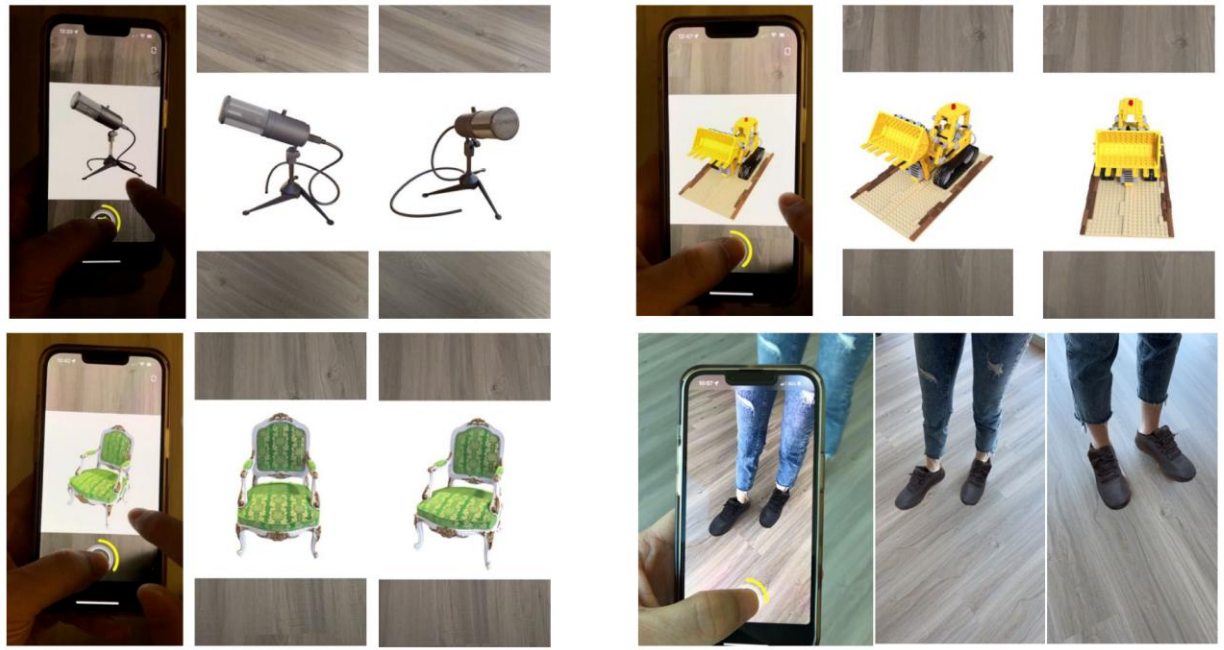
Real-Time Neural Light Field on Mobile Devices
(CVPR 2023)



한국과학기술연구원(KIST)

CVIPL 학생연구원 **김연욱**

한국과학기술연구원
Korea Institute of Science and Technology

Real-Time Neural Light Field on Mobile Devices

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

JUNLI CAO

팔로우

UCLA
ucla.edu의 이메일 확인됨
GenAI EfficientAI

제목	인용	연도
Real-time neural light field on mobile devices J Cao, H Wang, P Chemerys, V Shakhrai, J Hu, Y Fu, D Makoviichuk, ... Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern ...	75	2023
4Real: Towards Photorealistic 4D Scene Generation via Video Diffusion Models H Yu, C Wang, P Zhuang, W Menapace, A Siarohin, J Cao, LA Jeni, ... arXiv preprint arXiv:2406.07472	54	2024
Wonderland: Navigating 3d scenes from a single image H Liang, J Cao, V Goel, G Qian, S Korolev, D Terzopoulos, KN Plataniotis, ... Proceedings of the Computer Vision and Pattern Recognition Conference, 798-810	43	2025

