Gaussian Graph Network: Learning Efficient and Generalizable Gaussian Representations from Multi-view Images

Shengjun Zhang, Xin Fei, Fangfu Liu, Haixu Song, Yueqi Duan*
Tsinghua University
{zhangsj23, feix21}@mails.tsinghua.edu.cn, duanyueqi@tsinghua.edu.cn

Abstract

3D Gaussian Splatting (3DGS) has demonstrated impressive novel view synthesis performance. While conventional methods require per-scene optimization, more recently several feed-forward methods have been proposed to generate pixel-aligned Gaussian representations with a learnable network, which are generalizable to different scenes. However, these methods simply combine pixel-aligned Gaussians from multiple views as scene representations, thereby leading to artifacts and extra memory cost without fully capturing the relations of Gaussians from different images. In this paper, we propose Gaussian Graph Network (GGN) to generate efficient and generalizable Gaussian representations. Specifically, we construct Gaussian Graphs to model the relations of Gaussian groups from different views. To support message passing at Gaussian level, we reformulate the basic graph operations over Gaussian representations, enabling each Gaussian to benefit from its connected Gaussian groups with Gaussian feature fusion. Furthermore, we design a Gaussian pooling layer to aggregate various Gaussian groups for efficient representations. We conduct experiments on the large-scale RealEstate10K and ACID datasets to demonstrate the efficiency and generalization of our method. Compared to the state-of-the-art methods, our model uses fewer Gaussians and achieves better image quality with higher rendering speed.

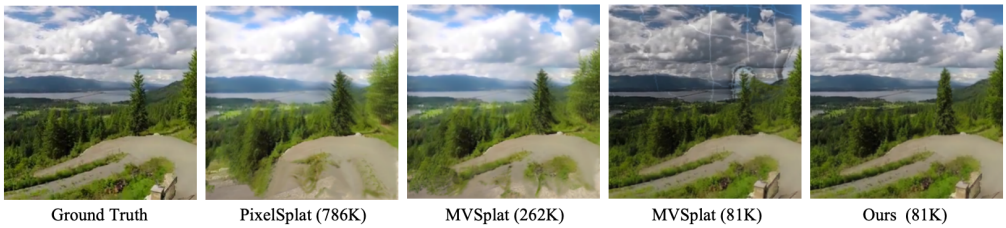
1 Introduction

Novel view synthesis is a fundamental problem in computer vision due to its widespread applications, such as virtual reality, augmented reality, robotics and so on. Remarkable progress has been made using neural implicit representations [33, 42, 43], but these methods suffer from expensive time consumption in training and rendering [28, 13, 1, 40, 65, 11, 17, 34]. Recently, 3D Gaussian Splatting (3DGS) [21] has drawn increasing attention for explicit Gaussian representations and real-time rendering performance. Benefiting from rasterization-based rendering, 3DGS avoids dense points querying in scene space, so that it can maintain high efficiency and quality.

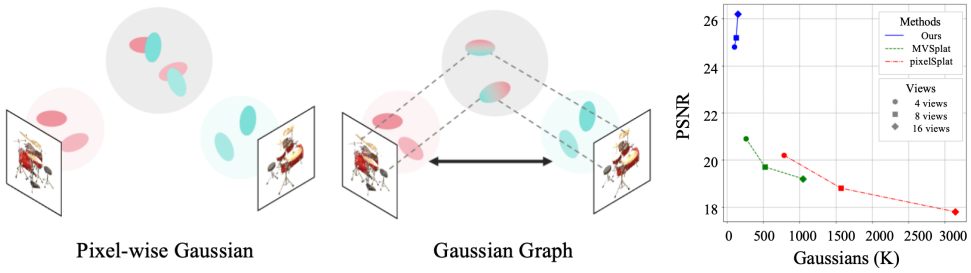
Since 3DGS relies on per-subject [21] or per-frame [31] parameter optimization, several generalizable methods [68, 6, 4, 45] are proposed to directly regress Gaussian parameters with feed-forward networks. Typically, these methods [6, 4] generate pixel-aligned Gaussians with U-Net architectures, epipolar transformers or cost volume representations for depth estimation and parameter predictions, and directly combine Gaussian groups obtained from different views as scene representations. However, such combination of Gaussians leads to superfluous representations, where the overlapped regions are covered by similar Gaussians predicted separately from multiple images. While a simple solution is to delete redundant Gaussians, it ignores the connection among Gaussian groups. As

*Corresponding author.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).



(a)



(b) (c)

Figure 1: Comparison of previous methods and ours. (a) We visualize the rendering results of various methods and report the number of Gaussians in parentheses. (b) Previous pixel-wise methods can be considered as a degraded case of Gaussian Graphs without edges. (c) We report PSNR as well as the number of Gaussians for pixelSplat [4], MVSplat [6] and GNN under different input settings.

illustrated in Figure 1a, pixelSplat [4] and MVSplat [6] suffer from artifacts with several times as many Gaussians as ours, while the deletion of similar Gaussians hurts rendering quality.

To tackle the challenge, we propose Gaussian Graphs to model the relations of Gaussian groups from multiple views. Based on this structure, we present Gaussian Graph Network (GGN), extending conventional graph operations to Gaussian domain, so that Gaussians from different views are not independent but can learn from their neighbor groups. Precisely, we reformulate the scalar weight of an edge to a weight matrix to depict the interactions between two Gaussian groups, and introduce a Gaussian pooling strategy to aggregate Gaussians. Under this definition, previous methods [4, 6] can be considered as a degraded case of Gaussian Graphs without edges. As shown in Figure 1b, our GGN allows message passing and aggregation across Gaussians for efficiency representations.

We conduct extensive experiments on both indoor and outdoor datasets, including RealEstate10K [69] and ACID [26]. While the performance of previous methods declines as the number of input views increases, our method can benefit from more input views. As shown in Figure 1c, our model outperforms previous methods under different input settings with higher rendering quality and fewer Gaussian representations. Our main contributions can be summarized as follows:

- We propose Gaussian Graphs to construct the relations of different Gaussian groups, where each node is a set of pixel-aligned Gaussians from an input view.
- We introduce Gaussian Graph Network to process Gaussian Graphs by extending the graph operations to Gaussian domain, bridging the interaction and aggregation across Gaussian groups.
- Experimental results illustrate that our method can generate efficient and generalizable Gaussian representations. Our model requires fewer Gaussians and achieves better rendering quality.

2 Related Works

2.1 Neural Implicit Representations

Early researches focus on capturing dense views to reconstruct scenes, while neural implicit representations have significantly advanced neural processing for 3D data and multi-view images, leading to high reconstruction and rendering quality [32, 38, 64, 43]. In particular, Neural Radiance Fields (NeRF) [33] has garnered considerable attention with a fully connected neural network to represent

complex 3D scenes. Subsequently, following works have emerged to address NeRF's limitations and enhance its performance. Some studies aim to solve the long-standing problem of novel view synthesis by improving the speed and efficiency of training and inference [17, 13, 34, 40, 28, 65, 11]. Other research focuses on modeling complex geometry and view-dependent effects to reconstruct dynamic scenes [8, 22, 39, 48, 55, 25, 52, 47, 23, 51]. Additionally, some studies [50, 46, 61, 56] have worked on reconstructing large urban scenes to avoid blurred renderings without fine details, which is a challenge for NeRF-based methods due to their limited model capacity. Furthermore, other works [36, 49, 54, 59] apply NeRF to novel view synthesis with sparse input views by incorporating additional training regularizations, such as depth, correspondence, and diffusion models.

2.2 3D Gaussian Splatting

More recently, 3D Gaussian Splatting (3DGS) [21] has drawn significant attention in the field of computer graphics, especially in the realm of novel view synthesis. Different from the expensive volume sampling strategy in NeRF, 3DGS utilizes a much more efficient rasterization-based splatting approach to render novel views from a set of 3D Gaussian primitives. However, 3DGS may suffer from artifacts, which occur during the splatting process. Thus, several works [62, 12, 19, 24] have been proposed to enhance the quality and realness of rendered novel views. Since 3DGS requires millions of parameters to represent a single scene, resulting in significant storage demands, some researches [30, 35, 14, 10, 20] focus on reducing the memory usage to ensure real-time rendering while maintaining the rendering quality. Other works [70, 58] are proposed to reduce the amount of required images to reconstruct scene. Besides, to extend the capability of 3D Gaussians from representing static scenes to 4D scenarios, several works [53, 9, 63, 57] have been proposed to incorporate faithful dynamics which are aligned with real-world physics.

