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

Huan Wang[1,2,*], Jian Ren[1,†], Zeng Huang[1,‡], Kyle Olszewski[1], Menglei Chai[1], Yun Fu[2], and Sergey Tulyakov[1]

[1] Snap Inc.
[2] Northeastern University, USA
Project: https://snap-research.github.io/R2L

(a) NeRF *vs.* our NeLF method

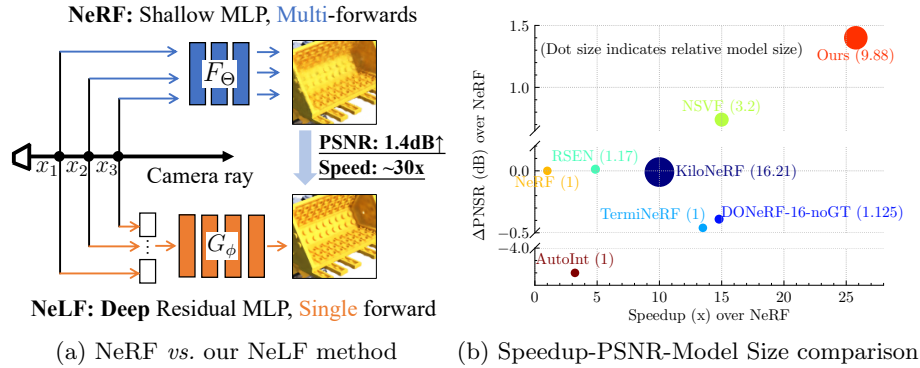(b) Speedup-PSNR-Model Size comparison

Fig. 1. (a) Our neural light field (NeLF, bottom) method improves the rendering quality by 1.40 PSNR over neural radiance field (NeRF, top) [34] 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

---

*Work done when Huan was an intern at Snap

†Corresponding author: jren@snapchat.com

‡Now at Google

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 $26 \sim 35\times$ FLOPs reduction (per camera ray) and $28 \sim 31\times$ runtime speedup, meanwhile delivering *significantly better* ($1.4 \sim 2.8$ dB average PSNR improvement) rendering quality than NeRF without any customized parallelism requirement.

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 [10,32,36,49] and differential neural renders [34] 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 [34], thus has inspired plenty of follow-up works [6,11,27,60].

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 [56] drop in [42]) while the reduction of inference time is only limited. Other research efforts focus on decreasing the number of sampled points [28,35]. However, this does not fundamentally resolve the sampling issue. Some work [35] 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

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 [34]. Repre. size measures the required storage of a neural network or cached files to represent a scene. $\mathit{\Delta}$PSNR 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	$\mathit{\Delta}$PSNR (dB)↑
NeRF [34]	$1\times$	$1\times$	$1\times$	No	0
PlenOctrees [59]	-	$3000\times$	$\sim 600\times$	No	+0.02
DONeRF-8 [35]	$27.60\times$	-	$1.125\times$	Depth data	-0.14
KiloNeRF [43]	$\sim 0.6\times$	$692\times$	$16.21\times$	Parallelism	-0.01
NSVF [30]	-	$\sim 15\times$	$\sim 3.2\times$	No	+0.74
AutoInt [28]	-	$3.22\times$	$\sim 1\times$	No	-4.2
TermiNeRF [41]	-	$13.49\times$	$\sim 1\times$	No	-0.46
RSEN [4]	-	$4.86\times^{*}$	$1.17\times$	No	+0.013
Ours	$26 \sim 35\times$	$28 \sim 31\times$	$4 \sim 10\times$	No	+1.40

