DropoutGS: Dropping Out Gaussians for Better Sparse-view Rendering

Yexing Xu[1*] Longguang Wang[1*] Minglin Chen[1] Sheng Ao[2] Li Li[3] Yulan Guo[1†]
[1]The Shenzhen Campus, Sun Yat-Sen University [2]Xiamen University [3]University of Macau
xuyx55@mail2.sysu.edu.cn, wanglg9@mail.sysu.edu.cn, guoyulan@sysu.edu.cn

Abstract

Although 3D Gaussian Splatting (3DGS) has demonstrated promising results in novel view synthesis, its performance degrades dramatically with sparse inputs and generates undesirable artifacts. As the number of training views decreases, the novel view synthesis task degrades to a highly under-determined problem such that existing methods suffer from the notorious overfitting issue. Interestingly, we observe that models with fewer Gaussian primitives exhibit less overfitting under spare inputs. Inspired by this observation, we propose a Random Dropout Regularization (RDR) to exploit the advantages of low-complexity models to alleviate overfitting. In addition, to remedy the lack of high-frequency details for these models, an Edge-guided Splitting Strategy (ESS) is developed. With these two techniques, our method (termed DropoutGS) provides a simple yet effective plug-in approach to improve the generalization performance of existing 3DGS methods. Extensive experiments show that our DropoutGS produces state-of-the-art performance under sparse views on benchmark datasets including Blender, LLFF, and DTU. The project page is at: https://xuyx55.github.io/DropoutGS/.

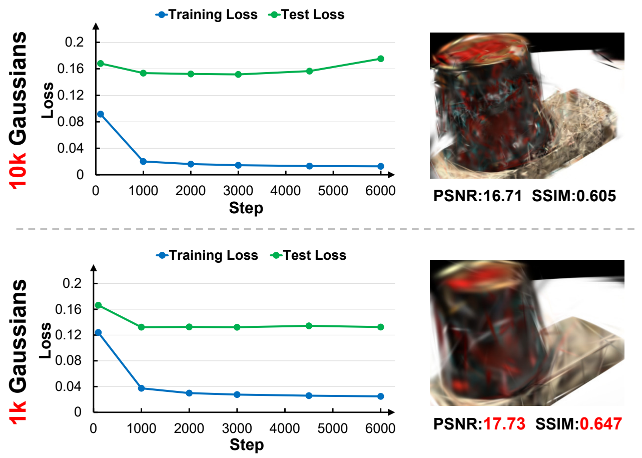


Figure 1. **Results produced by 3DGS with different numbers of Gaussians.** The training loss curves and rendered results are visualized for comparison. Compared with the model with 1k Gaussians, the one with 10k Gaussians suffers overfitting and produces inferior results.

1. Introduction

The computer vision community has witnessed incredible advances of Novel View Synthesis (NVS), which aims at synthesizing novel views of a scene from a set of observed views. Traditional methods employ explicit scene representations, such as point clouds [10], voxels [6], and meshes [38], to represent the 3D scene for novel view synthesis. Recently, learning-based methods have achieved remarkable progress compared to previous ones. Specifically, Neural Radiance Field (NeRF) [27] represents a scene with a neural network, achieving high-quality rendering and lightweight storage. 3D Gaussian Splatting (3DGS) [21] decomposes a scene into a set of discrete Gaussian primitives and utilizes a splat-based rendering technique to generate novel views in real time. Nevertheless, these methods require a large number of training views and suffer from notable performance drops with insufficient views, which limits their applications in real-world scenarios.

To achieve novel view synthesis under sparse views, DRGS [7] encourages Gaussian primitives to align with the object surface by using the depth information obtained from a pre-trained monocular depth estimator. Then, DNGaussian [23] improves the depth regularization by further considering the smoothness of the predicted depth map. In addition, DNGaussian proposes a global-local depth normalization to address the scale inconsistency in monocular depth estimation. However, these methods are sensitive to the accuracy of the depth map and the depth error may be amplified to produce unsatisfactory artifacts.

In this paper, we take a step to investigate the performance degradation of 3DGS under sparse views. Interestingly, we observe a notorious overfitting issue during the training of 3DGS, as illustrated in Fig. 1. While the training

*Equal contribution.
†Corresponding author.

loss of the model with 10k Gaussians continues to decrease, the test loss begins to increase after 3000 iterations. Meanwhile, the rendered image suffers severe hollow artifacts. In contrast, this phenomenon is weakened when the number of Gaussians is reduced to 1k and the rendered images are smoother. Nevertheless, the edge and texture become blurry. In summary, under sparse views, 3DGS is easily over-parameterized and prone to overfitting the training data without capturing the 3D structure.

Inspired by this observation, we propose DropoutGS that combines an effective yet efficient dropout technique with 3DGS to address the overfitting issue under sparse views. Specifically, our DropoutGS randomly dropouts several Gaussians during training to alleviate overfitting. To remedy the detail loss caused by the dropout technique, we further introduce an Edge-guided Splitting Strategy (ESS) to encourage the Gaussians to focus more on edge regions. While previous sparse view synthesis methods commonly rely on exploring additional constraints (e.g., depth maps) to produce satisfactory performance, our DropoutGS attempts to handle this issue from another perspective of overfitting. Extensive experiments show that our DropoutGS produces competitive performance against previous methods on diverse benchmark datasets.

Overall, the main contributions are summarized as follows:

- We study the performance degradation of 3DGS under sparse views and attribute it to the overfitting issue. From this point of view, we propose DropoutGS to leverage the dropout technique to alleviate this problem.
- We propose a Random Dropout Regularization (RDR) to alleviate overfitting and obtain smooth rendering results by randomly dropping out Gaussians during optimization.
- We introduce an Edge-guided Splitting Strategy (ESS) to encourage 3DGS to focus more on edge regions during optimization for finer details.
- Our DropoutGS is compatible with diverse 3DGS-based methods and achieves state-of-the-art performance on multiple benchmark datasets.