2.3 Generalizable Novel View Synthesis

Optimization-based methods train a separate model for each individual scene and require dozens of images to synthesis high-quality novel views. In contrast, feed-forward models learn powerful priors from large-scale datasets, so that 3D reconstruction and view synthesis can be achieved via a single feed-forward inference. A pioneering work, pixelNeRF [66], proposes a feature-based method to employ the encoded features to render novel views. MVSNeRF [5] combines plane-swept cost volumes which is widely used in multi-view stereo with physically based volume rendering to further improve the quality of neural radiance field reconstruction. Subsequently, many researches [2, 3, 15, 18, 37, 41, 27] have developed novel view synthesis methods with generalization and decomposition abilities. Several generalizable 3D Gaussian Splatting methods have also been proposed to leverage sparse view images to reconstruct novel scenes. While Splatter Image [45] and GPS-Gaussian [68] regress pixel-aligned Gaussian parameters to reconstruct single objects or humans instead of complex scenes, pixelSplat [4] takes sparse views as input to predict Gaussian parameters by leveraging epipolar geometry and depth estimation. MVSplat [6] constructs a cost volume structure to directly predict depth from cross-view features, further improving the geometric quality.

However, these methods follow the paradigm of regressing pixel-aligned Gaussians and combine Gaussians from different views directly, resulting in an excessive number of Gaussians when the model processes multi-view inputs. In comparison, we construct Gaussian Graphs to model the relations of Gaussian groups. Furthermore, we introduce Gaussian Graph Network with specifically designed graph operations to ensure the interaction and aggregation across Gaussian groups. In this manner, we can obtain efficient and generalizable Gaussian representations from multi-view images.

3 Method

The overall framework is illustrated in Figure 2. After predicting positions and features of Gaussian groups from input views with extracted feature maps, we build a Gaussian Graph to model the relations. Then, we process the graph with Gaussian Graph Network via our designed graph operations on Gaussian domain for information exchange and aggregation across Gaussian groups. We leverage the fused Gaussians features to predict other Gaussian parameters, including opacity α, covariance matrix Σ and colors c.

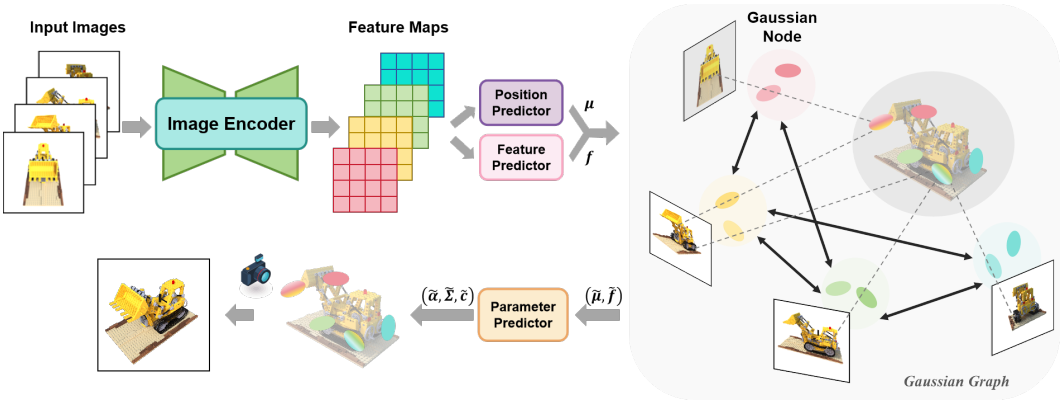


Figure 2: Overview of Gaussian Graph Network. Given multiple input images, we extract image features and predict the means and features of pixel-aligned Gaussians. Then, we construct a Gaussian Graph to model the relations between different Gaussian nodes. We introduce Gaussian Graph Network to process our Gaussian Graph. The parameter predictor generates Gaussians parameters from the output Gaussian features.

3.1 Building Gaussian Graphs

Given N input images $\mathcal{I} = \{I_i\} \in \mathbb{R}^{N\times H\times W\times 3}$ and their corresponding camera parameters $\mathcal{C} = \{c_i\}$, we follow the instructions of pixelSplat [4] and MVSplat [6] to extract image features:

$$\mathcal{F} = \Phi_{image}(\mathcal{I}), \quad \mathcal{F} = \{F_i\} \in \mathbb{R}^{N\times \frac{H}{4}\times \frac{W}{4}\times C}, \tag{1}$$

where Φ_{image} is a 2D backbone. We predict the means and features of pixel-aligned Gaussians:

$$\mu_i = \psi_{unproj}\left(\Phi_{depth}(F_i), c_i\right) \in \mathbb{R}^{HW\times 3}, \quad f_i = \Phi_{feat}(F_i) \in \mathbb{R}^{HW\times D}, \tag{2}$$

where Φ_{depth} and Φ_{feat} stand for neural networks to predict depth maps and Gaussian features, and ψ_{unproj} is the unprojection operation.

We build a Gaussian Graph G with nodes $V = \{v_i\} = \{(\mu_i, f_i)\}_{1\le i\le N}$. To model the relations between nodes, we define its adjacency matrix $A = [a_{ij}]_{N\times N}$ as:

$$a_{ij} = \begin{cases} 1, & i = j \\ \psi_{overlap}(v_i, v_j), & i \neq j \end{cases} \tag{3}$$

where $\psi_{overlap}(v_i, v_j)$ computes the overlap ratio between view i and view j. To limit further computational complexity, we prune the graph by preserving edges with top n weights and ignore other possible edges. The degree matrix $D = [d_{ij}]_{N\times N}$ satisfies $d_{ii} = \sum_j a_{ij}$. Thus, the scaled adjacency can be formulated as:

$$\tilde{A} = D^{-1}A \quad \text{or} \quad \tilde{A} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}. \tag{4}$$

3.2 Gaussian Graph Network

Linear Layers. Assuming that a conventional graph has N nodes with features $\{g_i\} \in \mathbb{R}^{N\times C}$ and the scaled adjacency matrix $A = [\tilde{a}_{ij}]_{N\times N}$, the basic linear operation can be formulated as:

$$\hat{g}_i = \sum_{j=1}^{N} \tilde{a}_{ij} g_j W \in \mathbb{R}^D, \tag{5}$$

where $W \in \mathbb{R}^{C\times D}$ is a learnable weight. Different from the vector nodes g_i in conventional graphs, each node of our Gaussian Graph contains a set of pixel-aligned Gaussians $v_i = \{\mu_i, f_i\}$. Therefore, we extend the scalar weight a_{sk} of an edge to a matrix $E^{s\to k} = [e_{ij}^{s\to k}]_{HW\times HW}$, which depicts the detailed relations at Gaussian-level between v_s and v_k. For the sake of simplicity, we define

$$e_{ij}^{s\to k} = \begin{cases} 1, & \psi_{proj}(\mu_{s,i}, c_s) = \psi_{proj}(\mu_{k,j}, c_s) \\ 0, & \psi_{proj}(\mu_{s,i}, c_s) \neq \psi_{proj}(\mu_{k,j}, c_s) \end{cases} \tag{6}$$

Algorithm 1 Gaussian Graph Network

Input: Multi-view images $\mathcal{I} = \{I_i\}$, camera parameters $\mathcal{C} = \{c_i\}$, the number of graph layers h
Output: Gaussian parameters (μ, Σ, α, c)

