Scaling View Synthesis Transformers

Evan Kim*
MIT
evnkim@mit.edu

Hyunwoo Ryu*
MIT
hwryu@mit.edu

Thomas W. Mitchel
Adobe
thomas.w.mitchel@gmail.com

Vincent Sitzmann
MIT
sitzmann@mit.edu

Abstract

Geometry-free view synthesis transformers have recently achieved state-of-the-art performance in Novel View Synthesis (NVS), outperforming traditional approaches that rely on explicit geometry modeling. Yet the factors governing their scaling with compute remain unclear. We present a systematic study of scaling laws for view synthesis transformers and derive design principles for training compute-optimal NVS models. Contrary to prior findings, we show that encoder–decoder architectures can be compute-optimal; we trace earlier negative results to suboptimal architectural choices and comparisons across unequal training compute budgets. Across several compute levels, we demonstrate that our encoder–decoder architecture, which we call the Scalable View Synthesis Model (SVSM), scales as effectively as decoder-only models, achieves a superior performance–compute Pareto frontier, and surpasses the previous state-of-the-art on real-world NVS benchmarks with substantially reduced training compute. https://www.evn.kim/research/svsm

1. Introduction

Given a set of images of a scene with known camera poses, the goal of Novel View Synthesis (NVS) is to render novel views of the scene from arbitrary viewpoints. Single scene approaches such as NeRF [16] and Gaussian Splatting [12] have achieved impressive fidelity by explicitly modeling 3D geometry and rendering. Feed-forward extensions of these frameworks train neural networks to reconstruct the 3D representation, achieving promising results [2, 3, 29, 31]. However, their formulation inherits handcrafted 3D structure, constraining their ability to scale and handle more complex artifacts such as reflections or transparency.

Typified by Large View Synthesis Model (LVSM) [10], a new class of view synthesis models have emerged which achieve state-of-the-art rendering quality using pure transformer architectures with fewer (if any) geometric inductive biases [9, 10, 17, 21]. However, this class of models is relatively new, and their design space remains unexplored. In particular, training a NVS model involves making design choices over a large number of variables, including the number of context and target views, camera pose parameterizations, attention mechanisms, etc., and there does not yet exist a rigorous investigation of how these choices affect the performance, training efficiency, and inference throughput. To this point, while there exist extensive scaling analyses in language modeling and 2D vision [7, 8, 11, 19, 30], there exists no analogue for 3D vision. Thus, the goal of this work is to provide such an analysis in addition to a compute-optimal training recipe for view synthesis transformers in terms of both architecture and training strategy.

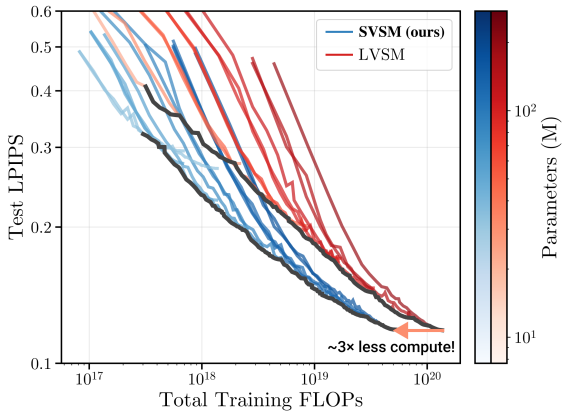

Figure 1. **Scaling Laws for View Synthesis Transformers.** Evaluated on RealEstate10K [34], our SVSM exhibits a 3× more compute-optimal Pareto frontier than LVSM while retaining the same scaling behavior (similar slope and curvature everywhere).

In particular, we first challenge the necessity of the decoder-only architecture proposed in LVSM [10]. While powerful, it requires passing all context images through the entire transformer each time a *single* target image is decoded. It is a *bidirectional* model where both the target view tokens *and* context view tokens are updated in each layer of the network. While this allows the model to consider only information in the context images that is relevant to the target view, it incurs substantial computational cost

*Indicates equal contribution.

due to repeated processing of context views.

Instead, we advocate for an encoder-decoder design which produces an intermediate scene latent representation. This approach is potentially far more efficient: the computational cost of constructing the scene representation is amortized through repeated calls to the decoder, which efficiently extracts information from the representation via *unidirectional* cross attention from scene to target. However, the scene representation also represents an information bottleneck. Without a proper training strategy that maximally leverages the efficiency of the encoder-decoder design, it can be challenging for encoder-decoder models to outperform decoder-only models. Here, we identify that the key for unlocking the potential of encoder-decoder models lies in the way target views (*i.e.* the reconstruction targets) are utilized during training. The implicit standard practice employed by prior work [2, 10, 21, 31] has been to reconstruct multiple different target views from a single scene during training. However, the consequences of this approach have never been fully analyzed. To this point, we propose and empirically validate the *effective batch hypothesis*, which argues that reconstructing multiple target views per scene effectively multiplies the batch size.

These insights yield a principled transformer view synthesis model, which we call the *Scalable View Synthesis Model* (SVSM), that fully capitalizes on the rendering efficiency of a unidirectional encoder-decoder architecture, maximizing training throughput without compromising performance or scalability. We demonstrate that our unidirectional model scales as efficiently as bidirectional models, which aligns with the scalability of causal, unidirectional attention in large language models [11]. As part of this analysis, we also reveal scaling relationships within view synthesis that parallel those observed in the Chinchilla language model family [8]. Finally, we demonstrate that SVSM achieves state-of-the-art results in real-world NVS tasks with significantly reduced compute, challenging the previous understanding that bidirectional attention is critical to high-fidelity view synthesis [10].

Key Contributions:

- We provide the first rigorous scaling analysis for novel view synthesis transformers.
- We propose and empirically confirm the *effective batch size hypothesis* that unlocks compute-optimal training.
- We show that bidirectional decoding is not critical for scalable view synthesis, contrasting recent work [10].
- Based on this analysis, we present a compute-optimal model that achieves a new state-of-the-art in real-world NVS tasks with substantially reduced training compute.

2. Related Work and Preliminaries

Generalizable novel view synthesis. In generalizable novel view synthesis, we are given a set of V_C context images paired with known camera poses and intrinsics $\mathfrak{C} = \{(I_i, g_i, K_i) \mid i = 1, \ldots, V_C\}$. The typical objective is to synthesize an *unseen* view of the same scene given a target camera configuration g_T, K_T:

$$\tilde{I}_T = \text{Render}\big[\mathfrak{C}, g_T, K_T\big]. \tag{1}$$

One line of work attempts to solve this problem with neural network architectures that explicitly model aspects of 3D image formation, for instance, via differentiable rendering or using epipolar line constraints [2, 23, 25, 26, 29]. In contrast, *geometry-free* methods avoid explicit geometric modeling in favor of flexibility and generality [6, 20–22, 24]. Here, we are primarily interested in a recently proposed subclass of these models which achieve state-of-the-art results with pure transformer architectures [9, 10, 17, 21]. In particular, we seek to study the "Large View Synthesis Model" (LVSM) [10], which achieves state-of-the-art NVS performance and serves as the prototypical instance of the view synthesis transformer. LVSM can be implemented in two ways: as either an encoder-decoder model or decoder-only model. The authors' proposed decoder-only variant is far more performant, so we primarily consider this architecture (pictured in Fig. 2, left) in our analysis.

Decoder-only LVSM consists of a single module: A decoder $\mathcal{D}$ which ingests the raw context $\mathfrak{C}$ along with a *single* target configuration, and aims to render a prediction of the target view $\tilde{I}_T = \mathcal{D}\big[\mathfrak{C}, g_T, K_T\big]$. This model is *bidirectional* as the context view tokens are updated with information about target pose and tokens in each layer. Thus, the processed context tokens cannot be reused and must be re-initialized and updated each time a new target view is rendered. As a consequence, rendering V_T target views requires V_T forward passes through the full network. Decoder-only LVSM consists of standard ViT layers [5] which apply self-attention (`Attn`) followed by an MLP (`MLP`). Therefore, the FLOPs on a forward pass scale linearly with the number of target views V_T

$$\begin{aligned} \chi_{\texttt{MLP}}^{(\text{LVSM})} &\propto V_T \times (V_C + 1) \\ \chi_{\texttt{Attn}}^{(\text{LVSM})} &\propto V_T \times (V_C + 1)^2 \end{aligned} \tag{2}$$

where $\chi_{\texttt{MLP}}^{(\text{LVSM})}$ and $\chi_{\texttt{Attn}}^{(\text{LVSM})}$ are the FLOPs consumed by the MLP and attention in the decoder-only LVSM. In our studies, $\chi_{\texttt{MLP}}^{(\text{LVSM})}$ is the dominating factor (for more details, see Supp 9). In what follows, we will continue to use χ to denote the compute metric, typically measured in FLOPs.

