong Jia, Yihang Sun, Junqi You, Songbur Wong, Zichen Zou, Junchi Yan, Zuxuan Wu, and Yu-Gang Jiang. Efficient-lvsm: Faster, cheaper and and better large view synthesis model via decoupled co-refinement attention. *arXiv preprint arXiv:2602.06478*, 2026.

[23] Hanwen Jiang, Hao Tan, Peng Wang, Haian Jin, Yue Zhao, Sai Bi, Kai Zhang, Fujun Luan, Kalyan Sunkavalli, Qixing Huang, and Georgios Pavlakos. Rayzer: A self-supervised large view synthesis model. In *ICCV*, 2025.

[24] Lihan Jiang, Yucheng Mao, Linning Xu, Tao Lu, Kerui Ren, Yichen Jin, Xudong Xu, Mulin Yu, Jiangmiao Pang, Feng Zhao, Dahua Lin, and Bo Dai. Anysplat: Feed-forward 3d gaussian splatting from unconstrained views. In *SIGGRAPH ASIA*, 2025.

[25] Haian Jin, Hanwen Jiang, Hao Tan, Kai Zhang, Sai Bi, Tianyuan Zhang, Fujun Luan, Noah Snavely, and Zexiang Xu. LVSM: A large view synthesis model with minimal 3d inductive bias. In *ICLR*, 2025.

[26] Haian Jin, Rundi Wu, Tianyuan Zhang, Ruiqi Gao, Jonathan T. Barron, Noah Snavely, and Aleksander Holynski. Zipmap: Linear-time stateful 3d reconstruction via test-time training. In *CVPR*, 2026.

[27] Justin Johnson, Alexandre Alahi, and Fei-Fei Li. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016.

[28] Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL `https://kellerjordan.github.io/posts/muon/`.

[29] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM TOG*, 2023.

[30] Taras Khakhulin, Denis Korzhenkov, Pavel Solovev, Gleb Sterkin, Andrei-Timotei Ardelean, and Victor Lempitsky. Stereo magnification with multi-layer images. In *CVPR*, 2022.

[31] Evan Kim, Hyunwoo Ryu, Thomas W. Mitchel, and Vincent Sitzmann. Scaling view synthesis transformers. In *CVPR*, 2026.

[32] Hanxue Liang, Jiawei Ren, Ashkan Mirzaei, Antonio Torralba, Ziwei Liu, Igor Gilitschenski, Sanja Fidler, Cengiz Oztireli, Huan Ling, Zan Gojcic, and Jiahui Huang. Feed-forward bullet-time reconstruction of dynamic scenes from monocular videos. *Adv. Neural Inform. Process. Syst.*, 2025.

[33] Chin-Yang Lin, Cheng Sun, Fu-En Yang, Min-Hung Chen, Yen-Yu Lin, and Yu-Lun Liu. Longsplat: Robust unposed 3d gaussian splatting for casual long videos. In *ICCV*, 2025.

[34] Chin-Yang Lin, Chung-Ho Wu, Chang-Han Yeh, Shih-Han Yen, Cheng Sun, and Yu-Lun Liu. Frugalnerf: Fast convergence for few-shot novel view synthesis without learned priors. In *CVPR*, 2025.

[35] Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, Xuanmao Li, Xingpeng Sun, Rohan Ashok, Aniruddha Mukherjee, Hao Kang, Xiangrui Kong, Gang Hua, Tianyi Zhang, Bedrich Benes, and Aniket Bera. DL3DV-10K: A large-scale scene dataset for deep learning-based 3d vision. In *CVPR*, 2024.

[36] Junchen Liu, Sven Elflein, Or Litany, Zan Gojcic, and Ruilong Li. Test-time training with KV binding is secretly linear attention. *arxiv preprint arxiv:2602.21204*, 2026.

[37] Yuan Liu, Sida Peng, Lingjie Liu, Qianqian Wang, Peng Wang, Christian Theobalt, Xiaowei Zhou, and Wenping Wang. Neural rays for occlusion-aware image-based rendering. In *CVPR*, 2022.

[38] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2017.

[39] Andreas Meuleman, Y. Liu, Chen Gao, Jia-Bin Huang, Changil Kim, Min H. Kim, and Johannes Kopf. Progressively optimized local radiance fields for robust view synthesis. In *CVPR*, 2023.

[40] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: practical view synthesis with prescriptive sampling guidelines. *SIGGRAPH*, 2019.

[41] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[42] Thomas W. Mitchel, Hyunwoo Ryu, and Vincent Sitzmann. True self-supervised novel view synthesis is transferable. In *ICLR*, 2026.

[43] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM TOG*, 2022.

[44] Julius Plucker. On a new geometry of space. *Philosophical Transactions of the Royal Society of London*, 1865.

[45] Jiawei Ren, Michal Jan Tyszkiewicz, Jiahui Huang, and Zan Gojcic. Tokengs: Decoupling 3d gaussian prediction from pixels with learnable tokens. In *CVPR*, 2026.

[46] Robin Rombach, Patrick Esser, and Björn Ommer. Geometry-free view synthesis: Transformers and no 3d priors. In *ICCV*, 2021.

[47] Mehdi S. M. Sajjadi, Henning Meyer, Etienne Pot, Urs Bergmann, Klaus Greff, Noha Radwan, Suhani Vora, Mario Lucic, Daniel Duckworth, Alexey Dosovitskiy, Jakob Uszkoreit, Thomas A. Funkhouser, and Andrea Tagliasacchi. Scene representation transformer: Geometry-free novel view synthesis through set-latent scene representations. In *CVPR*, 2022.

[48] Tim Salimans and Diederik P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *NeurIPS*, 2016.

[49] Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *ICML*, 2021.

[50] Noam Shazeer. Glu variants improve transformer. *arxiv preprint arXiv:2002.05202*, 2020.

[51] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR*, 2016.

[52] Oriane Sim'eoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michael Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Herv'e J'egou, Patrick Labatut, and Piotr Bojanowski. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

[53] Vincent Sitzmann, Semon Rezchikov, William T. Freeman, Joshua B. Tenenbaum, and Frédo Durand. Light field networks: Neural scene representations with single-evaluation rendering. In *Adv. Neural Inform. Process. Syst.*, 2021.

[54] Mohammed Suhail, Carlos Esteves, Leonid Sigal, and Ameesh Makadia. Light field neural rendering. *IEEE TPAMI*, 2021.

[55] Mohammed Suhail, Carlos Esteves, Leonid Sigal, and Ameesh Makadia. Generalizable patch-based neural rendering. In *ECCV*, 2022.

[56] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, 2022.

[57] Cheng Sun, Jaesung Choe, Charles Loop, Wei-Chiu Ma, and Yu-Chiang Frank Wang. Sparse voxels rasterization: Real-time high-fidelity radiance field rendering. In *CVPR*, 2025.

[58] Cheng Sun, Jaesung Choe, Min-Hung Chen, Ryo Hachiuma, and Yu-Chiang Frank Wang. Dvsm: Decoder-only view synthesis model done right. *arXiv preprint arXiv:2605.29891*, 2026.

[59] Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Ge Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Oluwasanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): Rnns with expressive hidden states. In *ICML*, 2025.

[60] Stanislaw Szymanowicz, Eldar Insafutdinov, Chuanxia Zheng, Dylan Campbell, João F. Henriques, Christian Rupprecht, and Andrea Vedaldi. Flash3d: Feed-forward generalisable 3d scene reconstruction from a single image. *3DV*, 2024.

[61] Stanislaw Szymanowicz, Minghao Chen, Jianyuan Wang, Christian Rupprecht, and Andrea Vedaldi. Lagernvs: Latent geometry for fully neural real-time novel view synthesis. In *CVPR*, 2026.

[62] Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. Lgm: Large multi-view gaussian model for high-resolution 3d content creation. In *ECCV*, 2024.

[63] Chen Wang, Hao Tan, Wang Yifan, Zhiqin Chen, Yuheng Liu, Kalyan Sunkavalli, Sai Bi, Lingjie Liu, and Yiwei Hu. tttlrm: Test-time training for long context and autoregressive 3d reconstruction. *arXiv preprint arXiv:2602.20160*, 2026.

[64] Haoru Wang, Kai Ye, Yangyan Li, Wenzheng Chen, and Baoquan Chen. The less you depend, the more you learn: Synthesizing novel views from sparse, unposed images without any 3d knowledge. In *ICLR*, 2026.

[65] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotný. VGGT: visual geometry grounded transformer. In *CVPR*, 2025.

[66] Peng Wang, Yuan Liu, Zhaoxi Chen, Lingjie Liu, Ziwei Liu, Taku Komura, Christian Theobalt, and Wenping Wang. F2-nerf: Fast neural radiance field training with free camera trajectories. In *CVPR*, 2023.

[67] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P. Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas A. Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *CVPR*, 2021.

[68] Xingzheng Wang, Kaiqiang Chen, Zixuan Wang, and Wenhao Huang. Pmsnet: Parallel multi-scale network for accurate low-light light-field image enhancement. In *Transactions on Multimedia*, 2024.

[69] Zhengqing Wang, Yuefan Wu, Jiacheng Chen, Fuyang Zhang, and Yasutaka Furukawa. Clift: Compressive light-field tokens for compute-efficient and adaptive neural rendering. In *NeurIPS*, 2025.

[70] Xinyue Wei, Kai Zhang, Sai Bi, Hao Tan, Fujun Luan, Valentin Deschaintre, Kalyan Sunkavalli, Hao Su, and Zexiang Xu. Meshlrm: Large reconstruction model for high-quality mesh. *arXiv preprint arXiv:2404.12385*, 2024.