2. Related Work

2.1. Novel View Synthesis

Novel View Synthesis (NVS) is a long-standing task aiming at generating novel views of a scene from a set of observed images. In this field, radiance fields are widely used and produce remarkable progress. Neural Radiance Fields (NeRF) [27] encodes scenes into a neural network and achieves high-fidelity view synthesis by volumetric rendering. However, the computationally intensive rendering processes limit its scalability and efficiency. Later, a number of methods are developed to improve the rendering speed [14, 24, 28, 39] and image quality [2, 3, 13] of NeRF. More recently, the unstructured radiance field 3DGS proposed by Bernhard *et al.* has achieved significant breakthroughs in novel view synthesis tasks. 3DGS represents complex scenes using a series of discrete Gaussian basis functions. In addition, efficient and differentiable splatting techniques are adopted for real-time rendering. Inspired by the success of 3DGS, subsequent methods further improve its representational capabilities [16, 18] and splitting strategies [11, 46, 52]. Despite remarkable results, these methods suffer severe performance drops under sparse views.

To achieve novel view synthesis under sparse views, early works have explored various regularizations for NeRF, including depth [8, 31, 37], normal [33], frequency [45] and cross-view consistency [22, 35]. As for recent 3DGS, depth prior [7, 23, 44, 54] has also been widely studied to stabilize the optimization of Gaussians under sparse views. Despite remarkable progress, these methods still suffer two limitations. *First*, these methods rely on an additional monocular depth estimation module to obtain the depth map, which introduces extra computational overhead. *Second*, as 3DGS is sensitive to depth accuracy, depth errors could be accumulated and amplified to produce undesirable artifacts in the rendered images. Different from the aforementioned methods, we propose an alternative to consider the performance degradation as an overfitting problem and introduce a DropoutGS method.

2.2. Overfitting in Deep Learning

Overfitting is a notorious problem in deep learning. It usually arises from limited training data, high model complexity, or insufficient regularization, leading to poor generalization on unseen samples. To address this issue, various strategies have been studied. Specifically, dropout [12, 34, 36, 43], as one of the most effective techniques, improves the robustness of the model by randomly deactivating a subset of neurons during training. Data augmentation strategies [9, 48, 49, 53] have been widely investigated to increase the diversity of training data. Adversarial training [15, 25, 30, 42] aims to introduce multiple perturbations like noise in the training process, encouraging the model to become more resilient and robust to outliers.

In this paper, we observe a similar overfitting issue for 3DGS under sparse training views. Motivated by the great success of dropout to address this issue, we develop DropoutGS by incorporating a Random Dropout Regularization (RDR) to smooth out the overfitting degradation and an Edge-guided Splitting Strategy (ESS) to compensate for the detail loss.

3. Method

Our DroupGS consists of a Random Dropout Regularization (RDR) and Edge-guided Splitting Strategy (ESS), as illustrated in Fig. 2. In this section, we first present the preliminaries of 3DGS in Sec. 3.1. Then, we conduct pilot experiments to illustrate our motivation in Sec. 3.2. Next, we detail our method designs separately. Finally, we present the loss function of our method in Sec. 3.5.

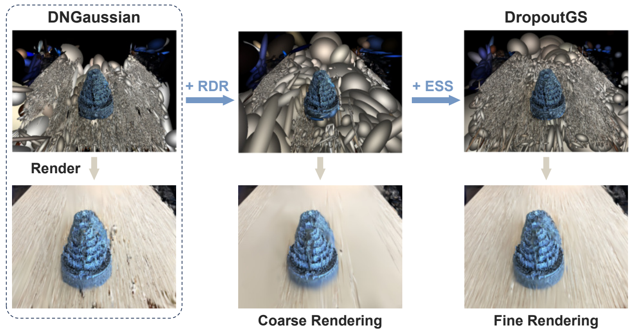


Figure 2. **An overview of our framework.** RDR is first employed to alleviate the overfitting issue. Then, ESS is adopted to split the large Gaussians to better capture high-frequency details.

3.1. Preliminaries

3DGS [21] represents a 3D scene with a set of Gaussian primitives, each with trainable parameters including mean position μ, 3D covariance Σ, scaling factor s, opacity o, and spherical harmonic features f. 3DGS employs the Gaussian basis functions to capture the spatial distribution of scene features. The rendering process blenders these basis functions to generate views, calculating the color as a weighted summation of Gaussian primitives:

$$C(x) = \sum_{i \in \mathcal{N}} \alpha_i \prod_{j=1}^{i-1} (1-\alpha_j) c_i = \sum_{i \in \mathcal{N}} w_i c_i, \tag{1}$$

where c_i is the decoded spherical harmonic features, and w_i measures the contribution of Gaussians to the rendering results.

3.2. Pilot Study

It is widely known that the imbalance between model complexity and training data amount is a primary reason for overfitting. For 3DGS, the number of Gaussian primitives determines the complexity of the model while the number of training views refers to the training data amount. From this point of view, we attribute the performance degradation of 3DGS under sparse views to the overfitting caused by an excessive number of Gaussian primitives.

To demonstrate this, we first investigate the performance of 3DGS with different numbers of Gaussian primitives under sparse training views. Specifically, we initialize the 3DGS using point clouds with different numbers of points and train these models on the DTU dataset. Note that, densification and pruning are not performed to maintain consistent model complexity throughout the training phase. As

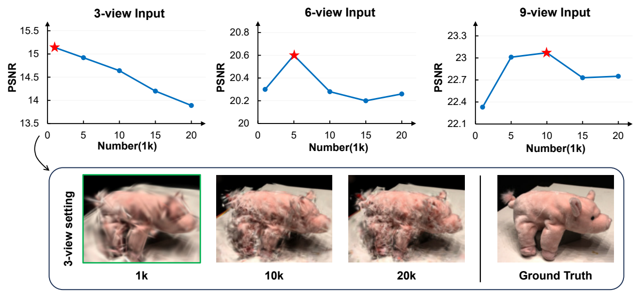


Figure 3. **The relationship between the amount of training data and model complexity.** We investigate the performance of 3DGS with different primitive settings under varying numbers of sparse views. The results show that models achieve optimal performance when their complexity matches the training data.