field faithfully. Notably, since the debut of NeRF [34], its MLP-based network architecture is inherited with few substantial changes [6,35,42,43]. 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 that large networks hunger for large sample sizes to curb overfitting [23,52]. 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 [34], many follow-up works have been improving its efficiency. One major direction is to skip the empty space and sample more wisely along a camera ray. NSVF [30] defines a set of voxel-bounded implicit fields organized in a sparse voxel octree structure, which enables skipping empty space in novel view synthesis. AutoInt [28] improves the rendering efficiency by reducing the number of evaluations along a ray through learned partial integrals. DeRF [42] 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 [43] 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 samples for training. Differently, KiloNeRF is still a *NeRF*-based method while ours is *NeLF*. Point sampling is still needed in KiloNeRF while our method *roots out* this problem. Besides, KiloNeRF results in *thousands of* small networks, making parallelism more challenging and requiring customized parallelism implementation, while our *single* network can get significant speedup simply using the vanilla PyTorch [39]. DONeRF [35] 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 NeRF [34]) to only 4 to 16 while maintaining comparable or even better quality. However, the depth oracle network is learned with *ground-truth depth* as target, which is typically unavailable in practice. Our method does not demand it. Another direction for faster NeRF rendering is to pre-compute and cache the representations per the idea of trading memory for computational efficiency. 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 sparse neural radiance grid that enables real-time rendering on commodity hardware. We consider this line of works *orthogonal* to ours.

Neural light field (NeLF). Light fields enjoy a long history as a scene representation in computer vision and graphics [1,2]. Levoy *et al.* [26] 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 [7,22,33]. One recent neural light field paper is Sitzmann *et al.* [46]. 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 [34] 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 subdivided *local* light fields instead of the global light field. In their experiments, they also employ a pre-trained NeRF teacher for regularization. A very recent work [48] proposes a two-stage transformer-based model that can represent view-dependent effects accurately. A concurrent work NeuLF [29] employs a two-plane parameterization of the light field and uses a vanilla MLP network to learn the NeLF mapping. Our NeLF 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 [46], 8 layers in [4]); **(2)** we propose to leverage a NeRF model to synthesize extra data for training, making our method a bridge from 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 [46]), or can achieve better efficiency while maintaining the rendering quality (*e.g.*, [4] achieves merely around 5× speedup *vs.* our 30× speedup 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 Buciluă *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 [9,20,54,55]. 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 [44,54], feature map attention [61], feature distribution [38], activation boundary [17], inter-sample distance relationship [31,37,40,51], and mutual information [50]. 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 NeRF [34], 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, $F_\Theta : \mathbb{R}^5 \mapsto \mathbb{R}^4$, where F refers to an MLP neural
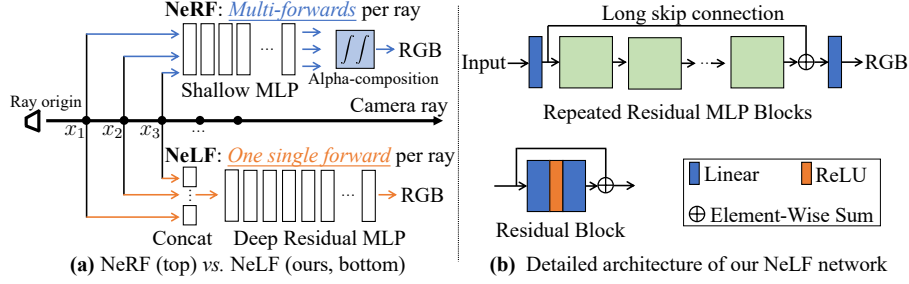
Fig. 2. (a) Comparison between our proposed NeLF network (*Deep Residual MLP*, bottom) and NeRF network (*Shallow MLP*, top). (b) Detailed architecture of the proposed *deep* light field network, which employs extensive repeated residual MLP blocks.

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 by 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 4D oriented ray to its target 3D RGB, $G_\phi : \mathbb{R}^4 \mapsto \mathbb{R}^3$. NeLF has several attractive advantages over NeRF. **(1)** Methodologically, it is more straightforward for 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 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 [34]), 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 supplementary material 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 [42], "*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.*, [42,43,35]) 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 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,

$$x_o \sim U(x_o^{\min}, x_o^{\max}), \; y_o \sim U(y_o^{\min}, y_o^{\max}), \; z_o \sim U(z_o^{\min}, z_o^{\max}),$$
$$x_d \sim U(x_d^{\min}, x_d^{\max}), \; y_d \sim U(y_d^{\min}, y_d^{\max}), \; z_d \sim U(z_d^{\min}, z_d^{\max}), \tag{1}$$