[71] Haofei Xu, Songyou Peng, Fangjinhua Wang, Hermann Blum, Daniel Barath, Andreas Geiger, and Marc Pollefeys. Depthsplat: Connecting gaussian splatting and depth. In *CVPR*, 2025.

[72] Jiayu Yang, Wei Mao, José Manuel Álvarez, and Miaomiao Liu. Cost volume pyramid based depth inference for multi-view stereo. In *CVPR*, 2020.

[73] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In *ECCV*, 2018.

[74] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *CVPR*, 2021.

[75] Wangbo Yu, Jinbo Xing, Li Yuan, Wenbo Hu, Xiaoyu Li, Zhipeng Huang, Xiangjun Gao, Tien-Tsin Wong, Ying Shan, and Yonghong Tian. Viewcrafter: Taming video diffusion models for high-fidelity novel view synthesis. *IEEE TPAMI*, 2024.

[76] Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. GS-LRM: large reconstruction model for 3d gaussian splatting. In *ECCV*, 2024.

[77] Tianyuan Zhang, Sai Bi, Yicong Hong, Kai Zhang, Fujun Luan, Songlin Yang, Kalyan Sunkavalli, William T Freeman, and Hao Tan. Test-time training done right. *arXiv preprint arXiv:2505.23884*, 2025.

[78] Qitao Zhao, Hao Tan, Qianqian Wang, Sai Bi, Kai Zhang, Kalyan Sunkavalli, Shubham Tulsiani, and Hanwen Jiang. E-rayzer: Self-supervised 3d reconstruction as spatial visual pre-training. *arXiv preprint arXiv:2512.10950*, 2025.

[79] Jensen Zhou, Hang Gao, Vikram S. Voleti, Aaryaman Vasishta, Chun-Han Yao, Mark Boss, Philip Torr, Christian Rupprecht, and Varun Jampani. Stable virtual camera: Generative view synthesis with diffusion models. In *ICCV*, 2025.

Supplementary material

Overview. This supplementary material expands on the main paper with implementation details (Sec. A.1–A.6), a method-level comparison with related decoder-only LVSM and TTT-based methods (Sec. B), additional quantitative results (Sec. C), and additional qualitative comparisons (Sec. D). We further provide turntable rendering videos and an interactive HTML viewer in the supplementary attachment.

A Additional details

A.1 Common implementation details

Unless otherwise specified, all of our final models share the configuration described in this section. The decoder consists of 12 blocks with a latent dimension of 768, organized as illustrated in main paper Fig. 3. Optimization is carried out with AdamW [38] using a weight decay of 0.05 and momentum coefficients $(0.9, 0.95)$. The learning rate is annealed along a cosine schedule preceded by 2.5k linear warm-up iterations, peaking at 4×10^{-4} during the lowest-resolution stage and dropping to 1×10^{-5} when finetuning at higher resolutions. The perceptual term in our rendering loss is weighted by $\lambda = 0.2$. Training runs on 64 A100 GPUs, and we cast the learnable layers to BFloat16 mixed precision during forward propagation.

Camera information is encoded with the Plücker parameterization [44]: each pixel is associated with a 6-dimensional vector $(\mathbf{r}_o \times \mathbf{r}_d,\ \mathbf{r}_d)$, where $\mathbf{r}_o$ denotes the camera origin, $\mathbf{r}_d$ the unit direction of the corresponding ray, and $\times$ the vector cross product. We further normalize the context-view poses following the convention of GS-LRM [76].

Inside each transformer block, we adopt the pre-norm layout combined with QK-normalization. Each attention layer uses 12 heads, and every fully-connected sublayer expands the hidden dimension by a factor of 4. All linear projections in our architecture are bias-free.

A.2 TTB layer details

This section elaborates on TTB layer details that were elided from the main paper for space. We provide the pseudocode of training our TTB model in Algo. 1 as a complementary reference for the descriptions below. The three fast-weight matrices of our SwiGLU-MLP, $\mathbf{w}_0, \mathbf{w}_2 \in \mathbb{R}^{H\times d_{\text{in}}\times d_h}$ and $\mathbf{w}_1 \in \mathbb{R}^{H\times d_h\times d_{\text{out}}}$, are stored as ordinary learnable parameters and are jointly optimized in the outer loop alongside the rest of the network. Consequently, the inner loop for every scene starts from a common, outer-loop-learned meta-initialization θ_0, which is then advanced by T Muon [28] update steps before being applied to the rendering tokens. When local-duplicate inference is enabled with $K > 1$ groups, every per-group reconstruction starts from the same meta-initialized θ_0 and is specialized only by the views assigned to its group, so all local copies share the same starting point and differ through their inner-loop trajectories.

We initialize TTB models from the corresponding DVSM [58] checkpoint with `strict=False`: the patch embeddings, input LayerNorms, Plücker encoding, intra-view attention layers, both FFNs in each block, and the RGB head are inherited directly, while every module inside the TTB layer—including its LayerNorm, query/key and value projections, output projection, per-token learning-rate head, output RMSNorm, and the fast weights $\mathbf{w}_0, \mathbf{w}_1, \mathbf{w}_2$—is freshly initialized. The cross-view attention projections of DVSM are not reused, since the inner-loop target of TTB is structurally different and warm-starting from the DVSM projections did not yield benefits in our preliminary experiments. To prevent the random-init TTB layer from disrupting the inherited DVSM features at the start of finetuning, we initialize the TTB output projection with a small standard deviation following the residual zero-init trick used in deep transformers, so that the TTB block contributes approximately zero at finetune step 0 and the entire forward pass reduces to the pretrained DVSM model plus an identity TTB layer. This zero-init together with the parameter inheritance above accounts for the $+1.02$ PSNR gain reported in Tab. 13.

For the TTB layer micro-architecture, we share a single linear projection between query and key construction: the inner-loop input $\mathbf{W}_k\mathbf{X}^{(\text{ctx-rend})}$ in main paper Eq. 6 and the rendering-time query $\mathbf{W}_k\mathbf{X}^{(\text{nv})}$ are both produced by passing the input through a single linear layer followed by SiLU and

Algorithm 1 TTB model training

Input: DVSM [58] model parameters ϕ with L blocks; TTB MLP fast-weight $\{\theta_\ell\}_{\ell=1}^{L}$; Outer lr α; Inner steps T; Training set $\mathcal{D}$
Output: Optimized model parameters ϕ^* and fast weight $\{\theta_\ell^*\}_{l=1}^{L}$

function INNEROPTIM($\theta^{(0)}, \mathbf{X}, \mathbf{Y}$)
  ▷ Bake the fast-weight MLP for a $\mathbf{X} \mapsto \mathbf{Y}$ mapping
  for $t = 1, \ldots, T$ **do**
    $\mathbf{Y}' = f_{\theta^{(t-1)}}(\mathbf{X})$ ▷ MLP feed-forwarding
    $\mathcal{L}_{\text{TTB}} = \|\mathbf{Y}' - \mathbf{Y}\|_2^2$ ▷ Inner-loop objective
    ▷ We follow LaCT [77] to use Muon [28] for inner-loop optimization
    $\theta^{(t)} = \theta^{(t-1)} - \text{Muon}\left(\nabla_{\theta^{(t-1)}} \mathcal{L}_{\text{TTB}}\right)$ ▷ Fast-weight updating (Eq.8)
  return $\theta^{(T)}$
end function

