Appendices

A. Additional Ablation Study on Augmentations

In the present study, we introduce a technique of random patch sampling designed to improve the training efficacy of the NeRF+SR pipeline. In addition to this, we extend the same framework to incorporate additional data augmentations conventionally employed in Convolutional Neural Networks (CNNs), such as random rotation or perspective transformation applied to the sampled patches. Our central hypothesis posits that these slight image transformations could potentially generate novel patterns absent from the training set but pertinent to the 3D spatial context. Through these augmentations, the SR module is hypothesized to further generalize, thereby facilitating the recovery of lost details in unobserved perspectives.

Mathematically, the NeRF+SR pipeline involves the sampling of a patch in the low-resolution (LR) ray space R^P_{LR} and its high-resolution (HR) counterpart R^P_{HR}. A transformation $\mathcal{A}$ is subsequently applied to these patches, yielding the transformed patches $R^{P'}_{\text{LR}}$ and $R^{P'}_{\text{HR}}$ as defined in Equation 5. These transformed patches are then integrated into Equation 3 for training the pipeline.

$$R^{P'}_{\text{LR}} = \mathcal{A}(R^P_{\text{LR}}), R^{P'}_{\text{HR}} = \mathcal{A}(R^P_{\text{HR}}) \tag{5}$$

To empirically validate our hypothesis, we implement two lightweight augmentations, random rotation and random horizontal flip, layered atop the random patch sampling technique. For synthetic datasets, the maximum rotation angle is set to 10 degrees, and the probability for a horizontal flip is set at 10%. Conversely, for the real-world scenes dataset LLFF, the maximum rotation angle is limited to 5 degrees, and the random horizontal flip is omitted since it's inappropriate with the forward-facing scenes.

The empirical results, presented in Table 5, reveal a marginal degradation in the output PSNR when using transformation-based augmentations as compared to utilizing random patch sampling exclusively. Consequently, random patch sampling remains as the most effective lightweight augmentation strategy for enhancing the NeRF+SR pipeline. We earmark the exploration of the effective utilization of transformation-based augmentations within the NeRF+SR pipeline for future research endeavors.

B. Additional Qualitative Results

We show additional qualitative results in Figure 4. Here we compare using different SR methods and different training procedures on the SR module. For the SR methods, we compare using bilinear interpolation and EDSR [27]. For different training procedures, we compare taking the pretrained EDSR from [4], finetuning the EDSR model with grid-based patch sampling and finetuning the EDSR model with random patch sampling

As discerned from Figure 4, reliance on bilinear interpolation culminates in outputs characterized by a lack of sharpness, rendering them blurry. In contrast, utilization of a pretrained SR module yields images of greater clarity, albeit with some loss of intricate details. Subsequent finetuning facilitates the recovery of nuanced patterns, such as shadows. Remarkably, the deployment of our proposed random patch sampling methodology further enhances performance, as evidenced by improvements in the Peak Signal-to-Noise Ratio (PSNR) when compared to traditional grid-based patch sampling.

Dataset	Data Aug	2x	4x	8x
NeRF-Synthetic	Grid-Patch	31.84	29.28	26.02
	Rand-Patch	**32.53**	**30.47**	**27.27**
	RP+RRot+Hflip	32.22	30.26	27.26
NSVF-Synthetic	Grid-Patch	34.34	30.45	26.26
	Rand-Patch	**35.39**	**32.04**	**27.93**
	RP+RRot+Hflip	35.03	31.78	27.79
LLFF	Grid-Patch	**26.2**	24.94	**21.68**
	Rand-Patch	26.04	**25.41**	21.3
	RP+RRot	25.94	25.37	21.29

Table 5. PSNR of using different augmentation techniques for SR rate 2x, 4x and 8x. The output resolution is 800x800 for NeRF-Synthetic and NSVF-Synthetic, and 1008x756 for LLFF. Grid-Patch stands for grid-based patch sampling and Rand-Patch stands for random patch sampling. RRot stands for random rotation and Hflip stands for random horizontal flip. The best results in each SR rate and dataset are highlighted in bold.

