A Lesson in Splats: Teacher-Guided Diffusion for 3D Gaussian Splats Generation with 2D Supervision

Supplementary Material

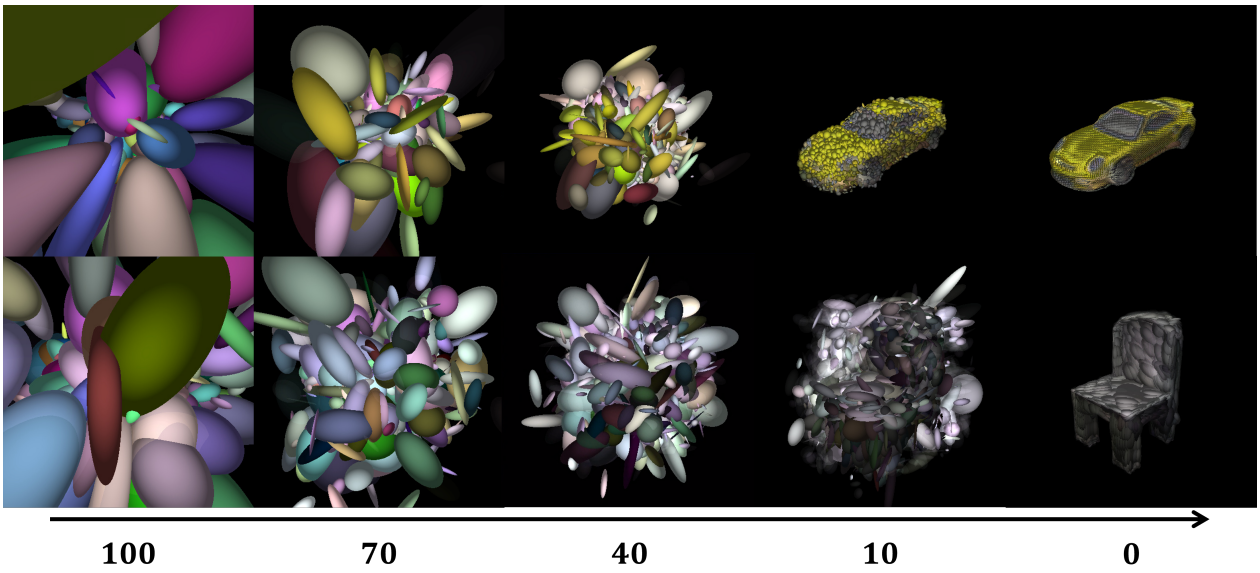

Figure 4. Visualization of the denoising process of our diffusion models, trained on the Car and Chair categories of ShapeNet-SRN dataset.

6. Additional results

6.1. Quantitative results

Co3D is an object-level dataset captured in the real world. We train our model on the Co3D hydrant class (with Splatter Image [55] as the teacher model) and compared it against ViewSet Diffusion [53] and Splatter Image [55] in Table 6. For both ViewSet and our model, we report average scores over seeds and across the test data.

Method	PSNR ↑	SSIM ↑	LPIPS ↓
ViewSet Diffusion [53]	21.24	0.79	0.201
Splatter Image [55]	21.77	0.78	0.154
Ours	**22.34**	**0.82**	**0.149**

Table 6. Comparison on Co3D hydrant dataset.

6.2. Qualitative results

We present visual comparisons of our method to PixelNeRF [68] and VisionNeRF [28] on ShapeNet-SRN Cars and Chairs in Fig. 5. Although our diffusion model is of smaller size (Medium) than the original Splatter Image (Large), we are still able to outperform it. More qualitative results from RealEstate10K dataset are in Fig. 6.

6.3. Ablations

Feedforward vs Diffusion Model. To evaluate whether the observed improvements originated from the diffusion framework or architectural modifications, we trained a feedforward model by removing the time-conditioning layers from the U-Net architecture while preserving its overall structure. For comparison, we predicted Gaussian parameters from a single input image following the splatter image. The feedforward model exhibited significantly worse performance, which we attribute to its reduced size, resulting in limited representational capacity. From Tab. 7, we conclude that the diffusion framework is more suitable for such generation tasks compared to deterministic models, producing better results even with a smaller model size.

Setting	PSNR ↑	SSIM ↑	LPIPS ↓
Splatter Image	24.1992	0.9213	0.0843
Feedforwad	19.9947	0.8613	0.1588

Table 7. Feedforward model vs Splatter Image.