for each training iteration **do**
  ▷ Let's assume one scene in a batch for brevity
  Sample scene: input views $\{(\mathbf{I}_i, \mathbf{P}_i, \mathbf{K}_i)\}_{i=1}^{N}$, target views $\{(\mathbf{I}'_i, \mathbf{P}'_i, \mathbf{K}'_i)\}_{i=1}^{M}$ from $\mathcal{D}$
  ▷ Construct the context pairs (Eq. 7). Omit layer index for brevity.
  $\mathbf{X}^{\text{(ctx-gt)}} = \text{LayerNorm}(\text{PE}^{\text{(ray)}}(\mathbf{P}, \mathbf{K}) + \text{PE}^{\text{(rgb)}}(\mathbf{I}))$ ▷ Context ground-truth tokens
  $\mathbf{X}^{\text{(ctx-rend)}} = \text{LayerNorm}(\text{PE}^{\text{(ray)}}(\mathbf{P}, \mathbf{K}))$ ▷ Context rendering tokens
  ▷ Reconstruction stage (context prefilling)
  for $\ell = 1, \ldots, L$ **do**
    ▷ Transformer block feed-forwarding (Eq. 3)
    $\mathbf{X}^{\text{(ctx-gt)}} = \mathbf{X}^{\text{(ctx-gt)}} + \text{IntraAttn}_\ell(\mathbf{X}^{\text{(ctx-gt)}})$
    $\mathbf{X}^{\text{(ctx-rend)}} = \mathbf{X}^{\text{(ctx-rend)}} + \text{IntraAttn}_\ell(\mathbf{X}^{\text{(ctx-rend)}})$
    $\mathbf{X}^{\text{(ctx-gt)}} = \mathbf{X}^{\text{(ctx-gt)}} + \text{FFN}_{\ell,1}(\mathbf{X}^{\text{(ctx-gt)}})$
    $\mathbf{X}^{\text{(ctx-rend)}} = \mathbf{X}^{\text{(ctx-rend)}} + \text{FFN}_{\ell,1}(\mathbf{X}^{\text{(ctx-rend)}})$
    ▷ **==== Start of test-time baking ====**
    $\mathbf{O} = \text{CrossAttn}_\ell(\mathbf{X}^{\text{(ctx-gt)}})$
    $\mathbf{Y} = \text{Linear}_{\ell,v}(\mathbf{O} - \mathbf{X}^{\text{(ctx-rend)}})$ ▷ The mapping target
    $\mathbf{X} = \text{Linear}_{\ell,k}(\mathbf{X}^{\text{(ctx-rend)}})$ ▷ The mapping source
    $\theta_\ell^{(T)} = \text{InnerOptim}(\theta_\ell, \mathbf{X}, \mathbf{Y})$ ▷ Inner-loop optimization for T steps
    $\mathbf{Y}' = f_{\theta_\ell^{(T)}}(\mathbf{X})$ ▷ Feed-forward with the baked MLP
    $\mathbf{X}^{\text{(ctx-gt)}} = \mathbf{X}^{\text{(ctx-gt)}} + \mathbf{O} + \mathbf{Y}'$ ▷ Apply residual to context ground-truth tokens
    $\mathbf{X}^{\text{(ctx-rend)}} = \mathbf{X}^{\text{(ctx-rend)}} + \mathbf{Y}'$ ▷ Apply residual to context rendering tokens
    ▷ **==== End of test-time baking ====**
    $\mathbf{X}^{\text{(ctx-gt)}} = \mathbf{X}^{\text{(ctx-gt)}} + \text{FFN}_{\ell,2}(\mathbf{X}^{\text{(ctx-gt)}})$
    $\mathbf{X}^{\text{(ctx-rend)}} = \mathbf{X}^{\text{(ctx-rend)}} + \text{FFN}_{\ell,2}(\mathbf{X}^{\text{(ctx-rend)}})$
  ▷ Rendering stage
  $\mathbf{X}^{\text{(nv)}} = \text{LayerNorm}(\text{PE}^{\text{(ray)}}(\mathbf{P}', \mathbf{K}'))$ ▷ Novel-view rendering tokens
  for $\ell = 1, \ldots, L$ **do**
    $\mathbf{X}^{\text{(nv)}} = \mathbf{X}^{\text{(nv)}} + \text{IntraAttn}_\ell(\mathbf{X}^{\text{(nv)}})$
    $\mathbf{X}^{\text{(nv)}} = \mathbf{X}^{\text{(nv)}} + \text{FFN}_{\ell,1}(\mathbf{X}^{\text{(nv)}})$
    $\mathbf{X}^{\text{(nv)}} = \mathbf{X}^{\text{(nv)}} + f_{\theta_\ell^{(T)}}\left(\text{Linear}_{\ell,k}(\mathbf{X}^{\text{(nv)}})\right)$ ▷ **Baked MLPs replace cross-view attentions**
    $\mathbf{X}^{\text{(nv)}} = \mathbf{X}^{\text{(nv)}} + \text{FFN}_{\ell,2}(\mathbf{X}^{\text{(nv)}})$
  $\bar{\mathbf{I}} = (\text{PixShuf} \circ \text{Sigmoid} \circ \text{Linear} \circ \text{LayerNorm})(\mathbf{X}^{\text{(nv)}})$ ▷ Color head (Eq.5)
  $\mathcal{J} = \text{MSE}(\bar{\mathbf{I}}, \mathbf{I}') + \lambda \cdot \text{Percep}(\bar{\mathbf{I}}, \mathbf{I}')$ ▷ Rendering losses (Eq.2)
  $\phi, \theta = \text{ADAM}(\{\phi, \theta\}, \{\nabla_\phi \mathcal{J}, \nabla_\theta \mathcal{J}\}, \alpha)$ ▷ Optimization step

an L_2 unit-norm, which constrains the writes and reads of the fast-weight MLP to a common spherical subspace and removes a magnitude degree of freedom from the inner-loop optimization. The per-token learning rate is predicted by an additional linear head that outputs $3H$ scalars per token—one for each of the three fast-weight matrices in every head—and reparameterized as $\mathrm{softplus}(\cdot + \mathrm{softplus}^{-1}(0.01))$, ensuring strictly positive learning rates and an effective initial value of 0.01 across all heads and matrices.

The outer loop differentiates fully through all T inner-loop steps without any first-order approximation, so that the rendering loss reflects the exact dependence of the final fast weights on the outer-loop parameters. Each Muon step orthogonalizes the inner-loop gradient with five Newton–Schulz quintic iterations executed in BFloat16; we transpose the gradient before iteration whenever $d_{\text{in}} > d_h$ so that the intermediate Gram matrix has size $\min(d_{\text{in}}, d_h)^2$, which keeps the per-step memory manageable. Even with this optimization, retaining T inner-step activations for backpropagation remains the dominant memory cost of TTB. We mitigate it with two layers of gradient checkpointing: the inner-loop activations of SwiGLU and Muon are recomputed within each step, and the outer-loop checkpointing is applied selectively to the intra-view attention and FFN sublayers but not to the TTB sublayer itself, the combination of which makes our 64-A100 training feasible at 960p resolution with $V = 32$ context views.

Finally, our two main paper variants differ only in the inner-loop mapping target. TTB instantiates $g(\cdot)$ in main paper Eq. 6b as the cross-view attention output of $\mathbf{X}^{(\text{ctx-gt})}$, while TTB– replaces this with a linear projection $\mathbf{W}_v\mathbf{X}^{(\text{ctx-gt})}$ acting directly on the context ground-truth tokens. Both variants share every other component described above, including the context pair construction, the QK-shared projection, the per-token learning-rate head, the residual target, and the feedback path that propagates the baked MLP output back to $\mathbf{X}^{(\text{ctx-gt})}$.

A.3 Two-view Re10K specific details

For the two-view setting, each GPU receives 32 scenes per iteration, with 2 context views and 3 target views per scene. The interval between the two context frames is sampled uniformly between 25 and 192, and target frames are drawn from the union of all intermediate frames between the two context views and the extrapolated frames within 25 frames on either side. We initialize the model from the corresponding DVSM [58] Re10K checkpoint following the strategy detailed in Sec. A.2, and finetune it on Re10K for 100K iterations at 256×256 followed by 10K iterations at 512×512. Note that doubling the context tokens through the context pair (main paper Eq. 7) negligibly affects this two-view setup, so we keep the same per-GPU batch composition as DVSM.

A.4 Many-view DL3DV [35] specific details

DL3DV is the main benchmark for our dense-view results in main paper Tab. 1 and Tab. 3, as well as for all the ablations in main paper Sec. 4.3. For training, we adopt a dynamic batching scheme that controls the per-GPU token budget rather than the number of scenes. Each scene draws either 16 or 32 context views uniformly at random, paired with the same number of target views; the number of scenes packed into a GPU is then adjusted on the fly so that the aggregate count of *context-pair* tokens (i.e., the union of $\mathbf{X}^{(\text{ctx-gt})}$ and $\mathbf{X}^{(\text{ctx-rend})}$ across all scenes in the batch) stays within a fixed budget. This compensates for the doubled token count introduced by the context pair and keeps the effective compute per iteration comparable to DVSM at the same memory ceiling. The frame interval between two adjacent context views is sampled uniformly from $[1, 16]$, and target views are drawn from $[\min(\mathcal{C})-2, \max(\mathcal{C})+2] \setminus \mathcal{C}$, where $\mathcal{C}$ is the index set of the sampled context views. We exclude from training a small subset of scenes whose aspect ratio or native resolution does not match the canonical 16:9 960p layout. The resolution curriculum starts from 340p and runs for 100K iterations, after which the model is finetuned for 10K iterations at each of 480p, 720p, and the final 960p (540×960).

For the controlled ablation setting used in main paper Tab. 4–6 and Tab. 13, we shrink the compute footprint to 8 A100 GPUs by fixing the context view count to 32, raising the patch size to 16 to lower the per-token resolution, and truncating the curriculum to 340p / 480p / 720p with 100K / 10K / 10K iterations respectively. All the design knobs ablated in those tables share this controlled setup; the rendering and reconstruction efficiency reported alongside is measured on a model with patch size 8 to remain consistent with the final main-table configuration.

A.5 Zero-shot evaluation details

The in-the-wild numbers in main paper Tab. 3 are produced by the DVSM-initialized 960p TTB model trained on DL3DV with patch size 8, evaluated zero-shot on three out-of-distribution datasets: the 9 scenes of MipNeRF360 [3], the 7 scenes of Free [66], and the 6 stable scenes from Hike [39] as identified by [33]. We hold out every 8th frame on MipNeRF360 and Free and every 10th frame on Hike as target views, and run k-means over the remaining frames to obtain context views. All input images are first resized—preserving their original aspect ratio—so that the total pixel count matches the $540{\cdot}960{=}518{,}400$ target used by our model, and predicted novel views are then resized back to the dataset's native resolution prior to metric computation. We apply the same image preprocessing pipeline to all baselines for a fair comparison. The LagerNVS [61] numbers are produced from our intermediate 480p checkpoint, since LagerNVS operates at 512 pixels along the longer image side and matching its native input resolution is the cleanest comparison.

A.6 Local-duplicate inference details

This section gives further details of the local-duplicate inference described in main paper Sec. **??**, which reconstructs the scene as K spatially-local groups of context views at inference time, each with its own copy of the baked fast weights, so that a single fast-weight MLP only has to absorb the views of one local group (about B on average) instead of all V views. It reuses the trained model as is, with no retraining and no additional learned parameters; the per-scene state grows to K sets of fast weights.

Grouping. Each context view i is summarized into a 6-dimensional descriptor $\mathbf{f}_i = [\mathbf{c}_i; \mathbf{d}_i] \in \mathbb{R}^6$, where $\mathbf{c}_i \in \mathbb{R}^3$ is the camera origin and $\mathbf{d}_i \in \mathbb{R}^3$ is the camera's unit optical axis, both expressed in the input world frame. The number of groups K is an inference-time hyper-parameter; $K = 1$ falls back to the single-MLP model. Unless stated otherwise we set $K = V/B$ with a nominal budget of $B = 16$ views per group, *i.e.*, $K = 2$ for $V = 32$ and $K = 4$ for $V = 64$. We then run Lloyd's algorithm for 20 iterations with the Euclidean distance to cluster the V descriptors into K groups. Centroids are initialized with the descriptors of K views whose indices are evenly spaced over $[0, V{-}1]$; since our context views are ordered by capture time, this spreads the initial centroids along the camera trajectory. If a centroid attracts no views during an iteration, we leave it unchanged for that round. Group sizes are neither balanced nor capped, and K must stay small relative to V.

