Supplementary Material for Efficient View Synthesis with Neural Radiance Distribution Field

Yushuang Wu Xiao Li Jinglu Wang Xiaoguang Han Shuguang Cui Yan Lu

1. Implementation Details

Network Structure. Fig. S1 shows the detailed MLP network structure used for the proposed NeRDF. The input is a vector with 1135=16×63(points)+63(origin)+64(direction) channels, which is obtained by encoding the origin, direction, and 16 on-the-path points of a given camera ray. We employ the same positional encoding method from the original NeRF [2] for the xyz positions, including one-the-path points and the ray origin. The output is a 96-dim (24×4) vector for each ray, which corresponds to the 24 frequencies for RGBσ (4 channels), respectively. We add residual connection after each layer as in [3], and add a long skip connection to concatenate the output feature in the 4-th layer with the input as in [2]. The output layer is a linear layer.

Training Time. On one RTX3090 GPU, training a typical teacher NeRF takes 4h. A NeRDF takes another 8.5h (NeRDF-8) or 10.5h (NeRDF-48) to converge (400k iterations). Compared with R2L, the online training in NeRDF saves 15h for pseudo image generation.

Optimization. The optimization is based on reimplementing the 8-layer MLP in NeRDF using the PyTorch extension of tiny-cuda-nn for inference, which accelerates the forwarding for small networks by a large margin (as used in InstantNGP).

2. Additional Experiments

Video Results. We include video results (and comparisons) on the Real Forward Facing dataset in the supplemental material. Please refer to the video file "video.mp4" in our zip package.

Results on high resolution. The result on 1008×756 resolution for NeRDF is shown below: on a higher resolution, NeRF gets a 1.5 dB drop, while the performance of NeRDF only drops by 0.9 dB.

Breakdown results. Tab. S4 lists the breakdown results of the teacher NeRF [2], R2L [3], and our NeRDF on each scene of the Real Forward Facing dataset (LLFF) [2] as well as the average PSNR among all data. We also give the rendering speed (in FPS) and memory cost (in model size, MB)

Table S1. Performance of NeRDF-8 on different resolutions.

Method	Res. 504×378		Res. 1008×756	
	PSNR(dB)	FPS	PSNR(dB)	FPS
NeRDF-8	26.53	369.0	25.64	100.9
NeRF	27.74	1.45	26.26	0.39

Table S2. The synthesis quality comparison of using trigonometric functions and GMM to approximate the radiance distribution and the impact of the number of frequencies/components used.

TrigF: K	4	8	12	16	24
PSNR(dB)	25.63	25.64	**25.69**	25.66	25.67
GMM: K'	2	4	8	16	32
PSNR(dB)	24.88	25.57	**25.63**	**25.63**	25.60

for a more detailed comparison. The FPSs are obtained by testing on an RTX3090 GPU. We use R2L-N and NeRDF-N ($N \in \{8, 16, 32, 48\}$) to denote different versions of R2L and NeRDF with N-layer MLPs, respectively. As seen in the table, our NeRDF can significantly outperform R2L on all 8 data at the same level of rendering speed (*e.g.* NeRDF-8 v.s. R2L-16 or NeRDF-16 v.s. R2L-32). Besides, our NeRDF can achieve a high-quality synthesis (26.53dB on average) even with an 8-layer MLP.

Table S3. The PSNR(dB) of NeRDF when using different number of sampling points for neural rendering in training (Train-n) and inference (Infer-n), with $n \in \{16, 32, 64, 128\}$.

	Infer-16	Infer-32	Infer-64	Infer-128
Train-16	25.62	25.62	25.62	25.62
Train-32	25.65	25.66	25.66	25.66
Train-64	25.50	25.65	25.69	25.67
Train-128	24.96	25.65	25.70	25.70
Infer. time (ms)	1.2	1.6	2.5	3.8

Radiance distribution approximation. We use discrete Fourier transformation to approximate the radiance distribution with a group of trigonometric functions (TrigF). About the distribution approximator, an alternative to TrigF is using a Gaussian Mixture Model (GMM), where the output of NeRDF in Eq. (5) of the main paper is changed to the parameters of a GMM, *i.e.*, the mean, variance, and weight of

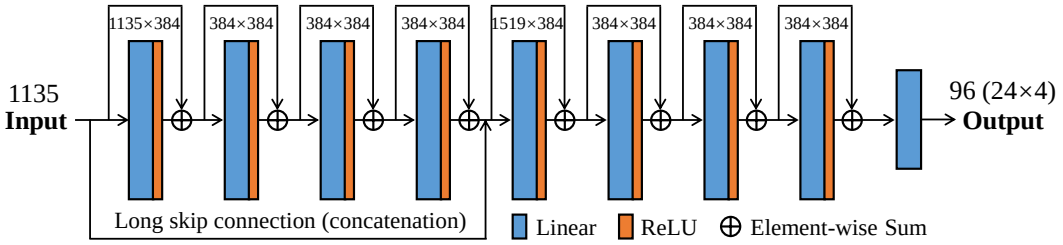

Figure S1. The network structure for NeRDF with 8 layers and 8 mix-of-gaussians modes.

