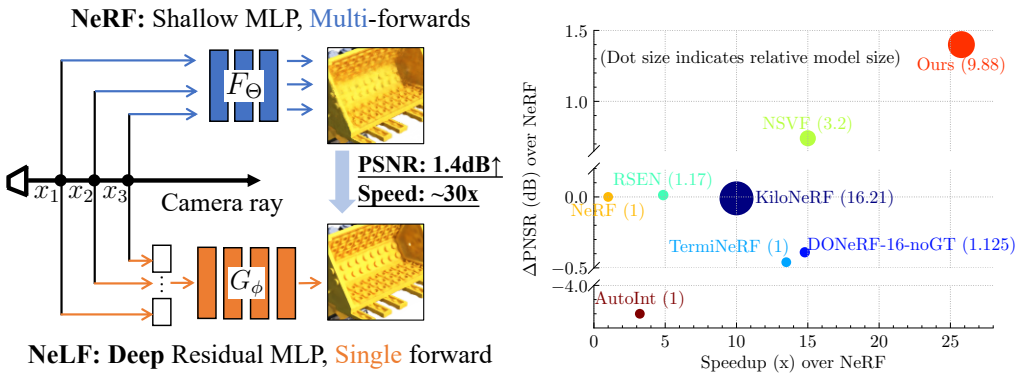
R2L: Distilling Neural *Radiance* Field to Neural *Light* Field for Efficient Novel View Synthesis

Huan Wang[1,†] **Jian Ren**[2] **Zeng Huang**[2] **Kyle Olszewski**[2]
Menglei Chai[2] **Yun Fu**[1] **Sergey Tulyakov**[2]
[1]Northeastern University [2]Snap Inc.
Project: https://snap-research.github.io/R2L



(a) NeRF *vs.* our NeLF method (b) Speedup-PSNR-Model Size comparison

Figure 1: (a) Our neural light field (NeLF, bottom) method improves the rendering quality by 1.40 PSNR over neural radiance field (NeRF, top) [32] on the NeRF synthetic dataset, while being around 30× faster. (b) Our method achieves a more favorable speedup-PSNR-model size tradeoff than other efficient novel view synthesis methods on the NeRF synthetic dataset. The number in the parentheses indicates the model size relative to the baseline NeRF model used in each paper (*best viewed in color*).

Abstract

Recent research explosion on Neural Radiance Field (NeRF) shows the encouraging potential to represent complex scenes with neural networks. One major drawback of NeRF is its prohibitive inference time: Rendering a single pixel requires querying the NeRF network hundreds of times. To resolve it, existing efforts mainly attempt to reduce the number of required sampled points. However, the problem of iterative sampling still exists. On the other hand, Neural *Light* Field (NeLF) presents a more straightforward representation over NeRF in novel view synthesis – the rendering of a pixel amounts to *one single forward pass* without ray-marching. In this work, we present a *deep residual MLP* network (88 layers) to effectively learn the light field. We show the key to successfully learning such a deep NeLF network is to have sufficient data, for which we transfer the knowledge from a pre-trained NeRF model via data distillation. Extensive experiments on both synthetic and real-world scenes show the merits of our method over other counterpart algorithms. On the synthetic scenes, we achieve 26 $\sim$ 35× FLOPs reduction (per camera ray) and 28 $\sim$ 31× runtime speedup, meanwhile delivering *significantly better* (1.4 $\sim$ 2.8 dB average PSNR improvement) rendering quality than NeRF without any customized implementation tricks.

[†]Preprint. This work was done when Huan was an intern at Snap Inc.

Table 1: Method comparison between our R2L approach and recent efficient novel view synthesis methods. Rendering speedup (measured by FLOPs reduction per ray and wall-time reduction) and representation (Repre.) size are relative to the original NeRF [32]. Repre. size measures the required storage of a neural network or cached files to represent a scene. ΔPSNR refers to the average PSNR improvement (on the NeRF synthetic dataset) over the baseline NeRF used in each paper. Note, ours and [4] are the only two neural *light* field methods here

Method	FLOPs speedup↑	Wall-time speedup↑	Repre. size↓	Extra design	ΔPSNR (dB)↑
NeRF [32]	1×	1×	1×	No	0
PlenOctrees [55]	-	3000×	∼ 600×	No	+0.02
DONeRF-8 [33]	27.60×	-	1.125×	Depth data	-0.14
KiloNeRF [41]	∼ 0.6×	692×	16.21×	Parallelism	-0.01
NSVF [28]	-	∼ 15×	∼ 3.2×	No	+0.74
AutoInt [27]	-	3.22×	∼ 1×	No	-4.2
TermiNeRF [39]	-	13.49×	∼ 1×	No	-0.46
RSEN [4]	-	4.86×*	1.17×	No	+0.013
Ours	26 ∼ 35×	28 ∼ 31×	4 ∼ 10×	No	+1.40

1 Introduction

Inferring the representation of a 3D scene from 2D observations is a fundamental problem in computer graphics and computer vision. Recent research innovations in implicit neural representations [10, 30, 34, 46] and differential neural renders [32] have remarkably advanced the solutions to this problem. Neural radiance field (NeRF) learned by a simple Multi-Layer Perceptron (MLP) network shows a great potential to store a complex scene into a compact neural network [32], thus has inspired plenty of follow-up works [6, 11, 26, 56].

Despite the success of NeRF and its extensions, the drawback is still apparent. The rendering time even for a single pixel is prolonged since the NeRF framework needs to aggregate the radiance of *hundreds of* sampled points via alpha-composition. It requires hundreds of network forwards, thus is prohibitively slow, especially on resource-constrained devices. One intuitive solution to the problem is to reduce the model size of NeRF MLP. However, apparent quality degradation of rendered images can be observed (*e.g.*, reducing the network width by only half causes around 0.01 SSIM [53] drop in [40]) while the reduction of inference time is only limited. Other research efforts focus on decreasing the number of sampled points [27, 33]. However, this does not fundamentally resolve the sampling issue. Some work [33] demands extra depth information for training, which is usually unavailable in most practical cases. Thus, a method that only requires *2D images* as input, represents the scene *compactly*, and enjoys a *fast* rendering speed with *high* image quality is highly desired. This paper aims to present such a method that can achieve all the four goals simultaneously by representing the scene as Neural *Light* Field (NeLF) instead of neural *radiance* field. In the neural light field, ray origin and direction are directly mapped into its associated RGB values, avoiding the need of sampling multiple points along the camera ray. Therefore, rendering a pixel requires only one single query, making it much faster than the radiance scene representation.

The idea of NeLF is attractive; however, realizing it for representing *complex real-world* scenes with better quality than NeRF is still challenging. Our first key technical innovation enabling this is a novel network architecture design for the neural light field network. Specially, we propose a deep (88 layers) residual MLP network with extensive residual MLP blocks. The *deep* network has much greater expressivity than the shallow counterparts, thus can represent the light field faithfully. Notably, since the debut of NeRF [32], its MLP-based network architecture is inherited with few substantial changes [6, 33, 40, 41]. To our best knowledge, this is the *first* attempt to address the NeRF rendering efficiency issue *from the network design perspective*. Although our network contains more parameters than the original NeRF, we only need *one* single network forward to render the color of a pixel, leading to much faster inference speed than NeRF.

The major technical problem is how to train the proposed deep residual MLP network. It is well-known in machine learning that large networks hunger for large sample sizes to curb overfitting [23, 49]. We can barely train such a large network using only the original 2D images (which are typically less than 100 in real-world applications). To tackle this problem, as the second key technical innovation of this paper, we propose to distill the knowledge [8, 18] from a *pretrained*

NeRF model to our network, by rendering pseudo data from random views using the pre-trained NeRF model. We name our method as **R2L** since we show distilling neural **R**adiance filed **to** neural **L**ight filed is an effective way to obtain a powerful NeLF network for efficient novel view synthesis. Empirically, we evaluate our method on both synthetic and real-world datasets. On the synthetic scenes, we achieve $26 \sim 35\times$ FLOPs reduction ($28 \sim 31\times$ wall-time speedup) over the original NeRF with significantly *higher* rendering quality. Comparison between ours and other efficient novel view synthesis approaches is summarized in Tab. 1.