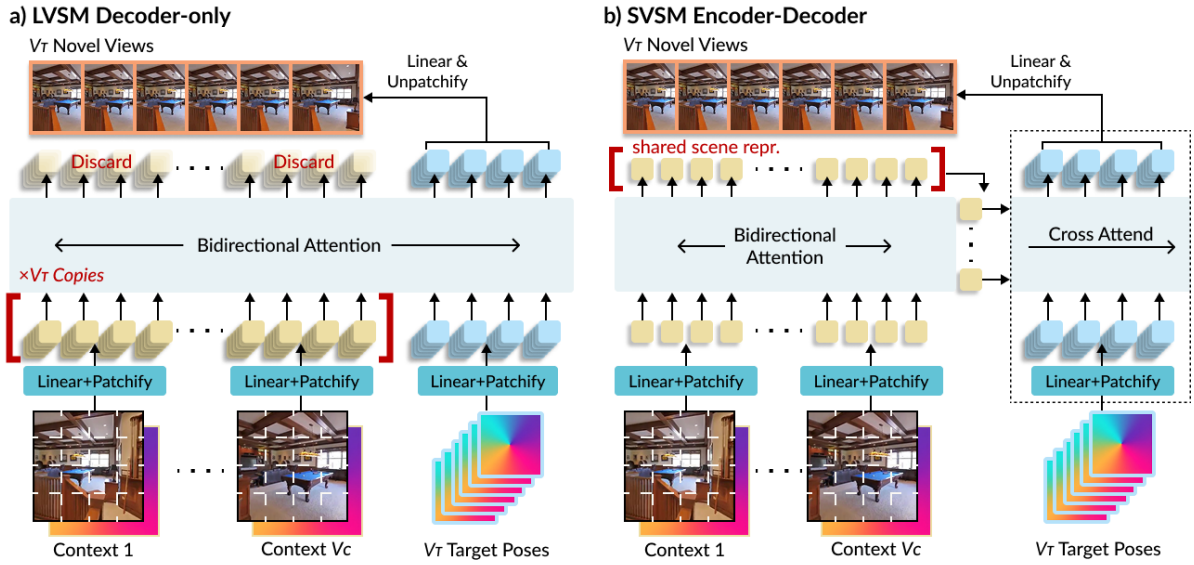

Figure 2. **Architectures of the current SOTA, the decoder-only LVSM [10] (a) and SVSM (ours, b).** Our cross-attention based decoder enables parallel rendering of multiple target views after a single scene encoding. Each target view is decoded independently given the shared scene representation, but the cross-attention allows these independent decodings to be executed in parallel.

Scaling Laws. As the scale of deep learning models continues to increase, it has become increasingly important to understand the relationship between performance and compute to ensure an efficient use of resources. To this end, scaling analyses have been conducted for language models [8, 11], vision transformers [30], and diffusion transformers [7, 19]. The general approach is straightforward: train models at different compute budgets and analyze performance as a function of compute.

Scaling studies are useful in two ways. First, they provide a predictable trend of performance with compute, essentially describing a performance metric P as function of compute χ which can reveal characteristic scaling behavior. For example, in language models, $P(\chi)$ has been found to approximately follow a power-law [11]. Second, they have revealed which hyperparameter choices are most effective as models scale. For instance, Chinchilla scaling laws [8] reveal the best way to trade-off between model size N, measured in parameter count, and the number of training samples used, D. Analysis is performed by sweeping across a wide range of N and D to discover for compute budget χ the optimal N and D. Then, taking this paired data and assuming a power law relation between N_{opt} and D_{opt} and χ, one can fit powers a and b of χ to the pairings

$$N_{\text{opt}}(\chi) \propto \chi^a, D_{\text{opt}}(\chi) \propto \chi^b. \tag{3}$$

Remarkably, experiments demonstrate $a \approx b$, suggesting N and D should scale proportionally. In our study, we will focus on the second of these two kinds of studies: replicating the Chinchilla study (Sec. 5) in the NVS domain and exploring other similar tradeoffs, such as the effective batch size (Sec. 4)

Extremely long context view synthesis. Recently, there has been a line of work aiming to develop view-synthesis and 3D-reconstruction models whose computational cost scales linearly with the number of context images [27, 33, 35]. Indeed, as in Eq. 2, as V_C grows, the quadratic cost of attention comes to dominate the compute and becomes infeasible. In that regime, such linear-cost models are promising alternatives. However, we restrict our focus to sparse to moderately sparse view synthesis, for which linear-cost models are currently not state of the art.

3. Encoder-Decoder View Synthesis

As discussed in Sec. 1 and Sec. 2, the decoder-only LVSM may not be the most compute-optimal due to the recomputation per-target view rendered. This motivates us to seek an alternative model with a fixed scene representation that can be decoded in a purely *unidirectional* manner (*i.e.* via cross attention) to reduce the cost of rendering and avoid redundant reprocessing of context information. To this extent, we introduce the Scalable View Synthesis Model (SVSM), which can be viewed as a simple modification to encoder-decoder LVSM. Specifically, our architecture implements Eq. 1 by first processing the context set $\mathfrak{C}$ with a transformer encoder, producing a set of latent tokens (a "scene representation") $\mathbf{z} = \mathcal{E}[\mathfrak{C}]$. The encoder $\mathcal{E}$ is standard transformer with full bidirectional self-attention. Unlike encoder-decoder LVSM, we do not employ a fixed-size scene representation, but instead take the set of encoded context image patch tokens as the scene representation to avoid introducing a bottleneck. To render a novel view, a cross-attention based decoder $\mathcal{D}$ ingests the target configuration and the fixed scene latent tokens $\mathbf{z}$ to render the target view, $\tilde{I} = D[\mathbf{z}, g_T, K_T]$. To render multiple images

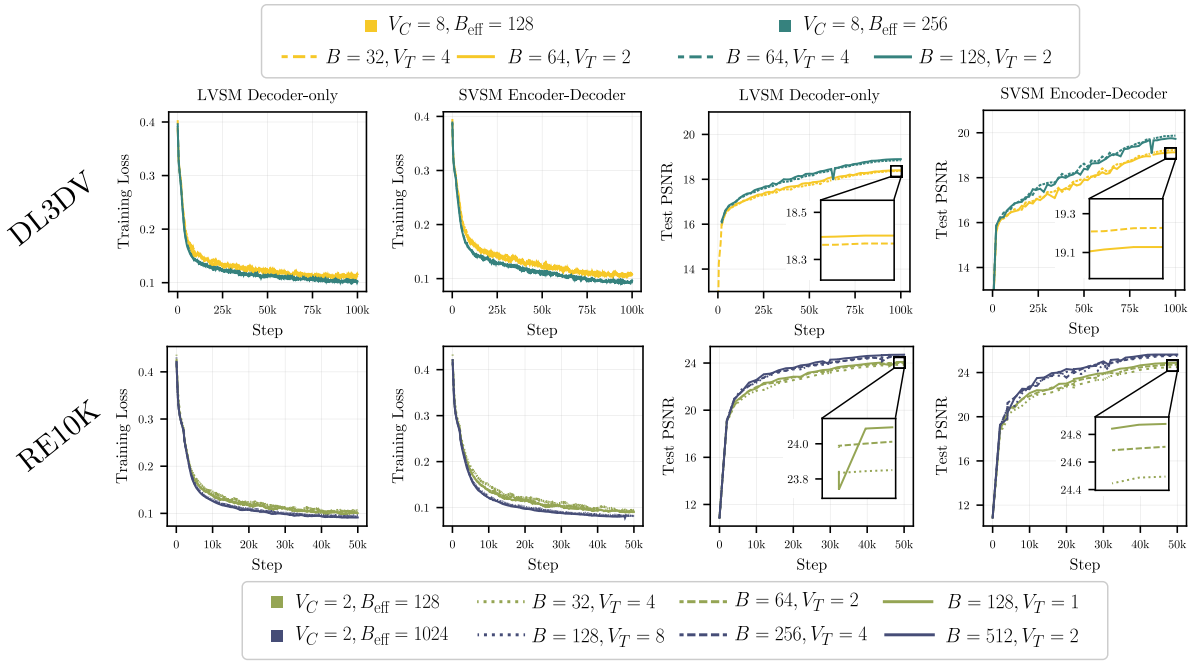

Figure 3. **Effective Batch Size**. Training loss (smoothed with a rolling-average) and test PSNR measured throughout training across various paired B and V_T runs provide evidence for our effective batch size hypothesis: Models trained with the same product of number of scenes in the batch B and number of reconstruction target views V_T, *i.e.* runs with the same *effective batch size* B_{eff}, perform the same and are colored identically. On $V_C = 8$ **(top)**, we sweep across $B_{\text{eff}} = 128, 256$ on DL3DV, and on $V_C = 2$ **(bottom)**, we sweep across $B_{\text{eff}} = 128, 1024$ on RealEstate10K.

of the same scene, we only require encoding the context set once, re-using the scene embedding $\mathbf{z}$. As with LVSM, each novel view is decoded independently given $\mathbf{z}$ (i.e., there is no interaction between target views). However, because the decoder uses cross-attention rather than bidirectional self-attention over all tokens, these independent target views can be decoded in parallel, without redundant recomputation of the scene representation.