$\mathcal{F} \leftarrow \Phi_{image}(\mathcal{I}, \mathcal{C})$
$\mu_i \leftarrow \psi_{unproj}\left(\Phi_{depth}(F_i), c_i\right), \quad f_i \leftarrow \Phi_{feat}(F_i) \quad i = 1, 2, \cdots, N$
$\tilde{A} \leftarrow D^{-1}A$
for $k \leftarrow 1$ to h **do**
  $f_i \leftarrow \sigma\left(\sum \tilde{a}_{ij} E^{j \to i} f_j W\right) \quad i = 1, 2, \cdots, N$
end for
$(\mu, f) \leftarrow \bigcup_{s=1}^{l} \phi_{pooling}(G^s)$
$R, S, \alpha, c \leftarrow \psi_R(f), \psi_S(f), \psi_\alpha(f), \psi_c(f)$
$\Sigma \leftarrow RSS^\top R^\top$

where $\mu_{s,i}$ is the center of the i-th Gaussian in Gaussian group v_s, and ψ_{proj} is the projection function which returns coordinates of the occupied pixel. In this manner, the linear operation on a Gaussian Graph can be defined as:

$$\hat{f}_i = \sum_{j=1}^{N} \tilde{a}_{ij} E^{j \to i} f_j W \in \mathbb{R}^D. \tag{7}$$

Pooling layers. If we have a series of connected Gaussian Graphs $\{G^s\}_{1 \le s \le l}$, where $G^s \subseteq G$, then we operate the specific pooling operation on each G^s and combine them together:

$$\phi_{pooling}(G) = \bigcup_{s=1}^{l} \phi_{pooling}(G^s). \tag{8}$$

If G^s only contains one node v^s, $\phi_{pooling}(G^s) = v^s$. Otherwise, we start with a random selected node $v_i^s = (\mu_i^s, f_i^s) \in G^s$. Since G^s is a connected graph, we can randomly pick up its neighbor v_j^s with corresponding camera parameters c_j^s. We define a function $\psi_{similarity}$:

$$\psi_{similarity}\left(v_{j,m}^s, v_i^s\right) = \max_n e_{mn}^{j \to i}, \tag{9}$$

where $v_{j,m}^s$ is the m-th Gaussian in node v_j^s. We define the merge function ψ_{merge}:

$$\psi_{merge}\left(v_{j,m}^s, v_i^s\right) = \begin{cases} \varnothing, & \psi_{similarity}\left(v_{j,m}^s, v_i^s\right) = 1 \\ v_{j,m}^s, & \text{otherwise} \end{cases} \tag{10}$$

Then, we merge two nodes together to get a new node

$$v_{new}^s = \bigcup_{m=1}^{HW} \psi_{merge}\left(v_{j,m}^s, v_i^s\right) \cup v_i^s \tag{11}$$

In this manner, we aggregate a connected graph G^s step by step to one node. Specifically, previous methods can be considered as a degraded Gaussian Graph without edges:

$$\phi_{pooling}(G) = \bigcup_{s=1}^{N} \phi_{pooling}(G^s) = \bigcup_{s=1}^{N} v^s. \tag{12}$$

where $\phi_{pooling}$ degenerates to simple combination of nodes.

Parameter prediction. After aggregating the Gaussian Graph $(\mu, f) = \psi_{pooling}(G)$, we predict the rotation matrix R, scale matrix S, opacity α and color c:

$$R = \phi_R(f), \quad S = \phi_S(f), \quad \alpha = \phi_\alpha(f), \quad c = \phi_c(f) \tag{13}$$

where ϕ_R, ϕ_S, ϕ_α and ϕ_c are prediction heads. The covariance matrix can be formulated as:

$$\Sigma = RSS^\top R^\top. \tag{14}$$

Our Gaussian Graph Network is illustrated in Algorithm 1, where σ stands for non-linear operations, such as ReLU and GeLU.

Table 1: Quantitative comparison on RealEstate10K (re10K) [69] and ACID [26] benchmarks. We evaluate all models with 4, 8, 16 input views and report PSNR as well as the number of Gaussians (K). † Models accept multi-view inputs, and only preserve Gaussians from two input views for rendering.

Datasets	Methods	4 views		8 views		16 views	
		PSNR	Gaussians	PSNR	Gaussians	PSNR	Gaussians
re10k	pixelSplat [4]	20.19	786	18.78	1572	17.80	3175
	pixelSplat†	20.84	393	20.79	393	20.75	393
	MVSplat [6]	20.86	262	19.69	524	19.18	1049
	MVSplat†	21.48	131	21.39	131	21.34	131
	Ours	24.76	102	25.15	126	26.18	150
ACID	pixelSplat [4]	20.15	786	18.84	1572	17.32	3175
	pixelSplat†	23.12	393	23.07	393	23.04	393
	MVSplat [6]	20.30	262	19.02	524	17.64	1049
	MVSplat†	23.78	131	23.72	131	23.70	131
	Ours	26.46	102	26.94	126	27.69	150

Table 2: Novel view synthesis results with two-view inputs. We report the average results of PSNR, SSIM and LPIPS on all test scenes.

Method	Param (M)	RealEstate10K [69]			ACID [26]		
		PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
pixelNeRF [66]	28.2	20.43	0.589	0.550	20.97	0.547	0.533
MuRF [59]	5.3	26.10	0.858	0.143	28.09	0.841	0.155
pixelSplat [4]	125.4	25.89	0.858	0.142	28.14	0.839	0.150
MVSplat [6]	12.0	26.39	0.869	0.128	28.25	0.843	0.144
Ours	12.5	26.42	0.870	0.126	28.40	0.845	0.143

4 Experiments

We validate our proposed Gaussian Graph Network for novel view synthesis. In Section 4.1, we compare various methods on RealEstate10K [69] and ACID [26] under different input settings. In Section 4.2, we analyze the efficiency of our representations. In Section 4.3, we further validate the generalization of our Gaussian representations under cross dataset evaluation. In Section 4.4, we ablate our designed operations of Gaussian Graph Network.

4.1 Multi-view Scene Reconstruction and Synthesis

Datasets. We conduct experiments on two large-scale datasets, including RealEstate10K [69] and ACID [26]. RealEstate10K dataset comprises video frames of real estate scenes, which are split into 67,477 scenes for training and 7,289 scenes for testing. ACID dataset consists of natural landscape scenes, with 11,075 training scenes and 1,972 testing scenes. For two-view inputs, our model is trained with two views as input, and subsequently tested on three target novel views for each scene. For multi-view inputs, we construct challenging subsets for RealEstate10K and ACID across a broader range of each scenario to evaluate model performance. We select 4, 8 and 16 views as reference views, and evaluate pixelSplat [4], MVSplat [6] and ours on the same target novel views.

Implementation details. Our model is trained with two input views for each scene on a single A6000 GPU for 300,000 iterations, utilizing the Adam optimizer. Following the instruction of previous methods [4, 6], all experiments are conducted on 256×256 resolutions for fair comparison. The image backbone is initialized by the feature extractor and cost volume representations in MVSplat [6]. The number of graph layer is set to 2. The training loss is a linear combination of MSE and LPIPS [67] losses, with loss weights of 1 and 0.05.

Results. As shown in Table 1 and Table 2, our Gaussian Graph benefits from increasing input views, whereas both pixelSplat [4] and MVSplat [6] exhibit declines in performance. This distinction highlights the superior efficiency of our approach, which achieves more effective 3D representation