Per-group reconstruction. For every group k we run the standard reconstruction pass of our model on its member views only. Inside this pass, all sublayers—intra-view attention, FFN, cross-view attention, and the TTB inner loop—see only the tokens of group k; in particular, the cross-view attention that provides the baking target attends across the views of the group rather than across all V views. The two streams $X^{(\text{ctx-gt})}$ and $X^{(\text{ctx-rend})}$ of a context view are created inside the same pass and are therefore always baked together. Every pass starts from the shared meta-initialization θ_0 and runs the same T inner-loop steps, producing the per-layer fast weights $\{\theta_{k,\ell}\}_{\ell=1}^{L}$ of copy k. The K passes are independent of each other and are executed sequentially in our implementation. At the end of context prefilling we cache the fast weights of every group together with the K cluster centroids.

Rendering. At rendering time, we do not re-run k-means. Each novel view computes its own 6-dimensional descriptor in the same input world frame and takes an $\arg\min$ of the Euclidean distance to the cached centroids to select a single group, and all of its tokens are rendered with the fast weights of that group. The switch between copies is hard, without blending. The per-token rendering cost is unchanged and independent of V; routing only adds one nearest-centroid search over K centroids per view, which is negligible. In our implementation every novel view is routed and rendered individually.

B Discussion: comparison with related methods

This section positions TTB within the recent landscape of feed-forward novel-view synthesis (NVS) and 3D reconstruction methods. We focus on the LVSM family [22, 25, 58, 77] together with three recent methods that adopt either TTT layers (LongLRM [12], tttLRM [63]) or a 3D-pretrained encoder (LagerNVS [61]), since these are the closest design points to TTB. The discussion is structured

around three axes: training settings (Sec. B.1), reconstruction/rendering complexity (Sec. B.2), and architectural composition (Sec. B.3).

B.1 Training settings across methods

Tab. 7 compares the training data, compute, and pretraining source across baselines and our method. All listed methods share AdamW as the outer-loop optimizer and a multi-stage resolution curriculum, so we restrict the table to the three axes that most directly affect comparability.

Table 7: **Training settings across methods.** "Multi-dataset" indicates training on a curated mixture beyond a single benchmark. "DVSM ckpt" refers to initializing from the corresponding DVSM [58] checkpoint released for that benchmark. "–" marks compute resources that are not explicitly reported by the corresponding paper.

Method	Training data	GPUs	Pretrain source
LVSM [25]	Objaverse / Re10K	64×A100	from scratch
Efficient-LVSM [22]	Re10K	64×A100	from scratch
DVSM [58]	DL3DV / Re10K	64×A100	from scratch
LaCT [77]	Objaverse / DL3DV	64×A100	from scratch
LongLRM [12]	DL3DV	64×A100	from scratch
tttLRM [63]	Objaverse / DL3DV	64×A100	LaCT [77] ckpt
LagerNVS [61]	Multi-dataset	—	VGGT [65] ckpt
TTB (ours)	DL3DV / Re10K	64×A100	DVSM [58] ckpt

The compute footprint ($64\times$A100) is broadly shared across the group when reported, so the differences mainly lie in data composition and pretraining strategy. Most LVSM-family methods train scene-only on Re10K or DL3DV, occasionally extending to Objaverse for object-level pretraining; LagerNVS is the outlier that explicitly relies on a multi-dataset mixture of fifteen sources to obtain its in-the-wild generalization. Three methods escape the from-scratch pattern, and they do so along two distinct lines: tttLRM and TTB warm-start from an NVS checkpoint of the same task family (TTT-LVSM and DVSM, respectively), reusing weights that were already optimized for novel-view rendering; LagerNVS instead transfers from VGGT, a model trained for 3D reconstruction, and jointly finetunes both the VGGT encoder and a freshly initialized cross-attention renderer with the rendering loss. TTB sits closer to tttLRM in this taxonomy—both inherit cross-view-attention features from a pretrained NVS backbone and reshape them into a TTT-style fast-weight representation—while LagerNVS represents an orthogonal direction of leveraging 3D-supervised priors.

B.2 Reconstruction and rendering complexity

Tab. 8 summarizes per-scene reconstruction and per-token rendering complexity for the methods above. The scene representation column reports the asymptotic memory footprint of the per-scene state that is consumed at rendering time.

The table makes the design space explicit. The decoder-only LVSM reconstructs nothing offline and pays the full attention cost at every novel-view query; LVSM enc.-dec. and Efficient-LVSM amortize the cost into a one-time encoding stage but the rendering cost remains linear in the number of context tokens. DVSM further removes the encoder-decoder decoupling but at the price of quadratic cross-view attention during prefilling. LagerNVS occupies the same asymptotic regime as LVSM enc.-dec. but with a heavier V-dependence because its VGGT encoder runs alternating frame-wise and global self-attention over all $V \cdot L$ tokens, yielding $\mathcal{O}(V^2 \cdot L^2)$ encoding and $\mathcal{O}(V \cdot L)$ per-token rendering through cross-attention to the encoder output. LaCT pushes both axes to their constant-complexity limits via TTT layers, whereas TTB occupies a previously empty cell on this map: it preserves DVSM's quadratic reconstruction cost—necessary because cross-view attention provides the inner-loop target—while inheriting LaCT's $\mathcal{O}(1)$ rendering and fixed-size scene representation. TTB– in turn shows that, when the cross-view attention target is replaced by a linear projection, reconstruction collapses back to linear in V, recovering LaCT's complexity profile but with the rest of the TTB design choices retained. LRM-style baselines such as LongLRM and tttLRM occupy a different regime altogether, since their per-scene state is an explicit 3D Gaussian set rendered by a

Table 8: **Asymptotic complexity comparison.** V denotes the number of context views and L the number of tokens per view. Reconstruction is reported per scene, rendering per novel-view token. "GS rasterizer" denotes that rendering is handled by a Gaussian splatting rasterizer rather than a neural module, which is not directly comparable in $\mathcal{O}(\cdot)$ form. We only list the complexity of the main model layer for global scene understanding. LongLRM still employs a few cross-view attention layers with $\mathcal{O}(V^2 \cdot L^2)$ complexity. LaCT, tttLRM, LagerNVS, and ours also employ intra-view attention with $\mathcal{O}(V \cdot L^2)$ reconstruction complexity and $\mathcal{O}(L)$ per-token rendering complexity.

Method	Reconstruction	Rendering (per token)	Scene representation
LVSM (decoder-only) [25]	—	$\mathcal{O}(V \cdot L)$	context tokens
LVSM (enc.-dec.) [25]	$\mathcal{O}(V^2 \cdot L^2)$	$\mathcal{O}(L_s)$	latent scene tokens
Efficient-LVSM [22]	$\mathcal{O}(V \cdot L^2)$	$\mathcal{O}(V \cdot L)$	KV-cache
DVSM [58]	$\mathcal{O}(V^2 \cdot L^2)$	$\mathcal{O}(V \cdot L)$	KV-cache
LaCT [77]	$\mathcal{O}(V \cdot L)$	$\mathcal{O}(1)$	fast weights
LongLRM [12]	$\mathcal{O}(V \cdot L)$	GS rasterizer	3D Gaussians
tttLRM [63]	$\mathcal{O}(V \cdot L)$	GS rasterizer	3D Gaussians
LagerNVS [61]	$\mathcal{O}(V^2 \cdot L^2)$	$\mathcal{O}(V \cdot L)$	latent scene tokens
TTB-- (ours)	$\mathcal{O}(V \cdot L)$	$\mathcal{O}(1)$	fast weights
TTB (ours)	$\mathcal{O}(V^2 \cdot L^2)$	$\mathcal{O}(1)$	fast weights

rasterizer rather than a neural module; we list them for completeness but do not directly compare $\mathcal{O}$-rendering with them.

B.3 Architectural composition

Tab. 9 compares the per-block composition and the total decoder depth. We restrict the table to depth, latent dim, and the cross-view sublayer type, since these are the most consequential differences relative to TTB. For LagerNVS we report depth in two parts since its model is composed of a 24-block VGGT encoder operating at 1024-dim and a 12-block cross-attention decoder operating at 768-dim.

Table 9: **Architectural composition.** "Cross-view sublayer" indicates the module that exchanges information across views in each block. The depth column is the number of decoder blocks, with the encoder depth listed in parentheses for encoder-decoder methods. The latent-dim column reports the decoder width, with the encoder width also given for encoder-decoder methods.