Choices of losses. Through experiments (Tab. 8), we found that it produces terrible results to directly train a diffusion model using rendering loss (both in stage 1 and stage 2), because the supervision indirectly comes from the rendered

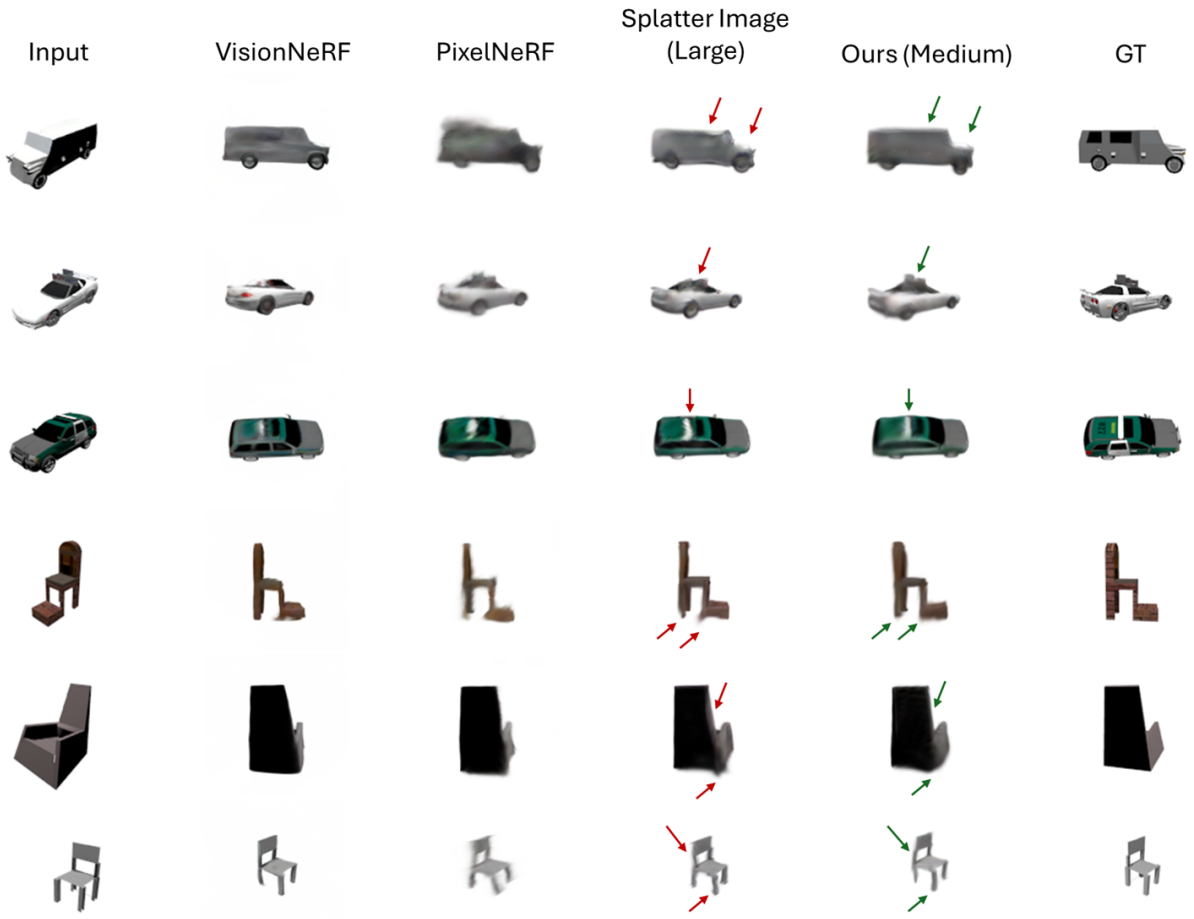

Figure 5. **Additional qualitative results.** Qualitative comparisons on the ShapeNet-SRN dataset for additional viewpoints and objects from the Car and Chair categories. Our model produces views that are more faithful to the source image and better maintain plausibility, while maintaining the fast rendering of Splatter Image. Note that while our diffusion model is of smaller size (Medium) than the original Splatter Image (Large), we are still able to outperform it.

image instead of the denoised spaltters, which makes it hard for the diffusion model to learn the accurate distribution.

Stage 1	Stage 2	PSNR ↑	SSIM ↑	LPIPS ↓
✗	R	16.7284	0.7836	0.3733
R	✗	18.8201	0.8415	0.1862
D	✗	21.3050	0.8965	0.1182
R + D	✗	22.6078	0.9046	0.1083
R + D	R + D	23.1323	0.9116	0.1061
R + D	R	24.4936	0.9264	0.0945

Table 8. Ablation of losses at two stages. 'R' and 'D' represent rendering loss and diffusion loss, respectively.

For stage 1 training, the performance improves using teacher model as guidance and it reports the best results using both rendering loss and diffusion loss.

For stage 2 training, if we continue to use the diffusion loss, the teacher model will limit the performance of our diffusion model. Therefore, we only use rendering loss at stage 2, allowing the model to explore how to minimize the rendering loss and improve the rendering performance.