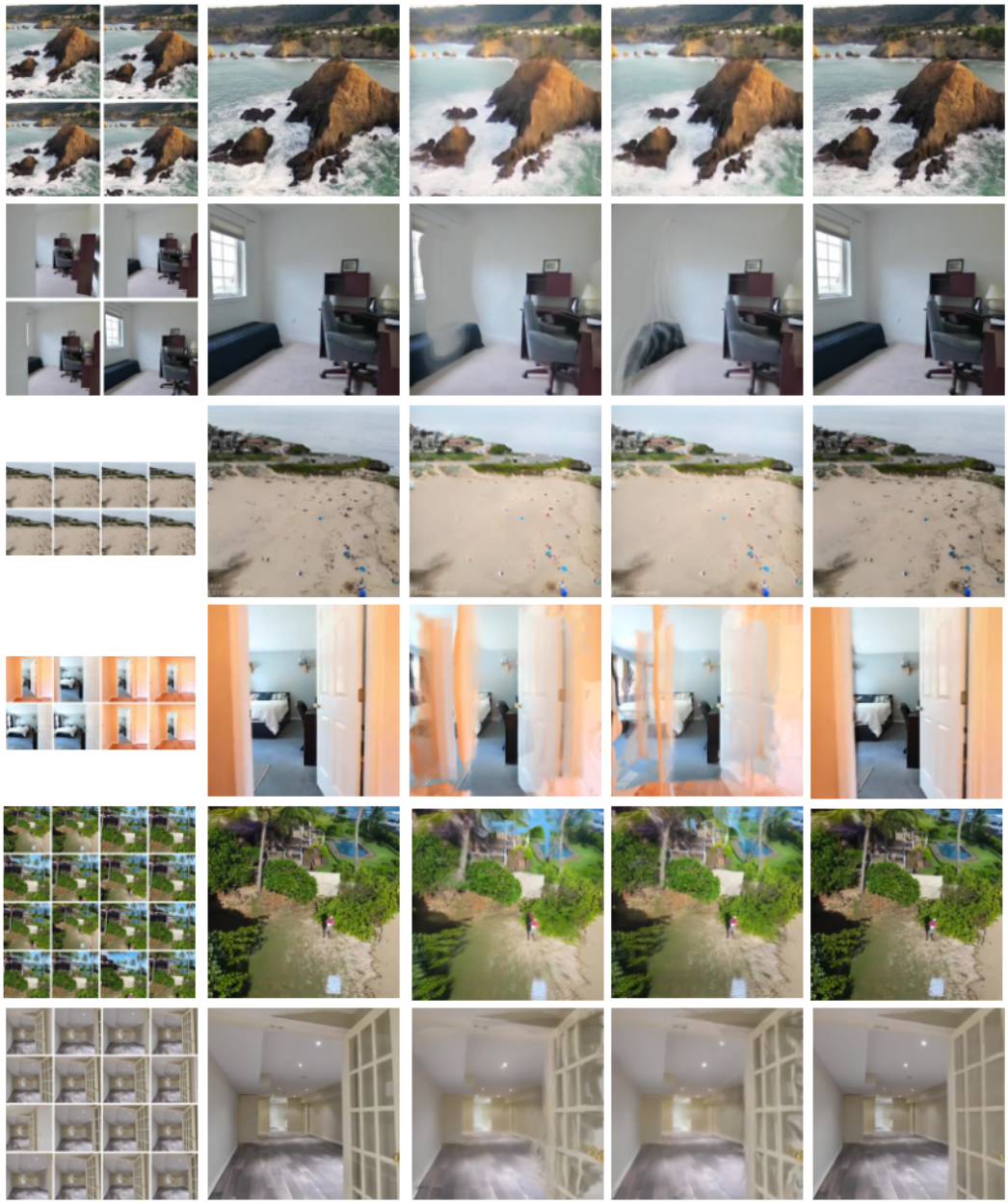


Figure 3: Visualization results on RealEstate10K [69] and ACID [26] benchmarks. We evaluate all models with 4, 8, 16 views as input and subsequently test on three target novel views.

with significantly fewer Gaussians compared to pixel-wise methodologies. For 4 view inputs, our method outperforms MVSplat [6] by about 4dB on PSNR with more than $2\times$ fewer Gaussians. For more input views, the number of Gaussians in previous methods increases linearly, while our GGN only requires a small increase. Considering that previous methods suffer from the redundancy of Gaussians, we adopt these models with multi-view inputs and preserve Gaussians from two input views for rendering. Under this circumstance, pixelSplat [4] and MVSplat [6] achieve better performance than before, because the redundancy of Gaussians is alleviated to some degree. However, the lack of interaction at Gaussian level limits the quality of previous methods. In contrast, our Gaussian Graph can significantly enhance performance by leveraging additional information from extra views. Visualization results in Figure 3 also indicate that pixelSplat [4] and MVSplat [6] tend to suffer from artifacts due to duplicated and unnecessary Gaussians in local areas, which increasingly affects image quality as more input views are added.

Table 3: Inference time comparison across different views. We train our model on 2 input views and report the inference time for 4 views, 8 views, and 16 views, respectively.

		2 views	4 views	8 views	16 views
pixelSplat [4]	125.4	137.3	298.8	846.5	2938.9
MVSplat [6]	12.0	60.6	126.4	363.2	1239.8
Ours	12.5	75.6	148.1	388.8	1267.5

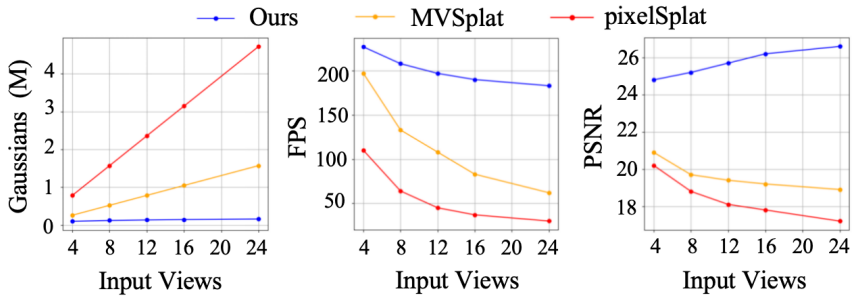


Figure 4: Efficiency analysis. We report the number of Gaussians (M), rendering frames per second (FPS) and reconstruction PSNR of pixelSplat [4], MVSplat [6] and our GGN.

4.2 Efficiency Analysis.

In addition to delivering superior rendering quality, our Gaussian Graph network also enhances efficiency in 3D representations. As illustrated in Figure 4, when using 24 input images, our model outperforms MVSplat by 8.6dB on PSNR with approximately one-tenth the number of 3D Gaussians and more than three times faster rendering speed. Additionally, we compare the average inference time of our model with pixel-wise methods in Table 3. Our GGN is able to efficiently remove duplicated Gaussians and enhance Gaussian-level interactions, which allows it to achieve superior reconstruction performance with comparable inference speed to MVSplat.

4.3 Cross Dataset Generalization.

To further demonstrate the generalization of Gaussian Graph Network, we conduct cross-dataset experiments. Specifically, all models are trained on RealEstate10K [69] or ACID [26] datasets, and are tested on the other dataset without any fine-tuning. Our method constructs the Gaussian Graph according to the relations of input views. As shown in Table 4, our GGN consistently outperforms pixelSplat [4] and MVSplat [6] on both benchmarks.

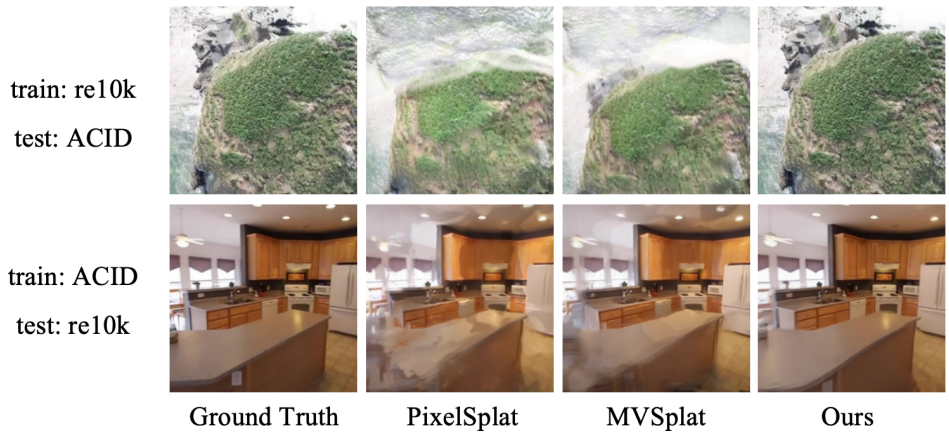


Figure 5: Visualization of model performance for cross-dataset generalization on RealEstate10K [69] and ACID [26] benchmarks.

Table 4: Cross-dataset performance and efficiency comparisons on RealEstate10K [69] and ACID [26] benchmarks. We assign eight views as reference and test on three target views for each scene.