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
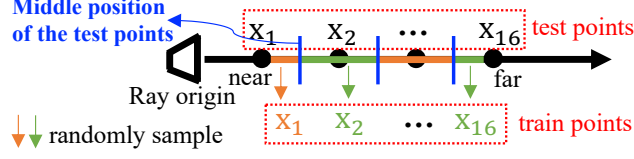
Fig. 3. Illustration of the point sampling in training and testing of our method. The orange and green colors denote the different *segments* of the ray. The blue color marks the *start* and *end* points of each segment. Each sampled train point is colored *based on the corresponding segment color*

shown in our supplementary material. 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:

$$(\hat{r}, \hat{g}, \hat{b}) = F_{\Theta^*}(x_o, y_o, z_o, x_d, y_d, z_d), \tag{2}$$

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,

$$\mathcal{L} = \text{MSE}(G_\phi(x_o, y_o, z_o, x_d, y_d, z_d), (\hat{r}, \hat{g}, \hat{b})). \tag{3}$$

3.4   Ray Representation and Point Sampling

It is critical to have a proper representation of a ray in NeLF. In this work, we propose a new simple and effective representation – we 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 (see the ablation of K in Fig. 6(a)) 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 [34], see Fig. 3). This design is critical to generalization. During testing, the K points are evenly spaced. We also tried changing the input to Plücker coordinates for our R2L network (inspired by [46]). Our representation achieves *better* test quality than Plücker (PSNR: 29.50 *vs.* 29.08, scene `Lego`, W181D88 network, trained with only pseudo data, $200K$ iters).

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 [16,45]. 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. (3)). 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).

3.6   Implementation Details

Our R2L 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 perform better, 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 [34], positional encoding [53] is used to enrich the input information.

4   Experiments

Datasets. We show experiments on the following datasets:

- **NeRF datasets** [34]. 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 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. Due to limited space, we defer the full-resolution (800×800) results to our supplementary material. 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 $1,000k$ iterations with the same learning rate schedule. The rays in a batch (batch size $98,304$

Table 2. PSNR↑, SSIM↑, and LPIPS↓ (AlexNet [25] is used for LPIPS) on the NeRF synthetic dataset (Realistic Synthetic 360°) and real-world dataset (Real Forward-Facing). R2L network: 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 their paper. The best results are in red, second best in blue

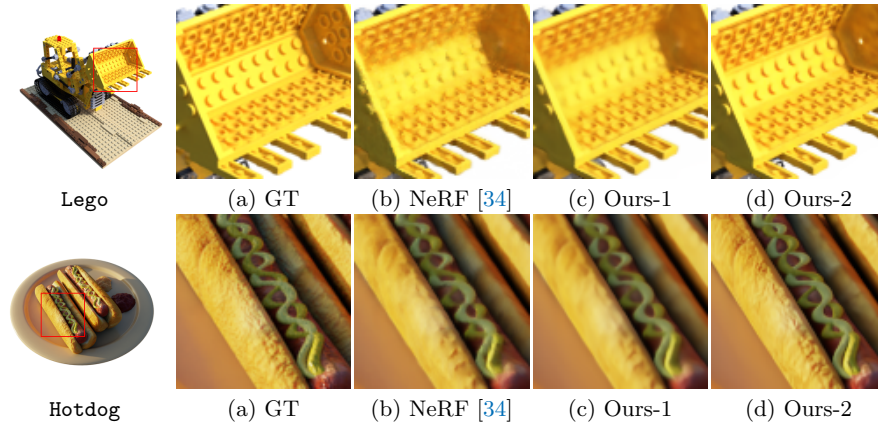| Method	Storage (MB)	FLOPs (M)	Synthetic			Real-world		
			PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Teacher NeRF [34]	2.4	303.82	30.47	0.9925	0.0391	27.68	0.9725	0.0733
Ours-1 (Pseudo)	23.7	11.79	30.48 (+0.01)	0.9939	0.0467	27.58 (-0.10)	0.9722	0.0997
Ours-2 (Pseudo+real)	23.7	11.79	31.87 (+1.40)	0.9950	0.0340	27.79 (+0.11)	0.9729	0.0968
Teacher NeRF in [43]	2.4	303.82	31.01	0.95	0.08	-	-	-
KiloNeRF [43]	38.9	∼500[†]	31.00 (-0.01)	0.95	0.03	-	-	-
Teacher NeRF in [4]	4.6	∼300	-	-	-	27.928	0.9160	0.065
RSEN [4]	5.4	67.2	-	-	-	27.941 (+0.013)	0.9161	0.060

Fig. 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 supplementary material for the visual comparison on the real-world dataset

rays) are randomly sampled from different images so that they do not share the same origin. This is found critical to achieving superior performance. Adam optimizer [24] is employed for all training. We use PyTorch 1.9 [39], referring to [58]. Experiments are conducted with 8 NVIDIA V100 GPUs.