To be more concrete, this architecture reduces the complexity of rendering V_T targets to

$$\begin{aligned} \chi^{(\text{SVSM})}_{\texttt{MLP}} &\propto V_T + V_C \\ \chi^{(\text{SVSM})}_{\texttt{Attn}} &\propto V_C \times (V_T + V_C). \end{aligned} \tag{4}$$

In other words, assuming most of the compute is due to the MLP layers (see Supp 9), rendering V_T target views requires $\mathcal{O}(V_T + V_C)$ FLOPs. In the limit of inference where $V_T \gg V_C$, this reduces to $\mathcal{O}(V_T)$, in stark contrast to the $\mathcal{O}(V_T V_C + V_T)$ of LVSM (see Eq. 2). Further, the benefit of this paradigm extends beyond inference. As long as we are training with multiple target views, as is standard practice [2, 10, 21, 31], the parallel nature of unidirectional decoding can save substantial training compute.

However, this reduction comes with a cost. Unlike LVSM, our encoder cannot proactively discard information irrelevant to the target view but instead needs to encode all necessary information for rendering *any* target view. Indeed, parameter count and training steps being equal, SVSM performs worse than LVSM. However, as we will show, SVMS's amortized rendering enables us to dramatically increase its size and training steps, such that when normalized by compute budget, SVSM significantly outperforms LVSM.

4. The Effective Batch Size for View Synthesis

As the cost of a forward pass scales both with the number of different scenes (the batch size B) as well as the number of target views (V_T) that we seek to render per scene, this introduces an additional hyperparameter into the training regime: What is the *optimal* trade-off between the number of target views and the number of different scenes? We study this question empirically and reveal that what matters is the *product* of target views and batch size, which we call the *effective batch size* of a NVS model.

Analysis Setup. We define effective bath size as $B_{\text{eff}} \equiv B \cdot V_T$, where B is the number of scenes in a training batch, and V_T is the number of rendering targets used per training scene. We train both decoder-only LVSM and the proposed SVSM models across two datasets — DL3DV [15] and RealEstate10K [34] (RE10K) — with $V_C = 8$ and $V_C = 2$ while holding B_{eff} constant and varying B and V_T. Specifically, we test $B_{\text{eff}} = 128$ on both datasets, and additionally test $B_{\text{eff}} = 1024$ on RE10K and $B_{\text{eff}} = 256$ on

DL3DV. For DL3DV we use the official test-train split, and for RE10K, we use the pixelSplat [2] test-train split. Further training details are outlined in Supp 10.

Effective Batch Size is What Matters. Results for all training runs are shown in Fig. 3. Remarkably, in all cases – across both models, both V_C settings, and all B_{eff} sets – the test metric and the training loss behavior remain approximately constant along a B_{eff}-level set. This effect is especially clear in the $V_C = 8$ case, where the test PSNR varies by at most ± 0.1 and remains present in the $V_C = 2$ case, where the variation is at most ± 0.2 PSNR. Tuning B and V_T within the same effective batch size B_{eff} does not result in significant difference in the final performance outcome, we exclude this degree of freedom from our subsequent analyses and treat B_{eff} as the true batch size.

SVSM Enables Compute-Optimal Tradeoff. How can we interpret this result through the lens of compute-optimiality? For the LVSM decoder-only model, the training compute scales as

$$\chi^{(\text{LVSM})} \propto BV_T(V_C + 1) = B_{\text{eff}}(V_C + 1). \tag{5}$$

Thus, any training settings within constant B_{eff} not only achieve within-noise results (as per our effective batch result), but also require the same number of FLOPs. This means for the decoder-only model there is *no advantage to be gained by tuning* V_T. In contrast, for the SVSM model, training compute is proportional to

$$\chi^{(\text{SVSM})} \propto B(V_C + V_T) = B_{\text{eff}} + BV_C. \tag{6}$$

Therefore, by reducing B and increasing V_T, one can achieve the same effective batch size – and consequently, the same performance – with lower compute cost. This justifies our original motivation for a model design that efficiently decodes multiple V_T.

5. Scaling Laws for Stereo (V_C=2) NVS

Analysis Setup. We first experiment in the most classical setting for feed-forward Novel View Synthesis – stereo synthesis with two context views. All training and evaluation for the $V_C = 2$ case is done on RealEstate10K [34]. As before, we follow the test-train split of pixelSplat [2], along with the same evaluation framework. For training, we use $V_T = 6$ target views per training example, following the setup of [10]. We use a batch size of 256 for all experiments. We use a patch size of $p = 16$ for all experiments, except in the case of table 2, where we use $p = 8$ to compare against the reported state-of-the-art numbers from LVSM. To ensure stable scaling of both models, we also apply a $1/\sqrt{L}$ multiplier to the residuals, where L is the depth

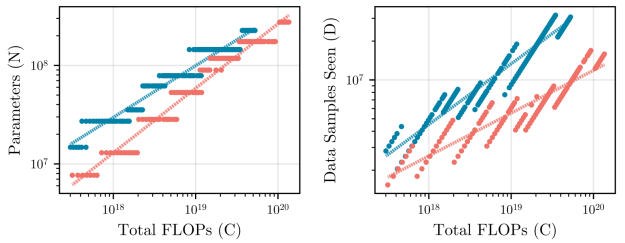

Figure 4. **Data and Model Scaling Plots.** While our model (blue) is optimal when sufficient data is available, decoder-only LVSM (red) performs better with less data. The Pareto frontier analysis shows that our model is more data-hungry. Our model is also less parameter-efficient, although the gap closes as we increase the training compute. However, with sufficient data and compute, our model (blue) is overall superior in terms of training compute-optimality and rendering speed.

of the transformer, following ideas from depth-μP [1, 28]. Further training details can be found in the supplementary material. We use the test LPIPS [32] loss as our primary performance metric, as it produces near linear trends on log-log plots against FLOPs.

Scaling Laws. We now follow the approach in language modeling [8] to answer the question: *for a given compute-budget, what is the optimal performance that can be attained?* For both model families, we sweep across a range of models from around 7M to 300M parameters, training each model for 3-4 different sample counts to densely cover the FLOP range [8]. Our training runs span a compute range of 10^3 magnitudes: 100 petaflops to 100 exaflops. From this data, we are then able to determine a mapping from compute budgets C to minimum test LPIPs – the Pareto frontier.

We plot results in Fig. 1, with their Pareto frontiers marked in dark gray. Plotted on a log-log scale, both models exhibit a consistent downward trend on test LPIPS with more compute. More significantly, the Pareto frontiers of both families have approximately the same slope at points of the same performance (see Supp 11), and SVSM's frontier is shifted left by a factor of 3. Thus, as an initial result, our scaling laws show us that our encoder-decoder architecture *scales exactly the same as the decoder-only LVSM while using* $3\times$ *less training-compute.* Qualitative results of this scaling are shown in Fig. 5, and we see that when FLOP-matched, SVSM has better rendering quality.

Optimal Model Choice. From our scaling experiments, we can further extract a compute-optimal training recipe for our view synthesis transformers, as demonstrated by Hoffmann et al. [8]. For each compute budget χ, we determine the corresponding optimal model size N and the amount of training data D used at that point. Then, plotting N and D against χ, we can extract the Chinchilla scaling equations (Eq. 3) by fitting lines onto the log-log plots in Fig. 4. The recovered coefficients are shown in Tab. 3, which inform

Model	Scale Parameters			Reconstruction Quality			Rendering FPS (↑)		
	Model Size	Train Iters	Train FLOPs (↓)	PSNR (↑)	SSIM (↑)	LPIPS (↓)	V_C=2	V_C=4	V_C=8
LVSM Encoder-Decoder [10]	173M	100k	2.53 zflops	28.58	0.893	0.114	53.7	52.9	**52.7**
LVSM Decoder-Only [10]	171M	100k	1.60 zflops	29.67	0.906	0.098	37.9	19.5	8.6
SVSM Enc-Dec (**ours**, Iter-matched)	740M	100k	**0.74 zflops**	29.80	0.907	0.098	48.6	42.7	35.0
SVSM Enc-Dec (**ours**, Pareto-optimal)	416M	170k	**0.77 zflops**	**30.01**	**0.910**	**0.096**	**71.0**	**61.8**	49.7

Table 1. **Stereo ($V_C = 2$) NVS Results of the Largest Models.** All models use a patch size of 8 with input images at 256×256 resolution. Our models achieve the highest reconstruction metrics while using less than half of the training compute. The rendering FPS of both SVSM models is also much faster than that of the LVSM decoder-only model, though both are slower than the LVSM encoder-decoder when V_C is large.