Method	# blocks	Latent dim	Cross-view sublayer
LVSM [25]	24	768	cross-view attention
Efficient-LVSM [22]	24	1024	cached KV cross-attention
DVSM [58]	12	768	cross-view attention (cached KV)
LaCT [77]	24	768	large-chunk TTT layer
LongLRM [12]	24	768	Mamba2 + attention hybrid
tttLRM [63]	24	768	large-chunk TTT layer
LagerNVS [61]	12 (+24 enc.)	768 (+1024 enc.)	VGGT alt. frame/global attn. (enc.) + cross-attn. (dec.)
TTB-- (ours)	12	768	TTB layer (linear target)
TTB (ours)	12	768	cross-view attn. + TTB layer

A consistent pattern across the methods is a 24-block decoder for from-scratch models, while methods that initialize from a pretrained backbone are able to reduce decoder depth. DVSM and our method use a 12-block decoder by sharing weights between reconstruction and rendering stages; LagerNVS reaches the same 12-block decoder width and depth (768, 12 blocks) by offloading representational capacity to a 24-block, 1024-dim VGGT encoder. TTB inherits the 12-block depth directly from DVSM: every TTB layer additionally houses a fast-weight MLP whose inner loop is supervised by the cross-view attention output, but the surrounding intra-view attention and FFN sublayers are unchanged from DVSM. The net effect is that TTB matches or surpasses both 24-block from-scratch baselines and the much heavier LagerNVS on DL3DV (main paper Tab. 1, Tab. 3) while operating with comparable parameter counts to DVSM (244M for TTB– and 271M for TTB, against 163M for DVSM, 379M for LaCT, and over 1B for LagerNVS due to the included VGGT encoder). We attribute

Table 10: **In-the-wild generalization at 16 input views.**

Method	Size	MipNerf360 [3] PSNR↑	MipNerf360 [3] SSIM↑	MipNerf360 [3] LPIPS↓	Free [66] PSNR↑	Free [66] SSIM↑	Free [66] LPIPS↓	Hike [39] PSNR↑	Hike [39] SSIM↑	Hike [39] LPIPS↓	Efficiency Time↓	Efficiency FPS↑
Input: 16 views × 518,400 pixels per view (960p)												
LVSM	165M	19.58	0.459	0.564	19.95	0.528	0.441	18.48	0.454	0.492	3.8s	20
LaCT	379M	20.10	0.494	0.498	20.77	0.588	0.350	18.35	0.512	0.403	1.9s	10
DVSM	163M	21.51	0.553	0.427	22.04	0.664	0.285	20.24	0.591	0.363	4.4s	4
Ours--	244M	21.46	0.540	0.442	22.21	0.638	0.321	19.25	0.549	0.391	1.9s	23
Ours	271M	22.03	0.563	0.424	22.93	0.673	0.285	19.94	0.585	0.368	5.4s	23
Input: 16 views × 512 pixels at image longer side												
LagerNVS	1052M	20.41	0.469	0.523	19.33	0.490	0.412	17.98	0.426	0.457	0.4s	21
DVSM	163M	21.47	0.531	0.482	21.71	0.637	0.336	20.23	0.586	0.347	0.4s	33
Ours--	244M	21.45	0.519	0.501	21.65	0.605	0.374	19.26	0.542	0.380	0.5s	67
Ours	271M	21.75	0.531	0.486	22.36	0.637	0.346	19.92	0.572	0.361	0.8s	67

this efficiency to the combination of a shallow shared decoder and the inheritance of pretrained DVSM weights, which Sec. A.2 discusses in detail.

C Additional results

C.1 Few-view lower-resolution results

We additionally train and evaluate our models on 2-view setup at a lower resolution. This is not our intended scenario as the rendering FPS speedup by replacing cross-view attentions with our baked MLPs become marginal: 64→72 at 256×256 and 45→52 at 512×512.

We follow previous works to train and test on Re10k [30] dataset. We start from the most capable pretrained checkpoint from DVSM [58] which employs DINO [52] injection and patch size of 8. We first train at 256×256 resolution for 100K iterations with learning rate 4×10^{-4}, and then finetune at 512×512 resolution for 10K iterations with learning rate 1×10^{-4}. In each iteration on a GPU, we sample 32 scenes with 2 context views and 3 target views. The distance between the 2 views is randomly sampled between 25–192 frame skips following previous works.

The results comparison is presented in Tab. 16. Under such few views, our TTBs can faithfully reproduce the cross-view attentions results, as reflected by the comparable results with our pretrained DVSM.

C.2 In-the-wild generalization at 16 input views

We complement the 64-view in-the-wild evaluation in main paper Tab. 3 with a 16-view counterpart in Tab. 10, covering the same three out-of-domain datasets (MipNeRF360 [3], Free [66], Hike [39]) at both 960p and 512-pixel longer-side settings. The trends mirror our in-domain DL3DV results: TTB closely matches DVSM [58] quality across all three datasets while consistently surpassing LaCT [77] and LVSM [25], and outperforms LagerNVS [61] by a clear margin under the 512-pixel setting. Efficiency-wise, TTB retains its constant-FPS rendering at 16 views just as at 64 views, confirming that the rendering speedup is independent of context view count. These results indicate that TTB's quality and speed advantages generalize beyond the dense-view regime to fewer-view, out-of-domain scenarios.

C.3 Inference at scale on DL3DV.

A single fast-weight MLP per TTB layer has fixed capacity, which becomes a bottleneck as the number of context views V grows. Tab. 11 compares this single-MLP baseline against our local-duplicate variant, which maintains K local copies of the per-layer fast-weight state, each updated only by views routed to its spatially-local group via k-means over per-view camera features (results reported with our ablation model at 720p). Up to $V{=}32$, the two variants are indistinguishable and both match DVSM [58]. Beyond $V{=}64$, the single-MLP variant saturates and plateaus around 25.8 PSNR, while the local-duplicate variant continues to scale and reaches 27.74 at $V{=}256$, on par with DVSM's 27.56. Throughout, rendering FPS stays in the 67 to 72 range, in stark contrast to DVSM's drop from 63.6 at $V{=}16$ to 12.6 at $V{=}256$. Local duplication thus resolves the capacity ceiling of

Table 11: **Inference at scale on DL3DV.** As the number of context views V grows, *TTB-single*—a single fast-weight MLP per TTB layer that absorbs all V views—suffers **capacity saturation**: its PSNR plateaus around $V{=}64$ and even regresses thereafter. *TTB-duplicate*, which maintains K local copies of the per-layer fast-weight state, each baked from a spatially-local group of views, prevents this saturation and tracks DVSM's quality scaling while preserving $\mathcal{O}(1)$ per-frame rendering. Best per column is shaded red, second-best orange.

	Method	$V{=}16$	$V{=}32$	$V{=}64$	$V{=}128$	$V{=}256$
PSNR↑	DVSM	22.83	24.68	26.23	27.20	27.56
	TTB-single	23.26	24.84	25.60	25.82	25.78
	TTB-duplicate	23.26	24.84	26.13	27.14	27.74
Recon-time↓	DVSM	0.136	0.389	1.258	4.815	18.106
	TTB-single	0.337	0.740	1.911	6.065	20.549
	TTB-duplicate	0.333	1.048	3.578	13.557	52.064
Rend-fps↑	DVSM	63.6	61.9	40.1	23.2	12.6
	TTB-single	73.2	72.6	72.3	72.2	72.2
	TTB-duplicate	72.0	67.9	68.0	67.0	68.5

Table 12: **Capacity scaling.** Left: PSNR of local-duplicate inference over the number of copies K and context views V (ablation setting, 720p); every row peaks at $K{=}V/16$, i.e. a fixed budget of 16 views per copy (bold). Right: scaling the fast-weight MLP width instead (DL3DV, $V{=}64$, 340p); quality rises monotonically and a $16\times$ width matches DVSM.

$V\backslash K$	1	2	4	8	16
16	**23.26**	22.86	22.07	20.73	19.02
32	**24.84**	**24.84**	24.44	23.53	22.00
64	25.60	26.11	**26.13**	25.76	24.93
128	25.82	26.60	27.07	**27.14**	26.85
256	25.78	26.64	27.26	27.66	**27.74**

Method	Width	PSNR↑	SSIM↑	LPIPS↓
DVSM	–	25.60	0.801	0.181
TTB ($2\times$)	1536	25.18	0.773	0.203
TTB ($4\times$, ours)	3072	25.36	0.780	0.197
TTB ($8\times$)	6144	25.52	0.786	0.192
TTB ($16\times$)	12288	25.61	0.792	0.188

a single fast-weight MLP without compromising the constant-complexity rendering that motivates TTB.

C.4 Capacity scaling

Tab. 12 studies the two knobs that set the capacity of the baked state. Sweeping K jointly with V shows that the best K is always $V/16$, so local duplication is governed by a single scene-independent budget of $B{=}16$ views per copy rather than a per-scene heuristic; we use this budget for all *Ours (duplicate)* entries in Tab. 1, and the $K{=}1$ column and the $K{=}V/16$ diagonal reproduce the two TTB rows of Tab. 11. Widening the fast-weight MLP closes the gap as well and reaches DVSM at $16\times$ width, indicating that the remaining gap is a capacity limit rather than a limitation of baking. Since width enlarges the model and requires retraining, we keep the $4\times$ width and scale capacity through K.

C.5 Effect of pretraining.

Tab. 13 compares three frameworks trained from scratch versus initialized from a DVSM [58] checkpoint (results reported with our ablation model at 720p). The PSNR gain from pretraining grows

Table 13: **Pretraining ablation.**

Method	w/o pretrain			w/ pretrain		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
TTT	22.84	0.688	0.307	23.12	0.704	0.286
TTB--	23.51	0.720	0.274	24.22	0.747	0.248
TTB	**23.82**	**0.731**	**0.264**	**24.84**	**0.768**	**0.230**

Table 14: **Efficiency comparison on DL3DV at 544×960 resolution with 32 input views.** Recon-time is the processing time of a single scene. Rend-fps indicates the number of frames rendered per second without batching. Infer-mem is the peak GPU memory usage when measuring the reconstruction and rendering. Train-time/it is the sum of model forward and backward time per iteration under training mode with a batch containing 32 context views. Train-mem is the peak memory usage when measuring training time. The actual training randomly samples different number of input views per scene rather than a fixed number. We highlight cells with numbers similar to the most efficient one under each column.