Comparison methods. We compare with with the original NeRF [34] to show that we can achieve significantly better rendering quality while being much faster. Meanwhile, we also compare with DONeRF [35], NSVF [30], and NeX [57] since they also target efficient NVS as we do. Other efficient NVS works such as AutoInt [28] and X-Fields [7] have been shown less favorable than RSEN [4]. Therefore, we only compare with RSEN [4]. KiloNeRF [43], another closely related work apart from RSEN, will also be compared to. Similar to [4], we do not compare to baking-based methods [15,59,12]) as they trade memory footprint for speed while our method aims to maintain the compact representation.

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 [30]	8.3	187.52	30.01 (-2.66)	0.078
NeX-MLP [57]	89.0	42.71	30.55 (-2.12)	0.076
DONeRF-16-noGT [35]	3.6	14.29	32.25 (-0.42)	0.065
DoNeRF-8 [35]	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

Pavillon   (a) GT   (b) NeRF   (c) DONeRF-8   (d) Ours-2

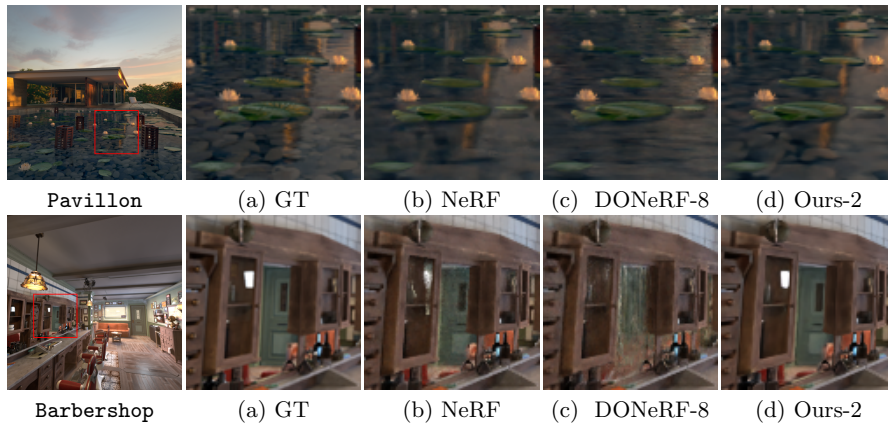Barbershop   (a) GT   (b) NeRF   (c) DONeRF-8   (d) Ours-2

Fig. 5. Visual comparison of ours, NeRF [34], DONeRF [35] on the DONeRF dataset

4.1 NeRF Synthetic and Real-World Dataset

The quantitative comparisons (PSNR, SSIM [56], LPIPS [62]) 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. Two primary reasons answer for this remarkable performance. First, our R2L network is *much deeper* than the NeRF network, which bestows a much greater capacity to represent scenes with fine-grained details. Second, we propose *hard-example training* (Sec. 3.5), which makes the network focus more on regressing the fine-grained details. **(3)** For the related works KiloNeRF and RSEN, their baseline NeRF models have different PSNRs due to different

Table 4. Average time (s) comparison among our R2L network (W181D88), DONeRF, and NeRF. The benchmark is conducted 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$\times$)	0.4162 (14.26$\times$)	0.3524 (14.16$\times$)	9.9344 (14.32$\times$)
Ours	**6.00 (35.24$\times$)**	**0.2103 (28.22$\times$)**	**0.1629 (30.63$\times$)**	**5.0198 (28.34$\times$)**

Table 5. Ablation study of different network and data schemes when learning a light field. Scene: `Lego`. All models are trained for $200k$ iterations. Note, the train PSNR of our method is lower than test PSNR because we use the hard examples (Sec. 3.5) *i.e.*, examples with small PSNR, for training.