Overall, our contributions can be summarized into the following aspects:

- Methodologically, we present a brand-new deep residual MLP network aiming for compact neural representation, fast rendering, without extra demand besides 2D images, for efficient novel view synthesis. This is the *first* attempt to improve the rendering efficiency via network architecture optimization.
- Our network represents complex real-world scenes as neural light fields. To resolve the data shortage problem when training the proposed deep MLP network, we propose an effective training strategy by distilling knowledge from a pre-trained NeRF model, which is the key to enabling our method.
- Practically, our approach achieves $26 \sim 35\times$ FLOPs reduction ($28 \sim 31\times$ wall-time speedup) over the original NeRF with even better visual quality, which also performs favorably against existing counterpart approaches.

2 Related Work

Efficient neural scene representation and rendering. Since the debut of NeRF [33], many follow-up works have been attempting to improve its efficiency. One major direction is to skip the empty space and sample more wisely along a camera ray. NSVF [28] defines a set of voxel-bounded implicit fields organized in a sparse voxel octree structure, which enables skipping empty space in novel view synthesis. It achieves 10 times faster than NeRF at inference time with improved quality. AutoInt [27] improves the rendering efficiency by reducing the number of evaluations along a ray through learned partial integrals. DeRF [40] spatially decomposes the scene into Voronoi diagrams, each learned by a small network. They achieve 3 times rendering speedup over NeRF with similar quality. Similarly, KiloNeRF [41] also spatially decomposes the scene, but into thousands of *regular* grids. Each of them is tackled by a tiny MLP network. Their work is similar to ours as a pre-trained NeRF model is also used to generate pseudo targets for training. Differently, they generate both the density and color as training targets, which makes their method still belong to the neural *radiance* field; while our model only regresses the color, as a neural *light* field network. Besides, their efficiency comes from the shrinkage of model size (thousands of *tiny* MLPs) while ours comes from the fundamentally saving of sampling. DONeRF [33] is proposed recently to reduce sampling through a depth oracle network learned with the ground-truth depth as supervision. It decimates the sampled points from hundreds (*i.e.*, 256 in the original NeRF) to only 4 to 16 while maintaining comparable or even better quality. However, the depth oracle network is learned with *ground-truth depth* as the target, which is typically unavailable in practice. Our method does not demand it, akin to the original NeRF [32].

Another direction for faster NeRF rendering is to pre-compute and cache the representations per the idea of trading memory for computational efficiency. In this line, FastNeRF [12] employs a factorized architecture to independently cache the position-dependent and ray direction-dependent outputs and achieves 3000 times faster than the original NeRF at rendering. Baking [15] precomputes and stores NeRF as a new representation (Sparse Neural Radiance Grid) that enables real-time rendering on commodity hardware. We consider these as an *orthogonal* direction to our work since they trade memory for speed while our method is to achieve faster rendering meanwhile keeping the *lightweight* scene representation as the original NeRF.

Neural light field (NeLF). Light fields enjoy a long history as a scene representation in computer vision and graphics [1, 2]. Levoy *et al*. [25] and Gortler *et al*. [13] introduced light fields in computer graphics as 4D scene representation for fast image-based rendering. With them, novel view synthesis can be realized by simply extracting 2D slices in the 4D light field, yet with two major drawbacks. First, they tend to cause considerable storage costs. Second, it is hard to achieve a full 360° representation without concatenating multiple light fields. In the era of deep learning, neural

light fields based on convolutional networks have been proposed [7, 22, 31]. One recent neural light field paper is Sitzmann *et al.* [44]. They employ Plücker coordinates to parameterize 360° light fields. In order to ensure multi-view consistency, they propose to learn a prior over the 4D light fields in a meta-learning framework. Despite intriguing ideas, their method is only evaluated on toy datasets, not as comparable to NeRF [32] in representing complex real-world scenes. Another recent NeLF work is RSEN [4]. To tackle the insufficient training data issue, they propose to learn a voxel grid of *local* light fields, which are much simpler to learn than the global light field. In their experiments, they also include a pre-trained NeRF teacher for regularization.

Our neural light field network is different from these in that, **(1)** methodologically, we propose a *deep* residual MLP (88 layers) to learn the light field, while these NeLF works still employ the NeRF-like shallow MLP networks (*e.g.*, 6 layers in [44], 8 layers in [4]); **(2)** we propose to leverage a NeRF model to synthesize extra data for training, making our method a bridge from neural radiance field to light field; **(3)** thanks to the abundant capacity, our R2L network can achieve better rendering quality (*e.g.*, our method can represent complex real-world scenes against [44]), or can achieve better efficiency when maintaining the rendering quality (*e.g.*, [4] achieves merely around $5\times$ speedup *vs.* $30\times$ speedup of ours over the baseline NeRF method).

Knowledge distillation (KD). The general idea of knowledge distillation is to guide the training of a student model through a larger pre-trained teacher model. Pioneered by Buciluǎ *et al.* [8] and later refined by Hinton *et al.* [18] for image classification, knowledge distillation has seen extensive application in vision and language tasks [9, 20, 51, 52]. Many variants have been proposed regarding the central question in knowledge distillation – how to define the *knowledge* that is supposed to be transferred from the teacher to the student, examples including output distance [5, 18], internal feature distance [42], feature map attention [57], feature distribution [36], activation boundary [17], inter-sample distance relationship [29, 35, 38, 48], and mutual information [47]. The distillation method in this work is to regress the output of the NeRF model with extra data labeled by the teacher (akin to [5, 8]), which is the most straightforward way of distillation for the numerical target. Yet we will show this simple scheme can work powerfully to train a deep neural light field network.

3 Methodology

3.1 Background: Neural Radiance Field (NeRF)

In the neural radiance field pioneered by Mildenhall *et al.* [32], the 3D scene is implicitly represented by an MLP network, which learns to map the 5D coordinate (spatial location (x, y, z) and viewing direction (θ, ϕ)) to the 1D volume density and 3D view-dependent emitted radiance at that spatial location,

$$F_\Theta : \mathbb{R}^5 \mapsto \mathbb{R}^4, \tag{1}$$

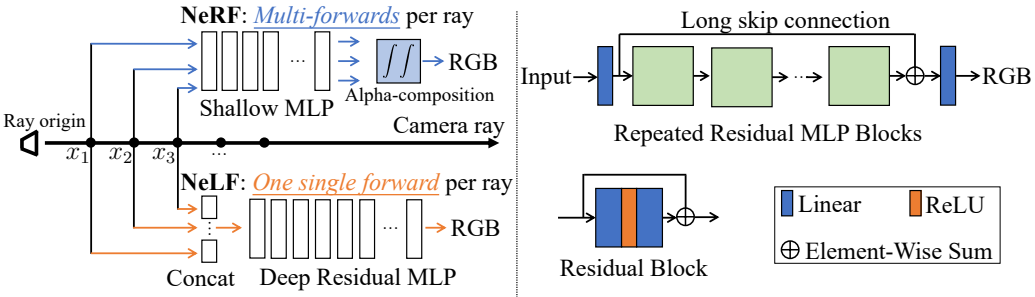
where F refers to an MLP neural network (parameterized by Θ) to represent a scene. For rendering, the classic volume rendering technique [21] is adopted in NeRF to obtain the desired color for an oriented ray. Volume rendering is differential thus making NeRF end-to-end trainable simply using the captured 2D images as supervision. For novel view synthesis, given an oriented ray, NeRF first samples several locations along the camera ray, predicts their emitted radiance by querying the MLP network F_Θ, and then aggregates the radiance together by alpha composition to output the final color. As sampling at vacuum points contributes nothing to the final color, a sufficient number of sampled points is critical to NeRF's performance so as to cover the worthy locations (such as those near the object surface). However, increased sampling incurs linearly increased query cost of the MLP network.