Ju Hu

팔로우

다른 이름 ▸

Snap Inc.
snapchat.com의 이메일 확인됨

학술자료 인용 공개 액세스

제목	인용	연도
Efficientformer: Vision transformers at mobilenet speed Y Li, G Yuan, Y Wen, J Hu, G Evangelidis, S Tulyakov, Y Wang, J Ren Advances in Neural Information Processing Systems 35, 12934-12949	651	2022
Rethinking vision transformers for mobilenet size and speed Y Li, J Hu, Y Wen, G Evangelidis, K Salahi, Y Wang, S Tulyakov, J Ren Proceedings of the IEEE/CVF international conference on computer vision ...	368	2023

Jian Ren

팔로우

소속을 알 수 없음
rutgers.edu의 이메일 확인됨 - 홈페이지
computer vision machine learning high-performance computing

학술자료 인용 공개 액세스

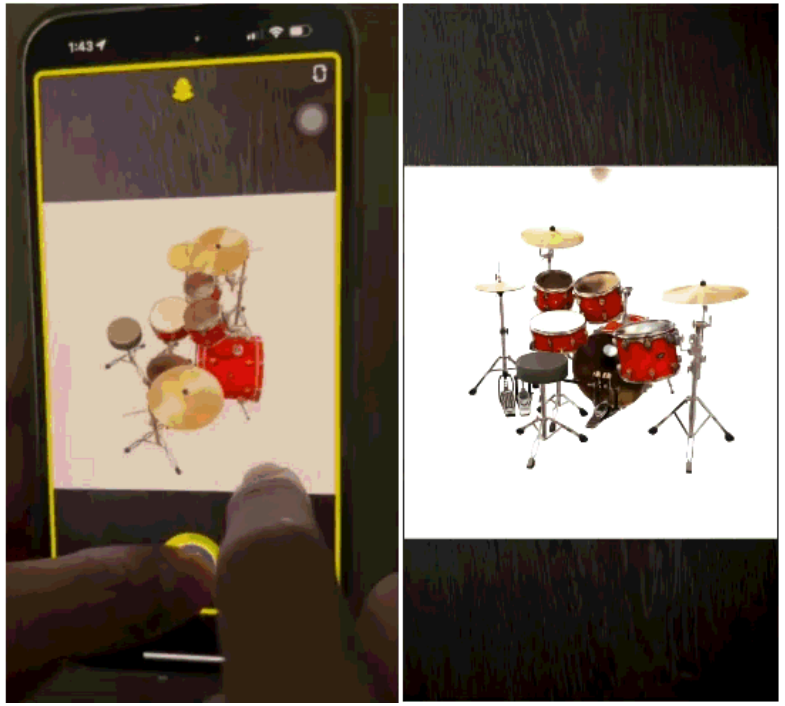
제목	인용	연도
Efficientformer: Vision transformers at mobilenet speed Y Li, G Yuan, Y Wen, J Hu, G Evangelidis, S Tulyakov, Y Wang, J Ren Advances in Neural Information Processing Systems 35, 12934-12949	653	2022
Magic123: One Image to High-Quality 3D Object Generation Using Both 2D and 3D Diffusion Priors G Qian, J Mai, A Hamdi, J Ren, A Siarohin, B Li, HY Lee, I Skorokhodov, ... arXiv preprint arXiv:2306.17843	443	2023