Train Data	Test Data	Method	PSNR↑	SSIM↑	LPIPS↓	Gaussians (K)	FPS
ACID	re10k	pixelSplat	18.65	0.715	0.276	1572	64
		MVSplat	19.17	0.731	0.270	524	133
		Ours	24.75	0.759	0.252	126	208
re10k	ACID	pixelSplat	19.75	0.724	0.264	1572	64
		MVSplat	20.58	0.735	0.239	524	133
		Ours	25.21	0.762	0.234	126	208

Table 5: Ablation study results of GGN on RealEstate10K [69] benchmarks. Each scene takes eight reference views and renders three novel views.

Models	PSNR↑	SSIM↑	LPIPS↓	Gaussians (K)
Gaussian Graph Network	25.15	0.786	0.232	126
w/o Gaussian Graph linear layer	24.72	0.778	0.246	126
w/o Gaussian Graph pooling layer	20.25	0.725	0.272	524
Vanilla	19.74	0.721	0.279	524

4.4 Ablations

To investigate the architecture design of our Gaussian Graph Network, we conduct ablation studies on RealEstate10K [69] benchmark. We first introduce a vanilla model without Gaussian Graph Network. Then, we simply adopt Gaussian Graph linear layer to model relations of Gaussian groups from multiple views. Furthermore, we simply introduce Gaussian Graph pooling layer to aggregate Gaussian groups to obtain efficient representations. Finally, we add the full Gaussian Graph Network model to both remove duplicated Gaussians and enhance Gaussian-level interactions.

Gaussian Graph linear layer. The Gaussian Graph linear layer serves as a pivotal feature fusion block, enabling Gaussian Graph nodes to learn from their neighbor nodes. The absence of linear layers leads to a performance drop of 0.43 dB on PSNR.

Gaussian Graph pooling layer. The Gaussian Graph pooling layer is important to avoid duplicate and unnecessary Gaussians, which is essential for preventing artifacts and floaters in reconstructions and speeding up the view rendering process. As shown in Table 5, the introduction of Gaussian Graph pooling layer improves the rendering quality by 4.9dB on PSNR and reduces the number of Gaussians to nearly one-fourth.

5 Conclusion and Discussion

In this paper, we propose Gaussian Graph to model the relations of Gaussians from multiple views. To process this graph, we introduce Gaussian Graph Network by extending the conventional graph operations to Gaussian representations. Our designed layers bridge the interaction and aggregation between Gaussian groups to obtain efficient and generalizable Gaussian representations. Experiments demonstrate that our method achieves better rendering quality with fewer Gaussians and higher FPS.

Limitations and future works. Although GGN produces compelling results and outperforms prior works, it has limitations. Because we predict pixel-aligned Gaussians for each view, the representations are sensitive to the resolution of input images. For high resolution inputs, *e.g.* 1024×1024, we generate over 1 million Gaussians for each view, which will significantly increase the inference and rendering time. GGN does not address generative modeling of unseen parts of the scene, where generative methods, such as diffusion models can be introduced to the framework for extensive generalization. Furthermore, GGN focuses on the color field, which does not fully capture the geometry structures of scenes. Thus, a few directions would be focused in future works.

Acknowledgements. This work was supported by the National Natural Science Foundation of China under Grant 62206147. We thank David Charatan for his help on experiments in pixelSplat.

References

[1] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV*, pages 5855–5864, 2021.

[2] Shengqu Cai, Anton Obukhov, Dengxin Dai, and Luc Van Gool. Pix2nerf: Unsupervised conditional p-gan for single image to neural radiance fields translation. In *CVPR*, pages 3981–3990, 2022.

[3] Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *CVPR*, pages 5799–5809, 2021.

[4] David Charatan, Sizhe Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. *arXiv preprint arXiv:2312.12337*, 2023.

[5] 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*, pages 14124–14133, 2021.

[6] 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. *arXiv preprint arXiv:2403.14627*, 2024.

[7] Yilun Du, Cameron Smith, Ayush Tewari, and Vincent Sitzmann. Learning to render novel views from wide-baseline stereo pairs. In *CVPR*, pages 4970–4980, 2023.

[8] Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B Tenenbaum, and Jiajun Wu. Neural radiance flow for 4d view synthesis and video processing. In *ICCV*, pages 14304–14314. IEEE Computer Society, 2021.

[9] Bardienus P Duisterhof, Zhao Mandi, Yunchao Yao, Jia-Wei Liu, Mike Zheng Shou, Shuran Song, and Jeffrey Ichnowski. Md-splatting: Learning metric deformation from 4d gaussians in highly deformable scenes. *arXiv preprint arXiv:2312.00583*, 2023.

[10] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. *arXiv preprint arXiv:2311.17245*, 2023.

[11] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, pages 5501–5510, 2022.

[12] Jian Gao, Chun Gu, Youtian Lin, Hao Zhu, Xun Cao, Li Zhang, and Yao Yao. Relightable 3d gaussian: Real-time point cloud relighting with brdf decomposition and ray tracing. *arXiv preprint arXiv:2311.16043*, 2023.

[13] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. In *ICCV*, pages 14346–14355, 2021.

[14] Sharath Girish, Kamal Gupta, and Abhinav Shrivastava. Eagles: Efficient accelerated 3d gaussians with lightweight encodings. *arXiv preprint arXiv:2312.04564*, 2023.

[15] Jiatao Gu, Lingjie Liu, Peng Wang, and Christian Theobalt. Stylenerf: A style-based 3d-aware generator for high-resolution image synthesis. *arXiv preprint arXiv:2110.08985*, 2021.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[17] Tao Hu, Shu Liu, Yilun Chen, Tiancheng Shen, and Jiaya Jia. Efficientnerf efficient neural radiance fields. In *CVPR*, pages 12902–12911, 2022.

[18] Wonbong Jang and Lourdes Agapito. Codenerf: Disentangled neural radiance fields for object categories. In *ICCV*, pages 12949–12958, 2021.

[19] Yingwenqi Jiang, Jiadong Tu, Yuan Liu, Xifeng Gao, Xiaoxiao Long, Wenping Wang, and Yuexin Ma. Gaussianshader: 3d gaussian splatting with shading functions for reflective surfaces. *arXiv preprint arXiv:2311.17977*, 2023.

[20] Kai Katsumata, Duc Minh Vo, and Hideki Nakayama. An efficient 3d gaussian representation for monocular/multi-view dynamic scenes. *arXiv preprint arXiv:2311.12897*, 2023.

[21] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ToG*, 42(4):1–14, 2023.

[22] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *CVPR*, pages 6498–6508, 2021.

[23] Zhengqi Li, Qianqian Wang, Forrester Cole, Richard Tucker, and Noah Snavely. Dynibar: Neural dynamic image-based rendering. In *CVPR*, pages 4273–4284, 2023.

[24] Zhihao Liang, Qi Zhang, Ying Feng, Ying Shan, and Kui Jia. Gs-ir: 3d gaussian splatting for inverse rendering. *arXiv preprint arXiv:2311.16473*, 2023.

[25] Haotong Lin, Sida Peng, Zhen Xu, Yunzhi Yan, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Efficient neural radiance fields for interactive free-viewpoint video. In *SIGGRAPH Asia*, pages 1–9, 2022.

[26] Andrew Liu, Richard Tucker, Varun Jampani, Ameesh Makadia, Noah Snavely, and Angjoo Kanazawa. Infinite nature: Perpetual view generation of natural scenes from a single image. In *ICCV*, pages 14458–14467, 2021.

[27] Fangfu Liu, Chubin Zhang, Yu Zheng, and Yueqi Duan. Semantic ray: Learning a generalizable semantic field with cross-reprojection attention. In *CVPR*, pages 17386–17396, 2023.

[28] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *NeurIPS*, 33:15651–15663, 2020.

[29] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, pages 10012–10022, 2021.

[30] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. *arXiv preprint arXiv:2312.00109*, 2023.

[31] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. *arXiv preprint arXiv:2308.09713*, 2023.

[32] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, pages 4460–4470, 2019.

[33] 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.

[34] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ToG*, 41(4):1–15, 2022.

[35] KL Navaneet, Kossar Pourahmadi Meibodi, Soroush Abbasi Koohpayegani, and Hamed Pirsiavash. Compact3d: Compressing gaussian splat radiance field models with vector quantization. *arXiv preprint arXiv:2311.18159*, 2023.