3.2 R2L: Distilling NeRF to NeLF

On the other hand, a scene can also be represented as a *light* field instead of *radiance* field, parameterized by a neural network. The network G_ϕ learns a mapping function directly from a 5D oriented ray to its target 3D RGB,

$$G_\phi : \mathbb{R}^5 \mapsto \mathbb{R}^3. \tag{2}$$

NeLF has several attractive advantages over NeRF. **(1)** Methodologically, it is more straightforward for the task of novel view synthesis, in that the output of the NeLF network is already the wanted color, while the output of a NeRF network is the radiance of a sampled point; the desired color has to



(a) NeRF (top) *vs.* NeLF (ours, bottom) **(b)** Detailed architecture of our NeLF network

Figure 2: (a) Comparison between our proposed NeLF network (*Deep Residual MLP*, bottom) and NeRF network (*Shallow MLP*, top). (b) Detailed architecture of the proposed *deep* light field network, which employs extensive repeated residual MLP blocks.

been obtained through an extra step of ray marching (see Fig. 2(a)). **(2)** Practically, given the same input ray (origin coordinate and direction), rendering in a light field simply amounts to a *single query* of the light field function. It *fundamentally* obviates the need for point sampling along a ray (which is the speed bottleneck in NeRF [32]), thus can be orders-of-magnitude faster than NeRF. Despite these intriguing properties, not many successful attempts have crystallized NeLF *with comparable quality to NeRF* up to date. To our best knowledge, only one recent NeLF method [4] achieves comparable quality to NeRF, but its speedup is relatively limited (around 5×wall-time speedup). In this paper, we propose a novel network architecture to make NeLF as effective as NeRF (meanwhile being much faster). Intuitively, the light field is *harder* to learn than radiance field – radiance at neighbor space locations does not change dramatically given the radiance field in the physical world is typically continuous; while two neighbor rays can point to starkly different colors because of occlusion. That is, the light field is intrinsically *less smooth* (sharply changing) than the radiance field. To capture the inherently more complex light field, we need a more *powerful* network. Per this idea, the 11-layer MLP network used in NeRF can hardly represent a complex light field by our empirical observation (see Tab. 5). We thereby propose to employ a *deep* MLP network to parameterize the above G function. Then, the foremost technical question is how to design the deep network.

Network design. Different from the NeRF network, we propose to employ intensive residual blocks [14] in our network. The resulted network architecture is illustrated in Fig. 2(b). Residual connections were shown critical to enable the much greater network depth in [14], which also applies here for learning the light field. The merit of having a *deeper* network will be justified in our experiments (see Fig. 6(b)). We also study an underperformance case in the Appendix when the residual connections are *not* used in a deep MLP network.

Notably, enabling a deep network for neural radiance/light field parameterization is *non-trivial*. Noted by DeRF [40], "*there are diminishing returns in employing larger (deeper and/or wider) networks*". As a result, notably, most NeRF follow-up works for improving rendering efficiency (*e.g.*, [40, 41, 33]) actually inherit the MLP architecture in NeRF with *few* substantial innovations. To our best knowledge, we are the *first* to address the efficiency issue of NeRF *through the network architecture optimization perspective*. Despite the residual structure is not new itself (due to ResNets [14]), its necessity and potential have not been fully recognized and exploited in the NVS task. Our paper is meant to make a step forward in this direction.

3.3 Synthesize Pseudo Data

Deep networks hunger for excessive data to be powerful. Unfortunately, this is not the case in novel view synthesis, where a user typically captures fewer than 100 images. To overcome this problem, we propose to employ a pre-trained NeRF model to synthesize extra data for training. This makes our method a bridge from neural *radiance* field to neural *light* field.

We need to decide where to sample to synthesize the pseudo data to avoid unnecessary waste. Specifically, with the original training data (images and their associated camera poses), we know

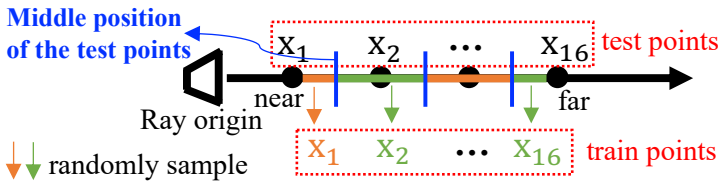


Figure 3: Illustration of the point sampling in training and testing of our method

the bounding box of the camera locations and their orientations. Then we *randomly* sample the ray origins (x_o, y_o, z_o) and normalized directions (x_d, y_d, z_d) obeying a uniform distribution U *within the bounding box* to make a 6D input as follows,

$$\begin{aligned} x_o &\sim U(x_o^{\min}, x_o^{\max}),\ y_o \sim U(y_o^{\min}, y_o^{\max}),\ z_o \sim U(z_o^{\min}, z_o^{\max}), \\ x_d &\sim U(x_d^{\min}, x_d^{\max}),\ y_d \sim U(y_d^{\min}, y_d^{\max}),\ z_d \sim U(z_d^{\min}, z_d^{\max}), \end{aligned} \tag{3}$$

where the viewing bounding box can be inferred from the training data. An example illustration of the pseudo data origins and directions in our method is shown in our Appendix. Note, since we can control the generated data, we explicitly demand the pseudo data completely cover the original training data, implying they are in the same domain, which is critical to the performance.

For a trained NeRF model F_{Θ^*}, the target RGB value can be queried as:

$$(\hat{r}, \hat{g}, \hat{b}) = F_{\Theta^*}(x_o, y_o, z_o, x_d, y_d, z_d), \tag{4}$$

where Θ^* stands for the converged model parameters. Then a slice of training data is simply a vector of these 9 numbers: $(x_o, y_o, z_o, x_d, y_d, z_d, \hat{r}, \hat{g}, \hat{b})$. To have an effective neural light field network F_Θ, we feed abundant pseudo data into the proposed deep R2L network and train it by the MSE loss function,

$$\mathcal{L} = \text{MSE}(G_\phi(x_o, y_o, z_o, x_d, y_d, z_d), (\hat{r}, \hat{g}, \hat{b})). \tag{5}$$

3.4 Ray Representation and Point Sampling

It is critical to have a proper representation of a ray in NeLF, *e.g.*, in [44], they use Plücker coordinates to parameterize 360° light fields. In this work, we use a *much simpler* way of representation – we simply concatenate the spatial coordinates of K sampled points along a ray to form an input vector ($3K$-d), fed into the NeLF network. Mathematically, we need at least two points to define a ray. More points will make the representation more precise. In this paper, we choose $K = 16$ points (based on our empirical study) along a ray. A critical design here is that we expect the network not to overfit the K points but to capture the underlying ray information. Thus, during training the K points are *randomly* sampled along the ray using the stratified sampling (same as NeRF [32], see Fig. 3). This design is critical to generalization (in our Appendix, we will show the performance drops significantly if the K points are *fixed* during training). During testing, the K points are evenly spaced.

3.5 Training with Hard Examples

Given that we randomly sample the camera locations and orientations, the rays are likely to point to the trivial parts of a scene (*e.g.*, the white background of a synthetic scene). Also, during training, some easy-to-regress colors will be well-learned early. Feeding these pixels again to the network barely increases its knowledge. We thus propose to tap into the idea of hard examples [16, 43]. That is, we want the network to pay more attention to the rays that are harder to regress (typically corresponding to the high-frequency details) during learning.

Specially, we maintain a *hard example pool*. A *harder* example is defined by a *larger* loss (Eq. (5)). In each iteration, we sort the losses for each sample in a batch in ascending order and add the top r (a pre-defined percentage constant) into the hard example pool. Meanwhile, in each iteration, the same amount r of hard examples are randomly picked out of the pool to augment the training batch. This design can accelerate the network convergence significantly as we will show in the experiments (see Fig. 6).