Network	Data	Train PSNR (dB)	Test PSNR (dB)
NeRF [34]	Original ($0.1k$ imgs)	25.61	19.81
NeRF+dropout [47]	Original ($0.1k$ imgs)	25.56	19.83
NeRF+BN [19]	Original ($0.1k$ imgs)	25.43	19.76
NeRF [34]	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

settings, so the PSNR results cannot be directly compared. Instead, we compare the *PSNR change* over the baseline NeRFs. KiloNeRF gets 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) with much fewer FLOPs.

4.2   DONeRF Synthetic Dataset

DONeRF [35] 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. 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.

Visual results in Fig. 5 show that our method delivers better visual quality than the baseline NeRF. On 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
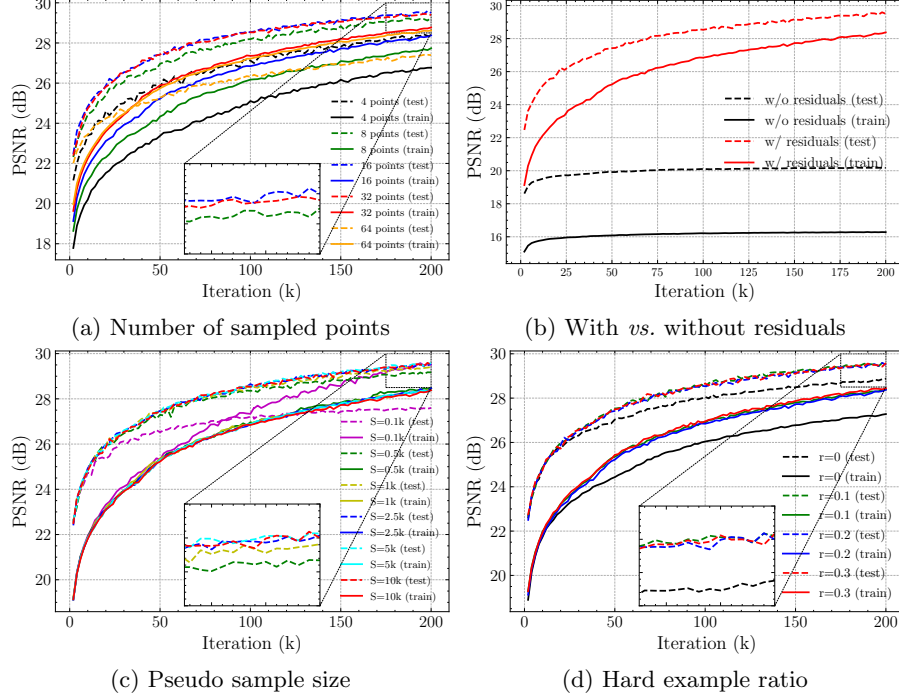
(a) Number of sampled points

(b) With *vs.* without residuals

(c) Pseudo sample size

(d) Hard example ratio

Fig. 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 sampled points in our R2L network (W181D88). Default: 16 points (blue lines) **(b)** PSNR comparison between two network designs: using residuals or not for our R2L network. **(c)** PSNR comparison under different pseudo sample sizes. Default: $S = 10k$. **(d)** PSNR comparison under different hard example ratios $r \in \{0, 0.1, 0.2, 0.3\}$. Default: $r = 0.2$

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 (and the original 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.

4.3   Ablation Study

More **data and** *deep* **network are critical**. Tab. 5 shows 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 with only 19.81 test PSNR. Note, this overfitting cannot be resolved by common regularization techniques like dropout [47] and

BN [19]. Only when the data size is greatly inflated (with 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 reason encouraging us to use deep networks is that we empirically find trading width for depth under the same FLOPs budget can consistently lead to performance gains (see our supplementary material).

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 $S = 0.1k$) 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.

