Recent Advances in Efficient Neural Light Field

Huan Wang

Northeastern University, Boston, USA

Talk @Zhidx, 07/07/2023, Fri

Short Bio: Huan Wang, BE'16@ZJU, MS'19 @ZJU, now 4th-year Ph.D. candidate @NEU. Interned at Alibaba/MERL/Snap. Now interning at Google. Work on **efficient deep learning** (network sparsity & distillation, *etc.*), primarily in the vision domain.

Outline: 2 Papers

- R2L: Distilling NeRF to NeLF (ECCV 2022)
- MobileR2L: Run R2L on mobile devices via arch. optimization (CVPR 2023)

R2L: Distilling Neural Radiance Field to Neural Light Field for Efficient Novel View Synthesis (ECCV 2022)

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

Project: https://snap-research.github.io/R2L

Talk @Zhidx, 07/07/2023, Fri

Background & Motivation

What is NVS (Image-Based Rendering) and NeRF?

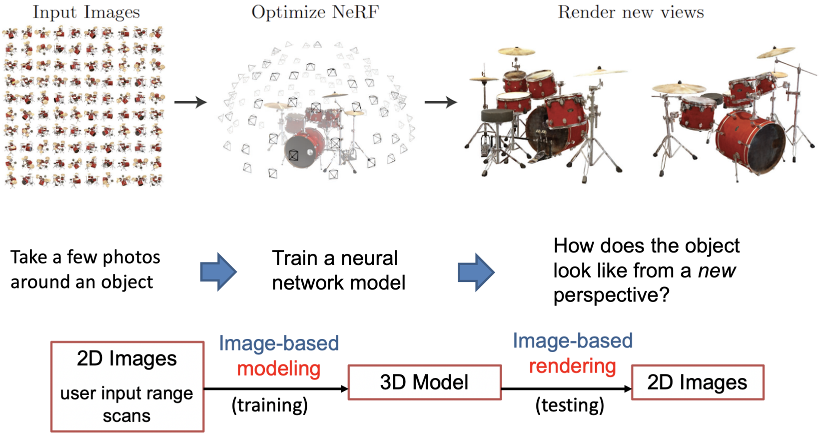


Figure: Illustration of NeRF framework. Src: (Mildenhall et al., 2020); edited.

NeRF, Implicit Scene Representation

- NeRF (or neural radiance field) (Mildenhall et al., 2020) is an *implicit* representation of a scene – a buzz word since 2020, Best Paper HM in ECCV'20.
- It opens the new doors of representing complex scenes with a simple MLP network.
- So simple but working! $(x, y, z; \theta, \phi) \rightarrow$ (NeRF network) $\rightarrow (r, g, b; \sigma) \rightarrow$ (Accumulation) $\rightarrow (r, g, b)$

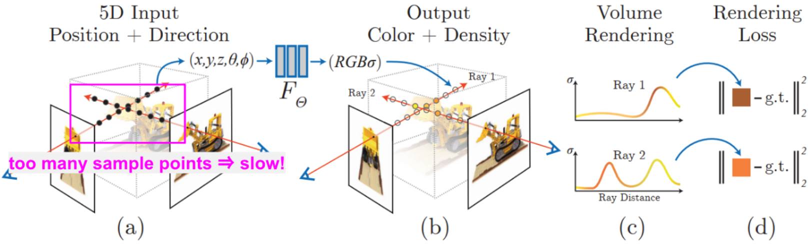


Figure: Illustration of NeRF framework. Src: (Mildenhall et al., 2020); edited.

Aside, Background: 3D Representations

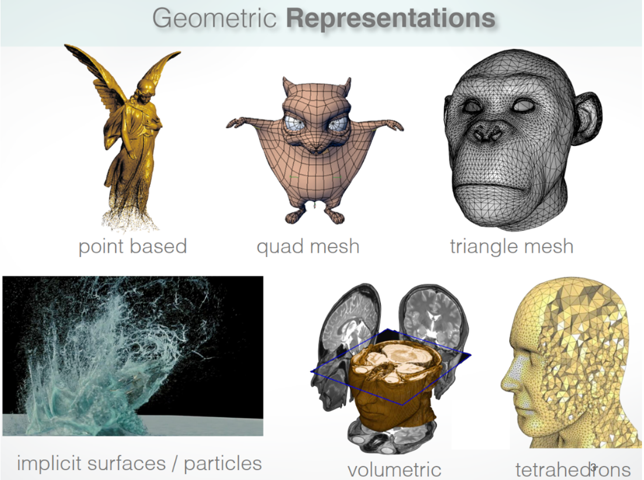


Figure: Courtesy: Prof. Hao Li, USC, CSCI 621: Digital Geometry Processing, http://cs621.hao-li.com. *Different representations require different processing methods.*