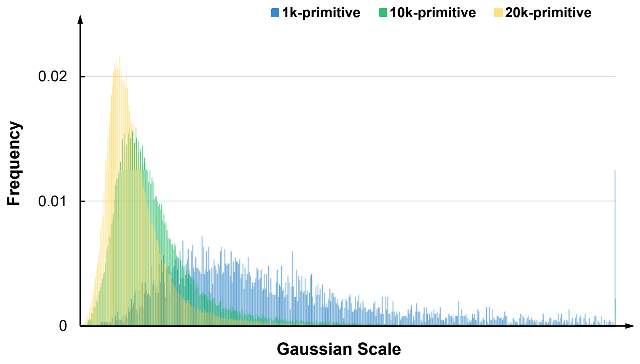


Figure 4. **The scale distribution of the Gaussians learned by models with different complexities.** The models with 10k and 20k primitives have a large portion of small-scale Gaussians. In contrast, the model with 1k primitives obtains more Gaussians with larger scales.

illustrated in Fig. 3, the model with 1k primitives achieves the best performance under 3-view input while models with higher complexity suffer from performance degradation. Meanwhile, as the number of input views increases, the optimal model size grows from 1k to 10k. These observations demonstrate that an excessive number of Gaussian primitives leads to the overfitting issue and inferior performance.

On top of the above experiments, we further conduct experiments to study the Gaussian primitives in the trained models. Specifically, we visualize the scales of the Gaussians across different models in Fig. 4. As we can see, the higher the model complexity is, the smaller the learned Gaussians are. Under this condition, the model is prone to overfitting the details in the training views without understanding the 3D structure. In contrast, the model with fewer Gaussian is encouraged to enlarge the Gaussian primitives to cover the contents in the training views. As a result, the 3D structure can be better modeled but high-frequency details cannot be well captured.

From the observations above, we can draw two challenges that limit the performance of 3DGS under sparse training views. *First*, over-parameterized models suffer from the overfitting issue and produce degraded performance. *Second*, decreasing the number of Gaussians alleviates the overfitting problem at the cost of the inferior capability to capture high-frequency details. To address these two challenges, we propose a coarse-to-fine framework with a random dropout regularization (RDR) to alleviate overfitting and an edge-guided splitting strategy (ESS) to remedy the missing details.

3.3. Random Dropout Regularization (RDR)

During the training phase, the Gaussian primitives are randomly deactivated with a probability p and the remaining Gaussian primitives are optimized to fit the observed views. Mathematically, the rendered color $\hat{C}$ for a pixel can be obtained as:

$$\begin{cases} z \sim \text{Uniform}\,(0,1) \\ r = [z > p] \\ \hat{C} = \sum_{i \in \mathcal{N}} r_i \cdot \alpha_i \prod_{j=1}^{i-1} (1 - r_j \cdot \alpha_j)\, c_i \end{cases}, \tag{2}$$

where Uniform $(0,1)$ denotes a uniform distribution, and $[\cdot]$ refers to Iverson bracket. During inference, all Gaussian primitives are activated for novel view synthesis. Particularly, we minimize the differences between the rendered results of the full model and the sub-model after dropout in each training iteration for optimization:

$$\mathcal{L}_{RDR} = \left\| C - \hat{C} \right\|_1 + \text{SSIM}(C, \hat{C}). \tag{3}$$

Note that, instead of using the groundtruth color in the observed image for supervision, the rendered color C produced by the full model is employed. To demonstrate the effectiveness of the loss function above, we splat the gradients of Gaussians to obtain gradient maps for comparison in Fig. 5a and 5b.

If the rendered image of the full model is adopted as supervision, only neighboring Gaussian primitives centered at the dropouted one are optimized. As a result, the gradients only exist locally (Fig. 5a) and encourage the model to focus on specific regions. In contrast, when the groundtruth color is employed for supervision, all Gaussian primitives are optimized and the gradients can be observed in all regions (Fig. 5b). As a result, the gradients in different parts of a 3DGS model may counteract the effects of each other [46], producing inferior results.

Although inspired by dropout, our method excludes its compensation strategy. Experiments show that the compensation strategy does not significantly enhance model performance and may alter the blending colors of the pixels unaffected by dropout, introducing potential adverse effects.

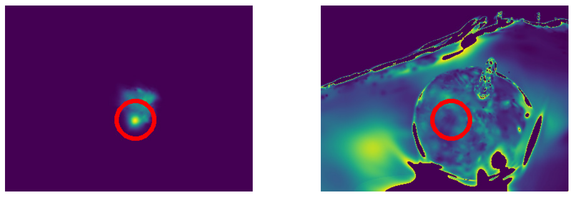

(a) gradients from $\mathcal{L}_{RDR}$ (b) gradients from $\mathcal{L}_{gt}$

Figure 5. **Visualization of the gradient maps.** The dropouted Gaussian is annotated with a red circle. Thus brighter regions correspond to higher gradients.

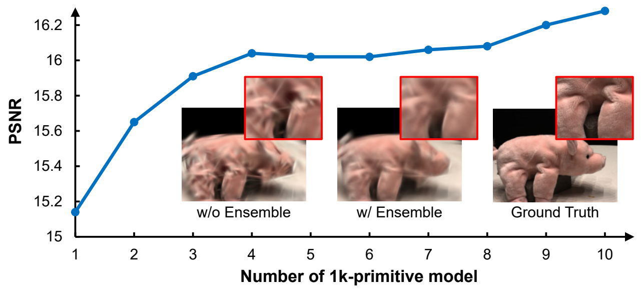


Figure 6. **Explaining the effectiveness of dropout from an ensemble learning perspective.** The quality of the rendered image significantly improves after integration, as evidenced by smoother appearances and fewer artifacts. This effect becomes more pronounced as the number of integrated sub-models increases.

Discussion. Previous studies [1, 17, 41] have revealed that dropout mitigates overfitting by approximating the geometric mean over an ensemble of potential sub-networks. From this point of view, the optimization process of our RDR can be considered as training of several low-complexity sub-models. Then, the inference process can be viewed as an ensemble of the trained sub-models such that superior performance can be achieved. As shown in Fig. 3, by integrating more low-complexity models, the ensemble model (*i.e.*, the full model) produces consistent accuracy gains.

3.4. Edge-guided Splitting Strategy (ESS)

Although RDR facilitates 3DGS to better capture the 3D structure of the scene, the optimized Gaussian primitives are prone to being enlarged with a weakened capability to model high-frequency details. To remedy this, we propose an edge-guided splitting strategy (ESS) to further refine the Gaussian primitives by splitting the large ones into smaller ones to better fit the high-frequency details.