References

1. Adelson, E.H., Bergen, J.R., et al.: The plenoptic function and the elements of early vision, vol. 2. MIT Press (1991) 4
2. Adelson, E.H., Wang, J.Y.: Single lens stereo with a plenoptic camera. TPAMI **14**(2), 99–106 (1992) 4
3. Andersson, P., Nilsson, J., Akenine-Möller, T., Oskarsson, M., Åström, K., Fairchild, M.D.: Flip: A difference evaluator for alternating images. In: Proceedings of the ACM in Computer Graphics and Interactive Techniques (2020) 12
4. Attal, B., Huang, J.B., Zollhoefer, M., Kopf, J., Kim, C.: Learning neural light fields with ray-space embedding networks. In: CVPR (2022) 3, 5, 6, 10
5. Ba, J., Caruana, R.: Do deep nets really need to be deep? In: NeurIPS (2014) 5
6. Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. arXiv preprint arXiv:2103.13415 (2021) 2, 3
7. Bemana, M., Myszkowski, K., Seidel, H.P., Ritschel, T.: X-fields: Implicit neural view-, light-and time-image interpolation. ACMTOG **39**(6), 1–15 (2020) 5, 10
8. Buciluǎ, C., Caruana, R., Niculescu-Mizil, A.: Model compression. In: SIGKDD (2006) 3, 5
9. Chen, G., Choi, W., Yu, X., Han, T., Chandraker, M.: Learning efficient object detection models with knowledge distillation. In: NeurIPS (2017) 5
10. Chen, Z., Zhang, H.: Learning implicit fields for generative shape modeling. In: CVPR (2019) 2
11. Dellaert, F., Yen-Chen, L.: Neural volume rendering: Nerf and beyond. arXiv preprint arXiv:2101.05204 (2020) 2
12. Garbin, S.J., Kowalski, M., Johnson, M., Shotton, J., Valentin, J.: Fastnerf: High-fidelity neural rendering at 200fps. arXiv preprint arXiv:2103.10380 (2021) 4, 10
13. Gortler, S.J., Grzeszczuk, R., Szeliski, R., Cohen, M.F.: The lumigraph. In: Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques (1996) 4
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016) 7, 14
15. Hedman, P., Srinivasan, P.P., Mildenhall, B., Barron, J.T., Debevec, P.: Baking neural radiance fields for real-time view synthesis. arXiv preprint arXiv:2103.14645 (2021) 4, 10
16. Henriques, J.F., Carreira, J., Caseiro, R., Batista, J.: Beyond hard negative mining: Efficient detector learning via block-circulant decomposition. In: CVPR (2013) 9
17. Heo, B., Lee, M., Yun, S., Choi, J.Y.: Knowledge transfer via distillation of activation boundaries formed by hidden neurons. In: AAAI (2019) 5
18. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. In: NeurIPS Workshop (2014) 3, 5
19. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: ICML (2015) 12, 14
20. Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., Wang, F., Liu, Q.: Tinybert: Distilling bert for natural language understanding. arXiv preprint arXiv:1909.10351 (2019) 5
21. Kajiya, J.T., Von Herzen, B.P.: Ray tracing volume densities. SIGGRAPH **18**(3), 165–174 (1984) 6
22. Kalantari, N.K., Wang, T.C., Ramamoorthi, R.: Learning-based view synthesis for light field cameras. ACM Transactions on Graphics **35**(6), 1–10 (2016) 5