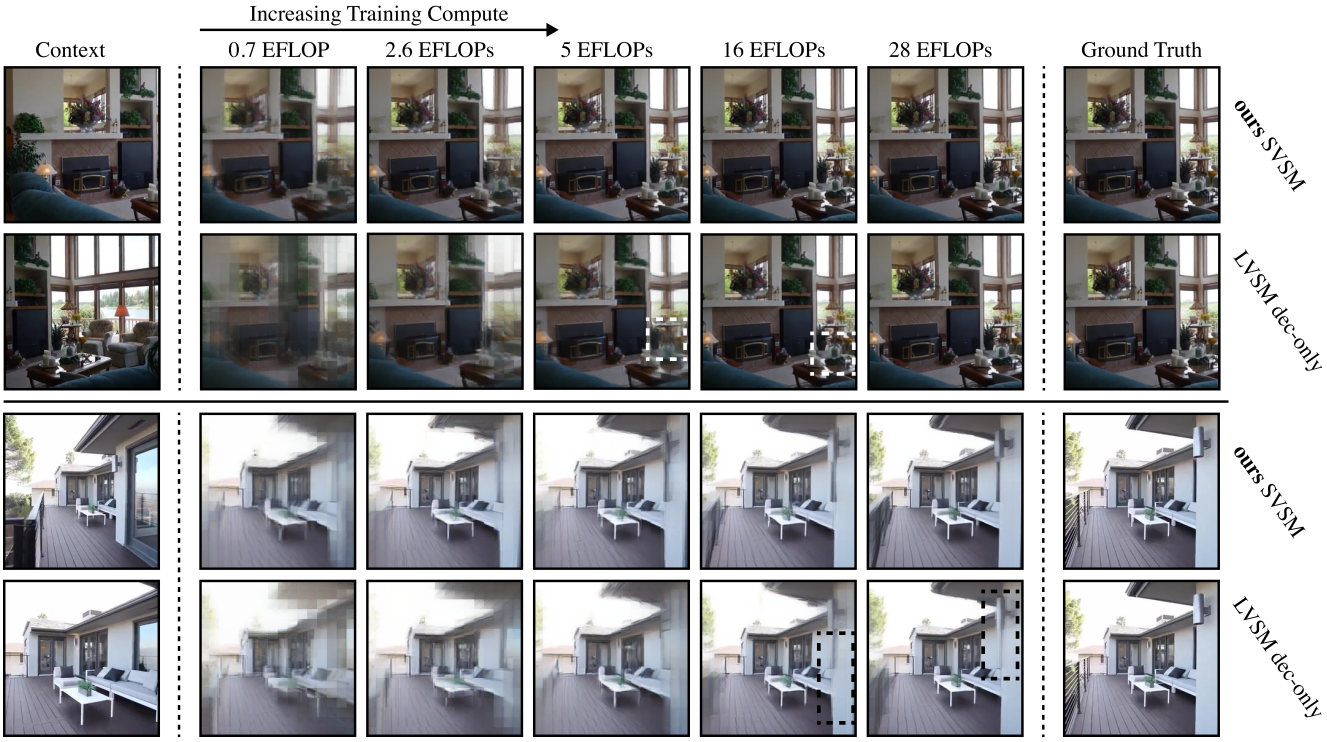

Figure 5. **Qualitative Scaling Behavior**, $V_C = 2$. From left to right, both models steadily increase in rendering quality until reaching near photo-realistic results. Compared vertically, for a given compute-budget, SVSM renderings consistently contain less artifacts.

Model	Recon Quality		
	PSNR (↑)	SSIM (↑)	LPIPS (↓)
pixelNeRF [29]	20.43	0.589	0.550
pixelSplat [2]	26.09	0.863	0.136
MVSplat [3]	26.39	0.869	0.128
GS-LRM [31]	28.10	0.892	0.114
SVSM Enc-Dec (**ours**)	**30.01**	**0.910**	**0.096**

Table 2. **Comparison to Geometry-Aware Methods.** Our method achieves a new state-of-the-art on RealEstate10K [34] with the set from Charatan et al. [2], outperforming not only LVSM, but also prior work with explicit 3D structure.

how to train models which end on the frontier.

From these results it follows that for SVSM, if we increase our compute budget by a factor of $k\times$, it should approximately be equally allocated between increasing the model size by $\sqrt{k}$ and increasing data sample count by $\sqrt{k}$ as $a_{\text{SVSM}} \approx b_{\text{SVSM}}$, matching the findings of the Chinchilla

Model	Parameter Coeff. a	Data Coeff. b
LVSM	0.65	0.33
SVSM	0.52	0.47

Table 3. **Parameter and Data Scaling Coefficients.** As regressed from the plots in Fig. 4, we find power law coefficients for scaling models and data with respect to compute.

scaling laws [8] for language models. For LVSM this relationship does not seem to hold exactly, but the fit still shows that requires significant scaling of data with respect to compute, though to a smaller power.

Notably our data sample counts include *repeated scene data*, as we only have access to small, pose-labeled datasets. This differs from standard scaling practice, in which models are typically trained for less than one full epoch [8, 11, 30]. Although we have not yet seen evidence of overfitting in

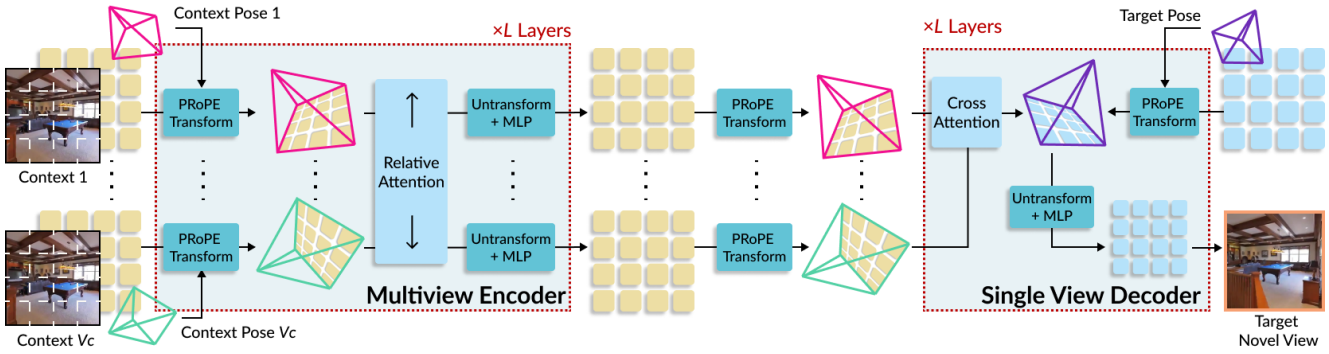

Figure 6. **Multiview PRoPE.** We find that multiview projective RoPE embeddings [13, 14, 18] are critical for our model to scale with compute and data in the multiview setting ($V_C > 2$). For each layer of the multiview transformer encoder, PRoPE embeddings use context camera poses to transform context view tokens into a common coordinate frames before the attention layer, and apply the inverse transformation before each MLP. To render, both context features and query tokens of the target view are transformed by PRoPE before cross-attention.

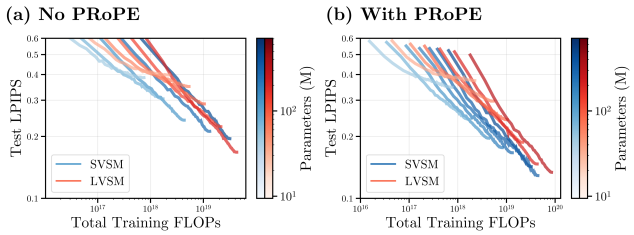

Figure 7. **Multiview Scaling Behavior.** Conducted on DL3DV [15]. **(a)** For V_C>2, without PRoPE, SVSM saturates and stops scaling much more quickly than LVSM. **(b)** When PRoPE is added, SVSM continues scaling with a better Pareto-frontier.

our experiments – perhaps due to the diversity of view sequences which are sampled during training – we have shown that increasing scale requires increasing the number data samples. Thus, having access to larger amounts of diverse posed data will be essential for developing large-scale generalizable NVS models.