Edge Score. We first define an edge score to identify the edge regions. For an input view I, an edge detection method is employed to obtain pixel-wise probability $E(I)$, ranging from 0 to 1. Then, we project Gaussians onto the edge map and compute the single-view edge score $\mathcal{E}'$ by aggregating the probabilities of each primitive according to its contribu-

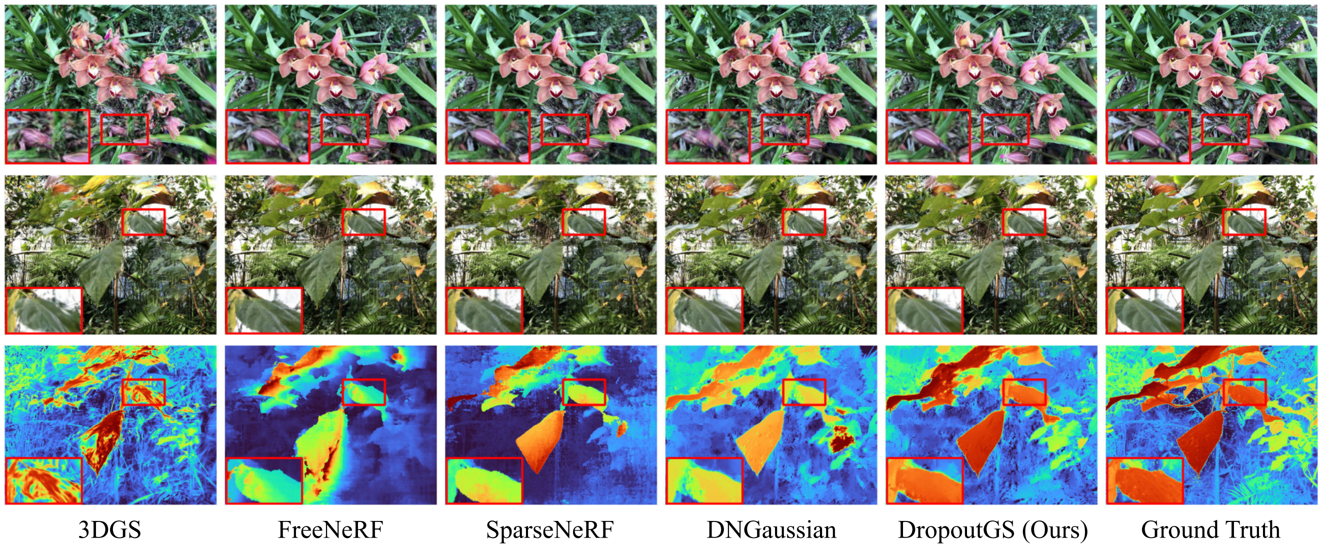


Figure 7. **Visual results on the LLFF dataset.** The ground truth of depth is obtained from the 3DGS trained with dense views. 3DGS fails to render novel views with correct geometry. DNGaussian can better represent the scene structure in comparison, but it cannot get smooth surfaces. The rendering results of FreeNeRF and SparseNeRF have a loss of details. Our model can learn complex details while obtaining smooth surfaces.

tion to pixels:

$$\mathcal{E}_i' = \alpha_i \prod_{j}^{i-1}(1-\alpha_j) \cdot \sum_p E(I)\mathcal{M}^i(p) \tag{4}$$

where $\mathcal{M}^i(p)$ is a binary mask indicating whether the p-th pixel is covered by the i-th Gaussian. Considering Gaussians cover different numbers of pixels in multiple viewpoints, we adopt a viewpoint accumulation approach to obtain the final edge score $\mathcal{E}_i$:

$$\mathcal{E}_i = \sum_{k=1}^{M} \frac{\mathcal{E}'_{i,k}}{\sum_p \mathcal{M}_k^i} \tag{5}$$

where $\mathcal{E}'_{i,k}$ indicates the edge score at the k-th viewpoint out of a total of M.

Splitting Strategy. After obtaining the edge scores, Gaussian primitives with large scales and high edge scores are split into smaller ones to better fit the edge details. Specifically, the mask for the target primitives is calculated as below:

$$\mathcal{M}_{edge} = \{S_i \geq \mathcal{S}_{thr}\} \cap \{\mathcal{E}_i \geq \mathcal{E}_{thr}\}, \tag{6}$$

where S_i denotes the size of i-th primitives while $\mathcal{S}_{thr}$ and $\mathcal{E}_{thr}$ respectively denote the size threshold and edge threshold. By splitting large Gaussian primitives into smaller ones, finer high-frequency details can be modeled, as illustrated in Fig. 2.

3.5. Loss Function

Since DropoutGS only makes slight modifications to the rendering process and does not rely on additional modules, it can be seamlessly integrated with existing 3DGS techniques. In particular, our approach remains effective and provides further improvements when applied to methods with depth regularization. In this case, the overall loss function of DropoutGS is expressed as:

$$\mathcal{L} = \mathcal{L}_{gs} + \lambda_{depth}\mathcal{L}_{depth} + \lambda_{RDR}\mathcal{L}_{RDR}, \tag{7}$$

where $\mathcal{L}_{gs}$ denotes the loss function in original 3DGS [21]. $\mathcal{L}_{depth}$ is the depth regularization used in DNGaussian and $\mathcal{L}_{RDR}$ denotes the proposed RDR loss. λ_{depth} and λ_{RDR} are the coefficients used to balance different constraints.

4. Experiments

4.1. Setups

Datasets. We evaluate our method on three benchmark datasets: the LLFF [26] dataset, the DTU [20] dataset and the Blender [27] dataset. We follow the 3-view setting used in previous works [23, 37, 45] with the same split of LLFF and DTU. Additionally, we adopt the same object masks as [45] for fair comparison on the rendering quality of the target objects rather than the background in DTU. For Blender, we train our model with 8 views and evaluate it on the other 25 views following the setting used in [19, 45].