Method	Venue	PSNR↑	SSIM↑	LPIPS↓	#params↓	Recon-time↓	Rend-fps↑	Infer-mem↓	Train-time/it↓	Train-mem↓
LVSM [25]	ICLR'25	22.24	0.691	0.352	165M	15s	20	9G	73s	42G
LaCT [77]	ICLR'26	26.90	0.837	0.185	379M	3.7s	10	23G	36s	72G
DVSM $_{ps8}$	arXiv'26	29.13	0.893	0.129	163M	15s	2	20G	133s	65G
TTB– (ours)	–	28.12	0.867	0.158	244M	3.7s	23	18G	26s	68G
TTB (ours)	–	28.92	0.882	0.145	271M	18s	23	18G	90s	76G

Table 15: **End-to-end time for rendering N novel views** (seconds, DL3DV at 960p), computed as reconstruction time plus N/FPS from Tab. 1. TTB overtakes DVSM after 5–7 views at every V, while TTB-- leads from $N{=}1$.

V	Method	Recon	FPS	$N{=}1$	$N{=}2$	$N{=}5$	$N{=}10$	$N{=}30$	$N{=}100$
16	DVSM	4.4	4	4.7	4.9	5.7	6.9	11.9	29.4
	TTB--	1.9	23	1.9	2.0	2.1	2.3	3.2	6.2
	TTB	5.4	23	5.4	5.5	5.6	5.8	6.7	9.7
32	DVSM	15.0	2	15.5	16.0	17.5	20.0	30.0	65.0
	TTB--	3.7	23	3.7	3.8	3.9	4.1	5.0	8.0
	TTB	18.0	23	18.0	18.1	18.2	18.4	19.3	22.3
64	DVSM	60.0	1	61.0	62.0	65.0	70.0	90.0	160.0
	TTB--	7.5	23	7.5	7.6	7.7	7.9	8.8	11.8
	TTB	65.0	23	65.0	65.1	65.2	65.4	66.3	69.3

monotonically with cross-view attention preservation: smallest for TTT (no cross-view attention), larger for TTB-- (linear target), and largest for our full TTB (cross-view attention target), confirming that preserving cross-view attention is key to re-using decoder-only pretrained weights. Notably, TTB from scratch already surpasses pretrained TTT, indicating that its advantage is structural.

C.6 Efficiency analysis

We provide a detailed efficiency comparison in Tab. 14, reporting parameter counts, per-scene reconstruction time, per-view rendering FPS, and peak GPU memory at both inference and training time. All numbers are measured at 544×960 resolution with $V{=}32$ input views.

End-to-end time. Since TTB shifts computation from rendering to reconstruction, Tab. 15 reports the total time to reconstruct a scene and render N novel views. TTB overtakes DVSM after only 5–7 views at every V, a threshold that any interactive viewer or trajectory rendering exceeds, while TTB-- leads from the first view at the quality cost reported in Tab. 1.

C.7 Scene representation size

We further quantify the memory footprint of the per-scene state that must be retained between reconstruction and rendering at 544×960 resolution. Fig. 7 compares the KV-cache of DVSM [58] against the baked fast-weight state of TTB as the number of context views V grows. The DVSM KV-cache scales linearly with V, reaching tens of GBs at $V{=}256$, whereas TTB's scene representation is a fixed-size set of fast-weight matrices whose footprint is independent of V (324 MB across all L blocks in FP32). At $V{=}64$ this is an $85\times$ reduction over DVSM, growing to $340\times$ at $V{=}256$; local-duplicate inference stores $K{=}V/16$ copies and stays a constant $21\times$ smaller than the KV-cache at every V. This compact, V-independent state is what enables TTB's $O(1)$ per-token rendering and is also a practical advantage for deployment scenarios with limited GPU memory.

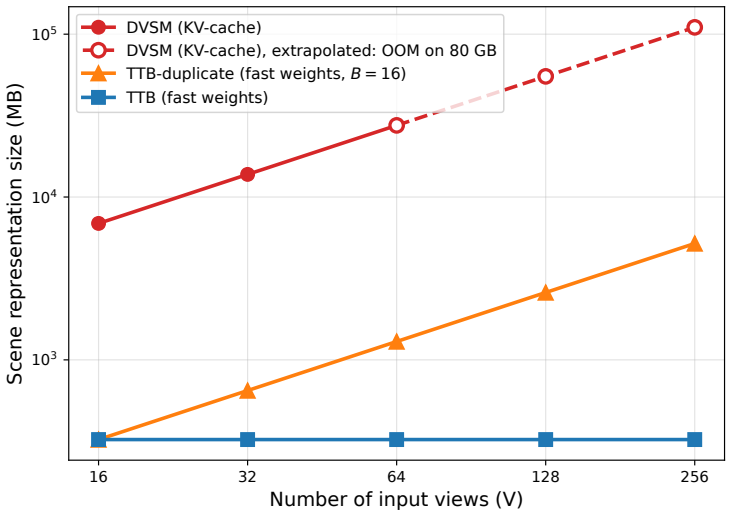

Figure 7: **Scene representation size as V grows** at 544×960 resolution, measured on a single A100. The DVSM [58] KV-cache grows linearly in V (hollow points: linear extrapolation where DVSM ran out of 80 GB memory), TTB's baked fast weights stay constant, and TTB-duplicate grows with the number of copies $K{=}V/16$ while remaining $21\times$ smaller than the KV-cache.

D Qualitative results

This section provides additional qualitative results that complement the quantitative comparisons in the main paper. We include in-domain comparisons on DL3DV [35] (Sec. D.1), two-view comparisons on Re10K [30] (Sec. D.2), and zero-shot comparisons on three in-the-wild datasets [3, 39, 66] (Sec. D.3).

D.1 In-domain results on DL3DV

Fig. 8 extends the comparison in main paper Fig. 5 to additional held-out DL3DV scenes spanning input view counts of $V{=}16$, 32, and 64, complementing the three-column quantitative results in main paper Tab. 1. TTB consistently preserves thin geometric structures and high-frequency texture where LaCT [77] blurs and LVSM [25] loses detail.

D.2 Two-view results on Re10K

Fig. 9 compares TTB against DVSM [58] and LVSM [25] under the two-view setting, complementing main paper Tab. 2. TTB matches DVSM's quality at the same rendering speed-up regime, and outperforms LVSM in regions involving moderate viewpoint extrapolation.

D.3 Zero-shot results on in-the-wild datasets

Figs. 10–12 report zero-shot results on MipNeRF360 [3], Free [66], and Hike [39], all using the DL3DV-trained TTB model with $V{=}64$ input views. These figures complement the in-the-wild numerical results in main paper Tab. 3. Despite a clear domain gap from DL3DV, TTB recovers fine details such as small text and thin structures that LaCT [77] and LVSM [25] blur out.

E Broader impacts

TTB shares its broader impacts with feed-forward novel-view synthesis methods in general. What is more specific to TTB is the methodological perspective it offers: rather than replacing attention with a test-time-trained fast-weight MLP as in prior TTT-style architectures, TTB shows that the fast-weight MLP can be supervised to *collaborate* with attention by baking its outputs into a constant-size state. This baking perspective is potentially applicable beyond novel-view synthesis, in any

Table 16: **Two input views on Re10K.**

Method	Venue	PSNR↑	SSIM↑	LPIPS↓
Resolution: 256×256				
pixelNeRF [74]	CVPR'21	20.43	0.589	0.550
AttnRend [13]	CVPR'23	24.78	0.820	0.213
ViewCrafter [75]	TPAMI'24	21.63	0.642	0.175
GPNR [55]	ECCV'24	24.11	0.793	0.255
PixelSplat [6]	CVPR'24	26.09	0.863	0.136
MVSplat [10]	ECCV'24	26.39	0.869	0.128
GS-LRM [76]	ECCV'24	28.10	0.892	0.114
DepthSplat [71]	CVPR'25	27.46	0.889	0.115
SEVA [79]	ICCV'25	25.66	0.841	0.139
LVSM [25]	ICLR'25	28.58	0.893	0.114
Dec.-only [25]	ICLR'25	29.67	0.906	0.098
Btimer [32]	NeurIPS'25	26.49	0.886	0.096
Eff. LVSM [22]	ICLR'26	28.93	0.895	0.102
TokenGS [45]	CVPR'26	28.41	0.903	0.135
SVSM [31]	CVPR'26	30.01	0.910	0.096
DVSM [58]	arXiv'26	31.23	0.925	0.085
Ours--		31.04	0.921	0.086
Ours		31.35	0.926	0.083
Resolution: 512×512				
LVSM [25]	ICLR'25	28.55	0.894	0.173
Dec.-only [25]	ICLR'25	29.53	0.904	0.141
Eff. LVSM [22]	ICLR'26	29.86	0.905	0.147
DVSM [58]	arXiv'26	31.17	0.924	0.129
Ours--		31.00	0.921	0.130
Ours		31.28	0.926	0.127

setting where a powerful but expensive attention module needs to be amortized into a state of fixed size. Long-context language models, where attention KV-caches grow linearly with sequence length, could similarly benefit from baking attention outputs into a fixed-capacity fast weight to enable constant-memory inference. World models and video generators that maintain a long temporal context face the same memory-growth issue and may adopt analogous baking strategies for the temporal-attention component. Physical 3D perception systems that operate on streaming sensor inputs, such as embodied agents and autonomous driving stacks, could likewise use a baked fast-weight state to maintain a compact running memory of the environment without unbounded KV growth. We hope the baking perspective inspires further exploration of how attention and test-time training can be combined across these domains.

F Declaration of LLM usage

Large language models were used as writing and editing aids during manuscript preparation—paraphrasing, grammar correction, and improving readability. We additionally used Claude Code as a coding assistant during the development of our training and evaluation pipeline; all code generated with its help was reviewed and tested by the authors before being incorporated into the final implementation. LLMs did not contribute to the research ideation, model design, experimental protocol, or analysis of results.