Aside, Background: Implicit Representations

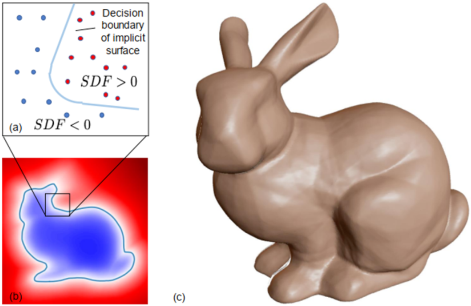


Figure 2: Our DeepSDF representation applied to the Stanford Bunny: (a) depiction of the underlying implicit surface $SDF = 0$ trained on sampled points inside $SDF < 0$ and outside $SDF > 0$ the surface, (b) 2D cross-section of the signed distance field, (c) rendered 3D surface recovered from $SDF = 0$. Note that (b) and (c) are recovered via DeepSDF.

Figure: DeepSDF (Park et al., 2019) uses SDF (signed distance function) parameterized by a neural network to represent the surface / shape of an object.

Motivation: NeRF is great but *slow* in inference

- However, one primary downside of NeRF is the **prohibitively slow** inference: Rendering one 400×400 image takes 6.7s on an NVIDIA V100 GPU.
- Our goal: *Faster inference* of NeRF for efficient NVS (novel view synthesis)

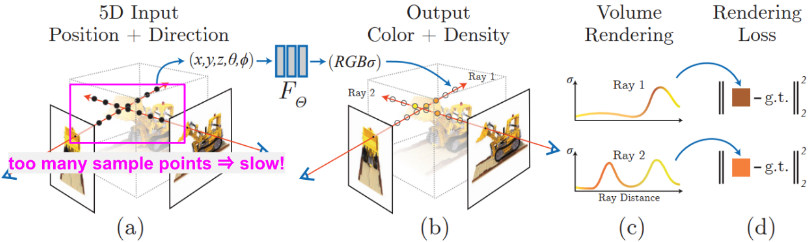


Figure: Illustration of NeRF framework. Src: (Mildenhall et al., 2020); edited.

Speed Bottleneck: Too Many Sampled Points

- NeRF is slow because of too many sampled points along the ray. Tradeoff here: quality vs. speed.
- Our goal: *Faster inference* of NeRF for efficient NVS (novel view synthesis) $\Rightarrow$ Technical goal: **Reduce the #sampled points**

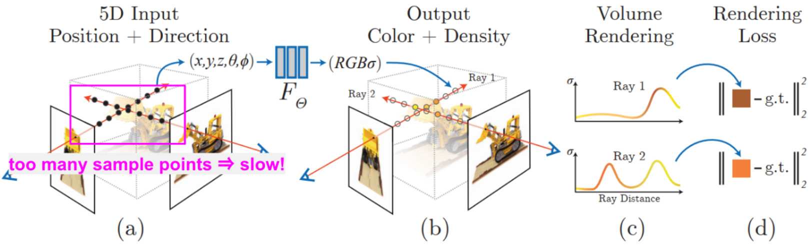
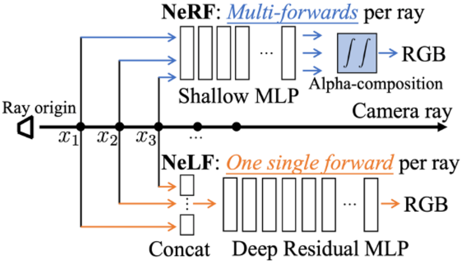


Figure: Illustration of NeRF framework. Src: (Mildenhall et al., 2020); edited.