C. Additional Comparison with more NeRF models.

Beyond the data presented in Table 1, we extend our comparative analysis to encompass additional NeRF models specifically optimized for efficiency, incorporating both super-resolution (SR) based and non-SR-based approaches, as enumerated in Table 6. The empirical results delineated in Table 6 affirm that our proposed pipeline not only maintains high-quality output but also excels in terms of efficiency. This efficiency is observed across multiple metrics including training duration, rendering velocity, and model compactness, all achieved without necessitating specialized CUDA support on GPUs.

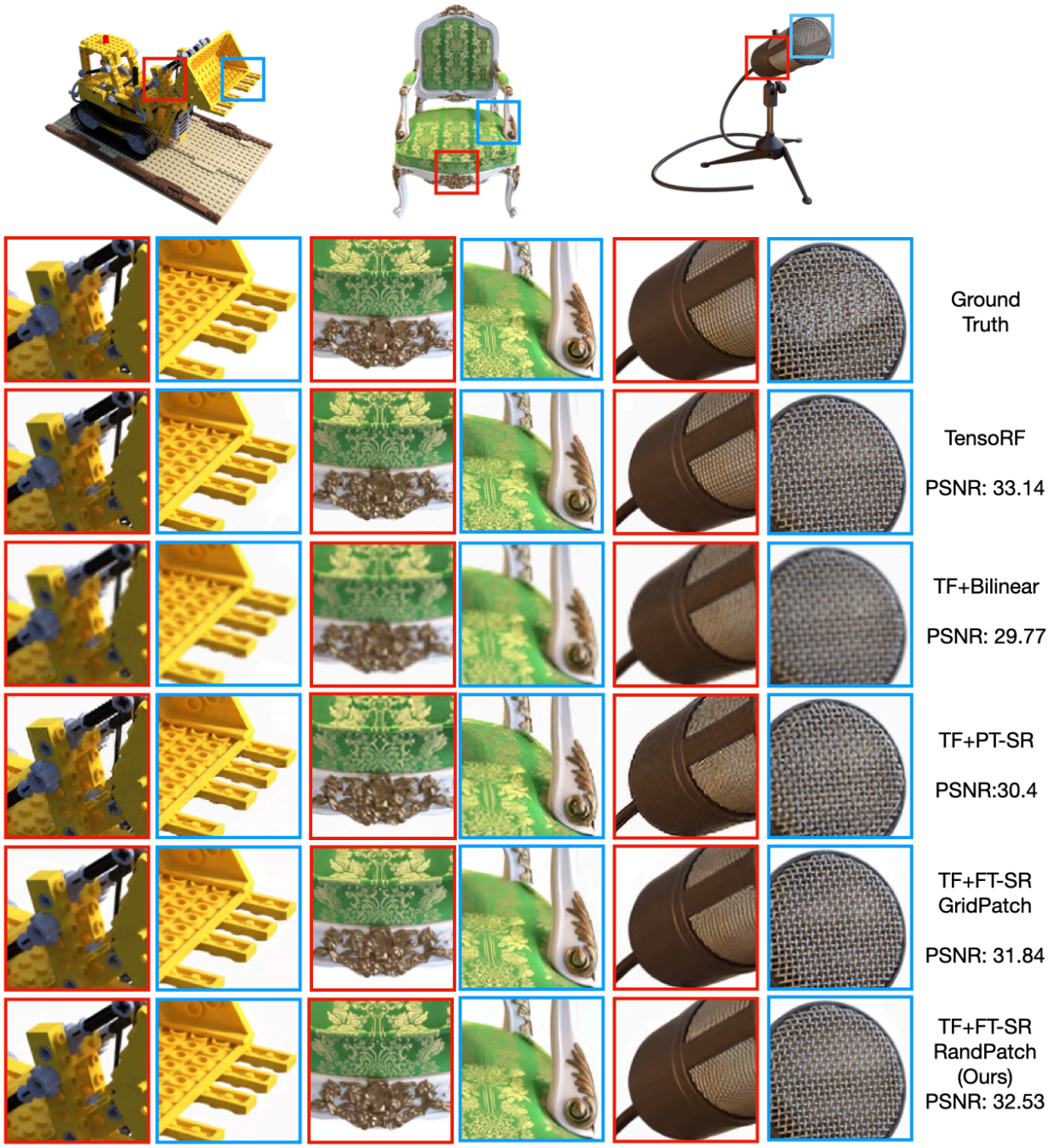

Figure 4. Qualitative results on lego, chair and mic scenes in NeRF-Synthetic. We show comparison on TensoRF at HR, and using bilinear, pretrained SR, finetuned SR with grid-based patch sampling and finetuned SR with random patch sampling to upsample output from TensoRF. The SR rate is $2\times$, and the SR module is EDSR [27]. We show the average PSNR on NeRF-Synthetic dataset of each method.