[36] 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*, pages 5480–5490, 2022.

[37] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *CVPR*, pages 11453–11464, 2021.

[38] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, pages 165–174, 2019.

[39] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *CVPR*, pages 10318–10327, 2021.

[40] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *ICCV*, pages 14335–14345, 2021.

[41] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. *NeurIPS*, 33:20154–20166, 2020.

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

[43] Vincent Sitzmann, Michael Zollhofer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *NeurIPS*, 32, 2019.

[44] Mohammed Suhail, Carlos Esteves, Leonid Sigal, and Ameesh Makadia. Generalizable patch-based neural rendering. In *ECCV*, pages 156–174. Springer, 2022.

[45] Stanislaw Szymanowicz, Christian Rupprecht, and Andrea Vedaldi. Splatter image: Ultra-fast single-view 3d reconstruction. *arXiv preprint arXiv:2312.13150*, 2023.

[46] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. In *CVPR*, pages 8248–8258, 2022.

[47] Fengrui Tian, Shaoyi Du, and Yueqi Duan. Mononerf: Learning a generalizable dynamic radiance field from monocular videos. In *ICCV*, pages 17903–17913, 2023.

[48] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *ICCV*, pages 12959–12970, 2021.

[49] Prune Truong, Marie-Julie Rakotosaona, Fabian Manhardt, and Federico Tombari. Sparf: Neural radiance fields from sparse and noisy poses. ieee. In *CVPR*, volume 1, 2023.

[50] Haithem Turki, Deva Ramanan, and Mahadev Satyanarayanan. Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs. In *CVPR*, pages 12922–12931, 2022.

[51] Haithem Turki, Jason Y Zhang, Francesco Ferroni, and Deva Ramanan. Suds: Scalable urban dynamic scenes. In *CVPR*, pages 12375–12385, 2023.

[52] Liao Wang, Jiakai Zhang, Xinhang Liu, Fuqiang Zhao, Yanshun Zhang, Yingliang Zhang, Minye Wu, Jingyi Yu, and Lan Xu. Fourier plenoctrees for dynamic radiance field rendering in real-time. In *CVPR*, pages 13524–13534, 2022.

[53] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. *arXiv preprint arXiv:2310.08528*, 2023.

[54] Jamie Wynn and Daniyar Turmukhambetov. Diffusionerf: Regularizing neural radiance fields with denoising diffusion models. In *CVPR*, pages 4180–4189, 2023.

[55] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *CVPR*, pages 9421–9431, 2021.

[56] Yuanbo Xiangli, Linning Xu, Xingang Pan, Nanxuan Zhao, Anyi Rao, Christian Theobalt, Bo Dai, and Dahua Lin. Bungeenerf: Progressive neural radiance field for extreme multi-scale scene rendering. In *ECCV*, pages 106–122. Springer, 2022.

[57] Tianyi Xie, Zeshun Zong, Yuxin Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. *arXiv preprint arXiv:2311.12198*, 2023.

[58] Haolin Xiong, Sairisheek Muttukuru, Rishi Upadhyay, Pradyumna Chari, and Achuta Kadambi. Sparsegs: Real-time 360 {\deg} sparse view synthesis using gaussian splatting. *arXiv preprint arXiv:2312.00206*, 2023.

[59] Haofei Xu, Anpei Chen, Yuedong Chen, Christos Sakaridis, Yulun Zhang, Marc Pollefeys, Andreas Geiger, and Fisher Yu. Murf: Multi-baseline radiance fields. *arXiv preprint arXiv:2312.04565*, 2023.

[60] Haofei Xu, Jing Zhang, Jianfei Cai, Hamid Rezatofighi, Fisher Yu, Dacheng Tao, and Andreas Geiger. Unifying flow, stereo and depth estimation. *TPAMI*, 2023.

[61] Linning Xu, Yuanbo Xiangli, Sida Peng, Xingang Pan, Nanxuan Zhao, Christian Theobalt, Bo Dai, and Dahua Lin. Grid-guided neural radiance fields for large urban scenes. In *CVPR*, pages 8296–8306, 2023.

[62] Zhiwen Yan, Weng Fei Low, Yu Chen, and Gim Hee Lee. Multi-scale 3d gaussian splatting for anti-aliased rendering. *arXiv preprint arXiv:2311.17089*, 2023.

[63] Zeyu Yang, Hongye Yang, Zijie Pan, Xiatian Zhu, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. *arXiv preprint arXiv:2310.10642*, 2023.

[64] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *NeurIPS*, 33:2492–2502, 2020.

[65] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *ICCV*, pages 5752–5761, 2021.

[66] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *CVPR*, pages 4578–4587, 2021.

[67] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, pages 586–595, 2018.

[68] Shunyuan Zheng, Boyao Zhou, Ruizhi Shao, Boning Liu, Shengping Zhang, Liqiang Nie, and Yebin Liu. Gps-gaussian: Generalizable pixel-wise 3d gaussian splatting for real-time human novel view synthesis. *arXiv preprint arXiv:2312.02155*, 2023.

[69] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*, 2018.

[70] Zehao Zhu, Zhiwen Fan, Yifan Jiang, and Zhangyang Wang. Fsgs: Real-time few-shot view synthesis using gaussian splatting. *arXiv preprint arXiv:2312.00451*, 2023.

A Gaussian Graph Network

In this section, we explain the definition of several functions in our method.

Unprojection ψ_{unproj} and projection ψ_{proj}. The camera parameter c_i includes the extrinsic matrix M_E, the intrinsic matrix $M_I \in \mathbb{R}^{3\times3}$ and camera origin $\mathbf{o}$. Assuming that $u_I \in \mathbb{R}^2$ is pixel coordinates from I_i and $d_{depth} \in \mathbb{R}$ is the estimated depth, the mean $\mu \in \mathbb{R}^3$ of pixel-aligned Gaussian is:

$$\mu = o + d_{depth}u_w, \quad [u_w, 1]^\top = M_E[u_c, 1]^\top, \quad [u_c, 1]^\top = M_I^{-1}[u, 1]^\top. \tag{15}$$

The projection function ψ_{proj} can be considered as the inverse process of unprojection, which projects 3D coordinates to pixel coordinates.

Overlap $\psi_{overlap}$. For view $v_i = (\mu_i, f_i)$ and view $v_j = (\mu_j, f_j)$, we project μ_i on view j and obtain M occupied pixels and $HW - M$ unoccupied pixels. We define $\psi_{overlap}(v_i, v_j) = \beta\frac{M}{HW}$, where β is a hyperparameter. The β is set to 0 at 0 iteration, 0.1 at 10,000 iterations and 0.2 at 50,000 iterations, in order to gradually enhance the influence of neighbor nodes.

B More Experiment Settings

B.1 Baselines

We choose NeRF-based methods [66, 44, 7, 59] and Gaussian-based methods [4, 6] as our baselines. We report performance of these methods according to the results in previous articles [6]. For new input settings, we employ their released checkpoints for comparison.

NeRF-based methods. pixelNeRF [66] is trained for 500,000 iterations with a batch size of 12. The hyperparameters are set to the same as previous experiments in the original paper. The near and far planes is set to be 0.1 and 10.0. GPNR [44] is trained for 250,000 iterations with a batch size of 4,098. The learning rate is set to 10^{-4}. AttnRend [7] is trained for 300,000 iterations with a batch size of 32. After 150,000 iterations, LPIPS loss is added to the total loss for further training.

Gaussian-based methods. pixelSplat [4] uses a pre-trained ResNet-50 and a ViT-B/8 vision transformer for image feature extraction. Is is trained for 300,000 steps using a batch size of 7 with an MSE loss for the first 150,000 iterations and supplement it with an LPIPS loss with weight 0.05 starting at 150,000 steps. pixelSplat predicts 3 Gaussians for each pixel and divides each Gaussian's opacity value by 3. MVSplat [6] initializes the image backbone with the UniMatch [60] pre-trained weight, and is trained for 300,000 iterations with a batch size of 14.