Evaluation Metrics. We report PSNR, SSIM [40] and LPIPS [51] metrics to evaluate the rendering quality. Besides, we also adopt the Average Error (AVGE) [2] as a

	Setting	LLFF				DTU			
		PSNR ↑	LPIPS ↓	SSIM ↑	AVGE ↓	PSNR ↑	LPIPS ↓	SSIM ↑	AVGE ↓
SRF [5]	Trained on DTU	12.34	0.591	0.250	0.313	15.32	0.304	0.671	0.171
PixelNeRF [47]	Trained on DTU	7.93	0.682	0.272	0.461	16.82	0.270	0.695	0.147
MVSNeRF [4]	Trained on DTU	17.25	0.356	0.557	0.171	18.63	0.197	0.769	0.113
SRF ft [5]	Trained on DTU Fine-tuned per Scene	17.07	0.529	0.436	0.203	15.68	0.281	0.698	0.162
PixelNeRF ft [47]	Trained on DTU Fine-tuned per Scene	16.17	0.512	0.438	0.217	18.95	0.269	0.710	0.125
MVSNeRF ft [4]	Trained on DTU Fine-tuned per Scene	17.88	0.327	0.584	0.157	18.54	0.197	0.769	0.113
Mip-NeRF [2]	Optimized per Scene	14.62	0.495	0.351	0.246	8.68	0.353	0.571	0.323
DietNeRF [19]	Optimized per Scene	14.94	0.496	0.370	0.240	11.85	0.314	0.633	0.243
RegNeRF [29]	Optimized per Scene	19.08	0.336	0.587	0.149	18.89	0.190	0.745	0.112
FreeNeRF [45]	Optimized per Scene	19.63	0.308	0.612	0.134	19.92	0.182	0.787	0.098
SparseNeRF [37]	Optimized per Scene	**19.86**	0.328	**0.624**	**0.127**	19.55	0.201	0.769	0.102
3DGS [21]	Optimized per Scene	15.52	0.405	0.408	0.209	10.99	0.313	0.585	0.252
3DGS†	Optimized per Scene	16.46	0.401	0.440	0.192	14.74	0.249	0.672	0.169
DNGaussian [23]	Optimized per Scene	19.12	0.294	0.591	0.132	18.91	0.176	0.790	0.102
DropoutGS (Ours)	Optimized per Scene	19.35	**0.282**	0.622	0.128	**20.22**	**0.150**	**0.830**	**0.086**

† with the same hyperparameters and the neural color renderer as DNGaussian.

Table 1. **Quantitative comparison on the LLFF and DTU datasets.** The best, second-best, and third-best entries are marked in **red**, orange, and yellow, respectively.

Method	PSNR ↑	SSIM ↑	LPIPS ↓
NeRF	14.934	0.687	0.318
NeRF (Simplified)	20.092	0.822	0.179
DietNeRF	23.147	0.866	0.109
DietNeRF + ft	23.591	0.874	0.097
FreeNeRF	24.259	0.883	0.098
SparseNeRF	22.410	0.861	0.119
3DGS	22.226	0.858	0.114
DNGaussian	24.305	0.886	0.088
DropoutGS (Ours)	**24.476**	**0.889**	**0.085**

Table 2. **Quantitative Comparison on Blender for 8 input views.** The best, second-best, and third-best entries are marked in **red**, orange, and yellow, respectively.

metric, which is the geometric mean of MSE, $\sqrt{1-\text{SSIM}}$ and LPIPS. All results are averaged over three repeated experiments.

Baselines. We compare our DropoutGS with 9 state-of-the-art methods, including one 3DGS-based method (DNGaussian [23]), 5 NeRF-based methods (FreeNeRF [45], SparseNeRF [37], RegNeRF [29], DietNeRF [19], and Mip-NeRF [2]) and 3 generative methods (MVSNeRF [4], PixelNeRF [47], SRF [5]).

Implementations. Our approach is built on the PyTorch framework. We train our model for 6k iterations on the LLFF, DTU, and Blender datasets. Following DNGaussian [23], we initialize 3DGS and DropoutGS with a randomly generated point cloud.

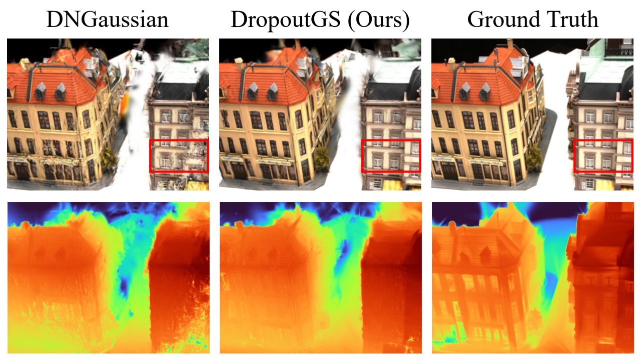
4.2. Performance Evaluation

The LLFF Dataset. We evaluate our method on eight complex scenes of the LLFF dataset with 3 views as input. Quantitative results are presented in Table 1 while visual results are shown in Fig. 7.

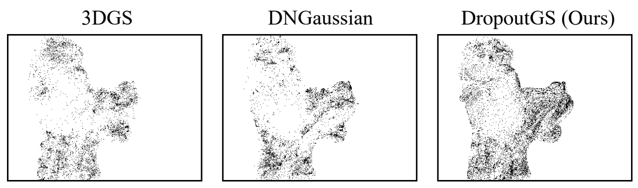
It can be observed from Table 1 that our method outperforms other methods in terms of LPIPS and achieves the second-best performance in SSIM and AVGE. This demonstrates the superior quantitative performance of our method. Moreover, as shown in Fig. 7, our DropoutGS produces results with finer details and fewer artifacts, while FreeNeRF [45] and SparseNeRF [37] generate blurry results with inferior perceptual quality. Although DNGaussian also captures fine details, it suffers from severe hollow artifacts. Benefiting from our RDR and ESS strategies, our DropoutGS fills up the hollows caused by overfitting without sacrificing details. Additionally, we further visualize the depth map produced by different methods. As compared to DNGaussian, our DropouGS generates more accurate depth maps with smooth surfaces, which means the 3D structure is better modeled.