Proposed Method: R2L (NeRF to NeLF)

NeLF *vs.* NeRF

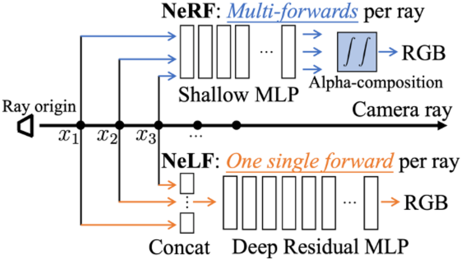
- We seek fast inference through another representation of a scene: **NeLF (or neural light field)**.
- NeLF *vs.* NeRF – **Upside**: Fast! Rendering one pixel amounts to one network query in NeLF vs. hundreds of network queries in NeRF, as shown below.



(a) NeRF (top) *vs.* NeLF (ours, bottom)

NeLF *vs.* NeRF

- We seek fast inference through another representation of a scene: **NeLF (or neural light field)**.
- NeLF *vs.* NeRF – **Upside**: Fast! Rendering one pixel amounts to one network query in NeLF vs. hundreds of network queries in NeRF, as shown below.



(a) NeRF (top) *vs.* NeLF (ours, bottom)

Downsides: (1) NeLF is inherently harder to learn than NeRF. (2) Given the same batch of images, the data for NeLF is much fewer than NeRF.
In short, to learn NeLF successfully, we must overcome two problems: #1 The target function is harder to learn. #2 Even worse, the data sample size shrinks.

R2L: ResMLP

To overcome problem #1, we make two contributions: (1) ResMLP, (2) Better ray representation.

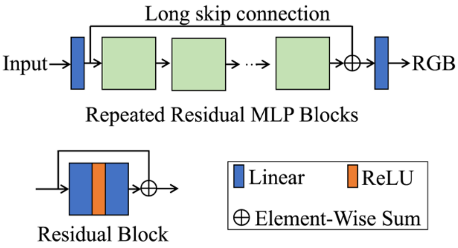


Figure: Architecture of our proposed ResMLP network. Note, this is the first attempt that deep residuals are used in NeRF-like models.

R2L: Ray Representation - PointCat

To overcome problem #1, we make two contributions: (1) ResMLP, (2) Better ray representation.

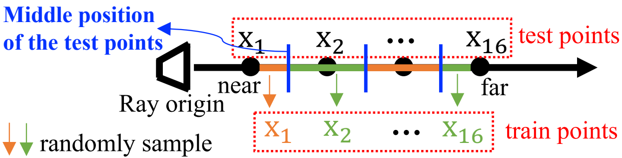


Figure: Illustration of the point sampling in training and testing of our method. The orange and green colors denote the different *segments* of the ray. The blue color marks the *start* and *end* points of each segment. Each sampled train point is colored *based on the corresponding segment color*.

The sampled points are concatenated into a long vector to make the input of our NeLF network – very simple!

During training, the points are randomly sampled; during testing, they are fixed.

Other Ray Representations - Two Planes

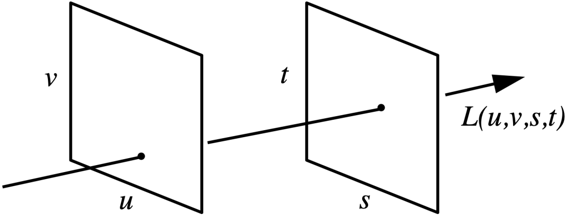


Figure 1: The light slab representation.

Figure: Two-plane parameterization (Levoy & Hanrahan, 1996) represents each ray in a 3D space with its intersection points on two planes.