Contents

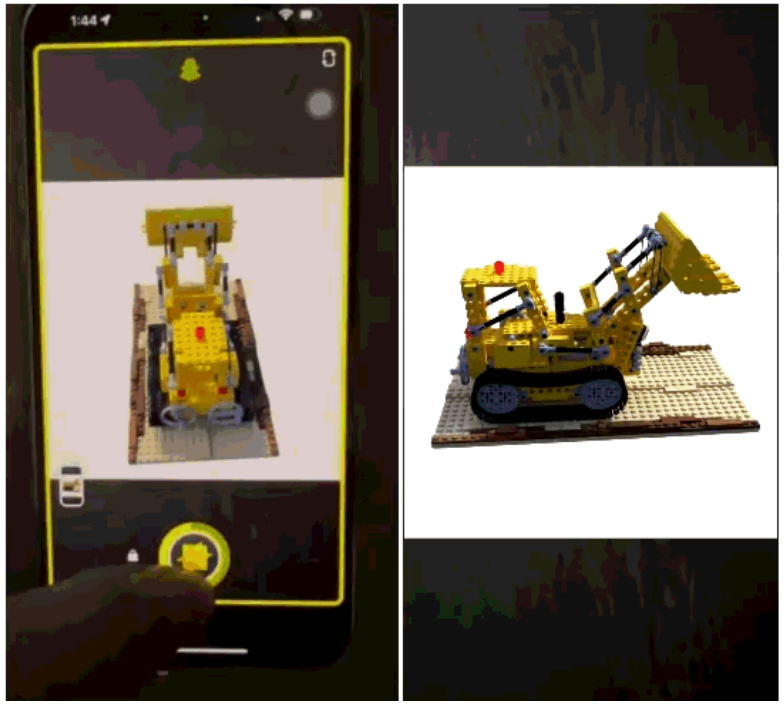
OverView

OverView

iPhone Rendering and try-on application Demo



Demo1

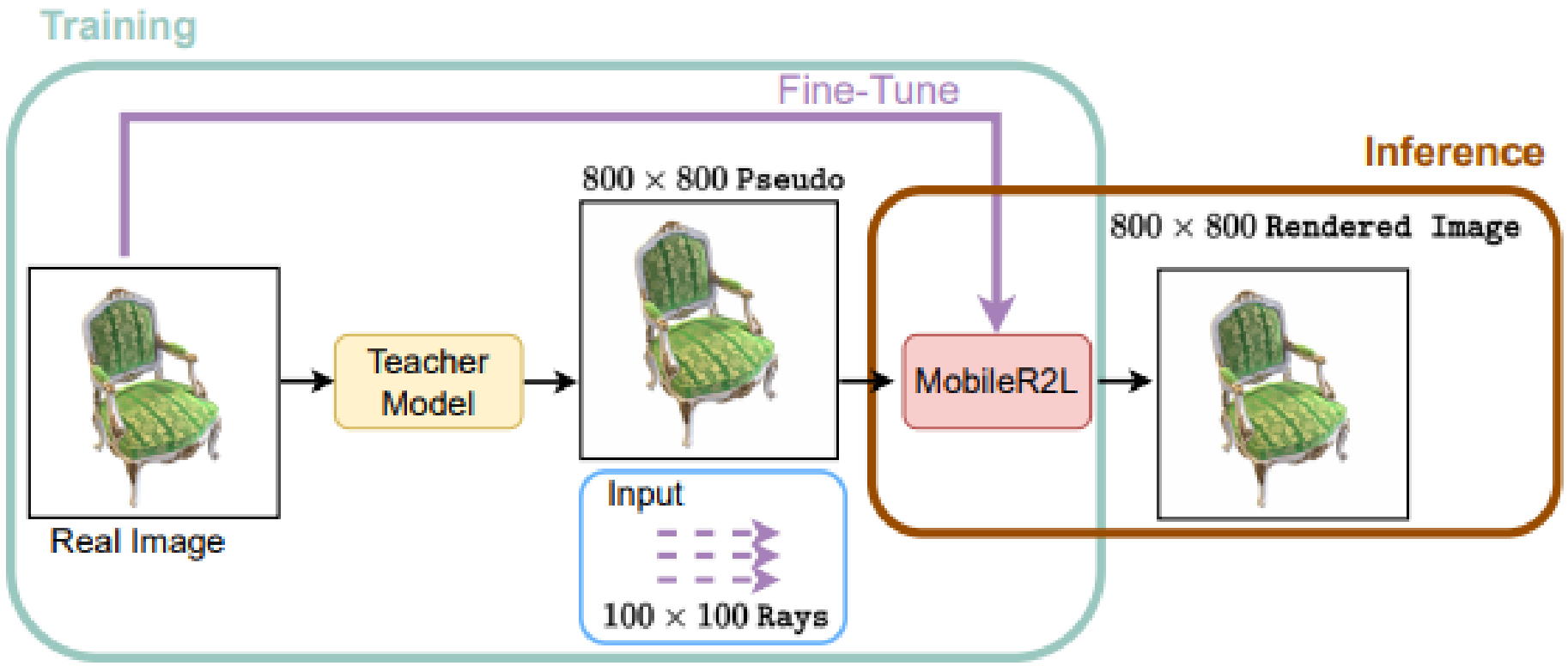


Demo2

Virtual Try-on Demo

OverView

Architecture Overview



Introduction

Introduction

- Recent efforts in **Neural Rendering Fields (NeRF)** have shown impressive results on **novel view synthesis**
- To be made **widely available**, this requires **devices** to run on, such as **mobile phones**
- limitations in computing, wireless connectivity, and hard drive capacity.

Introduction

- Unfortunately, the impressive image quality and capabilities of NeRF come with a price of **slow rendering speed**.
- Because **hundreds of points** need to be **sampled** along the ray to return the color of the quried **pixel**.

➔ Image를 렌더링하려면 number of image pixel * hundreds of points
➔ (mobile devise 및 일반 컴퓨터에는 없는) High-end GPU 필요

Introduction

MobileNeRF

- **Trade** rendering **speed** with **storage**.
- While showing promising **acceleration results**, their method requires **storage** for **texturing saving**.
- So, for **a single** real-world **scene** from the **forward-facing dataset**, MobileNeRF requires **201.5MB of storage**.

Introduction

Efficient NeRF Rendering

1. **Trades speed with space.**
   - precompute and cache scene representations
2. **Reduce the number of sampled points along the camera ray during redering.**