B.2 Training Details

The image backbone Φ_{image} consists of a ResNet [16] for per-image feature extraction and a Swin Transformer [29] for cross-view interaction. For Φ_{depth}, we adopt cost volume representations for depth estimation and a 2D UNet for depth refinement, following the instructions of MVSplat [6]. For Φ_{feat}, we employ a simple CNN network and upsampling operations.

Table 6: Details of network architecture.

Config	GGN
ResNet layers	[4, 4, 4]
ResNet channels	[32, 64, 128]
Transformer layers	[2, 2, 2, 2, 2, 2]
Transformer channels	[128, 128, 128, 128, 128, 128]
Cost volume UNet	[128, 128, 128]
Depth refine UNet	[64, 64, 64]

Table 7: Details of training settings.

Config	GGN
optimizer	Adam
scheduler	OneCycleLR
learning rate	2×10^{-4}
weight decay	10^{-4}
batch size	4
iterations	300,000

C Additional Results

We provide additional quantitative results and visualization results. As illustrated in Table 8 and Table 9, our GGN outperforms previous methods on all metrics with high FPS and fewer Gaussians.

Visualization results draw the same conclusion that our method generates efficient and generalizable Gaussian representations.

D Societal Impact

Our method focuses on generalizable novel view synthesis which can be used for application ranging from virtual reality to robotics. However, it can also have potential negative societal impact. Our method relies on large amounts of data and with large computational resources, which potentially has a negative impact on global climate change. What's more, accurate rendering of a scene may raise privacy concerns that need to be addressed carefully.

Table 8: Model performance and Gaussian numbers (K) of multi-view inputs on RealEstate10K [69]. † Models accept multi-view inputs, and only preserve Gaussians from two input views for rendering.

Views	Methods	PSNR↑	SSIM↑	LPIPS↓	Gaussians (K)	FPS↑
4 views	pixelSplat	20.19	0.742	0.224	786	110
	pixelSplat†	20.84	0.765	0.2217	393	175
	MVSplat	20.86	0.763	0.217	262	197
	MVSplat†	21.48	0.768	0.213	131	218
	Ours	24.76	0.784	0.172	102	227
8 views	pixelSplat	18.78	0.690	0.304	1572	64
	pixelSplat†	20.79	0.754	0.243	393	175
	MVSplat	19.69	0.768	0.238	524	133
	MVSplat†	21.39	0.766	0.215	131	218
	Ours	25.15	0.793	0.168	126	208
16 views	pixelSplat	17.80	0.647	0.320	3175	37
	pixelSplat†	20.75	0.754	0.245	393	175
	MVSplat	19.18	0.753	0.250	1049	83
	MVSplat†	21.34	0.765	0.215	131	218
	Ours	26.18	0.825	0.154	150	190

Table 9: Model performance and Gaussian numbers (K) of multi-view inputs on ACID [26]. † Models accept multi-view inputs, and only preserve Gaussians from two input views for rendering.

Views	Methods	PSNR↑	SSIM↑	LPIPS↓	Gaussians (K)	FPS↑
4 views	pixelSplat	20.15	0.704	0.278	786	110
	pixelSplat†	23.12	0.742	0.219	393	175
	MVSplat	20.30	0.739	0.246	262	197
	MVSplat†	23.78	0.742	0.221	131	218
	Ours	26.46	0.785	0.175	102	227
8 views	pixelSplat	18.84	0.692	0.304	1572	64
	pixelSplat†	23.07	0.738	0.232	393	175
	MVSplat	19.02	0.705	0.280	524	133
	MVSplat†	23.72	0.744	0.223	131	218
	Ours	26.94	0.793	0.170	126	208
16 views	pixelSplat	17.32	0.665	0.313	3175	37
	pixelSplat†	23.04	0.694	0.279	393	175
	MVSplat	17.64	0.672	0.313	1049	83
	MVSplat†	23.70	0.709	0.278	131	218
	Ours	27.69	0.814	0.162	150	190

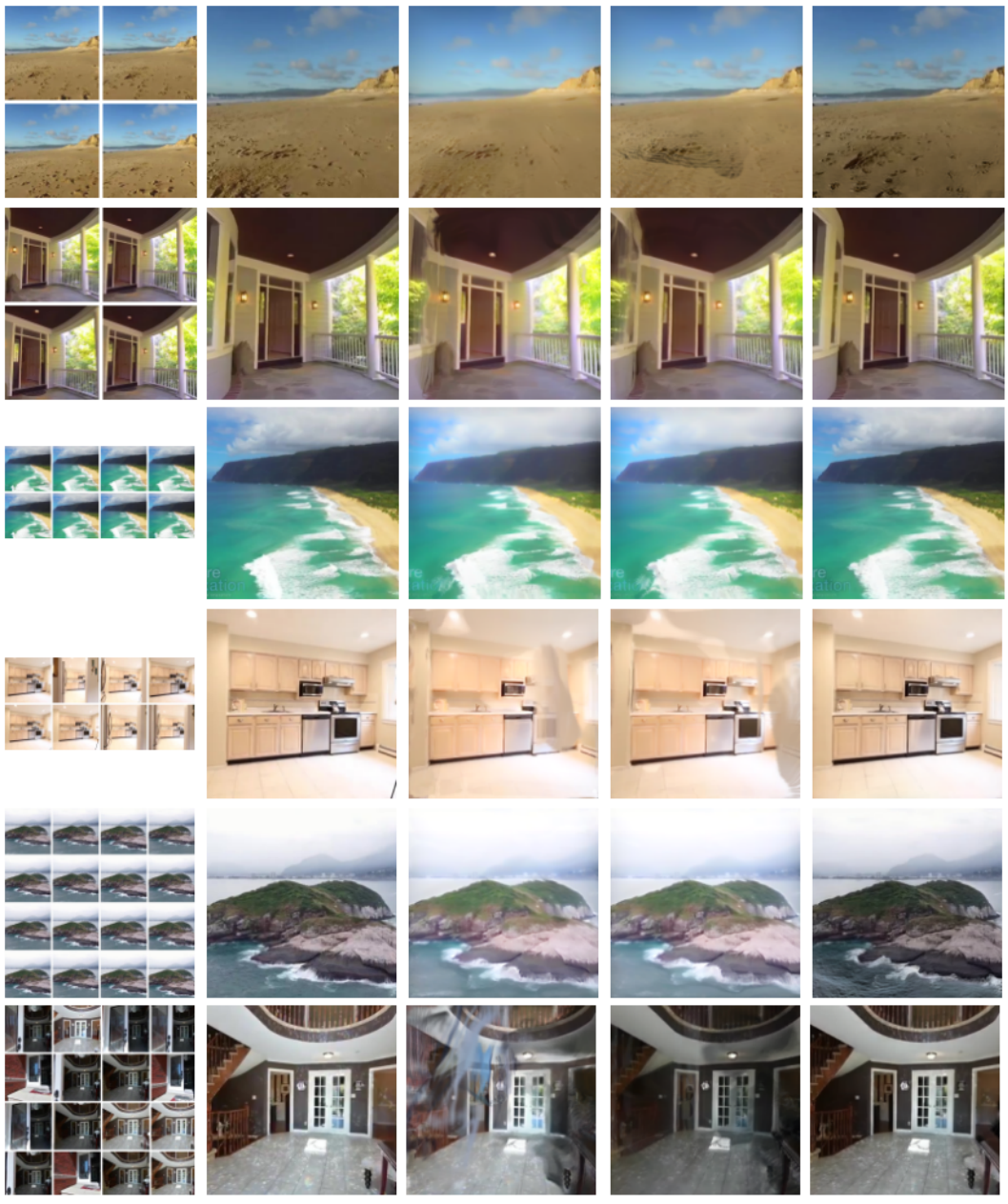


Figure 6: Additional visualization results on RealEstate10K [69] and ACID [26] benchmarks. We evaluate all models with 4, 8, 16 views as input and subsequently test on three target novel views.