Other Ray Representations - Plücker

mathematically convenient. In particular, we propose to leverage the 6D Plücker parameterization of the space of light rays $\mathcal{L}$ for LFNs. The Plücker coordinates (see [49] for an excellent overview) of a ray ℓ through a point $\mathbf{p}$ in a normalized direction $\mathbf{d}$ are

$$\mathbf{r} = (\mathbf{d}, \mathbf{m}) \in \mathbb{R}^6 \text{ where } \mathbf{m} = \mathbf{p} \times \mathbf{d}, \text{ for } \quad \mathbf{d} \in \mathbb{S}^2, \mathbf{p} \in \mathbb{R}^3. \tag{4}$$

where $\times$ denotes the cross product. While Plücker coordinates are a-priori 6-tuples of real numbers, the coordinates of any ray lie on a 4-dimensional subspace $\mathcal{L}$. Plücker coordinates uniformly represent

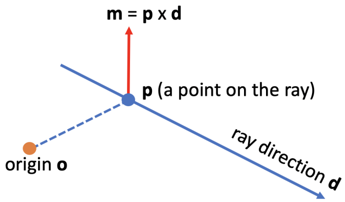


Figure: Plücker representation (Sitzmann et al., 2021) is a tuple of 6 real numbers, made with a normalized direction vector (3D) and a cross-product (3D) of the direction and a point on the line. **Not as good as our Point-Concat scheme** (see ablation study in the paper).

R2L: Distilling w/ Pseudo Data & Finetuning w/ Real

Network larger, more data hungry? (*i.e.*, problem #2) ⇒ Use a *pretrained* NeRF to synthesize a large amount of data.

Training loss: MSE

$$\mathcal{L} = \mathrm{MSE}(\underbrace{\boxed{G_\phi}}_{\textbf{R2L network}}(\overbrace{\underbrace{\boxed{x_o, y_o, z_o}}_{\textbf{Ray origin}}, \underbrace{\boxed{x_d, y_d, z_d}}_{\textbf{Ray direction}}), \underbrace{\boxed{(\hat{r}, \hat{g}, \hat{b})}}_{\textbf{Ray RGB}}}^{\textbf{Pseudo data (via querying a trained NeRF)}})$$

Finetune: After training our ResMLP model on the pseudo data, it can achieve comparable performance to the NeRF teacher. Then we *finetune* the student on the original real data, which *significantly* boosts the student's performance.

Final Question: How to Get the Psuedo Inputs?

Sampling from joint distribution ⇒ marginal distribution product (*i.e.*, assuming independent): $\mathscr{D}(X, Y, Z) \approx \mathscr{D}(X) \times \mathscr{D}(Y) \times \mathscr{D}(Z)$

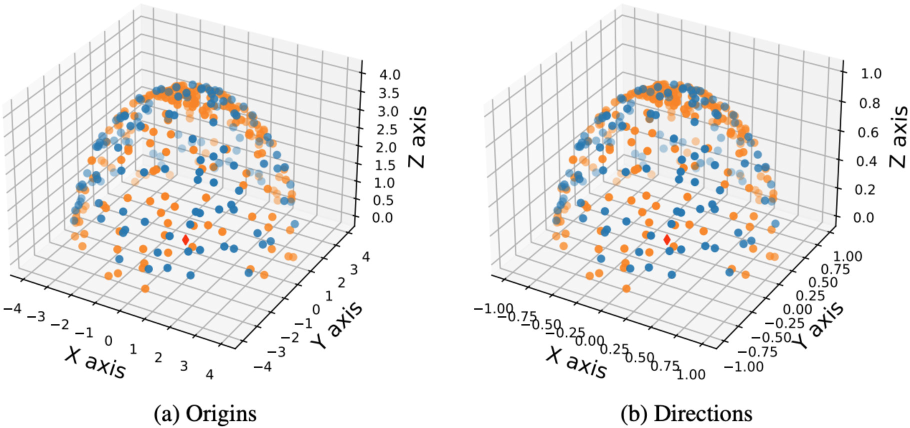
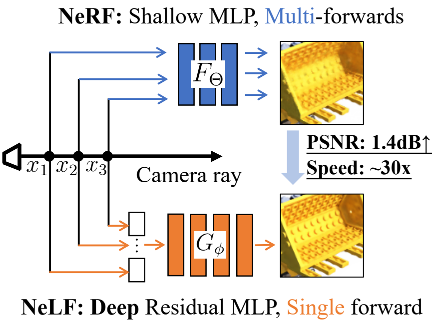


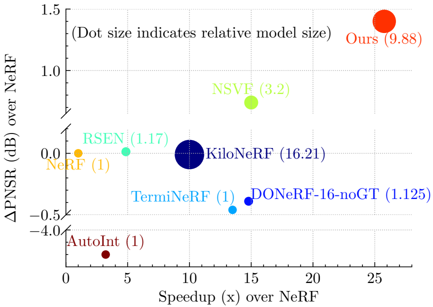
Figure: Blue: original data. Orange: pseudo data. Psuedo data needs to overlap with the original data – same domain!