Table 2: PSNR↑ and SSIM↑ on the NeRF synthetic dataset (Realistic Synthetic 360°) and real-world dataset (Real Forward-Facing). Training with pseudo and real data (ours-2) gives us better results. Our R2L network here is W256D88. †KiloNeRF adopts Empty Space Skipping and Early Ray Termination, so the FLOPs is scene-by-scene. We estimate the average FLOPs based on the description in the paper. The best results are in red, second best in blue

Method	Storage (MB)	FLOPs (M)	Synthetic		Real-world	
			PSNR↑	SSIM↑	PSNR↑	SSIM↑
Teacher NeRF [32]	2.4	303.82	30.47	0.9925	27.68	0.9725
Ours-1 (Pseudo data)	23.7	11.79	30.48 (+0.01)	0.9939	27.58 (-0.10)	0.9722
Ours-2 (Pseudo + real data)	23.7	11.79	31.87 **(+1.40)**	0.9950	27.79 **(+0.11)**	0.9729
Teacher NeRF in [41]	2.4	303.82	31.01	0.95	-	-
KiloNeRF [41]	38.9	∼500†	31.00 (-0.01)	0.95	-	-
Teacher NeRF in [4]	4.6	∼ 300	-	-	27.928	0.9160
RSEN [4]	5.4	∼ 60	-	-	27.941 (+0.013)	0.9161

3.6 Implementation Details

Our R2L method can lead to different networks under different FLOPs budgets. In this paper, we mainly have two: 6M and 12M FLOPs (per ray). They result in a bunch of networks: 12M: W256D88, 6M: W181D88, W256D44, W363D22 (W stands for width, D for depth). Obviously, a larger network is expected to deliver better performance, so W256D88 is used for obtaining better quality; ablation studies will be conducted on the 6M-budget networks since they are faster to train. Following NeRF [32], positional encoding [50] is also used to enrich the input information.

4 Experiments

Datasets. We show experiments on the following datasets:

- **NeRF datasets** [32]. We evaluate our method on two datasets: synthetic dataset (Realistic Synthetic 360°) and real-world dataset (Real Forward-Facing). Realistic Synthetic 360° contains path-traced images of 8 objects that exhibit complicated geometry and realistic non-Lambertian materials. 100 views of each scene are used for training and 200 for testing, with a spatial resolution of 800×800. Real Forward-Facing also contains 8 scenes, captured with a handheld cellphone. There are 20 to 62 images for each scene with $1/8$ held out for testing. All images have a resolution of 1008×756.
- **DONeRF dataset** includes their synthetic data. Images are rendered using Blender and their Cycles path tracer to render 300 images for each scene, which are split into train/validation/test sets at a 70%, 10%, 20% ratio.

Training settings. All images in the synthetic dataset are down-sampled by $2\times$ during training and testing. The original NeRF model is trained with a batch size of $1,024$ and initial learning rate as 5×10^{-4} (decayed during training) for $200k$ iterations. We synthesize $10k$ images using the pre-trained NeRF model. Our proposed R2L model is trained for $600k$ iterations with the same learning rate schedule. The rays in a batch are randomly sampled from different images so that they do not share the same origin. We empirically observe that the batch diversity is important to achieve superior performance. Adam optimizer [24] is employed for all training. We use PyTorch 1.9 [37] to implement our method, referring to the widely-used NeRF PyTorch code*. Experiments are conducted with 8 NVIDIA V100 GPUs. *Our code and trained models will be released*†.

Comparison methods. We compare with with the original NeRF [32] to show that we can achieve significantly better rendering quality while being much faster. Meanwhile, we also compare with DONeRF [33], NSVF [28], and NeX [54] since they also target efficient NVS as we do. Other efficient NVS works such as AutoInt [27] and X-Fields [7] have been shown less favorable than RSEN [4]. Therefore, we only compare with RSEN [4]. KiloNeRF [41], another closely related

*https://github.com/yenchenlin/nerf-pytorch
†https://github.com/snap-research/R2L

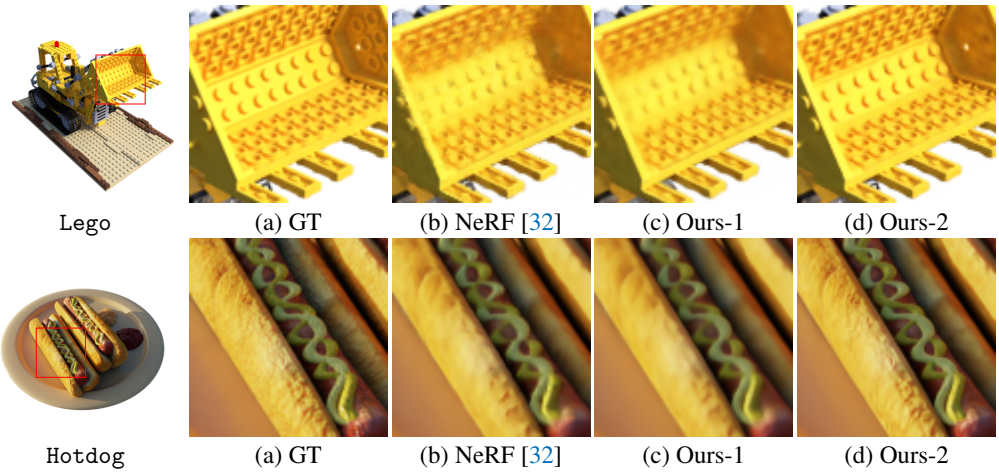


Figure 4: Visual comparison between our R2L network (W256D88) and NeRF on the synthetic scene `Lego` and `Hotdog`. Ours-1 is trained sorely on pseudo data, ours-2 on pseudo + real data. Please refer to our Appendix for the visual comparison on the real-world dataset

Table 3: PSNR↑ and FLIP↓ comparison on the DONeRF synthetic dataset. All the PSNR and FLIP results except ours and NeRF are directly cited from the DONeRF paper since we are using exactly the same dataset here. Training with pseudo and real data (ours-2) gives us better results. The best results are in red, second best in blue

Method	Storage (MB)	FLOPs (M)	PSNR↑	FLIP↓
Teacher NeRF (log+warp)	3.2	211.42	32.67	0.070
NSVF-large [28]	8.3	187.52	30.01 (-2.66)	0.078
NeX-MLP [54]	89.0	42.71	30.55 (-2.12)	0.076
DONeRF-16-noGT [33]	3.6	14.29	32.25 (-0.42)	0.065
DoNeRF-8 [33]	3.6	7.66	32.50 (-0.17)	0.064
Ours-1 (Pseduo data)	12.1	6.00	32.67 (+0.00)	0.071
Ours-2 (Pseduo + real data)	12.1	6.00	35.45 **(+2.78)**	0.047

work apart from RSEN, will also be compared to. Similar to [4], we do not compare to baking-based methods [15, 55, 12]) as they trade memory footprint for speed while our method aims to maintain the compact representation.

4.1 NeRF Synthetic and Real-World Dataset

The quantitative comparisons (PSNR, SSIM [53]) on the NeRF synthetic and real-world dataset are presented in Tab. 2. Visual comparison is shown in Fig. 4. **(1)** Using the pseudo data alone, our R2L network achieves comparable performance to the original ray-marching NeRF model either quantitatively or qualitatively, with only $1/26$ FLOPs. The blurry parts of NeRF results usually also appear on our results, since our model learns from the data generated by the NeRF teacher model. **(2)** With the original data included for training, our R2L network *significantly* improves the test PSNR **by** 1.40 over the teacher NeRF model. This means that the performance of our method is *not* upper-bounded by the teacher model. **(3)** For the two related works KiloNeRF and RSEN, their baseline NeRF models have different PSNRs due to various different settings (*e.g.*, KiloNeRF tests on 800×800 images while ours on 400×400 images), so the PSNR results cannot be directly compared. Instead, we compare the *PSNR change* over the baseline NeRFs. KiloNeRF achieves 0.01 dB PSNR drop *vs.* ours 1.40 dB PSNR boost. RSEN improves the PSNR on the much more challenging real-world dataset marginally (by 0.013 dB). In comparison, our improvement is more significant (0.11 dB), and with much fewer FLOPs.