SVSM-420M/740M Results. Finally, combining our scaling law findings, we train two separate models — SVSM-420M and SVSM-740M, aptly named to denote their parameter counts — to compare against the original results of LVSM's largest model on RealEstate10K. Due to compute-constraints, we train our models at a lower total budget of around 10^{21} FLOPs and a batch size of 256, approximately half the FLOPs and exactly half the batch size used by LVSM. We train two models under this budget: (1) a flop-matched model with the 24 layer LVSM model for a forward pass of a single training sample with $V_C = 2, V_T = 6$ and; (2) A model whose parameter count is given by plugging the budget[1] χ and the coefficients from Tab. 3 to Eq. 3.

[1]To be more specific, we plug $\chi/4$ into the scaling law, in order to adjust for the scaling law being derived off of 16×16 patch experiments, while this final budget is under 8×8 patches, which requires roughly $4\times$ as much compute.

While we train with under half the compute, our scaling laws in Sec. 5 predict equal performance with three times less compute. Thus, as predicted by our scaling laws and validated empirically by the results in 1 both SVSM models outperform decoder-only LVSM. Notably, SVSM-420M, the model trained in accordance with our scaling laws performs the best. For completeness, we also show reported results from prior work on this benchmark in Tab. 2.

Furthermore, we also benchmark the *rendering speed*, which is calculated with $V_T = 1$ to simulate real-time online rendering with respect to a stream of input poses. Additional details can be found in Supp 10.2. As seen in Tab. 1, SVSM generally renders much faster than decoder-only LVSM, and is eclipsed only once V_C increases to 8.

6. Scaling Laws for Multiview (V_C>2) NVS

Analysis Setup. We continue to experiment in the multiview paradigm (V_C>2), which necessitates the reconciliation of scene information across many views to produce quality renderings. Specifically, we focus on $V_C = 4$ regime. For training and evaluation, we choose DL3DV [15], a real-world dataset with wider baselines and more complex camera trajectories and subject matter, making it more suitable for multiview experiments. We follow the official test-train split and use V_T=4, and scene batch size of 64 for all experiments to save resources. All other settings follow those described in Sec. 5 and Supp 10.

Scaling Law Does Not Hold. Unfortunately, we find that naively extending our SVSM architecture to the multiview scenario does not result in a similar scaling trend. As can be seen from Figure 7a, the Pareto frontier of our unidirectional model saturates much quicker than bidirectional LVSM as we increase the train compute.

Relative Camera Attention Re-establishes Scaling Law. We hypothesize that this is not a fundamental problem of

Model	Scale Parameters			Reconstruction Quality			Rendering FPS (↑)		
	Model Size	Train Iters	Train FLOPs (↓)	PSNR (↑)	SSIM (↑)	LPIPS (↓)	V_C=4	V_C=8	V_C=16
LVSM Decoder-only + PRoPE [10, 14]	171M	100k	43 eflops	26.19	0.830	0.145	104.7	52.6	23.8
SVSM Enc-Dec (**ours**, ≈ Iter-matched)	711M	100k	32 eflops	26.29	0.835	0.141	280.4	261.2	230.4
SVSM Enc-Dec (**ours**, Pareto-optimal)	400M	233k	44 eflops	**26.87**	**0.853**	**0.129**	**411.1**	**381.1**	**333**

Table 4. **Multiview (V_C>2) NVS Results of the Largest Models.** Our compute-matched model achieves significantly better rendering quality (+0.68 PSNR, −0.016 LPIPS), while maintaining nearly four times the rendering speed at inference-time.

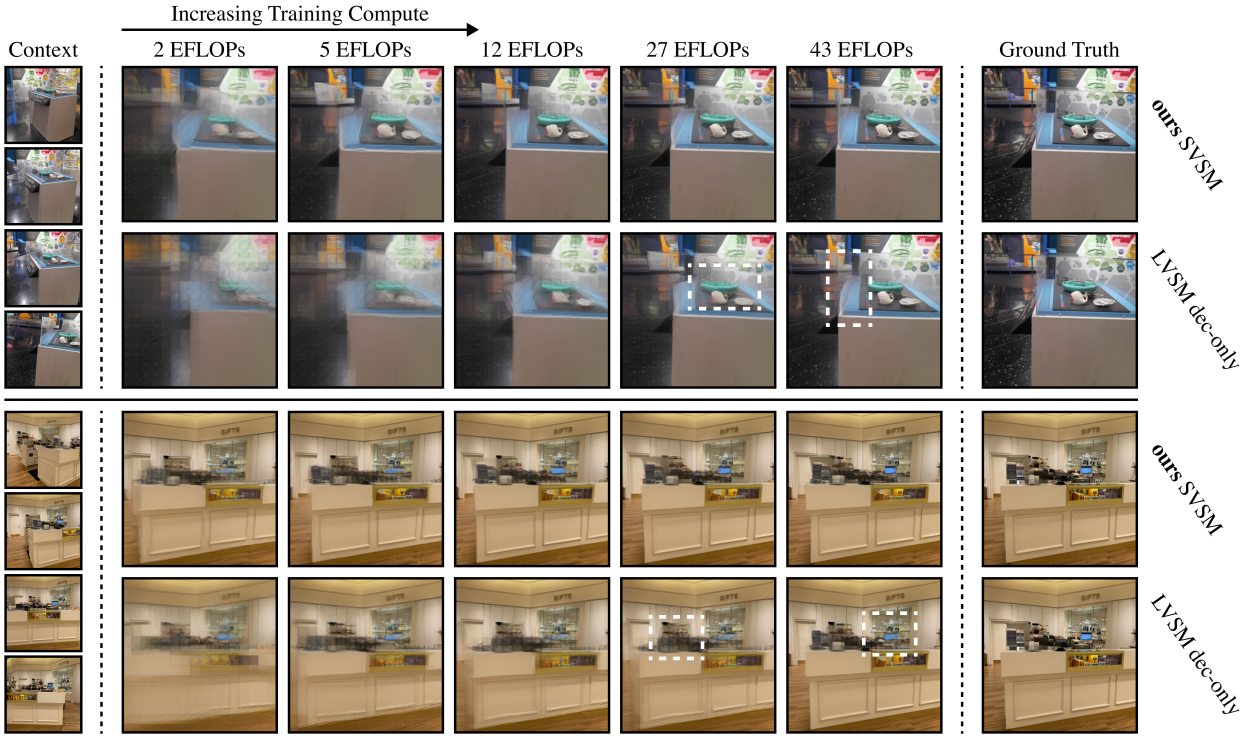

Figure 8. **Qualitative Scaling Behavior**, $V_C = 4$. The performance of both LVSM and our method steadily increases with compute (left to right). Compared vertically, for a given compute budget SVSM renderings are consistently less blurry.

encoder-decoder paradigm, but a problem caused by the way our model utilizes the pose information. Specifically, we find that adding a form of relative camera attention [13, 14, 18] resolves this issue.

Let g_i be the camera pose of the view that the i-th token belongs to. Relative camera attention models leverage attention mechanisms that only depend on the relative camera poses $g_{ij}{=}g_i^{-1}g_j$. This is typically achieved by 1) mapping the query, key, and value vectors to an arbitrary global reference frame, 2) performing attention there, and then 3) mapping back the results to each token's own reference frame. This mechanism embeds the pose information directly into the attention layers, ensuring that it isn't lost after the initial embedding. This potentially explains the efficacy of the method for our model, which may lose the pose information through the bottleneck otherwise.

For our model, we adopt the recently proposed PRoPE [14] embedding as the relative camera attention mechanism. We illustrate SVSM's architecture with the incorporation of PRoPE embeddings in Fig. 6. After adding PRoPE embeddings to both LVSM and SVSM, we retrain all models. Results are shown in Fig. 7. The equivalent scaling is re-established, and SVSM again maintains a tighter Pareto-frontier. Qualitative scaling results for both models are shown in Fig. 8. Note that while both models benefit from PRoPE embeddings, the advantage is far more pronounced in our encoder-decoder SVSM.

Final Models. Equipped with PRoPE embeddings, we again train larger models to compare directly against the 24-layer LVSM Decoder-only model, also equipped with PRoPE. As we did in the stereo case, we train a naive forward pass-matched model along with a Pareto-optimal model from a $N(\chi)$ fit to the data. The performance of both models are listed in Tab. 4 and their test loss curves are plot-

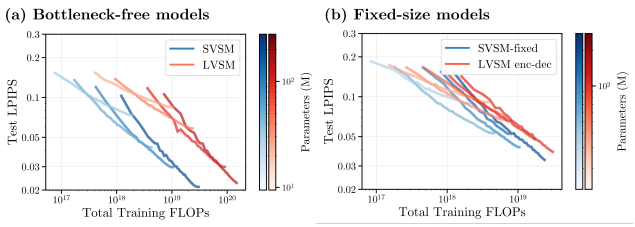

Figure 9. **Fixed-size Latent Scaling Experiments.** Conducted on Objaverse [4]. **(a)** For V_C=8, SVSM and LVSM decoder-only scale equally while SVSM's frontier is shifted by $5\times$ on the compute axis. **(b)** When a fixed latent bottleneck is used, SVSM-fixed and LVSM encoder-decoder scale equally, but significantly worse than the unbottlenecked designs. SVSM again maintains a superior pareto frontier.

ted in Fig. 7. Again, the version of SVSM which follows the scaling laws outperforms both models, with significant 0.7 PSNR and -0.016 LPIPS gaps. Beyond the superior reconstruction quality, the efficiency of SVSM becomes clear in the multi-view case, with a rendering FPS that is $4\times$ that of the decoder-only model, increasing to $14\times$ when extrapolated to larger context view counts.