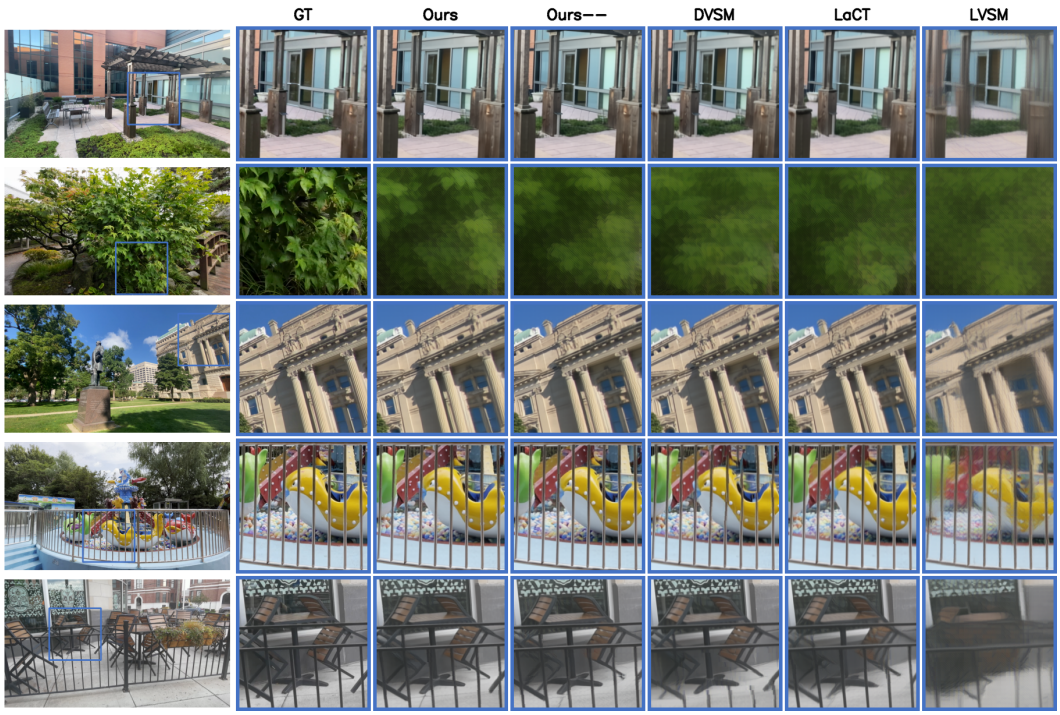

Figure 8: **Additional qualitative comparison on DL3DV.** 64 context views.

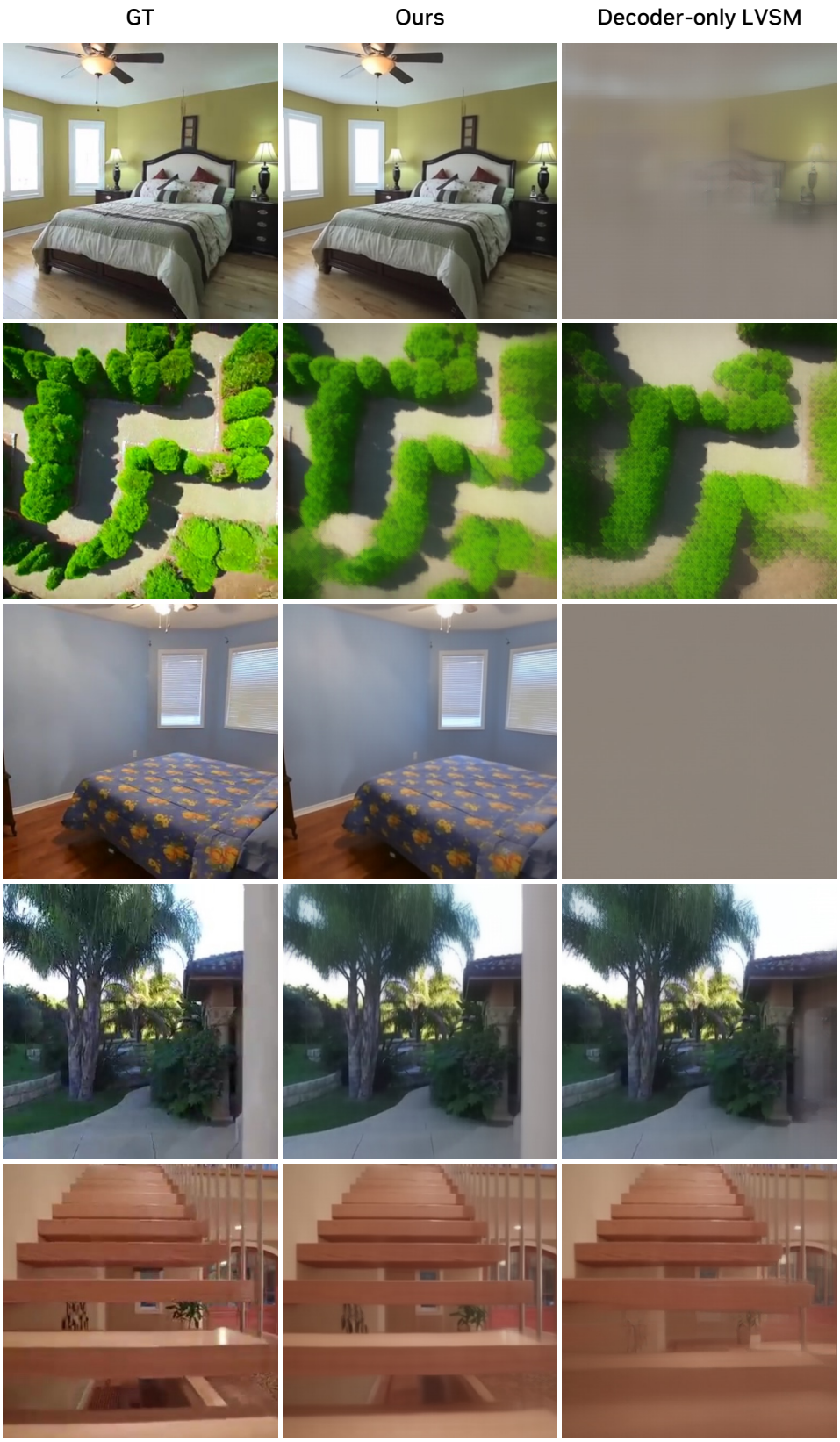

Figure 9: **Qualitative comparison on Re10K.** Two-view setting with sparse input.

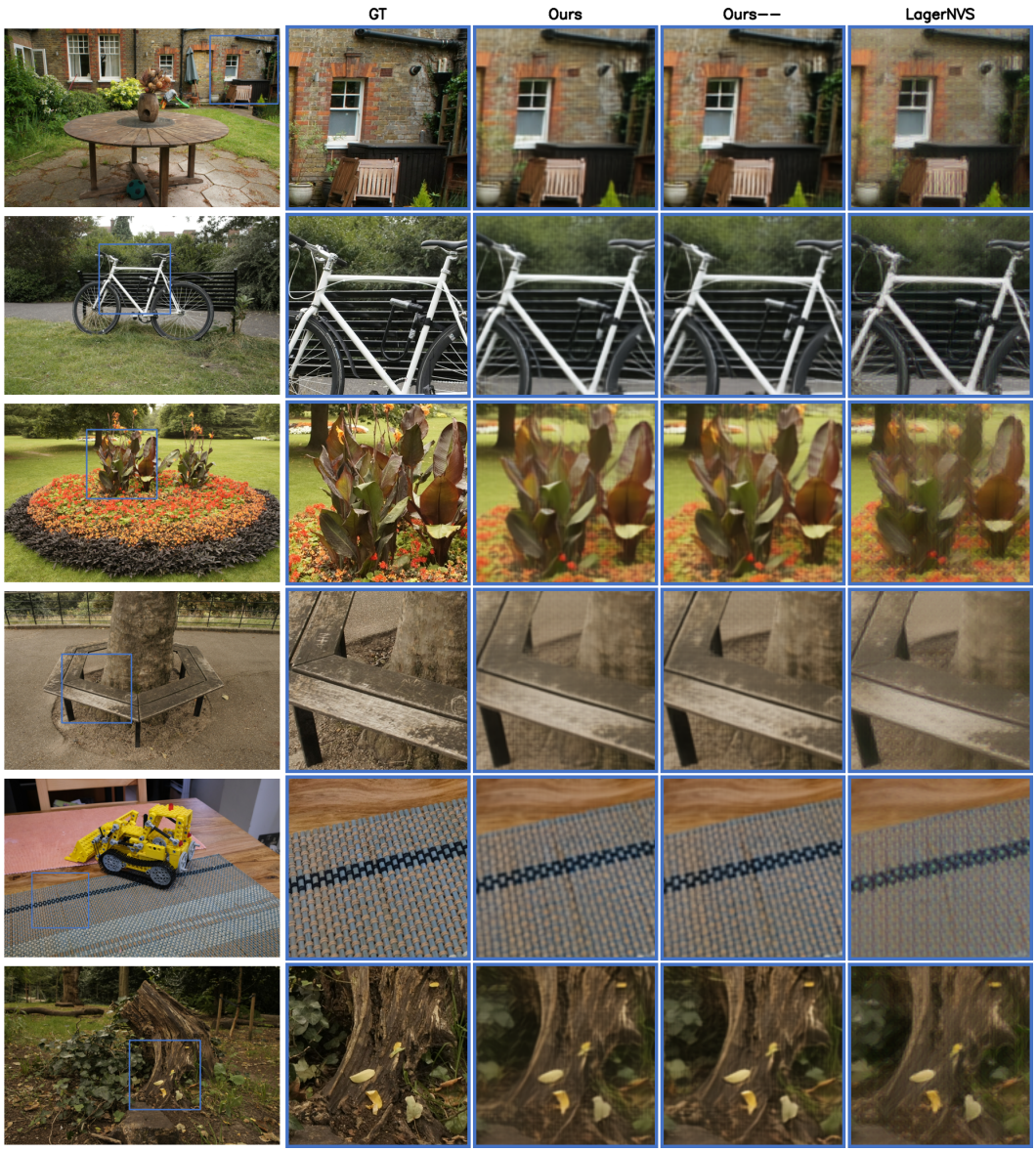

Figure 10: **Zero-shot qualitative results on MipNeRF360.** 64 context views.