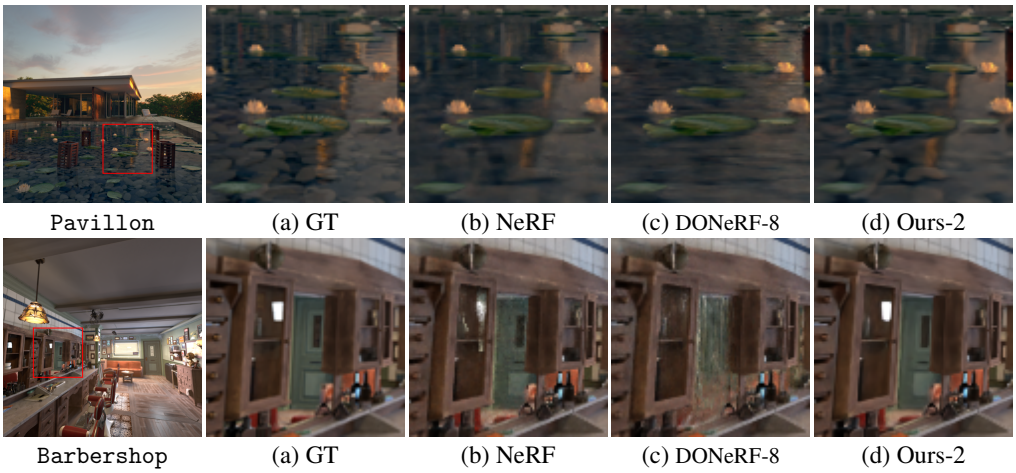


Figure 5: Visual comparison of ours, NeRF [32], DONeRF [33] on the DONeRF dataset

Table 4: Average time (s) comparison among our R2L network (W181D88), DONeRF, and NeRF. The benchmark is conducted on the platforms of two NVIDIA GPU and one Intel CPU under the *same* hardware and software. The speedup of ours and DONeRF is relative to the running time of NeRF. Results are averaged by 60 frames

Method	FLOPs (M)	GeForce 2080Ti	Tesla V100	CPU
NeRF	211.42	5.9343	4.9902	142.2612
DONeRF-16	14.29 (14.79×)	0.4162 (14.26×)	0.3524 (14.16×)	9.9344 (14.32×)
Ours	**6.00 (35.24×)**	**0.2103 (28.22×)**	**0.1629 (30.63×)**	**5.0198 (28.34×)**

4.2 DONeRF Synthetic Dataset

DONeRF [33] achieves fast rendering using *ground-truth depth* for training. However, the ground-truth depth is *not* available in most practical cases. As a remedy, they propose to use a pre-trained NeRF model to estimate depth as a proxy for the ground-truth depth. The approach of DONeRF without ground-truth depth (*e.g.*, DONeRF-16-noGT) is very relevant to ours since we both do not require the ground-truth depth and employ a pre-trained NeRF model for help. Thus, we compare with it using the synthetic dataset collected by the DONeRF paper.

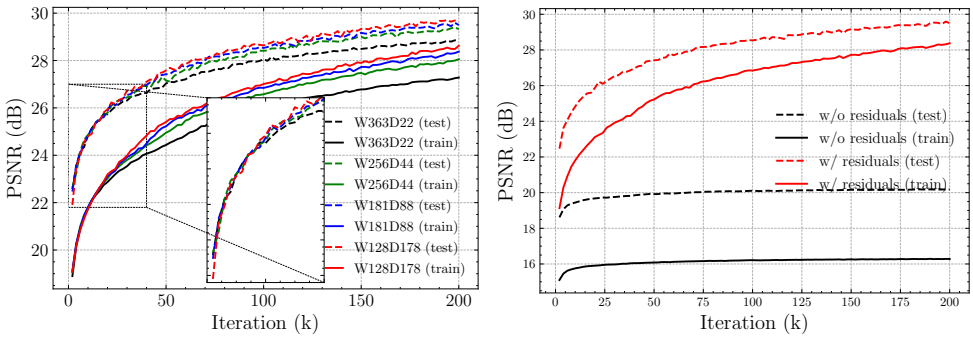
The quantitative results (PSNR and FLIP [3]) are presented in Tab. 3. **(1)** Trained purely with pseudo data, our method already outperforms DONeRF-16-noGT and DONeRF-8 (which even demands the ground-truth depth as input). **(2)** Similar to the case (Tab. 2) on the NeRF synthetic dataset, including the original real images for training significantly boosts the performance, by 2.78 dB. This improvement is even more obvious than the case in Tab. 2. We think the reason is that the DONeRF synthetic dataset has *more* training images (210 images) than the NeRF synthetic dataset (100 images). These real images are especially informative. Thus, more of them will lead to quality improvement.

Visual results are presented in Fig. 5, where our method delivers better visual quality than the baseline NeRF. In the scene `Pavillon` and `Barbershop`, our R2L network achieves *better* rendering quality than DONeRF-8 despite not using the ground-truth depth. Particularly note the reflection surfaces (*e.g.*, water in `Pavillon` and mirror in `Barbershop`), DONeRF cannot learn the reflection surfaces well because the ground-truth depth does not apply to the depth in the reflections, while our method (along with NeRF) still performs well.

Actual speed comparison. We further report the benchmark results of wall-time speed in Tab. 4 to demonstrate the FLOPs reduction is well-aligned with actual speedup. Our R2L network (W181D88) is $28 \sim 31\times$ faster than NeRF and $2\times$ faster than DONeRF-16-noGT.

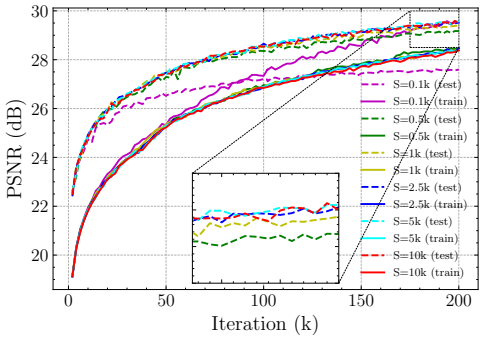
Table 5: Ablation study of different network and data schemes when learning a light field. Scene: Lego. All models are trained for $200k$ iterations

Network	Data	Train PSNR (dB)	Test PSNR (dB)
NeRF [32]	Original ($0.1k$ imgs)	25.61	19.81
NeRF+dropout [45]	Original ($0.1k$ imgs)	25.56	19.83
NeRF+BN [19]	Original ($0.1k$ imgs)	25.43	19.76
NeRF [32]	Pseudo ($10k$ imgs)	23.82	26.67
R2L (W181D88)	Pseudo ($10k$ imgs)	28.38	29.50
R2L (W181D88)	Pseudo + Original ($10.1k$ imgs)	29.85	30.09

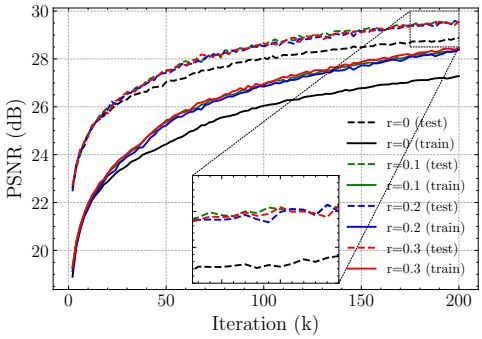


(a) Depth-width tradeoff

(b) With *vs.* without residuals



(c) Pseudo sample size



(d) Hard example ratio

Figure 6: Ablation studies. All networks are trained for $200k$ iterations, scene: Lego. Test PSNRs are plotted with dashed lines; train PSNRs are plotted with solid lines. **(a)** PSNR comparison of different network depth and width designs under (nearly) the same FLOPs and Params budget: W363D22 (FLOPs: 6.00M, Params: 3.01M), W256D44 (FLOPs: 6.02M, Params: 3.02M), W181D88 (FLOPs: 6.00M, Params: 3.02M), W128D178 (FLOPs: 6.02M, Params: 3.04M). **(b)** PSNR comparison between two network designs: using residuals or not for the deep R2L network. **(c)** PSNR comparison under different pseudo sample sizes. Default: $S = 10k$. **(d)** PSNR comparison under different hard example ratio $r \in \{0, 0.1, 0.2, 0.3\}$. Default: $r = 0.2$