Experimental Results

Comparison Overview (NeRF Synthetic Dataset)



(a) NeRF *vs.* our NeLF method



(b) Speedup-PSNR-Model Size comparis

Figure: (a) Our neural light field (NeLF, bottom) method improves the rendering quality by 1.40 PSNR over neural radiance field (NeRF, top) (Mildenhall et al., 2020) on the NeRF synthetic dataset, while being around $30\times$ 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.

Visual Comparison (NeRF Synthetic Dataset)

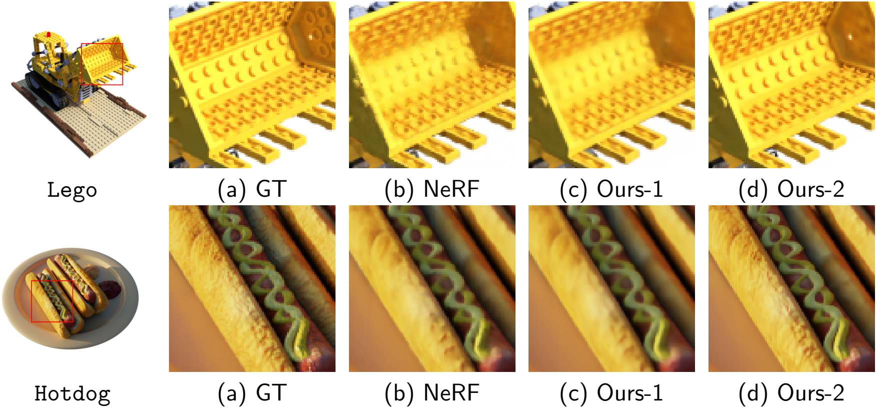
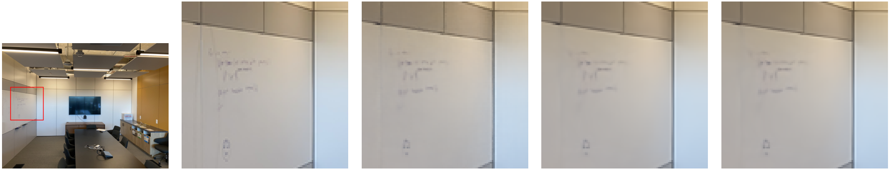


Figure: Ours *vs.* NeRF (Mildenhall et al., 2020). Ours-1 is trained sorely on pseudo data, Ours-2 on pseudo and real data. Much better quality with only 1/26 FLOPs. Please see our paper for more results.

Visual Comparison (Real-World Dataset)

Fern (a) GT (b) NeRF (c) Ours-1 (d) Ours-2



Room (a) GT (b) NeRF (c) Ours-1 (d) Ours-2

Figure: Ours *vs.* NeRF (Mildenhall et al., 2020). Ours-1 is trained sorely on pseudo data, Ours-2 on pseudo and real data. Comparable quality with only 1/26 FLOPs. Please see our paper for more results.

Conclusion

Conclusion

We present **R2L**, a new and novel **efficient NVS** method by **distilling a NeRF to NeLF**. Three major techniques are proposed to enable it:

- **Network**: We propose a much deeper ResMLP network to seek greater representative power – This is the *first attempt* that optimizes the NeRF speed from the network architecture level and the *first* time dense residuals are properly integrated into a NeRF network.
- **Data**: We propose to employ pretrained NeRF to synthesize abundant pseudo data. The model is later fintuned on the real data to further boost performance.
- **Ray representation**: We represent the ray with a simple concat scheme and propose to add noise during training to curb overfitting.

Empirically, to our knowledge, this is the *first NeLF* that runs round $30\times$ faster than NeRF while still being significantly better.

GitHub Code & Limitations

Thank you!