The DTU Dataset. To evaluate the rendering ability of our method on the object-centered scenes, we conduct our experiments on the DTU dataset. As shown in Table 1, our proposed method significantly outperforms all the baselines across multiple metrics in the 3-view setting, achieving state-of-the-art performance. Figure 8 illustrates that DropoutGS successfully renders novel views with accurate geometric structure and reduced overfitting artifacts such as hollows and blurring, while DNGaussian suffers from severe degradation in the foreground. Additionally, our DropoutGS produces denser point clouds that are better aligned with the surfaces of objects. This further demonstrates the effectiveness of our DropoutGS.

The Blender Dataset. We also evaluate our model on the Blender dataset with 8-view input, which contains eight synthetic scenes each with widely varying training and testing viewpoints. The quantitative results in Table 2 show that the proposed method achieves the best performance in terms of PSNR, SSIM, and LPIPS. From the qualitative results in



(a) Rendered images visualization



(b) Gaussian point cloud visualization

Figure 8. **Qualitative results on the DTU dataset.** The ground truth of depth is obtained from the 3DGS trained with dense views. Our method learns a denser primitive distribution improving the geometry accuracy.

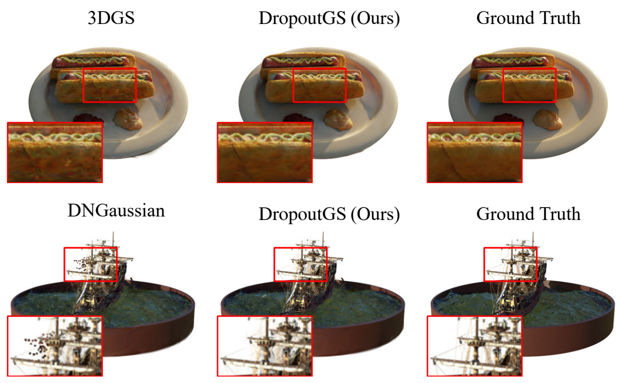


Figure 9. **Qualitative results on the Blender dataset.** Our DropoutGS can smooth out the artifacts that appear in 3DGS and DNGaussian, obtaining cleaner and smoother novel views.

Fig. 9, we can see that DropoutGS is able to generate views with fewer floaters compared to the vanilla 3DGS and the original DNGaussian. The *hotdog* scene also demonstrates the ability of our method to smooth out inadequately trained Gaussian primitives, resulting in a flatter surface. These experiments show that DropoutGS achieves competitive performance not only on the forward-facing dataset such as LLFF and DTU but also on the wide-baseline dataset with challenging reflective materials.

Compatibility. Our DropoutGS is compatible with different 3DGS methods and can be flexibly integrated with them to improve their performance under sparse views. To demonstrate this, we combine our DropoutGS with the original 3DGS, FSGS [54], and CoR-GS [50] for evaluation. As listed in Table 3, our approach facilitates these methods to produce notable gains in terms of all metrics. Unlike FSGS and CoR-GS which employ the point cloud obtained from [32] for initialization, Gaussians in 3DGS are randomly initialized so that they are more susceptible to the overfitting issue. As a result, our method introduces more significant improvements in 3DGS. Remarkably, our method is competitive with different initialization methods and produces consistent performance gains.

Method	w/ Ours	Setting	PSNR ↑	LPIPS ↓	SSIM ↑	AVGE ↓
3DGS†	✗	Random init.	16.46	0.401	0.440	0.192
	✓		**18.05**	**0.326**	**0.545**	**0.155**
3DGS†*	✗	MVS init.	19.86	0.222	0.670	0.112
	✓		**20.53**	**0.205**	**0.706**	**0.102**
CoR-GS* [50]	✗	Co-constraint	20.45	0.196	0.712	0.101
	✓		**20.59**	**0.195**	**0.716**	**0.097**
FSGS* [54]	✗	Depth constraint	20.43	0.248	0.682	0.108
	✓		**20.70**	**0.200**	**0.713**	**0.099**
DNGaussian*	✗	Depth constraint	19.94	0.228	0.682	0.116
	✓		**20.64**	**0.210**	**0.717**	**0.101**

Table 3. **Compatibility to different 3DGS methods with 3 input views on the LLFF dataset.** *init.* denotes initialization. * with the MVS point cloud as initialization.

4.3. Ablation Study

In this subsection, we conduct ablation experiments to study the effects of our DropoutGS. All experiments are conducted on the LLFF dataset with 3 views input. Quantitative and qualitative results are reported in Table 4 and Fig. 10.

RDR and ESS Strategies. We first conduct experiments to investigate the effectiveness of RDR and ESS strategies. Specifically, we introduce these two strategies to the baseline method separately and compare their performance with our DropoutGS. It can be observed from Table 4 that both RDR and ESS introduce notable performance improvement to the baseline. It is worth noticing that RDR produces a significant gain on PSNR and SSIM, but introduces a slight drop in LPIPS. With only RDR, Gaussian primitives are prone to be large such that high-frequency details cannot be well reconstructed. With both additional ESS, the large Gaussian primitives are split into small ones to better capture details in edge regions. As a result, the LPIPS score is improved together with PSNR and SSIM. Figure 10 further compares the visual results produced by methods with different settings. It can be seen that RDR helps to fill in the hollows caused by overfitting and ESS contributes to recovering the high-frequency details in the rendered views.

More View Settings. In addition to the 3-view input, we further evaluate the performance of DropoutGS in a more view setting on the LLFF dataset. We respectively use 6 views and 9 views as input following the default settings in DNGaussian and report the results in Table 5. Experiments show that the DropoutGS achieves better performance as the training view increases and outperforms the

	RDR	ESS	PSNR↑	SSIM↑	LPIPS↓	AVGE↓
Baseline			18.76	0.582	0.300	0.139
w/ ESS		✓	18.91	0.592	0.294	0.136
w/ RDR	✓		19.26	0.608	0.317	0.134
DropoutGS	✓	✓	**19.35**	**0.622**	**0.282**	**0.128**

Table 4. **Quantitative results of ablation study.** We ablate our model with 3-view input on the LLFF dataset. "Baseline" denotes DNGaussian.