4.3 Ablation Study

***More* data and *deep* network are critical**. In Tab. 5, we show the results of using the original 11-layer NeRF network to learn a light field on scene Lego. **(1)** Because of the severely insufficient data (only $0.1k$ training images), the network overfits to the training data while with only 19.81 test PSNR. Note, this overfitting cannot be alleviated by common regularization techniques in image classification like dropout [45], BN [19]. Only when the data size is greatly inflated (using the pseudo data) from $0.1k$ to $10k$, can we see a significant test PSNR improvement (from 19.81 to 26.67). This shows the (abundant) pseudo data is indispensable. **(2)** Compare our R2L to NeRF at the same setting of $10k$ pseudo images, our network design improves test PSNR by around 3 (from 26.67 to 29.50), which is a significant boost in terms of rendering quality. This justifies the

necessity of our *deep* network design. Another sign encouraging us to use deep networks is shown in Fig. 6(a), where we can consistently see performance gains when trading width for depth under the same FLOPs budget.

Ablation of residuals in our R2L network. Although the original NeRF network also employs skip connections (to add ray directions as input), it can hardly be considered as a typical residual network [14] in fact, as they do not use residuals in the internal layers. In comparison, we promote employing extensive residual blocks in the internal layers. Its necessity is justified by Fig. 6(b). As seen, without residuals, the network is barely trainable.

Ablation of pseudo sample size. The effect of pseudo sample size is of particular interest. As shown in Fig. 6(c), 100 images (see the $S = 0.1k$) are not enough to train our deep R2L network – note the test PSNR saturates early at around $50k$ iterations while its train PSNR keeps arising sharply. This is a typical case of overfitting, caused by the over-parameterized model not being fed with enough data. In contrast, with more data (see the cases of $S \geq 0.5k$), the train PSNR is held down and the test PSNR keeps arising. We observe no significant improvement starting from around $5k$ images.

Ablation of hard example ratio. Here we vary the hard example ratio r and see how it affects the performance. To make a fair comparison, we keep the training batch size always the same ($98,304$ rays per batch) when varying r. As shown in Fig. 6(d), using hard examples in each batch significantly improves the network learning in either train PSNR (*i.e.*, better optimization) or test PSNR (*i.e.*, better generalization) against the case of $r = 0$. There is no significant difference between hard example ratio $r = 0.1$, 0.2, and 0.3. In our experiments, we simply use a setting as $r = 0.2$.

5 Conclusion

We present the first *deep* neural light field network that can represent complex synthetic and real-world scenes. Starkly different from existing NeRF-like MLP networks, our R2L network is featured by an unprecedented depth and extensive residual blocks. We show the key to training such a deep network is abundant data, while the original captured images are barely sufficient. To resolve this, we propose to adopt a pre-trained NeRF model to synthesize excessive pseudo samples. With them, our proposed neural light field network achieves more than $26 \sim 35\times$ FLOPs reduction and $28 \sim 31\times$ wall-time acceleration on the NeRF synthetic dataset, with rendering quality improved significantly.

Future work. (1) Our method is only evaluated on static scenes in this paper akin to NeRF [32]. Extension to dynamic scenes (*e.g.*, [26]) is a worthy future direction. (2) Our R2L networks still demand a pre-trained NeRF model to synthesize pseudo data. This said, the proof-of-concept R2L networks already show the encouraging potential of NeLF representations. Completely avoiding the NeRF teacher is an obvious next step for our method.

References

[1] Edward H Adelson, James R Bergen, et al. *The plenoptic function and the elements of early vision*, volume 2. MIT Press, 1991. 3

[2] Edward H Adelson and John YA Wang. Single lens stereo with a plenoptic camera. *TPAMI*, 14(2):99–106, 1992. 3

[3] Pontus Andersson, Jim Nilsson, Tomas Akenine-Möller, Magnus Oskarsson, Kalle Åström, and Mark D Fairchild. Flip: A difference evaluator for alternating images. In *Proceedings of the ACM in Computer Graphics and Interactive Techniques*, 2020. 9

[4] Benjamin Attal, Jia-Bin Huang, Michael Zollhoefer, Johannes Kopf, and Changil Kim. Learning neural light fields with ray-space embedding networks. *arXiv preprint arXiv:2112.01523*, 2021. 2, 4, 5, 7, 8

[5] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *NeurIPS*, 2014. 4

[6] 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. *arXiv preprint arXiv:2103.13415*, 2021. 2

[7] Mojtaba Bemana, Karol Myszkowski, Hans-Peter Seidel, and Tobias Ritschel. X-fields: Implicit neural view-, light-and time-image interpolation. *ACMTOG*, 39(6):1–15, 2020. 4, 7

[8] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *SIGKDD*, 2006. 2, 4

[9] Guobin Chen, Wongun Choi, Xiang Yu, Tony Han, and Manmohan Chandraker. Learning efficient object detection models with knowledge distillation. In *NeurIPS*, 2017. 4

[10] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *CVPR*, 2019. 2

[11] Frank Dellaert and Lin Yen-Chen. Neural volume rendering: Nerf and beyond. *arXiv preprint arXiv:2101.05204*, 2020. 2

[12] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. *arXiv preprint arXiv:2103.10380*, 2021. 3, 8

[13] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. The lumigraph. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*, 1996. 3

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 5, 11

[15] Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. *arXiv preprint arXiv:2103.14645*, 2021. 3, 8

[16] Joao F Henriques, Joao Carreira, Rui Caseiro, and Jorge Batista. Beyond hard negative mining: Efficient detector learning via block-circulant decomposition. In *CVPR*, 2013. 6

[17] Byeongho Heo, Minsik Lee, Sangdoo Yun, and Jin Young Choi. Knowledge transfer via distillation of activation boundaries formed by hidden neurons. In *AAAI*, 2019. 4

[18] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NeurIPS Workshop*, 2014. 2, 4

[19] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 10

[20] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*, 2019. 4

[21] James T Kajiya and Brian P Von Herzen. Ray tracing volume densities. *SIGGRAPH*, 18(3):165–174, 1984. 4

[22] Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. Learning-based view synthesis for light field cameras. *ACM Transactions on Graphics*, 35(6):1–10, 2016. 4

[23] Michael J Kearns, Umesh Virkumar Vazirani, and Umesh Vazirani. *An introduction to computational learning theory*. MIT Press, 1994. 2

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 7

[25] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*, 1996. 3

[26] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *CVPR*, 2021. 2, 11

[27] David B Lindell, Julien NP Martel, and Gordon Wetzstein. Autoint: Automatic integration for fast neural volume rendering. In *CVPR*, 2021. 2, 3, 7

[28] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In *NeurIPS*, 2020. 2, 3, 7, 8, 15

[29] Yufan Liu, Jiajiong Cao, Bing Li, Chunfeng Yuan, Weiming Hu, Yangxi Li, and Yunqiang Duan. Knowledge distillation via instance relationship graph. In *CVPR*, 2019. 4

[30] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, 2019. 2

[31] 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. *ACM Transactions on Graphics*, 38(4):1–14, 2019. 4

[32] 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. In *ECCV*, 2020. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 15, 18

[33] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H. Mueller, Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, and Markus Steinberger. DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks. *Computer Graphics Forum*, 2021. 2, 3, 5, 7, 8, 9, 15

[34] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019. 2

[35] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In *CVPR*, 2019. 4

[36] Nikolaos Passalis and Anastasios Tefas. Learning deep representations with probabilistic knowledge transfer. In *ECCV*, 2018. 4