7. Scaling Laws for Fixed Latent Design

Analysis Setup. Lastly, we check both the scaling laws and the design space of view synthesis models with a fixed-size scene representation. This design has favorable rendering speeds for large context lengths, though it does not have favorable training compute as the encoder is still quadratic in the context length. For training and evaluation, we use Objaverse [4], with 8 context views (where the benefit of having a fixed latent starts to appear at inference time). We compare two designs: LVSM Encoder-Decoder and SVSM-fixed, which follows the same design of a unidirectional decoder but instead decodes off of a fixed latent. Both models utilize PRoPE with identity pose on the scene representation. We use $V_T = 8$ and scene batch size 64 for an effective batch size of 512 for all experiments. All other settings follow those described in Sec. 5 and Supp 10.

SVSM-Fixed Matches LVSM Scaling, but Both Scale Poorly. As shown in Fig. 9b, SVSM with a fixed latent and the LVSM encoder–decoder exhibit similar scaling behavior. However, SVSM-fixed consistently requires less compute to achieve the same performance, maintaining a Pareto advantage. This indicates that our unidirectional decoder remains more compute-efficient even when a fixed latent bottleneck is imposed, which is desirable when amortized rendering is required. Nevertheless, comparing to Fig. 9a makes clear that both fixed-latent designs scale substantially worse than their bottleneck-free counterparts.

8. Conclusion

In this work, we established a rigorous compute-normalized benchmark for transformer view synthesis models. Our empirical studies reveal the importance of the concept of *effective batch size* —the product of the number of scenes in a batch with the number of per-scene rendering target views — which redefines the notion of batch size for NVS training. Based on this insight, we propose the Scalable View Synthesis Model (SVSM), which features a unidirectional encoder-decoder architecture for favorable scaling with effective batch. We demonstrate that SVSM is dramatically more compute-efficient than the current SOTA architecture, LVSM, and consistently achieves the same performance with $2-3\times$ less training compute. We further demonstrate that relative camera pose embeddings in multi-view attention is the key to realizing favorable scaling behavior with increasing numbers of context views. Lastly, we show that even with a fixed-size latent representation, our unidirectional decoder is still more compute-efficient than the LVSM encoder–decoder architecture; however, both approaches scale substantially worse than the designs without a latent bottleneck. In sum, our findings establish a new framework for evaluating the performance and effectiveness of transformer view synthesis models.

Acknowledgements. This work was supported by the National Science Foundation under Grant No. 2211259, by the Singapore DSTA under DST00OECI20300823 (New Representations for Vision, 3D Self-Supervised Learning for Label-Efficient Vision), by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior/ Interior Business Center (DOI/IBC) under 140D0423C0075, by the Amazon Science Hub, by the MIT-Google Program for Computing Innovation, and by a 2025 MIT Office of Research Computing and Data Seed Grant.

References

[1] Blake Bordelon, Lorenzo Noci, Mufan Bill Li, Boris Hanin, and Cengiz Pehlevan. Depthwise hyperparameter transfer in residual networks: Dynamics and scaling limit. *arXiv preprint arXiv:2309.16620*, 2023. 5

[2] David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19457–19467, 2024. 1, 2, 4, 5, 6

[3] 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 *European Conference on Computer Vision*, pages 370–386. Springer, 2024. 1, 6

[4] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs,

Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. *arXiv preprint arXiv:2212.08051*, 2022. 9

[5] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 2

[6] SM Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S Morcos, Marta Garnelo, Avraham Ruderman, Andrei A Rusu, Ivo Danihelka, Karol Gregor, et al. Neural scene representation and rendering. *Science*, 360 (6394):1204–1210, 2018. 2

[7] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *International Conference on Machine Learning*, pages 12606–12633. PMLR, 2024. 1, 3

[8] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022. 1, 2, 3, 5, 6

[9] Hanwen Jiang, Hao Tan, Peng Wang, Haian Jin, Yue Zhao, Sai Bi, Kai Zhang, Fujun Luan, Kalyan Sunkavalli, Qixing Huang, et al. Rayzer: A self-supervised large view synthesis model. *arXiv preprint arXiv:2505.00702*, 2025. 1, 2

[10] 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. *arXiv preprint arXiv:2410.17242*, 2024. 1, 2, 3, 4, 5, 6, 8, 12, 13

[11] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. 1, 2, 3, 6

[12] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023. 1

[13] Xin Kong, Shikun Liu, Xiaoyang Lyu, Marwan Taher, Xiaojuan Qi, and Andrew J Davison. Eschernet: A generative model for scalable view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9503–9513, 2024. 7, 8

[14] Ruilong Li, Brent Yi, Junchen Liu, Hang Gao, Yi Ma, and Angjoo Kanazawa. Cameras as relative positional encoding. *arXiv preprint arXiv:2507.10496*, 2025. 7, 8, 14

[15] Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, et al. Dl3dv-10k: A large-scale scene dataset for deep learning-based 3d vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22160–22169, 2024. 4, 7

[16] 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. *Communications of the ACM*, 65(1):99–106, 2021. 1

[17] Thomas W Mitchel, Hyunwoo Ryu, and Vincent Sitzmann. True self-supervised novel view synthesis is transferable. *arXiv preprint arXiv:2510.13063*, 2025. 1, 2

[18] Takeru Miyato, Bernhard Jaeger, Max Welling, and Andreas Geiger. Gta: A geometry-aware attention mechanism for multi-view transformers. *arXiv preprint arXiv:2310.10375*, 2023. 7, 8, 14

[19] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023. 1, 3

[20] Robin Rombach, Patrick Esser, and Björn Ommer. Geometry-free view synthesis: Transformers and no 3d priors. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14356–14366, 2021. 2

[21] Mehdi SM Sajjadi, Henning Meyer, Etienne Pot, Urs Bergmann, Klaus Greff, Noha Radwan, Suhani Vora, Mario Lučić, Daniel Duckworth, Alexey Dosovitskiy, et al. Scene representation transformer: Geometry-free novel view synthesis through set-latent scene representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6229–6238, 2022. 1, 2, 4

[22] Mehdi SM Sajjadi, Aravindh Mahendran, Thomas Kipf, Etienne Pot, Daniel Duckworth, Mario Lučić, and Klaus Greff. Rust: Latent neural scene representations from unposed imagery. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17297–17306, 2023. 2

[23] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems*, 2019. 2

[24] Vincent Sitzmann, Semon Rezchikov, William T. Freeman, Joshua B. Tenenbaum, and Fredo Durand. Light field networks: Neural scene representations with single-evaluation rendering. In *Proc. NeurIPS*, 2021. 2

[25] Mohammed Suhail, Carlos Esteves, Leonid Sigal, and Ameesh Makadia. Generalizable patch-based neural rendering. In *European Conference on Computer Vision*, pages 156–174. Springer, 2022. 2

[26] Stanislaw Szymanowicz, Chrisitian Rupprecht, and Andrea Vedaldi. Splatter image: Ultra-fast single-view 3d reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10208–10217, 2024. 2

[27] Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A Efros, and Angjoo Kanazawa. Continuous 3d perception model with persistent state. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 10510–10522, 2025. 3

[28] Greg Yang, Dingli Yu, Chen Zhu, and Soufiane Hayou. Tensor programs vi: Feature learning in infinite-depth neural networks. *arXiv preprint arXiv:2310.02244*, 2023. 5

[29] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images.

2021 ieee. In *CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 2, 6

[30] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12104–12113, 2022. 1, 3, 6

[31] 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 *European Conference on Computer Vision*, pages 1–19. Springer, 2024. 1, 2, 4, 6

[32] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 5

[33] 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. 3

[34] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: learning view synthesis using multiplane images. *ACM Transactions on Graphics (TOG)*, 37(4):1–12, 2018. 1, 4, 5, 6

[35] Chen Ziwen, Hao Tan, Kai Zhang, Sai Bi, Fujun Luan, Yicong Hong, Li Fuxin, and Zexiang Xu. Long-lrm: Long-sequence large reconstruction model for wide-coverage gaussian splats. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025. 3

Scaling View Synthesis Transformers

Supplementary Material

9. FLOPs for View Synthesis Transformers

In this section, we explain how we calculated FLOP consumption for all models. Additionally, we show that in our regime, the MLP cost is most significant. In a vision transformer, there are three contributions to the FLOPs:

- Initial patchifying + tokenization layers (negligible)
- Transformer blocks: attention.
- Transformer blocks: MLP and projections.