Weighted loss at different timesteps. The difficulty of prediction at different timesteps varies. Therefore, during the stage 2 training, we assign different weights to the rendering loss obtained at different timesteps and accumulate them for back-propagation throughout the denoising steps. The ablation results are in Tab. 9.

7. Data details

7.1. ShapeNet-SRN Cars and Chairs

We adhere to the standard protocol for the ShapeNet-SRN dataset. Specifically, we use the provided images, camera

Figure 6. **Additional qualitative results.** Qualitative comparisons on RealEstate10K dataset.

Setting	PSNR ↑	SSIM ↑	LPIPS ↓
w/o weighted loss	22.8848	0.9116	0.1044
w weighted loss	24.4936	0.9264	0.0945

Table 9. Ablations of weighted loss at different timesteps

intrinsics, camera poses, and data splits provided by [51] with a resolution of 128×128. Our method is trained using relative camera poses. For single-view reconstruction, view 64 serves as the conditioning view, while for additional-view guidance, views 88 is used as guidance view. All remaining available views are treated as target views, where we compute novel view synthesis metrics.

7.2. RealEstate10K

We obtain 65,384 videos and their corresponding camera pose trajectories from the provided youtube links. Using these camera poses, we perform sparse point cloud reconstruction with COLMAP [48]. For evaluation, we adopt the test split provided by MINE [27] and follow prior work by assessing PSNR on novel frames that are 5 and 10 frames ahead of the source frame. Additionally, we evaluate on a randomly sampled frame within an interval of ± 30 frames, using the same frames employed in MINE's evaluation. For evaluation, we use a total of 3,205 frames. The results presented in Tab. 2 are sourced from Flash3D [54]. Our model is trained and tested at a resolution of 256×384.

8. Implementation details

Multi-step Denoising. We train the model on 4 NVIDIA A6000 GPUs. Our diffusion model is quite efficient . For bootstrapping at stage 1, we use a batch size of 100 on each GPU. After obtaining the diffusion model from the teacher model, we perform multi-step denoising with a DDIM sampler of 10 inference steps. The batch size for stage 2 reduces to 10. We assign different weights to the rendering loss obtained at different timesteps and accumulate them for back-propagation throughout the denoising steps.

Misc. We use Adam [24] as our optimizer with $\beta_1 = 0.9, \beta_2 = 0.999$. We use a total noising steps of 100, with a linear scheduling, starting from 0.0001 to 0.2. For the bootstrapping stage, we use the teacher model to provide both supervision and noised samples. Instead of predicting the noises added to the splatters, our diffusion model denoises the noised inputs to clean samples directly. We trained for 5000 epochs for the bootstrapping stage and then finetune for 1000 epochs. For a fair comparison, we further finetuned the Splatter Image model for 1000 epochs and found negligible improvement because the model has converged. We set t^* to be 20. For the architecture of diffusion model, we use the U-Net implementation from diffusers [1]. For the consistency branch, we use a denoising step of 10.

Method	GS optim	Guidance	PSNR	SSIM	LPIPS
Splatter Image	✗	✗	24.75	0.93	0.06
	✓	✗	25.24	0.94	0.06
Ours	✗	✗	25.18	0.93	0.06
	✓	✗	25.26	0.94	0.06
	✗	✓	25.36	0.94	0.06
	✓	✓	25.55	0.95	0.05

Table 10. **Additional-view guidance.** Evaluated on a subset of the car split, because per-sample GS optimization takes time.

Additional-view guidance Different from deterministic feedforward models, one significant advantage we gain from diffusion models is the ability of using guidance. We use one

[1] https://huggingface.co/docs/diffusers

input view as the condition to predict the Gaussian Splats parameters and then use a second view as guidance during the denoising process using the forward guidance from Universal Guidance [3].

Since we predict $\hat{s}_0$ directly, the noise can be calculated as follows:

$$\epsilon_t = \frac{s_t - \sqrt{\alpha_t}\hat{s}_0}{\sqrt{1-\alpha_t}} \tag{9}$$

Then we calculate the gradient using the guidance image x_{gd} and the corresponding view direction v:

$$\text{grad} \leftarrow \nabla_{s_t} \ell\big[x_{gd}, \mathcal{R}(\hat{s}_0, v)\big]. \tag{10}$$

With the guidance strength factor $s(t)$, we can obtain $\hat{\epsilon}_t$

$$\hat{\epsilon}_t = \epsilon_t + s(t) \cdot \text{grad} \tag{11}$$

At last, we can get s_{t-1} following DDIM sampling:

$$s_{t-1} = \sqrt{\alpha_{t-1}}\hat{s}_0 + \sqrt{1-\alpha_{t-1}} \cdot \hat{\epsilon}_t \tag{12}$$