[37] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019. 7

[38] Baoyun Peng, Xiao Jin, Jiaheng Liu, Dongsheng Li, Yichao Wu, Yu Liu, Shunfeng Zhou, and Zhaoning Zhang. Correlation congruence for knowledge distillation. In *ICCV*, 2019. 4

[39] Martin Piala and Ronald Clark. Terminerf: Ray termination prediction for efficient neural rendering. In *"3DV"*, 2021. 2

[40] Daniel Rebain, Wei Jiang, Soroosh Yazdani, Ke Li, Kwang Moo Yi, and Andrea Tagliasacchi. Derf: Decomposed radiance fields. In *CVPR*, 2021. 2, 3, 5

[41] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *ICCV*, 2021. 2, 3, 5, 7

[42] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. In *ICLR*, 2015. 4

[43] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *CVPR*, 2016. 6

[44] Vincent Sitzmann, Semon Rezchikov, William T Freeman, Joshua B Tenenbaum, and Fredo Durand. Light field networks: Neural scene representations with single-evaluation rendering. In *NeurIPS*, 2021. 4, 6

[45] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958, 2014. 10

[46] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural geometric level of detail: Real-time rendering with implicit 3d shapes. In *CVPR*, 2021. 2

[47] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. In *ICLR*, 2020. 4

[48] Frederick Tung and Greg Mori. Similarity-preserving knowledge distillation. In *CVPR*, 2019. 4

[49] Vladimir Vapnik. *The nature of statistical learning theory*. Springer Science & Business Media, 2013. 2

[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 7

[51] Huan Wang, Yijun Li, Yuehai Wang, Haoji Hu, and Ming-Hsuan Yang. Collaborative distillation for ultra-resolution universal style transfer. In *CVPR*, 2020. 4

[52] Lin Wang and Kuk-Jin Yoon. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. *TPAMI*, 2021. 4

[53] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *TIP*, 13(4):600–612, 2004. 2, 8

[54] Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. Nex: Real-time view synthesis with neural basis expansion. In *CVPR*, 2021. 7, 8, 15

[55] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *ICCV*, 2021. 2, 8

[56] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *CVPR*, 2021. 2

[57] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *ICLR*, 2017. 4

6 Appendix

6.1 Overview

In this appendix, we provide:

- the detailed per-scene quantitative results (PSNR, SSIM, FLIP, *etc*.) for NeRF and DONeRF datasets evaluated in the main paper (Sec. 6.2);
- more ablation studies about the number of sampled points and using or not using residuals in our R2L network (Sec. 6.3);
- a visual depiction of the pseudo data sampling used in our method (Sec. 6.4);
- the visual comparison on the NeRF real-world dataset, and rendered videos on both NeRF and DONeRF datasets (Sec. 6.5).

6.2 Per-Scene Quantitative Results

The detailed per-scene quantitative results (PSNR, SSIM, FLIP) for NeRF and DONeRF datasets are presented in Tabs. 6 and 7.

Table 6: Per-scene PSNR↑ and SSIM↑ comparison on the NeRF *synthetic* dataset (Realistic Synthetic 360°, Row 1) and *real-world* dataset (Real Forward-Facing, Row 2). Ours-1 is trained sorely on pseudo data and Ours-2 is trained on pseudo + real data. The best results are in red, second best in blue

Method	Chair		Drums		Ficus		Hotdog		Lego		Materials		Mic		Ship		Average	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
NeRF [32]	33.90	0.9985	25.56	0.9875	28.88	0.9958	34.64	0.9976	31.42	0.9922	29.22	0.9898	30.84	0.9950	29.30	0.9834	30.47	0.9925
Ours-1	34.02	0.9985	25.56	0.9876	28.48	0.9956	34.95	0.9977	31.26	0.9922	29.34	0.9903	31.02	0.9953	29.20	0.9834	30.48	0.9939
Ours-2	36.71	0.9992	26.03	0.9883	28.63	0.9957	38.07	0.9987	32.53	0.9939	30.20	0.9920	32.80	0.9969	29.98	0.9855	31.87	0.9950

Method	Room		Fern		Leaves		Fortress		Orchids		Flower		T-Rex		Horns		Average	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
NeRF [32]	33.07	0.9915	26.86	0.9815	22.40	0.9485	32.61	0.9969	21.29	0.9344	28.22	0.9730	28.10	0.9815	28.86	0.9897	27.68	0.9725
Ours-1	32.98	0.9916	26.87	0.9850	22.46	0.9491	32.61	0.9969	20.90	0.9287	28.25	0.9732	28.06	0.9811	28.50	0.9885	27.58	0.9722
Ours-2	33.30	0.9918	26.87	0.9850	22.71	0.9514	32.71	0.9970	21.01	0.9292	28.67	0.9747	28.12	0.9815	28.95	0.9894	27.79	0.9729

Table 7: Per-scene PSNR↑ and FLIP↓ of different methods on the DONeRF synthetic dataset. All the PSNR and FLIP results except ours and NeRF are directly cited from the DONeRF paper since we are using exactly the same dataset here

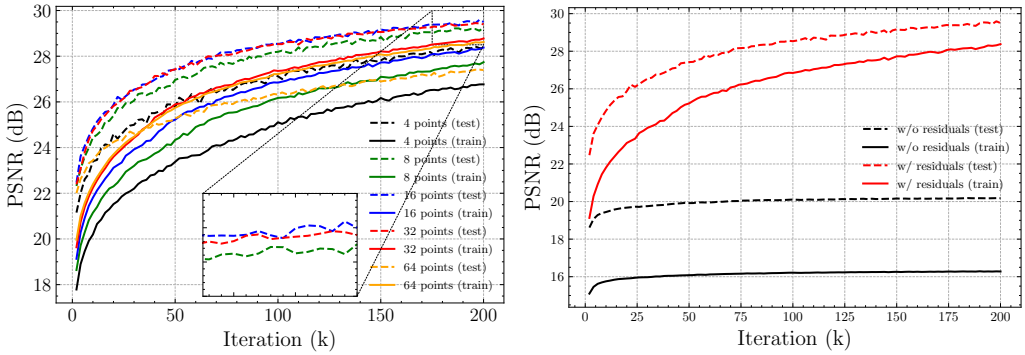
Method	Storage (MB)	FLOPs (M)	San Miguel		Pavillon		Classroom		Bulldozer		Forest		Barbershop		Average	
			PSNR	FLIP	PSNR	FLIP	PSNR	FLIP	PSNR	FLIP	PSNR	FLIP	PSNR	FLIP	PSNR	FLIP
NeRF (log+warp)	3.2	211.42	28.96	0.074	32.82	0.090	35.33	0.050	36.85	0.034	28.11	0.103	33.92	0.052	32.67	0.070
NSVF-large [28]	8.3	187.52	25.73	0.097	30.48	0.099	34.06	0.051	33.14	0.042	26.05	0.119	30.61	0.061	30.01	0.078
NeX-MLP [54]	89.0	42.71	30.68	0.060	30.41	0.102	34.10	0.046	34.03	0.046	24.65	0.125	29.45	0.075	30.55	0.076
DONeRF-16-noGT [33]	3.6	14.29	27.70	0.078	32.22	0.088	34.63	0.049	35.41	0.040	30.74	0.079	32.80	0.057	32.25	0.065
DONeRF-8 [33]	3.6	7.66	28.65	0.071	31.46	0.096	35.23	0.048	35.88	0.039	32.09	0.070	31.72	0.060	32.50	0.064
Ours-1	12.1	6.00	29.29	0.073	32.96	0.089	35.44	0.051	36.38	0.037	28.14	0.104	33.83	0.053	32.67	0.071
Ours-2	12.1	6.00	31.37	0.057	34.10	0.052	38.96	0.034	38.01	0.030	34.18	0.064	36.05	0.041	35.45	0.047

6.3 More Ablation Studies

(1) Effect of the number of sampled points and sample positions. Although our method is a light field network, which essentially is irrelevant to point sampling, we propose converting the ray origin and direction to multiple point coordinates along the ray as a simple and effective representation of the ray. We sample K points along each ray and *concatenate* all the K points together as a whole input to the proposed R2L network (see Sec. 3.4 in the main paper). The ablation study of the number of sampled points is shown in Fig. 7(a). Too few (*e.g.*, 4 and 8) or too many (*e.g.*, 64) sampled points are harmful to the performance. In our paper, we simply set K to 16.