Code: https://github.com/snap-research/R2L

To Improve:

- Slow training
- Still not real-time on mobile devices ⇒ MobileR2L
- Needs a pretrained teacher? (Can be an upside too! Better teacher, better student)
- Better random sampling (accelerate training)
- ...

Real-Time Neural Light Field on Mobile Devices (CVPR 2023)

Junli Cao[1] Huan Wang[2] Pavlo Chemerys[1] Vladislav Shakhrai[1] Ju Hu[1]
Yun Fu[2] Denys Makoviichuk[1] Sergey Tulyakov[1] Jian Ren[1]

[1]Snap Inc. [2]Northeastern University

Project: https://snap-research.github.io/MobileR2L

Talk @Zhidx, 07/07/2023, Fri

Motivation

Run R2L in real-time on mobile devices? Problems:

- Slow! 3 seconds for rendering 200x200 on iPhone 13.
- 800×800 input, 800×800 output? Too large, OOM!

Learn a Page from Image Super-Resolution (SR)

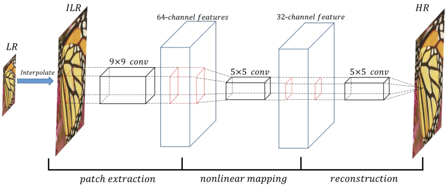
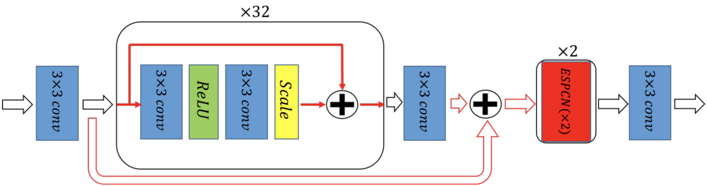


Figure 2: Sketch of the SRCNN architecture.



(e) EDSR

Figure: Illustration of SRCNN (Dong et al., 2014) (ECCV'14) and EDSR (Lim et al., 2017) (CVPRw'17). Src: (Yang et al., 2019). Want to be faster? Upsampling later.

Overview of MobileR2L

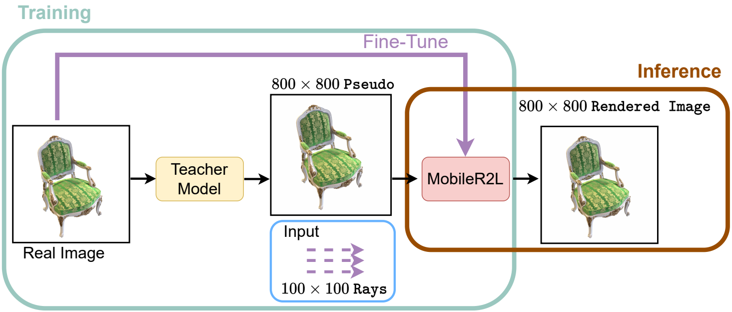


Figure: **Training and Inference Pipeline.** The training involves a teacher model to generate pseudo data, which is used to learn the MobileR2L. The teacher model, *e.g.*, NeRF, is trained on real images. Once we have the teacher model, we use it to generate pseudo images, *e.g.*, images with the resolution of 800×800, in addition to down-scaled rays, *e.g.*, rays with spatial size as 100×100, that share the same origin with the pseudo images to train the MobileR2L. After that, we use the real data to fine-tune MobileR2L. For inference, we directly forward the rays into the pre-trained MobileR2L to render images.

The Proposed Architecture of MobileR2L

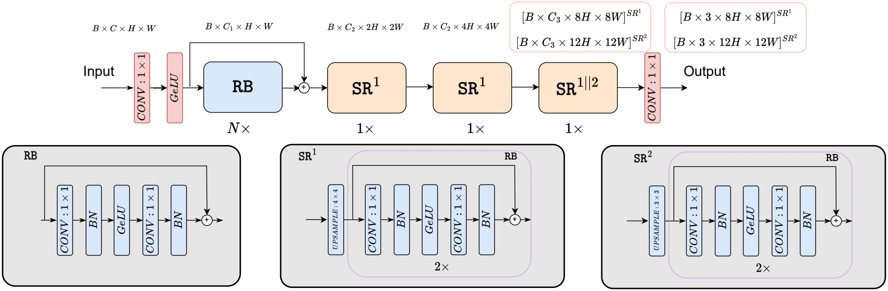


Figure: **Overview of Network.** The backbone includes residual blocks (RB) repeated that are 28 times ($N = 28$). Following the backbone, there are two types of super-resolution (SR) modules. The first SR module (SR^1) has kernel size 4×4 in the Transpose CONV layer that doubles the input H, W to $2H, 2W$, whereas the second SR module (SR^2) has kernel size 3×3, tripling the spatial size to $3H, 3W$.