23. Kearns, M.J., Vazirani, U.V., Vazirani, U.: An introduction to computational learning theory. MIT Press (1994) 3
24. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (2015) 10
25. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NeurIPS (2012) 10
26. Levoy, M., Hanrahan, P.: Light field rendering. In: Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques (1996) 4
27. Li, Z., Niklaus, S., Snavely, N., Wang, O.: Neural scene flow fields for space-time view synthesis of dynamic scenes. In: CVPR (2021) 2
28. Lindell, D.B., Martel, J.N., Wetzstein, G.: Autoint: Automatic integration for fast neural volume rendering. In: CVPR (2021) 2, 3, 4, 10
29. Liu, C., Li, Z., Yuan, J., Xu, Y.: Neulf: Efficient novel view synthesis with neural 4d light field. In: EGSR (2022) 5
30. Liu, L., Gu, J., Zaw Lin, K., Chua, T.S., Theobalt, C.: Neural sparse voxel fields. In: NeurIPS (2020) 3, 4, 10, 11
31. Liu, Y., Cao, J., Li, B., Yuan, C., Hu, W., Li, Y., Duan, Y.: Knowledge distillation via instance relationship graph. In: CVPR (2019) 5
32. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space. In: CVPR (2019) 2
33. Mildenhall, B., Srinivasan, P.P., Ortiz-Cayon, R., Kalantari, N.K., Ramamoorthi, R., Ng, R., Kar, A.: Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. ACM Transactions on Graphics **38**(4), 1–14 (2019) 5
34. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: ECCV (2020) 1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12
35. Neff, T., Stadlbauer, P., Parger, M., Kurz, A., Mueller, J.H., Chaitanya, C.R.A., Kaplanyan, A.S., Steinberger, M.: DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks. Computer Graphics Forum (2021) 2, 3, 4, 7, 10, 11, 12
36. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. In: CVPR (2019) 2
37. Park, W., Kim, D., Lu, Y., Cho, M.: Relational knowledge distillation. In: CVPR (2019) 5
38. Passalis, N., Tefas, A.: Learning deep representations with probabilistic knowledge transfer. In: ECCV (2018) 5
39. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. In: NeurIPS (2019) 4, 10
40. Peng, B., Jin, X., Liu, J., Li, D., Wu, Y., Liu, Y., Zhou, S., Zhang, Z.: Correlation congruence for knowledge distillation. In: ICCV (2019) 5
41. Piala, M., Clark, R.: Terminerf: Ray termination prediction for efficient neural rendering. In: "3DV" (2021) 3
42. Rebain, D., Jiang, W., Yazdani, S., Li, K., Yi, K.M., Tagliasacchi, A.: Derf: Decomposed radiance fields. In: CVPR (2021) 2, 3, 4, 7
43. Reiser, C., Peng, S., Liao, Y., Geiger, A.: Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In: ICCV (2021) 3, 4, 7, 10
44. Romero, A., Ballas, N., Kahou, S.E., Chassang, A., Gatta, C., Bengio, Y.: Fitnets: Hints for thin deep nets. In: ICLR (2015) 5

45. Shrivastava, A., Gupta, A., Girshick, R.: Training region-based object detectors with online hard example mining. In: CVPR (2016) 9
46. Sitzmann, V., Rezchikov, S., Freeman, W.T., Tenenbaum, J.B., Durand, F.: Light field networks: Neural scene representations with single-evaluation rendering. In: NeurIPS (2021) 5, 8
47. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. JMLR **15**(1), 1929–1958 (2014) 12, 13
48. Suhail, M., Esteves, C., Sigal, L., Makadia, A.: Light field neural rendering. In: CVPR (2022) 5
49. Takikawa, T., Litalien, J., Yin, K., Kreis, K., Loop, C., Nowrouzezahrai, D., Jacobson, A., McGuire, M., Fidler, S.: Neural geometric level of detail: Real-time rendering with implicit 3d shapes. In: CVPR (2021) 2
50. Tian, Y., Krishnan, D., Isola, P.: Contrastive representation distillation. In: ICLR (2020) 5
51. Tung, F., Mori, G.: Similarity-preserving knowledge distillation. In: CVPR (2019) 5
52. Vapnik, V.: The nature of statistical learning theory. Springer Science & Business Media (2013) 3
53. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: NeurIPS (2017) 9
54. Wang, H., Li, Y., Wang, Y., Hu, H., Yang, M.H.: Collaborative distillation for ultra-resolution universal style transfer. In: CVPR (2020) 5
55. Wang, L., Yoon, K.J.: Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. TPAMI (2021) 5
56. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. TIP **13**(4), 600–612 (2004) 2, 11
57. Wizadwongsa, S., Phongthawee, P., Yenphraphai, J., Suwajanakorn, S.: Nex: Real-time view synthesis with neural basis expansion. In: CVPR (2021) 10, 11
58. Yen-Chen, L.: Nerf-pytorch. https://github.com/yenchenlin/nerf-pytorch/ (2020) 10
59. Yu, A., Li, R., Tancik, M., Li, H., Ng, R., Kanazawa, A.: Plenoctrees for real-time rendering of neural radiance fields. In: ICCV (2021) 3, 10
60. Yu, A., Ye, V., Tancik, M., Kanazawa, A.: pixelnerf: Neural radiance fields from one or few images. In: CVPR (2021) 2
61. Zagoruyko, S., Komodakis, N.: Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In: ICLR (2017) 5
62. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: CVPR (2018) 11