Table S4. Per-scene breakdown results on the Real Forward Facing dataset of rendering an image of resolution 504×378.

Method	Memory ↓ (MB)	FPS ↑ (RTX 3090, PyTorch)	PSNR (dB)								
			Fern	Flower	Fortress	Horns	Leaves	Orchids	Room	T-Rex	Average
Teacher NeRF	3.8	0.17	26.81	28.44	31.68	29.62	22.38	21.37	33.13	28.61	27.75
R2L-8	2.2	33.67	22.61	22.01	25.85	22.02	18.15	17.30	26.37	22.06	22.05
R2L-16	4.6	20.12	24.57	25.03	28.19	24.73	20.26	19.10	29.15	24.33	24.42
R2L-32	8.6	11.40	25.79	26.59	29.87	26.11	21.42	20.52	30.96	25.68	25.87
R2L-48	13.0	7.89	26.28	26.98	30.43	26.77	21.88	21.01	31.50	26.17	26.37
NeRDF-8	5.1	21.18	25.69	27.85	30.66	26.98	21.68	20.90	31.76	26.72	26.53
NeRDF-16	9.6	11.88	25.93	28.18	31.01	27.44	22.03	21.09	31.98	26.96	26.82
NeRDF-32	19.0	6.32	25.81	28.20	30.93	27.48	22.06	21.10	32.01	26.98	26.82
NeRDF-48	28.0	4.34	26.15	28.34	31.37	28.13	22.15	21.40	32.43	27.57	27.19

each Gaussian component. We compare the synthesis quality of using these two kinds of approximators on the Fern data, when using different numbers of frequencies in TrigF (*i.e.*, K) and components in GMM (denoted as K'), with results shown in Tab. S2. As shown, using TrigF is better than GMM that brings a higher rendering quality, and is more robust for the choice of K. The best K and K' are 8 and 12, respectively, where the number of output channels in TrigF and GMM are equal, *i.e.,* 96 channels. Besides, we also investigate the effect of the number of sample points for volume rendering in Tab. S3, where we only give the time taken in the volume rendering stage in the last row of Tab. S3, implemented with Taichi [1] and test on an RTX3090 GPU. Considering the trade-off among training time, inference time, and synthesis quality, we choose 64 sample points for both training and inference.

Network width. The network width is 384 used in our NeRDF implementation. We conduct an additional ablation study on the network width. We provide the synthesis quality of NeRDF with an MLP of depth 8 and width $\in \{256, 320, 384, 448, 512\}$ on the Fern data and the corresponding rendering speed (in FPS) for a 504×378 image. As shown in Tab. S5, the choice of network width leads to a trade-off between the synthesis quality and the inference speed, and we choose 384 for a balanced performance. Note that even with the same network width as R2L-8, the synthesis quality of NeRDF-8 is still much better than R2L-8 (25.15dB v.s. 22.61dB). For the Tab. 5 in the main paper, experiments are conducted with network width 256. We further give the same group of experiment results with network width 384 to confirm the superiority of NeRDF. As demonstrated in Tab. S6, with width 384, the conclusion is consistent with using width 256: (i) each component in NeRDF is effective and (ii) learning the **radiance distribution** field is superior to get a more efficient representation than learning the light field.

Table S5. The synthesis quality and rendering speed of using different network width with an MLP with 8 layers.

Width	256	320	384	448	512
PSNR(dB)	25.15	25.43	25.69	25.80	25.97
FPS	31.06	24.27	21.18	16.61	14.66

Table S6. The synthesis quality (PSNR, dB) comparison between R2L (a) and different variants of NeRDF (b,c,d) on FERN and FLOWER, with network width 384.

Idx	Distribution	OVS	VDC	FERN	FLOWER
(a)				23.45	23.33
(b)	✓			25.58	24.85
(c)	✓	✓		25.65	27.74
(d)	✓	✓	✓	**25.69**	**27.85**

References

[1] Yuanming Hu, Tzu-Mao Li, Luke Anderson, Jonathan Ragan-Kelley, and Frédo Durand. Taichi: a language for high-performance computation on spatially sparse data structures. *TOG*, 2019. 2

[2] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthe-

sis. In *ECCV*, 2020. 1

[3] Huan Wang, Jian Ren, Zeng Huang, Kyle Olszewski, Menglei Chai, Yun Fu, and Sergey Tulyakov. R2l: Distilling neural radiance field to neural light field for efficient novel view synthesis. In *ECCV*, 2022. 1