Other Changes from R2L to MobileR2L

- Input Format:
  - R2L: Use MLP. Rays are mutually independent. No norm layers ⇒ Shuffled rays as input
  - MobileR2L: Use Conv. Rays are not mutually independent ⇒ Cropped patch as input (use BN for training stability)
- 1x1 MLP to Conv (27% faster! see below)
- ReLU to GeLU (PSNR slightly better, a minor change)

Table 4. **Analysis of Network Design.** For all the comparisons, we use the input tensor with the spatial size as 100×100 and render the image with spatial size. The latency (ms) is measured on iPhone 13 (iOS16) with models compiled with CoreMLTools [11]

	PSNR↑	SSIM↑	LPIPS↓	Latency↓
MLP	19.13	0.9759	0.6630	$\boxed{19.57}$
CONV2D	19.16	0.9759	0.6301	$\boxed{14.30}$
CONV2D + ReLU	26.82	0.9949	0.0282	16.20
CONV2D + GeLU	26.99	0.9949	0.0730	17.00
CONV2D + GeLU + BN	**27.18**	**0.9954**	**0.0259**	17.00

Experimental Results

Quantitative Comparison

Table: **Quantitative Comparison** on Synthetic 360° and Forward-facing. Our method obtains better results on the three metrics than NeRF for the two datasets. Compared with MoibleNeRF and SNeRG, we achieve better results on most of the metrics.

	Synthetic 360°			Forward-facing		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
NeRF (Mildenhall et al., 2020)	31.01	0.947	0.081	26.50	0.811	0.250
NeRF-Pytorch (Yen-Chen, 2020)	30.92	0.991	0.045	26.26	0.965	0.153
SNeRG (Hedman et al., 2021)	30.38	0.950	**0.050**	25.63	0.818	**0.183**
MoibleNeRF (Chen et al., 2023)	30.90	0.947	0.062	25.91	0.825	**0.183**
MobileR2L (Ours)	**31.34**	**0.993**	0.051	**26.15**	**0.966**	0.187
Our Teacher	33.09	0.961	0.052	26.85	0.827	0.226

Our Teacher here: Mip-NeRF (Barron et al., 2021) (Better teacher, better student).

Visual Comparison

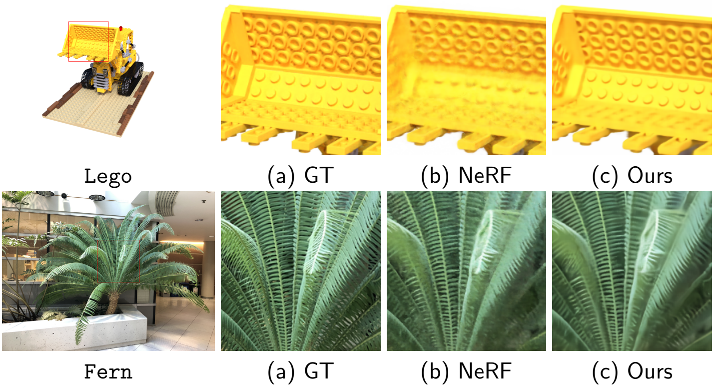


Figure: Visual comparison between our method and NeRF (Mildenhall et al., 2020) (trained via NeRF-Pytorch (Yen-Chen, 2020)) on the synthetic 360° Lego (size: 800×800×3) and real-world forward-facing scene `Fern` (size: 1008×756×3). *Best viewed in color*. Please refer to our webpage for more visual comparison results.