The first is negligible as it is a single linear layer at the start. Letting n be the number of tokens and d the transformer dimension, for each self-attention transformer block, the following are the FLOPs consumed by the attention, projection, and MLP layers:

$$\chi_{\texttt{MLP}}, \chi_{\texttt{Proj}} \propto nd^2 \tag{7}$$

$$\chi_{\texttt{Attn}} \propto n^2 d \tag{8}$$

Thus, the total FLOPs consumed for a transformer block are of the form

$$\chi = An^2 d + Bnd^2, \tag{9}$$

where A is 4 while B is 16 in the self-attention case, so even at the smallest model size list (Supp 10.3), $\frac{Bnd^2}{An^2 d} \approx 4 \cdot \frac{384}{512} \approx 3$, meaning in most cases our MLP / linear projection FLOPs take up a large majority. For SVSM's cross-attention decoder, the formula becomes a little more complicated as there are separate n_{ctxt} and n_{target}, but these remain on the same order of magnitude as n, so the same is true.

For the decoder-only model, the number of tokens n is given by

$$n = (V_C + 1) \cdot \frac{H}{p} \cdot \frac{W}{p}, \tag{10}$$

where p is the patch size and H and W are the width and height. For the SVSM encoder-decoder, the number of active tokens vary as

$$n_{\text{enc}} = V_C \cdot \frac{H}{p}\frac{W}{p}, n_{\text{dec}} = V_T \cdot \frac{H}{p} \cdot \frac{W}{p}, \tag{11}$$

So, for both models we can write down the MLP and attention complexities as:

$$\chi^{(\text{LVSM})}_{\texttt{MLP}} \propto V_T n \propto V_T(V_C + 1) \tag{12}$$

$$\chi^{(\text{SVSM})}_{\texttt{MLP}} \propto n_{\text{enc}} + n_{\text{dec}} \propto V_C + V_T \tag{13}$$

$$\chi^{(\text{LVSM})}_{\texttt{Attn}} \propto V_T n^2 \propto V_T(V_C + 1)^2 \tag{14}$$

$$\chi^{(\text{SVSM})}_{\texttt{Attn}} \propto n^2_{\text{enc}} + n_{\text{enc}} n_{\text{dec}} \propto V_C^2 + V_C V_T \tag{15}$$

	Encoder			Decoder		
Params	**dim**	**dim_head**	**n_layers**	**dim**	**dim_head**	**n_layers**
15M	384	64	3	384	64	3
27M	384	64	6	384	64	6
35M	384	64	8	384	64	8
62M	512	64	8	384	64	8
79M	512	64	8	512	64	12
145M	640	64	10	640	64	14
226M	768	64	10	768	64	16
316M	768	64	12	768	64	24
420M	768	64	16	768	64	32
740M	1024	64	16	1024	64	32

Table 5. **RealEstate10K V_C=2, SVSM Encoder-Decoder.** SVSM Encoder-Decoder model settings used to sweep scaling laws for the stereo (V_C=2) novel view synthesis setting. **Bolded** is the row for the compute-matched, Pareto-optimal SVSM model compared against the 24-layer LVSM Decoder-only model.

	Encoder			Decoder		
Params	**dim**	**dim_head**	**n_layers**	**dim**	**dim_head**	**n_layers**
8M	-	-	-	384	64	3
13M	-	-	-	384	64	6
22M	-	-	-	512	64	6
28M	-	-	-	512	64	8
53M	-	-	-	640	64	10
90M	-	-	-	768	64	12
118M	-	-	-	768	64	16
175M	-	-	-	768	64	24
275M	-	-	-	896	64	28

Table 6. **RealEstate10K V_C=2, LVSM Decoder-only.** LVSM Decoder-only model settings used to sweep scaling laws for the stereo (V_C=2) novel view synthesis setting.

For fixed V_C, we see that multiplying V_T by an amount k scales compute by exactly $k\times$ in LVSM, while it only scales compute by $\frac{kV_T+V_C}{V_T+V_C}$ in the SVSM case, which is where its advantage lies.

10. Further Experimental Details

10.1. Training Details

All models in this study are multi-view ViTs, following the design of LVSM [10]. The main deviation is no layer index-dependent initialization. All layers are initialized with the same standard deviation. Instead, for stability, we apply a $1/\sqrt{L}$ multiplier on the residuals of all layers where L is the depth of the transformer. Empirically, we find this to maintain stable training on the same learning rate across multiple transformer depths.

All models are trained with the AdamW optimizer, with

Params	Encoder			Decoder		
	dim	dim_head	n_layers	dim	dim_head	n_layers
15M	384	64	3	384	64	3
32M	384	64	6	384	64	8
85M	512	64	10	512	64	12
168M	640	64	12	640	64	16
280M	768	64	12	768	64	20
711M	1024	64	24	1024	64	24
400M	768	64	24	768	64	24

Table 7. **DL3DV V_C=4, SVSM Encoder-Decoder.** SVSM Encoder-Decoder model settings used to sweep scaling laws for the multi-view ($V_C > 2$) novel view synthesis setting. **Bolded** is the row for the compute-matched, Pareto-optimal SVSM model compared against the 24-layer LVSM Decoder-only model.

Params	Encoder			Decoder		
	dim	dim_head	n_layers	dim	dim_head	n_layers
8M	-	-	-	384	64	3
22M	-	-	-	512	64	6
43M	-	-	-	640	64	10
90M	-	-	-	768	64	12
175M	-	-	-	768	64	24
383M	-	-	-	1024	64	30

Table 8. **DL3DV V_C=4, LVSM Decoder-only.** LVSM Decoder-only model settings used to sweep scaling laws for the multi-view ($V_C > 2$) novel view synthesis setting.

a peak learning rate of 4e-4, $\beta_1 = 0.9$, $\beta_2 = 0.95$, and weight decay of 0.05 on all parameters except LayerNorm weights, all following LVSM. We additionally warmup the learning rate for 3000 steps in all models.

All models are trained on 256×256 resolution for both DL3DV and RealEstate10K. All models are trained with the same reconstruction metric used in [10]. In particular, the loss is

$$\mathcal{L} = \text{MSE}(I_T, \tilde{I}_T) + \lambda \cdot \text{Perceptual}(I_T, \tilde{I}_T), \tag{16}$$

where we choose $\lambda = 0.5$ as our perceptual loss weight.

We also have a few experiment specific details. For V_C=2, RealEstate10K, we sample our context and target views from a video index range from 25-192. For V_C>2, DL3DV, we sample our context and target views from a video index range of 16-24 as the baselines between consecutive frames in DL3DV are much wider. For effective batch tests under V_C=4 on DL3DV, we train each model for 100k iterations, while under V_C=2 on RealEstate10K, we train for less iterations (50k) to save compute resources. The trend is still clear even with the limited iterations.

10.2. Rendering Speed Benchmarking

We benchmark all rendering on a single A6000 GPU. We found that due to some hardware configurations of our setup, when benchmarking with batch size 1, all models would cap out at 30 fps, even when the width of each layer was increased or decreased, suggesting that there was some non-FLOP based bottleneck. To circumvent this, all models were tested with batch size 64, which allowed the rendering FPS to properly reflect the forward pass FLOPs for each of the models. We still used $V_T = 1$ for all models, and report

$$\text{FPS} = \frac{B \cdot V_T}{t_{\text{iter}}} = \frac{B}{t_{\text{iter}}}, \tag{17}$$

where t_{iter} is the iteration time for one batch. If higher V_T was used (offline rendering), the rendering speed difference between our model and LVSM would be even larger.

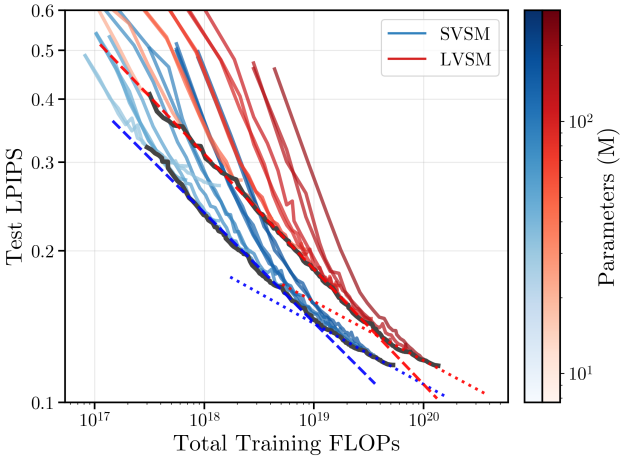

Figure 10. **Linear Power Scaling Laws.** We fit scaling laws onto sections of the Pareto-frontiers of the model families. We see that both models have approximately the same slope in each of their corresponding sections, indicating equal scaling.