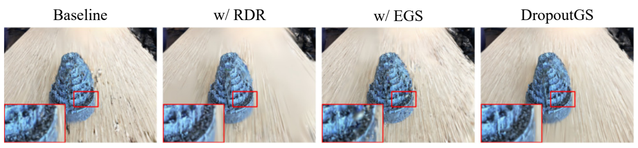


Figure 10. **Qualitative results of ablation study.** The model with only ESS can cause over-splitting, where too many insufficiently trained Gaussians are generated in high-frequency regions as artifacts. Only the full model can generate clean and sharp novel views.

Views	Method	PSNR ↑	LPIPS ↓	SSIM ↑	AVGE ↓
3	3DGS	15.52	0.405	0.408	0.209
	3DGS†	16.46	0.401	0.440	0.192
	DNGaussian	19.12	0.294	0.591	0.132
	DropoutGS (Ours)	**19.23**	**0.287**	**0.614**	**0.130**
6	3DGS	20.63	0.226	0.699	0.108
	3DGS†	21.09	0.229	0.699	0.103
	DNGaussian	22.18	0.198	0.755	0.088
	DropoutGS (Ours)	**23.35**	**0.177**	**0.791**	**0.076**
9	3DGS	20.44	0.230	0.697	0.108
	3DGS†	23.21	0.176	0.785	0.076
	DNGaussian	23.17	0.180	0.788	0.077
	DropoutGS (Ours)	**24.33**	**0.160**	**0.825**	**0.066**

Table 5. **Comparison with 3, 6, and 9 input views on LLFF.**

baselines at all training settings. However, it can be observed that the performance improvement of DropoutGS decreases at denser viewpoint settings. This is mainly because denser viewpoints provide more constraints to the model during training, thus reducing its dependence on external constraints.

Dropout Rate. We conduct ablation experiments on the dropout rate to investigate its impact on RDR performance. Table 6 reports the performance of DropoutGS with varying dropout rates using 3 views with visual results being shown in Fig. 11. As the dropout rate increases from 0.0 to 0.2, quantitative performance gains are achieved in terms of all metrics with fewer visual artifacts. When the dropout rate continues to increase, although higher PSNR is produced, the LPIPS score suffers a slight degradation due to excessive smoothing of the resultant images.

Edge Threshold. We further conduct experiments to investigate the impact of the edge threshold in our ESS strategy. Quantitative and qualitative results are presented in Table 7 and Fig. 12. As the edge threshold decreases, high-frequency details can be better preserved such that the LPIPS score is consistently improved from 0.317 to 0.275.

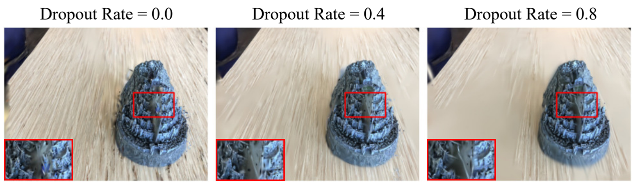


Figure 11. **Qualitative results of the ablation study on the random dropout rate.** A large dropout rate results in a reduction of artifacts but also a loss of details.

Dropout Rate	PSNR ↑	SSIM ↑	LPIPS ↓	AVGE ↓
0.0	18.76	0.582	0.300	0.139
0.2	19.20	0.616	0.279	0.130
0.4	19.35	0.622	0.282	0.128
0.6	19.37	0.622	0.298	0.130
0.8	19.44	0.621	0.302	0.129

Table 6. **Ablation results on the random dropout rate.** We conduct the reproduced performance of DNGaussian as the results without random dropout regularization.

Edge Threshold (ET.)	PSNR ↑	SSIM ↑	LPIPS ↓	AVGE ↓
INF	19.26	0.608	0.317	0.134
1×10^{-1}	19.22	0.608	0.316	0.134
1×10^{-2}	19.29	0.611	0.313	0.133
1×10^{-3}	19.35	0.622	0.282	0.128
1×10^{-4}	19.17	0.618	0.275	0.129

Table 7. **Ablation results on the edge threshold.** We explore the performance of different edge thresholds on the LLFF dataset with 3 views as input.

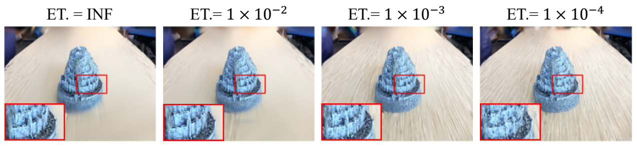


Figure 12. **The visualization result of the ablation study on the edge threshold.** The highlighted foreground region appears overfitting degradation at excessively small edge thresholds.

However, excessively low thresholds lead to notable distortion with degraded PSNR performance. Overall, 0.001 is selected as the default setting of the edge threshold.

5. Conclusion

In this paper, we present a novel coarse-to-fine framework, termed DropoutGS, for sparse view rendering. Specifically, we introduce the dropout technique into 3DGS and propose a random dropout regularization to alleviate overfitting. Then, an edge-guided splitting strategy is designed for further recovery of the high-frequency details. Extensive experiments validate the effectiveness and superiority of our approach over existing methods and demonstrate its compatibility with 3DGS-based techniques.

Acknowledgements

This work was partially supported by the National Natural Science Foundation of China (No. U20A20185, 62372491, 62301601), the Guangdong Basic and Applied Basic Research Foundation (No. 2022B1515020103, 2023B1515120087), the Shenzhen Science and Technology Program (No. RCYX20200714114641140), the Science and Technology Research Projects of the Education Office of Jilin Province (No. JJKH20251951KJ), and the SYSU-Sendhui Joint Lab on Embodied AI.