Zoom-in Comparison

Figure: Zoom-in comparisons. *Top row*: MobileNeRF (Chen et al., 2023). Results are obtained from the code and demo released by the authors. *Bottom row*: MobileR2L. Our approach renders high-quality images even for zoom-in views.

Storage (Representation Size) Comparison

Table: **Analysis of Storage (MB)** required for different rendering methods. Our method has a clear advantage over existing works with much less storage required, compared to MoibleNeRF (Chen et al., 2023) and SNeRG (Hedman et al., 2021).

	Synthetic 360°			Forward-facing		
	MoibleNeRF	SNeRG	Ours	MobileNeRF	SNeRG	Ours
Disk storage	125.8	86.8	**8.3**	201.5	337.3	**8.3**

15$\times$ $\sim$ **24**$\times$ **smaller!**

Latency Comparison on Mobile Devices

Table: **Analysis of Inference Speed.** Latency (ms) is obtained on iPhone with iOS 16. Following MobileNeRF (Chen et al., 2023), we use the notation $\frac{M}{N}$ to indicate that M out of N scenes in the Forward-facing dataset that can not run on devices. Specifically, MobileNeRF cannot render `Leaves` and `Orchids` in Forward-facing.

	Synthetic 360°		Forward-facing	
	MobileNeRF	Ours	MobileNeRF	Ours
iPhone 13	17.54	26.21	$27.15\frac{2}{8}$	18.04
iPhone 14	16.67	22.65	$20.98\frac{2}{8}$	16.48

Note, MobileNeRF is 1/12 Award Candidates in CVPR'23. MobileR2L presents an equally (if not more) promising alternative for efficient mobile neural rendering.

More Demo: Snap's AR Demo

- A fake lego bulldozer on a real desk.

Another Remark: Why MobileR2L is Special?

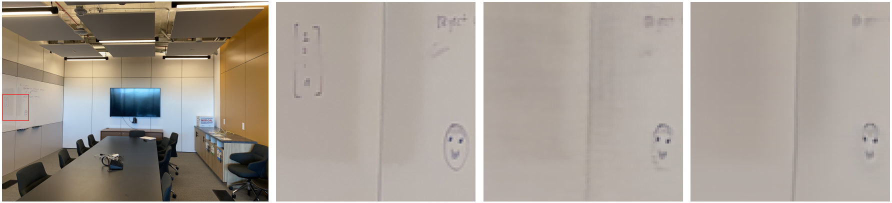
- It is the *first* to propose a 2D-3D hybrid architecture design for NVS, opening the new doors to marry the two areas together – just image so many techniques from the 2D image restoration area; many of them can be used here for improved quality in NeRF/NeLF.

Conclusion and Discussions

Conclusion

- We present MobileR2L, the very first NeLF that can run in real-time on mobile devices.
- MobileR2L is featured by a hybrid design of 3D light field network and 2D super-resolution architecture, enabling low-resolution (100×100) input with high-resolution (800×800) output.
- Empirically, MobileR2L achieves $\sim$**60fps** real-time rendering on iPhone14, while saving over 24$\times$ storage than MobileNeRF (one of the 12 Award Candidate papers in CVPR 2023).

Limitations / Future Work



Room (a) GT (b) NeRF (c) Ours

Figure: Visual comparison on the real-world scene Room. Both our model and NeRF fail to synthesize the whiteboard writings on the upper-left of the cutout patch.

- **Quality**. On some scenes, we still observe blurry rendered results, fine-grained details and textures are missing (see above).
- **Training speed**. R2L or MobileR2L training cost is $100 \sim 500\times$ of NeRF (so essentially, R2L or MobileR2L is trading inference cost with training cost).
- **Still need a teacher**. How to learn light field directly remains a question.

Thanks! Questions?

More materials:

- CVPR'23 Workshop: *Efficient Neural Networks: From Algorithm Design to Practical Mobile Deployments*, hosted by Snap. [Webpage]
- Slides can be found on my personal webpage.

Feel free to drop me a line (wang.huan@northeastern.edu) or raise a GitHub issue if you have any questions.

References

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

Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In *CVPR*, 2023.

Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *ECCV*, 2014.

Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *ICCV*, 2021.

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

Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *CVPR Workshop*, 2017.

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.

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