Scaling View Synthesis Transformers

Supplementary Material

8. 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 9.3), $\frac{Bnd^2}{An^2 d} \approx 4 \cdot \frac{384}{512} \approx 3$, 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.

9. Further Experimental Details

9.1. Training Details

All models in this study are multi-view ViTs, following the design of LVSM [9]. 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, following ideas from depth-μP [1, 27]. Empirically, we find this to maintain stable training on the same learning rate across multiple transformer depths. We also use a patch size of $p = 16$ for all experiments,

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.

except in the case of table 3, where we use $p = 8$ to compare against the reported state-of-the-art numbers from LVSM.

All models are trained with the AdamW optimizer, with 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 [9]. 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.

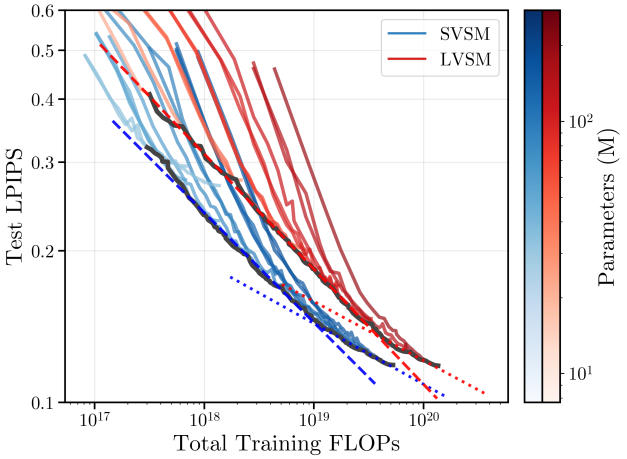

Figure 9. **Linear Power Scaling Laws.** We fit scaling laws onto sections of the Pareto-frontiers of the model families. We see that both models have approximately the same slope in each of their corresponding sections, indicating equal scaling.

9.2. 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.

9.3. 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.

10. 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. 9, 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. 9, 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.

11. PRoPE Ablations

We conduct a series of ablations on PRoPE [13] 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 [17] 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.

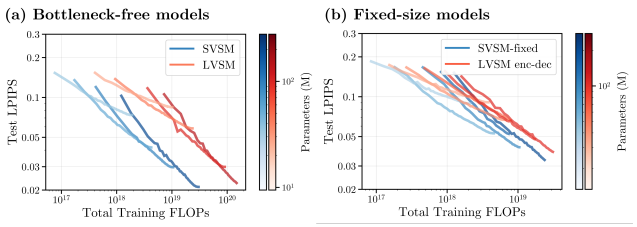

Figure 10. **Fixed-size Latent Scaling Experiments.** Conducted on Objaverse [4]. **(a)** For V_C=8, SVSM and LVSM decoder-only scale equally while SVSM's frontier is shifted by $5\times$ on the compute axis. **(b)** When a fixed latent bottleneck is used, SVSM-fixed and LVSM encoder-decoder scale equally, but significantly worse than the unbottlenecked designs. SVSM again maintains a superior pareto frontier.

12. Scaling Laws for Fixed Latent Design

Lastly, we check both the scaling laws and the design space of view synthesis models with a fixed-size scene representation. A typical example is the LVSM encoder-decoder design. 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.

Analysis Setup. 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 9.

SVSM-Fixed Matches LVSM Scaling, but Both Scale Poorly. As shown in Fig. 10b, 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. 10a makes clear that both fixed-latent designs scale substantially worse than their bottleneck-free counterparts.

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

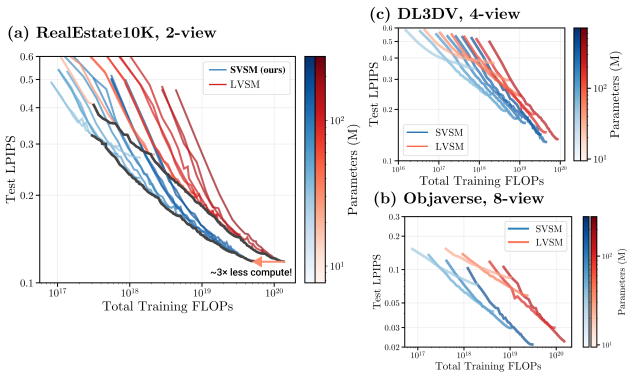

Figure 11. **All Scaling Laws.** We collect scaling laws across the three datasets we tested on, and in all cases, SVSM is compute-optimal.