References

[1] Pierre Baldi and Peter J Sadowski. Understanding dropout. *NeurIPS*, 2013. 4

[2] 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. 2, 5, 6

[3] 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. 2

[4] 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. 6

[5] Julian Chibane, Aayush Bansal, Verica Lazova, and Gerard Pons-Moll. Stereo Radiance Fields (SRF): Learning view synthesis for sparse views of novel scenes. In *CVPR*, 2021. 6

[6] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3D-R2N2: A unified approach for single and multi-view 3D object reconstruction. In *ECCV*, 2016. 1

[7] Jaeyoung Chung, Jeongtaek Oh, and Kyoung Mu Lee. Depth-regularized optimization for 3D gaussian splatting in few-shot images. In *CVPR*, 2024. 1, 2

[8] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised NeRF: Fewer views and faster training for free. In *CVPR*, 2022. 2

[9] Terrance DeVries. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017. 2

[10] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3D object reconstruction from a single image. In *CVPR*, 2017. 1

[11] Guangchi Fang and Bing Wang. Mini-Splatting: Representing scenes with a constrained number of gaussians. *arXiv preprint arXiv:2403.14166*, 2024. 2

[12] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, 2016. 2

[13] Yiming Gao, Yan-Pei Cao, and Ying Shan. SurfelNeRF: Neural surfel radiance fields for online photorealistic reconstruction of indoor scenes. In *CVPR*, 2023. 2

[14] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. FastNeRF: High-fidelity neural rendering at 200fps. In *ICCV*, 2021. 2

[15] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 2

[16] Abdullah Hamdi, Luke Melas-Kyriazi, Jinjie Mai, Guocheng Qian, Ruoshi Liu, Carl Vondrick, Bernard Ghanem, and Andrea Vedaldi. GES: Generalized exponential splatting for efficient radiance field rendering. In *CVPR*, 2024. 2

[17] Kazuyuki Hara, Daisuke Saitoh, and Hayaru Shouno. Analysis of dropout learning regarded as ensemble learning. In *ICANN*, 2016. 4

[18] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2D gaussian splatting for geometrically accurate radiance fields. In *SIGGRAPH*, 2024. 2

[19] Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting NeRF on a Diet: Semantically consistent few-shot view synthesis. In *ICCV*, 2021. 5, 6

[20] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *CVPR*, 2014. 5

[21] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D gaussian splatting for real-time radiance field rendering. *TOG*, 2023. 1, 3, 5, 6

[22] Min-Seop Kwak, Jiuhn Song, and Seungryong Kim. GeCoNeRF: Few-shot neural radiance fields via geometric consistency. *arXiv preprint arXiv:2301.10941*, 2023. 2

[23] Jiahe Li, Jiawei Zhang, Xiao Bai, Jin Zheng, Xin Ning, Jun Zhou, and Lin Gu. DNGaussian: Optimizing sparse-view 3D gaussian radiance fields with global-local depth normalization. In *CVPR*, 2024. 1, 2, 5, 6

[24] Sicheng Li, Hao Li, Yue Wang, Yiyi Liao, and Lu Yu. SteerNeRF: Accelerating nerf rendering via smooth viewpoint trajectory. In *CVPR*, 2023. 2

[25] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017. 2

[26] 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. *TOG*, 2019. 5

[27] 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. *COMMUN ACM*, 2021. 1, 2, 5

[28] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *TOG*, 2022. 2

[29] Michael Niemeyer, Jonathan T Barron, Ben Mildenhall, Mehdi SM Sajjadi, Andreas Geiger, and Noha Radwan. RegNeRF: Regularizing neural radiance fields for view synthesis from sparse inputs. In *CVPR*, 2022. 6

[30] Leslie Rice, Eric Wong, and Zico Kolter. Overfitting in adversarially robust deep learning. In *ICML*, 2020. 2

[31] Barbara Roessle, Jonathan T Barron, Ben Mildenhall, Pratul P Srinivasan, and Matthias Nießner. Dense depth priors for neural radiance fields from sparse input views. In *CVPR*, 2022. 2

[32] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 7

[33] Seunghyeon Seo, Yeonjin Chang, and Nojun Kwak. FlipNeRF: Flipped reflection rays for few-shot novel view synthesis. In *ICCV*, 2023. 2

[34] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 2014. 2

[35] Prune Truong, Marie-Julie Rakotosaona, Fabian Manhardt, and Federico Tombari. SPARF: Neural radiance fields from sparse and noisy poses. In *CVPR*, 2023. 2

[36] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *ICML*, 2013. 2

[37] Guangcong Wang, Zhaoxi Chen, Chen Change Loy, and Ziwei Liu. SparseNeRF: Distilling depth ranking for few-shot novel view synthesis. In *ICCV*, 2023. 2, 5, 6

[38] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3D mesh models from single RGB images. In *ECCV*, 2018. 1

[39] 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. 2

[40] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *TIP*, 2004. 5

[41] David Warde-Farley, Ian J Goodfellow, Aaron Courville, and Yoshua Bengio. An empirical analysis of dropout in piecewise linear networks. *arXiv preprint arXiv:1312.6197*, 2013. 4

[42] Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994*, 2020. 2

[43] Haibing Wu and Xiaodong Gu. Max-pooling dropout for regularization of convolutional neural networks. In *ICONIP*, 2015. 2

[44] Haolin Xiong, Sairisheek Muttukuru, Rishi Upadhyay, Pradyumna Chari, and Achuta Kadambi. SparseGS: Real-time 360° sparse view synthesis using gaussian splatting. *arXiv preprint arXiv:2312.00206*, 2023. 2

[45] Jiawei Yang, Marco Pavone, and Yue Wang. FreeNeRF: Improving few-shot neural rendering with free frequency regularization. In *CVPR*, 2023. 2, 5, 6

[46] Zongxin Ye, Wenyu Li, Sidun Liu, Peng Qiao, and Yong Dou. AbsGS: Recovering fine details in 3D gaussian splatting. In *ACM MM*, 2024. 2, 4

[47] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *CVPR*, 2021. 6

[48] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, 2019. 2

[49] Hongyi Zhang. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 2

[50] Jiawei Zhang, Jiahe Li, Xiaohan Yu, Lei Huang, Lin Gu, Jin Zheng, and Xiao Bai. CoR-GS: sparse-view 3D gaussian splatting via co-regularization. In *ECCV*, 2025. 7

[51] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 5

[52] Zheng Zhang, Wenbo Hu, Yixing Lao, Tong He, and Hengshuang Zhao. Pixel-GS: Density control with pixel-aware gradient for 3D gaussian splatting. *arXiv preprint arXiv:2403.15530*, 2024. 2

[53] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *AAAI*, 2020. 2

[54] Zehao Zhu, Zhiwen Fan, Yifan Jiang, and Zhangyang Wang. FSGS: Real-time few-shot view synthesis using gaussian splatting. In *ECCV*, 2025. 2, 7