Method	PSNR↑	SSIM↑	LPIPS↓	Train Time	Render Time(s)	Model Size(MB)
NeRF [31]	31.01	0.947	0.081	∼35h	20	5
MipNeRF [8]	33.09	0.961	0.043	∼35h	-	-
NSVF‡ [29]	31.74	0.953	0.047	>48h	3	-
KiloNeRF† [34]	31.00	0.950	0.030	>35h	0.026	-
SNeRG [21]	30.38	0.950	0.05	∼35h	0.012	86.8
MobileNeRF‡† [13]	30.90	0.947	0.062	>35h	0.0013	125.8
Efficient-NeRF [22]	31.68	0.954	**0.028**	6h	0.004	∼3000
TensoRF [12]	33.14	0.963	0.049	18m	1.4	71.8
DVGO [37]	31.95	0.958	0.053	14m	0.44	612
FastNeRF [18]	29.90	0.937	0.056	-	0.041	>7000
Plenoxel† [17]	31.71	0.958	0.049	11m	0.066	815
Instant-NGP† [32]	**33.18**	-	-	5m	0.016	16
MobileR2L [10]	31.34	**0.993**	0.051	>35h	-	8.3
NeRF-SR [38]	28.46	0.921	0.076	>35h	5.6	-
FastSR-NeRF (2×)	32.53	0.961	0.052	1.5h	0.309	20

(a) NeRF Synthetic Dataset results.

Method	PSNR↑	SSIM↑	LPIPS↓	Train Time	Render Time(s)	Model Size(MB)
NeRF [31]	30.81	0.952	-	∼35h	∼20	∼5
NSVF‡ [29]	35.13	**0.979**	-	>48h	∼3	-
DVGO [37]	35.18	**0.979**	-	∼20m	-	∼600
TensoRF [12]	**36.52**	0.959	**0.027**	15m	1.4	74
FastSR-NeRF (2×)	35.39	**0.979**	0.032	1.5h	0.302	26

(b) NSVF Synthetic Dataset results.

Method	PSNR↑	SSIM↑	LPIPS↓	Train Time	Render Time(s)	Model Size(MB)
NeRF [31]	26.5	0.811	0.250	∼48h	33	5
SNeRG [21]	25.63	0.818	0.183	∼48h	0.036	310
NeX [42]	27.26	0.904	0.178	20h	0.0033	-
Efficient-NeRF [22]	**27.39**	0.912	**0.082**	4h	0.005	4300
TensoRF [12]	26.6	0.832	0.207	28m	5.9	188
MobileR2L [10]	26.15	**0.966**	0.187	>48h	-	8.3
NeRF-SR [38]	27.26	0.842	0.103	>48h	39.1	-
RefSR-NeRF [23]	26.23	0.874	0.243	-	8.5	38
FastSR-NeRF (2×)	26.20	0.822	0.241	2.5h	0.786	26

(c) LLFF Dataset results.

Table 6. Quality and efficiency results on NeRF-Synthetic, NSVF-Synthetic, and LLFF datasets. The tables are organized into three sections: implicit MLP-based NeRFs, efficient fully-explicit or hybrid NeRFs, and SR-based NeRFs, including our approach. Top performance in each quantitative metric is marked in bold, and the second best is underlined. Clear efficiency disadvantages are highlighted in red. Our tests run on a NVIDIA V100 GPU, while other results are their GPU results cited from respective papers when available. ‡ notes such method requires 8 high-end GPUs to train. † notes the method relies on customized CUDA kernels. Our method produces **excellent quantitative results that is on par or only slightly less than the state-of-the-art** cross all the benchmarks. Our method further achieves **great efficiency results across training time, rendering speed and model size** without the need of customized CUDA kernels support, which is favorable for inexpensive consumer-grade devices.