   - Sampling is the root cause of slow rendering speed.
   - Fewer sampled points lead to **performance degradation**, so usually introduce **depth**, or **mesh** to maintain the visual quality.
3. **Takes a “divide and conquer” strategy.**
   - **decomposes** the scene spatially into thousands of **small grids** or Voronoi diagrams
   - and **learns** each diagram with a **small network**.
   - **parallelism implementation** to obtain speedup.
4. **Achieve rendering efficiency by representing the scene with NeLF instead NeRF.**
   - NeLF **avoids** the **dense sampling** on camera ray, resulting in a **faster rendering speed**.

Introduction

Efficient NeRF Rendering

1. **Trades speed with space.**
   - precompute and cache scene representations
2. **Reduce the number of sampled points along the camera ray during redering.**
   - Sampling is the root cause of slow rendering speed.
   - Fewer sampled points lead to performance degradation, so usually introduce depth, or mesh to maintain the visual quality.
3. **Takes a “divide and conquer” strategy.**
   - decomposes the scene spatially into thousands of small grids or Voronoi diagrams
   - and learns each diagram with a small network.
   - parallelism implementation to obtain speedup.
4. **Achieve rendering efficiency by representing the scene with NeLF instead NeRF.**
   - NeLF avoids the dense sampling on camera ray, resulting in a faster rendering speed.