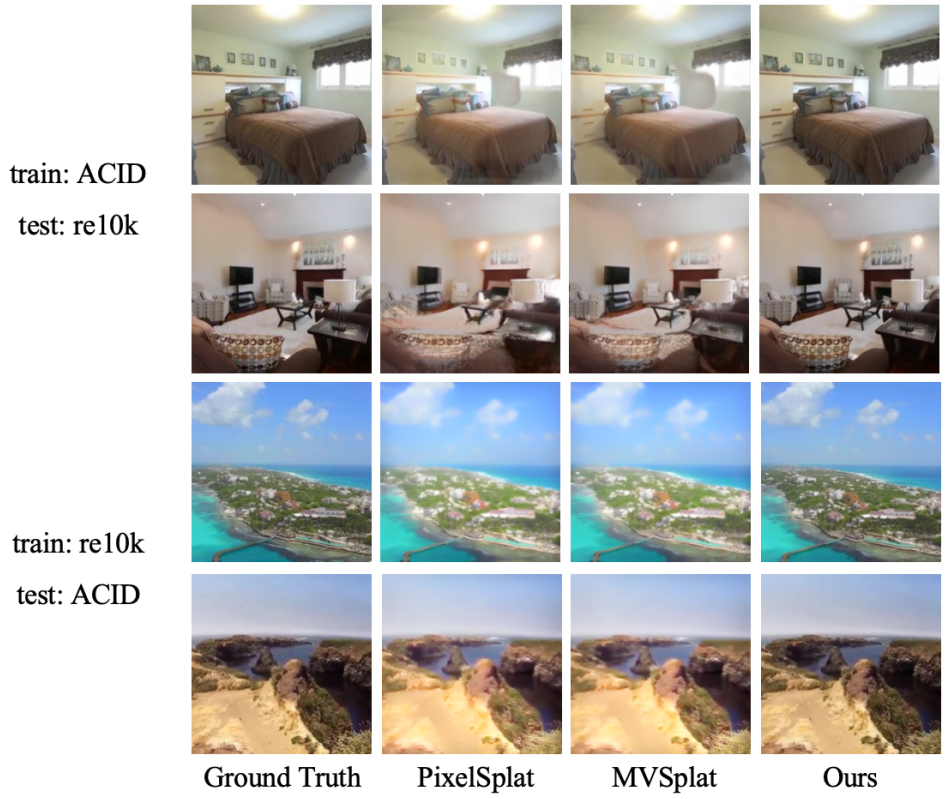


Figure 7: Additional visualization results of model performance for cross-dataset generalization on RealEstate10K [69] and ACID [26] benchmarks.

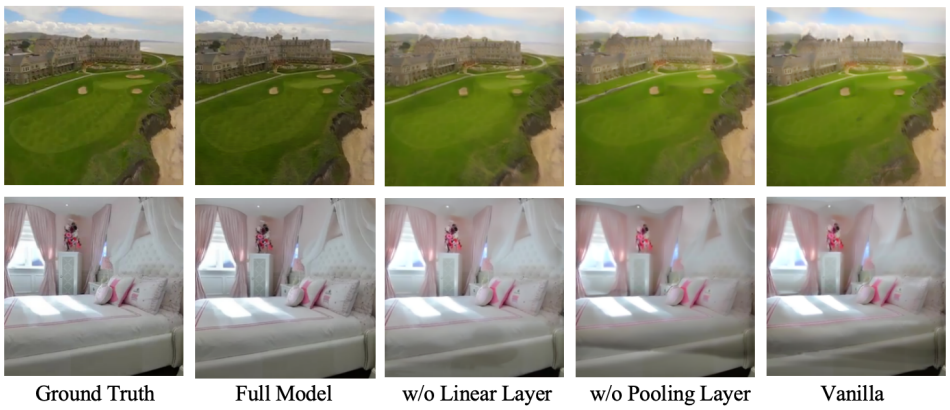


Figure 8: Visualization results of ablation study on RealEstate10K [69] and ACID [26] benchmarks.

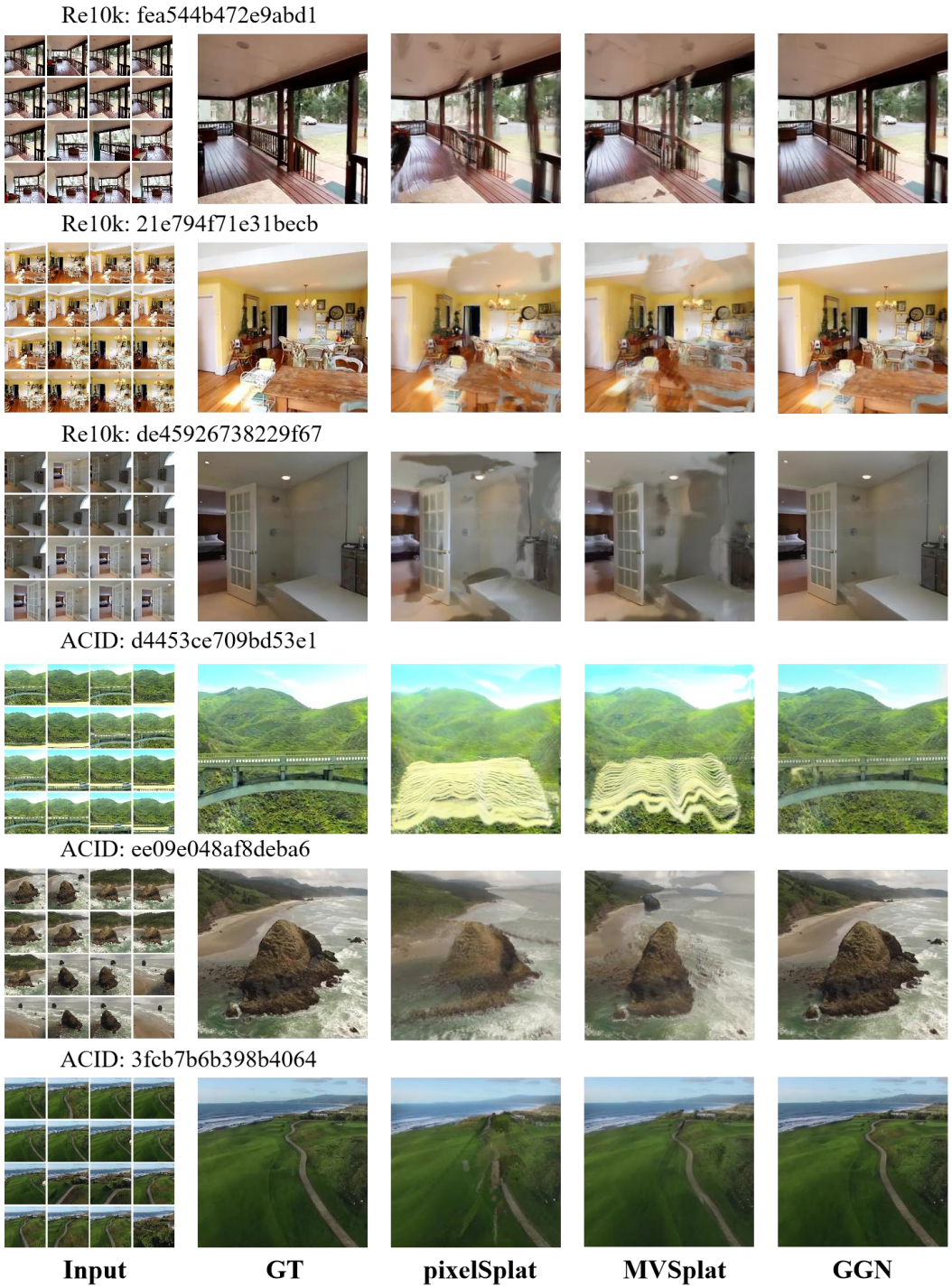


Figure 9: Visualization results of large-scale scenes from RealEstate10K and ACID benchmarks.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We provide claims made in the abstract and introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of our work in Section 5.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We provide assumptions and proofs both in Section 3 and appendix.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We propose a new network architecture and describe it clearly and fully in Section 3, Section 4 and appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We will release our codes when we prepare it well.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide implementation details in both Section 4 and appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We follow the experimental setting in previous studies, which do not include error bars in their experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report information on the computer resources in Section 4 and appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss potential societal impacts in appendix D.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [NA]

    Justification: Our paper does not use existing asserts.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: Our paper does not release new asserts.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.