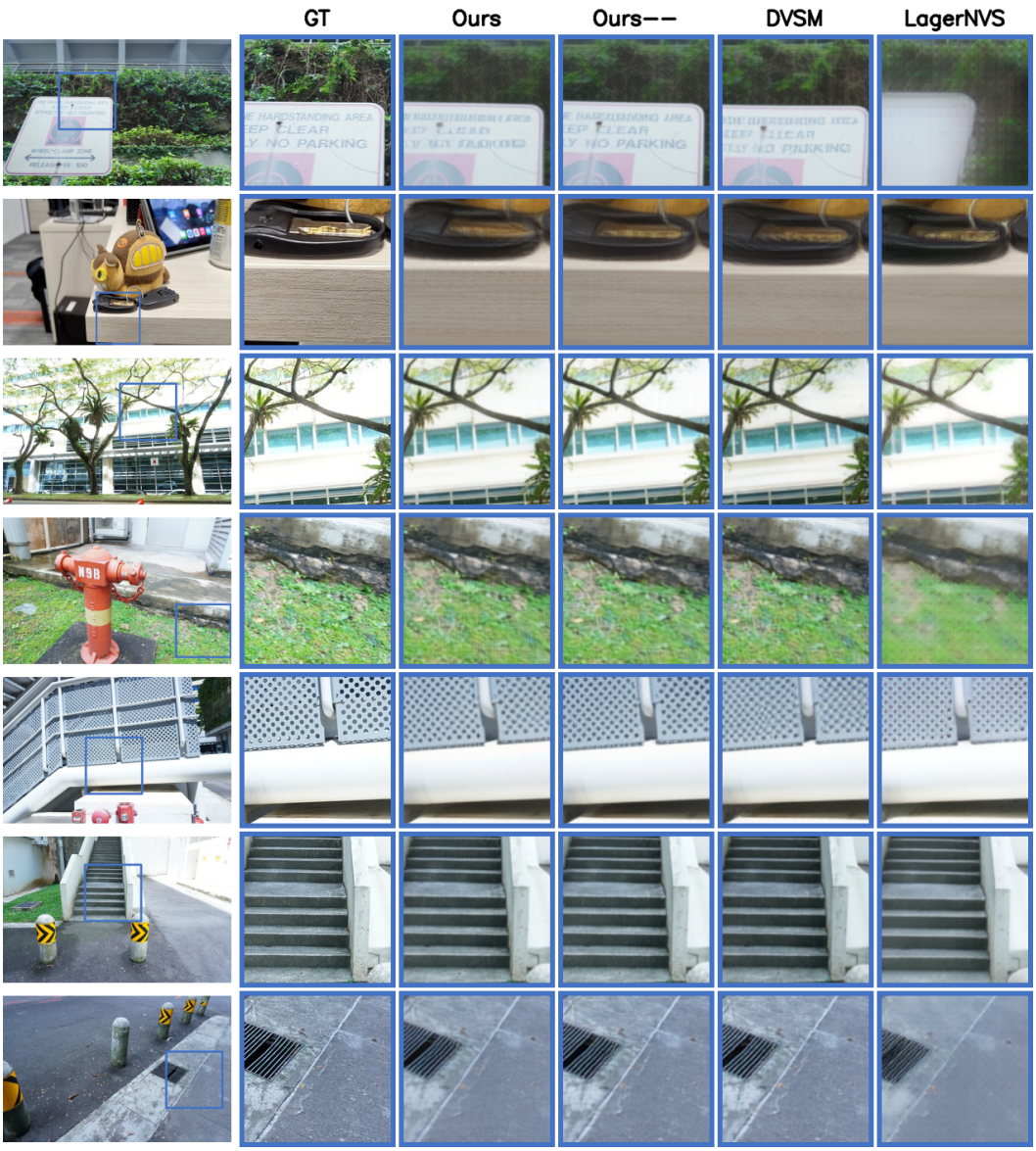

Figure 11: **Zero-shot qualitative results on Free.** 64 context views.

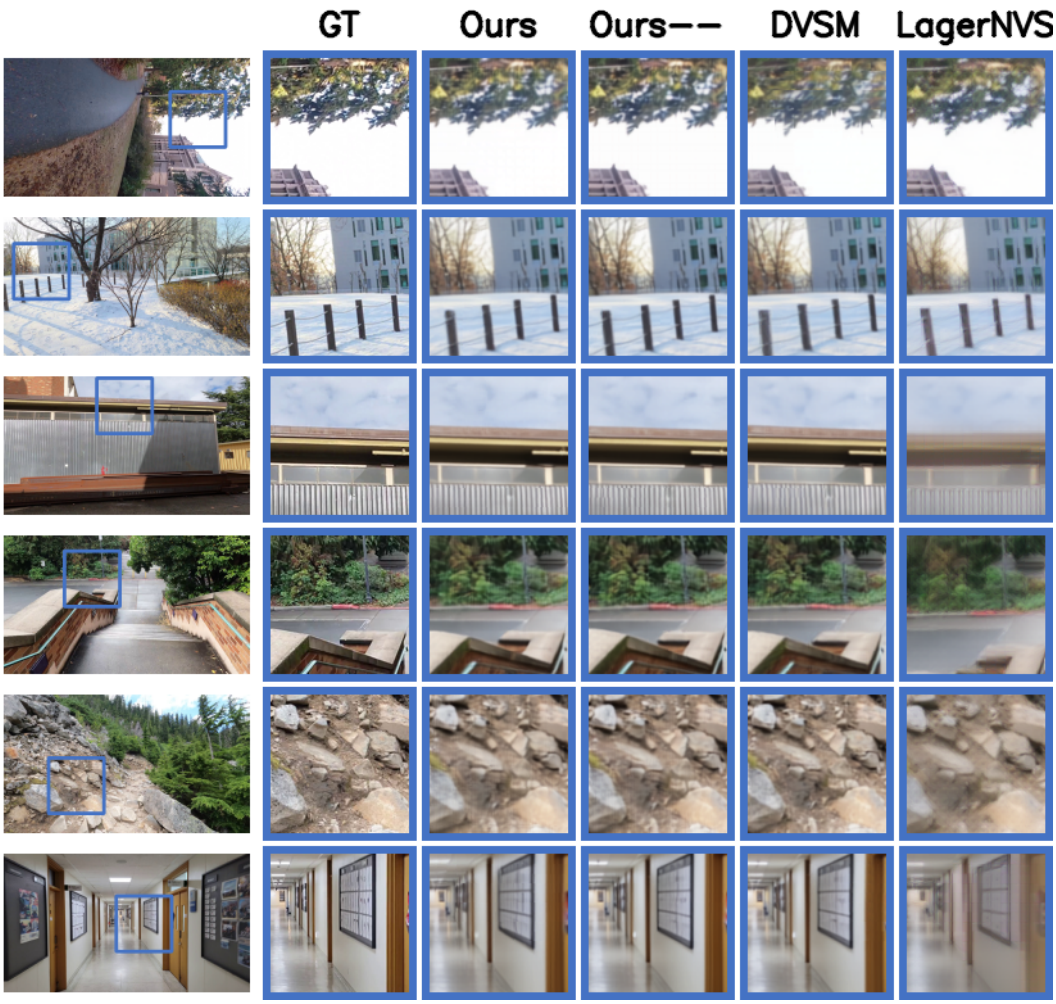

Figure 12: **Zero-shot qualitative results on Hike.** 64 context views.

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: The abstract and Sec. 1 state TTB's three contributions and 23× speedup; supported by Tabs. 1, 2, 3 and ablations Tabs. 4–6.

   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: Discussed at the end of Sec. 4.3: fixed MLP capacity, quadratic reconstruction in V, and reconstruction-only (no extrapolation) setting.

   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: [N/A]

Justification: The paper presents no formal theorems or proofs.

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: Sec. 3.2–3.3 fully describe the architecture and TTB layer; Sec. 4.1 and Appendix A.1 specify all hyperparameters, training recipe, and the 140-scene DL3DV split.

   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: Code and pretrained checkpoints are publicly available in the DVSM repository at `https://github.com/NVlabs/DVSM`, with instructions to reproduce the main results.

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: All training and evaluation settings are listed in Sec. 4.1, with full details in Appendix A.1–A.6.

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: Following standard practice in feed-forward NVS, we report metrics averaged over each benchmark's full test split without error bars due to the cost of training on 64 A100s.

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: Sec. 4.1 reports 64 A100s for main models and 8 A100s for ablations; per-method reconstruction time and rendering FPS are in Tabs. 1, 3, 4.

   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: The work uses only publicly released scene datasets and involves no human subjects or private data.

   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: Discussed in Appendix E.

    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: [N/A]

    Justification: The paper releases no high-risk assets; TTB is a reconstruction model trained on publicly released datasets.

    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, codebases, and prior models are cited at first use and used under their published terms.

    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: [Yes]

    Justification: Turntable videos and the interactive HTML viewer are documented and provided in the supplementary attachment.

    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: No crowdsourcing or 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: No 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: LLMs are not a core component; usage is limited to writing and coding aids, voluntarily documented in Appendix F.

    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.