(in blue) maintains a pareto advantage over LVSM decoder-only.

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.

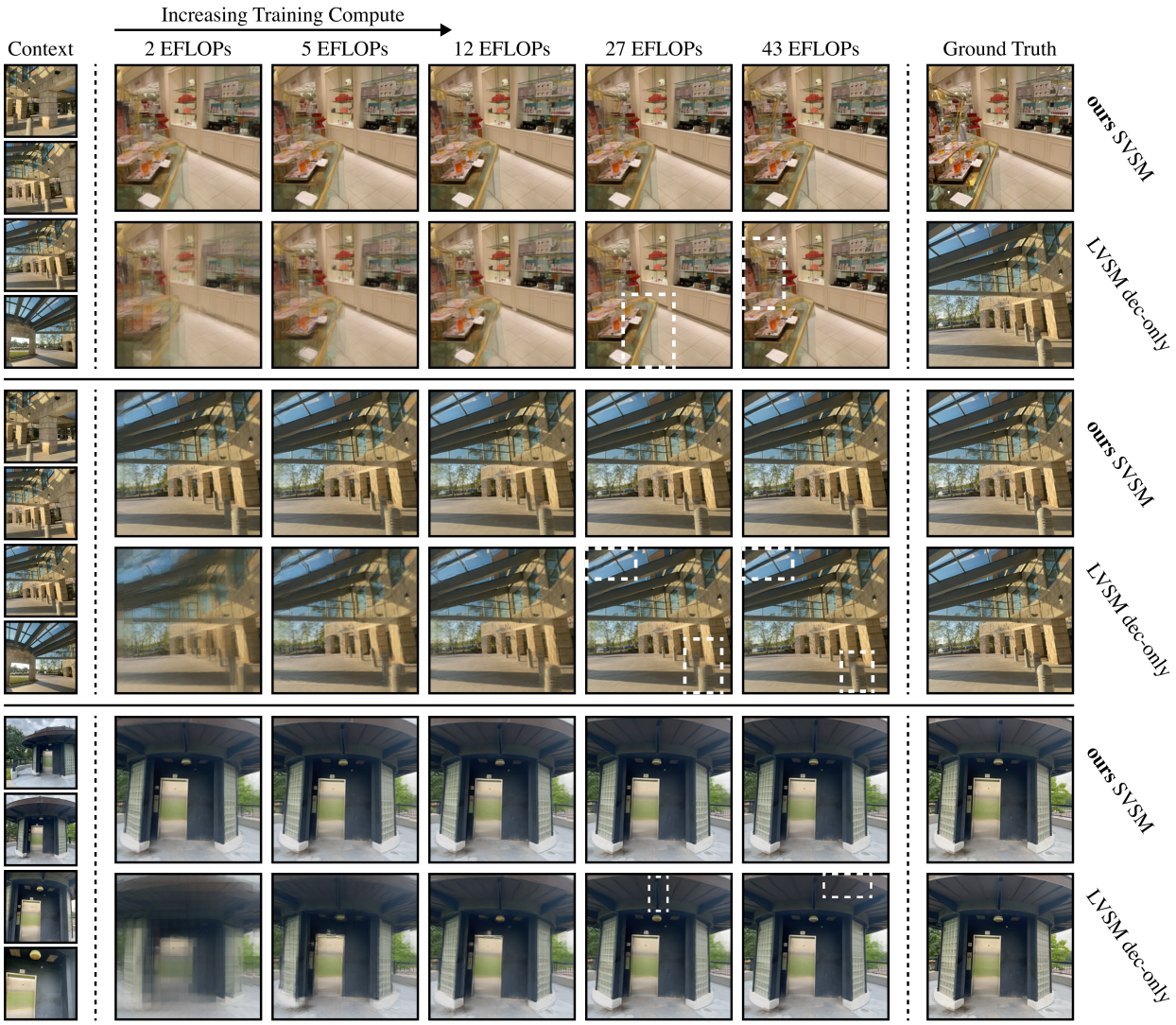

Figure 13. Qualitative results on DL3DV (V_C=4) across scale.

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