10.3. Model Size List for Scaling Sweeps

We trained a wide range of models across both families and both problem settings. Their sizes and hyperparameters are listed in tables 5, 6, 7, and 8. For the decoder-only model there is not much room for flexibility in terms of model hyperparameters – we simply scale dimension up along with the layer count. For SVSM encoder-decoder we can flexibly allocate different amounts of compute to the encoder and the decoder. For low context view cases, we allocate more to the decoder as there are less complex relations amongst the context views and for higher context views we allocate similar amounts to the decoder as to the encoder. Empirically, we roughly found this to have better performance-per-compute, but we did not thoroughly study this split, so we leave this to future work.

11. Linear Fits on the Loss vs. Compute Frontiers

Though the trend is not perfectly linear, we fit lines onto sections of $P(\chi) \propto \chi^c$ in Fig. 10, which show equal scaling between SVSM and LVSM. The actual coefficients are

reported in Tab. 9, which have almost identical power law coefficients across the two model families.

Model	c for $P > 0.14$	c for $P \leq 0.14$
LVSM	-0.23	-0.12
SVSM	-0.22	-0.12

Table 9. **LPIPS vs. compute scaling coefficients.** As regressed from Fig. 10, we find power law coefficients for LPIPS $P(\chi) \propto \chi^c$, and we report c in this table for the first half, which is determined by test LPIPS loss greater than 0.14 and the second half which is test LPIPS loss less than 0.14.

12. PRoPE Ablations

We conduct a series of ablations on PRoPE [14] in the multi-view setting in an attempt to elucidate the mechanism through which it enables scaling. One potential source of success is that the PRoPE SVSM design allows for the decoder to see the clean context poses, while vanilla SVSM does not. To test if this might be the cause for the success, we concatenated context plucker rays to the scene representation from the encoder. However, this has negligible impact (Tab. 10). Additionally, replacing PRoPE with GTA [18] showed negligible difference, indicating epipolar geometry is not crucial for viewcount scalability. Hence, we presume that the relative embeddings are the key, i.e., canonicalizing features to the target frame. Lastly, having PRoPE on just the decoder performs nearly as well as having PRoPE on both, indicating that the cross attention seems to benefit the most from the relative embedding inductive bias.

Model	PSNR (↑)	SSIM (↑)	LPIPS (↓)
Vanilla SVSM	23.50	0.727	0.254
Vanilla w/ concat. pose	23.47	0.726	0.254
PRoPE on Encoder	23.61	0.733	0.249
PRoPE on Decoder	24.31	0.771	0.220
PRoPE on both	24.62	0.782	0.210
GTA on both	24.63	0.782	0.207

Table 10. **PRoPE ablations.** We vary where we apply PRoPE, try a different relative attention method (GTA), and also test pose information flow. GTA varies negligibly from PRoPE, indicating that epipolar geometry is not crucial, and the skip pose connection also has negligible impact, indicating that pose information flow is not responsible.

13. Compiled Scaling Results

We provide a collection of scaling results across RealEstate10K, DL3DV, and Objaverse across 2, 4, and 8 context views respectively in Fig. 11. In all cases, SVSM (in blue) maintains a pareto advantage over LVSM decoder-only.

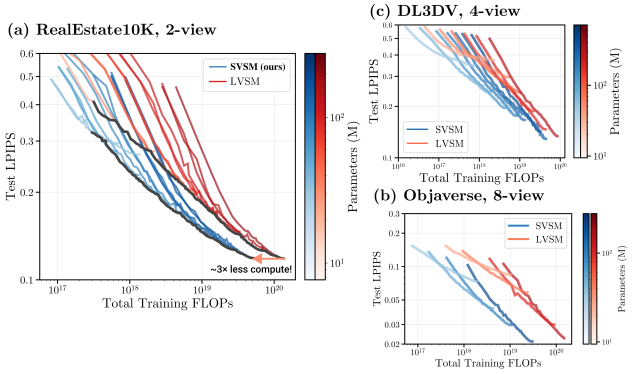

Figure 11. **All Scaling Laws.** We collect scaling laws across the three datasets we tested on, and in all cases, SVSM is compute-optimal.

14. Further Qualitative Results

Lastly, we provide more qualitative results across various training compute budgets of 2 context view evaluations on RealEstate10K in Fig. 12 and 4 context view evaluations on DL3DV in Fig. 13. Multiview consistency of outputs is shown in Fig. 14 on Objaverse.



Figure 12. Qualitative results on RE10K (V_C=2) across scale.

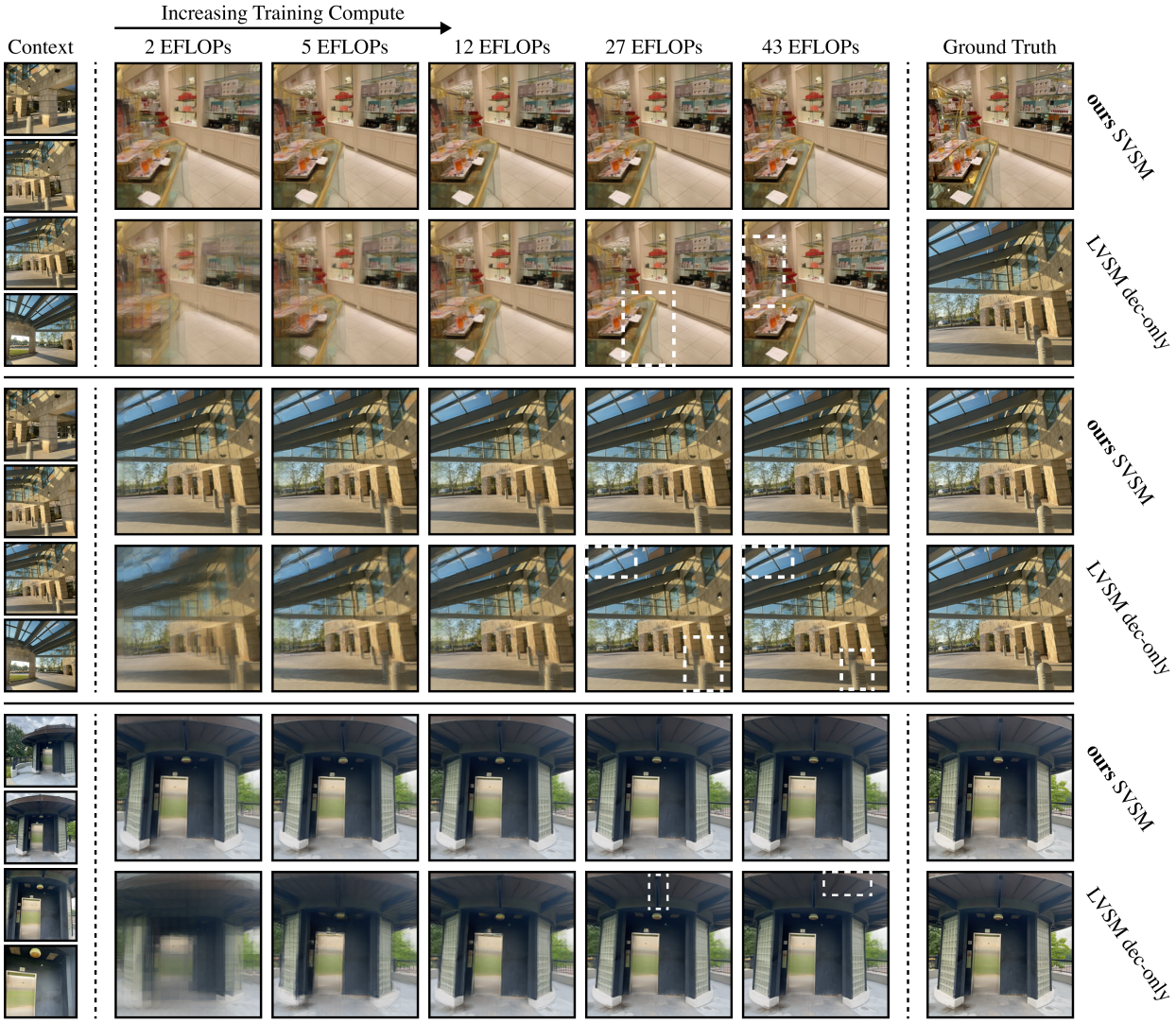

Figure 13. Qualitative results on DL3DV (V_C=4) across scale.



Figure 14. Multiview consistency results on Objaverse (V_C=8).