We mentioned in Sec. 3.4 that during training the K points are *randomly* sampled, which is critical to curbing overfitting. By our empirical study, using *fixed* (*vs.* random) sample positions will lead to *dramatic* test PSNR degradation (by more than 6 dB) on the scene `Lego`.

(2) Residuals are critical to performance. In our paper, we find that the residuals are critical to the strong performance of our R2L network. Here we employ a network with the same design as W181D88, only removing the skip connections in the internal blocks. Its performance compared to the residual W181D88 is shown in Fig. 7(b). As seen, without residuals (black lines), the network saturates very early, and *dramatically* underperforms its counterpart with residuals (red lines).

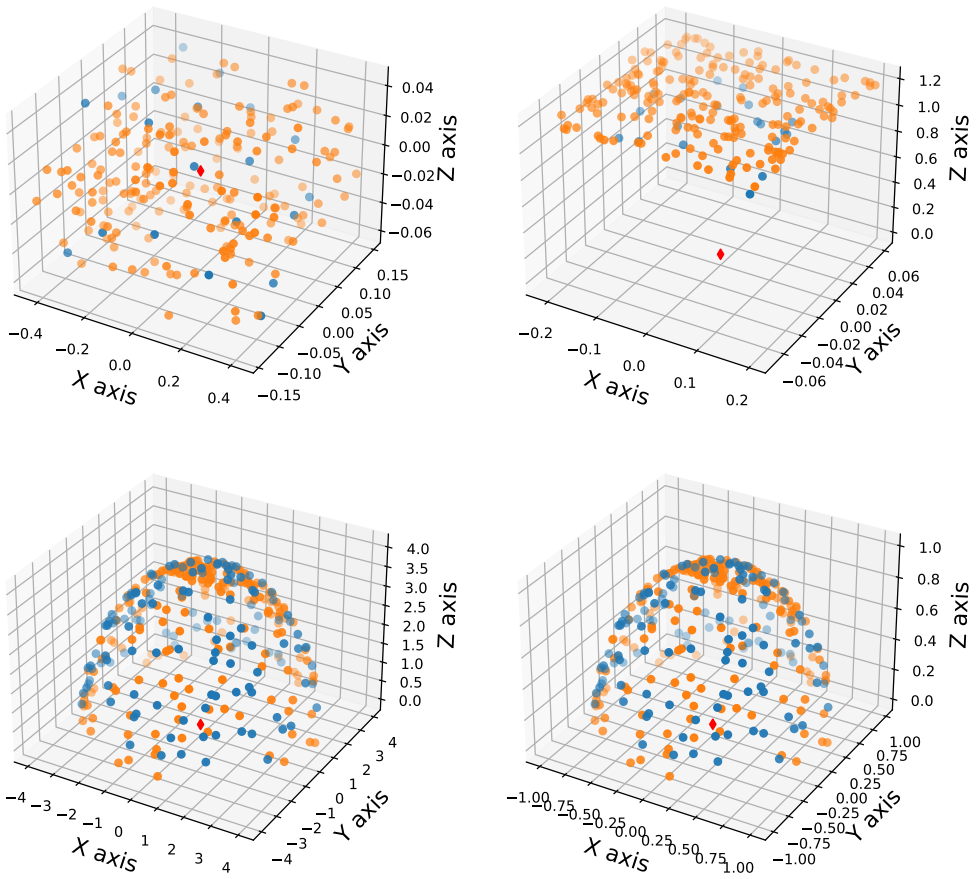


(a) Effect of number of sampled points (b) Effect of residuals in our R2L model

Figure 7: More ablation studies. All the networks are trained for $200k$ iterations on the scene `Lego`. Test PSNRs are plotted with dashed lines; train PSNRs are plotted with solid lines. **(a)** Test PSNR comparison of different numbers of sampled points in our R2L network (W181D88). Default: 16 points (blue lines). **(b)** PSNR comparison between using residuals (red lines) and not using residuals (black lines) in our R2L network (W181D88)

6.4 Visual Illustration of Ray Sampling in R2L Method

Given the training data of a scene, the viewing origin and direction bounding boxes can be inferred from the training data. In our R2L method, we randomly sample rays within the ray origin and direction bounding boxes. This is shown in Fig. 8. Note, since the pseudo data is *synthetic*, we have complete control over how the pseudo data is synthesized. We thereby explicitly demand the pseudo data completely cover the original training data, implying they are in the same domain, which is critical to the performance.



(a) Origins (b) Directions

Figure 8: Visualization of the origins and directions in 3D space of the pseudo samples (200 data points) generated in our method (Row 1: real-world scene `Fern`, Row 2: synthetic scene `Lego`). The origins and directions of the training data (around 20 data points for Row 1, around 100 for Row 2) are colored in *blue*; pseudo data origins and directions in *orange*. The red diamond marks the origin $(0, 0, 0)$ in the 3D coordinate system

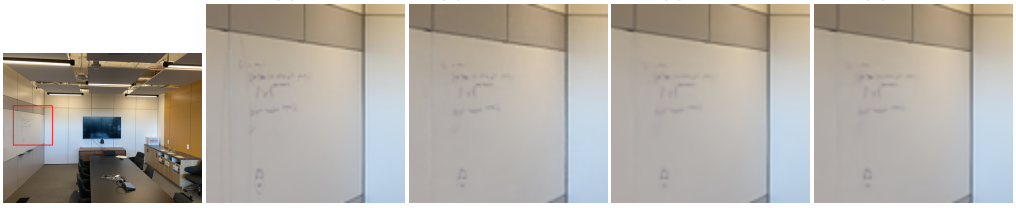
6.5 More Visual Results

(1) Visual comparison on NeRF real-world dataset. In Fig. 9, we present the visual comparison on the NeRF real-world dataset (Real Forward-Facing). As seen, our methods (Ours-1 and Ours-2) achieve comparable quality to NeRF (despite having only $\frac{1}{26}$ FLOPs).

One may have noted that Ours-2 achieves *significantly* better quality than Ours-1 and NeRF on the **synthetic** dataset (see Tab. 2 and Fig. 4 in the main paper), while on the **real-world** dataset, Ours-2 is not obviously better than Ours-1 and NeRF, either quantitatively (see Tab. 2 in the main paper) or qualitatively (see Fig. 9 here).

This is mainly because on the real-world dataset, the original real training set has only *dozens of* images – For the synthetic dataset, each scene has 100 training images; in comparison, for the real-world dataset, each scene has merely $17 \sim 54$ images. We observe an apparent positive correlation between data size and performance boost: *more real data, more performance boost*. E.g., on the real-world dataset, scene `Fern` has the fewest training samples (17 images) and scene `Horn` has the most (54 images). Note in Tab. 2, the test PSNR boost of Ours-2 over Ours-1 is also the smallest for scene `Fern` and the greatest for `Horn`. Conceivably, with more real data collected, Ours-2 will pose an even more pronounced advantage over Ours-1 and NeRF.

`Fern` (a) GT (b) NeRF [32] (c) Ours-1 (d) Ours-2



`Room` (a) GT (b) NeRF [32] (c) Ours-1 (d) Ours-2

Figure 9: Visual comparison between our R2L network (W256D88) and NeRF on the real-world scene `Fern` (Row 1) and `Room` (Row 2). Ours-1 is trained sorely on pseudo data and Ours-2 is trained on pseudo + real data

(2) Rendered videos of our method. Finally, we provide the rendered videos by our R2L method along with those by NeRF for reference. Please see "rendered_videos" on our project webpage.