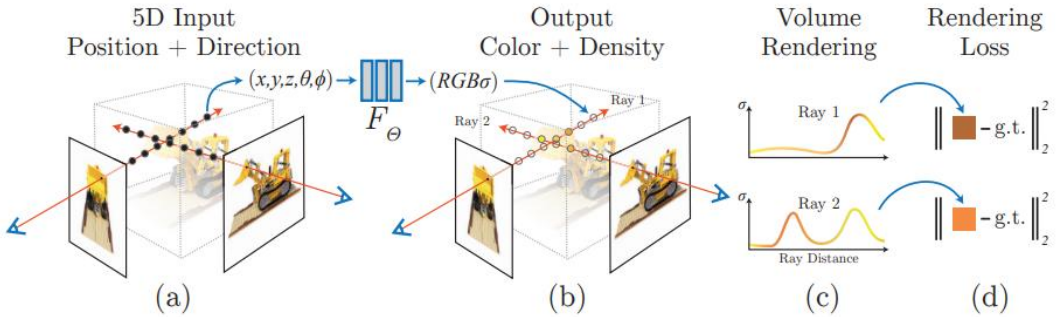
Preliminaries

Preliminaries

NeRF

- Maps the **5D coordinates**(location, view direction) to a **1D** volume **density** and **3D radiance**.
- the **position** and **direction** are encoded by **positional encoding** to enrich the input.

- $\hat{C}$: Color of a pixel
- F_Θ : MLP network
- **r** : Camera ray
- $\mathbf{r}(t_i) = \mathbf{o} + t_i\mathbf{d}$: Location of a point on the ray with origin **O** and direction **d**
- t_i : Euclidean distance
- $\delta_i = t_{i+1} - t_i$: Distance between two adjacent sampled points



$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i \cdot (1 - \exp(-\sigma_i\delta_i)) \cdot \mathbf{c}_i,$$

$$(\mathbf{c}_i, \sigma_i) = F_\Theta(\mathbf{r}(t_i), \mathbf{d}), \qquad (1)$$

$$T_i = \exp(-\sum_{j=1}^{i-1} \sigma_j\delta_j),$$

불투명도

투과율

Preliminaries

NeRF

- Maps the **5D coordinates**(location, view direction) to a **1D** volume **density** and **3D radiance**.
- the **position** and **direction** are encoded by **positional encoding** to enrich the input.

Problem:

- number of sampled points (**64(coarse) + 128(fine) = 196** in the original NeRF paper == Too large).
- the rendering computation for even a **single pixel** is **heavy**.

→ **distilling** the NeRF representation to **NeLF**. (**R2L**)

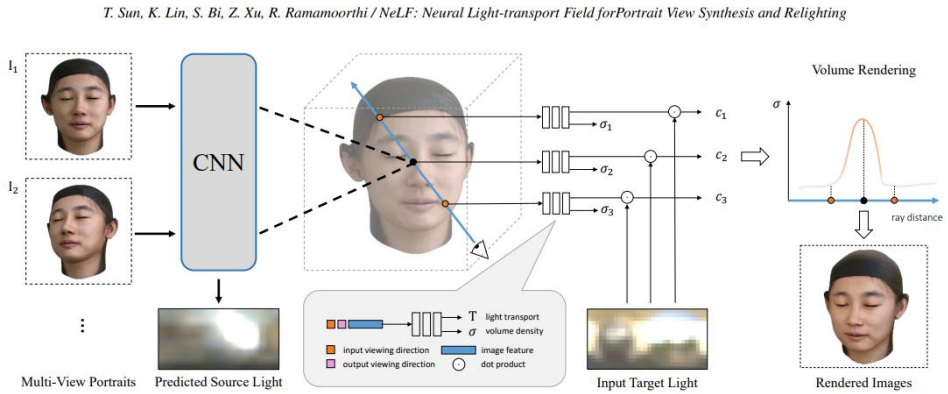
Preliminaries

NeLF(Neural Light Field)

Position과 View Direction을 입력으로 받아, 해당 광선이 닿는 픽셀의 **RGB**를 출력.(No density)

- **Sample points** along the ray just like NeRF.
- **Positional encoding** is also adopted.
- Differently, **concatenate** the **points** to **one vector.**
  - Ex) V = [PE(x1), PE(y1), PE(z1), PE(x2), PE(y1), PE(z1) …]
- Feed into a neural network to learn the **RGB**.

➔ No *alpha composition, one MLP for one pixel* → *fast rendering*



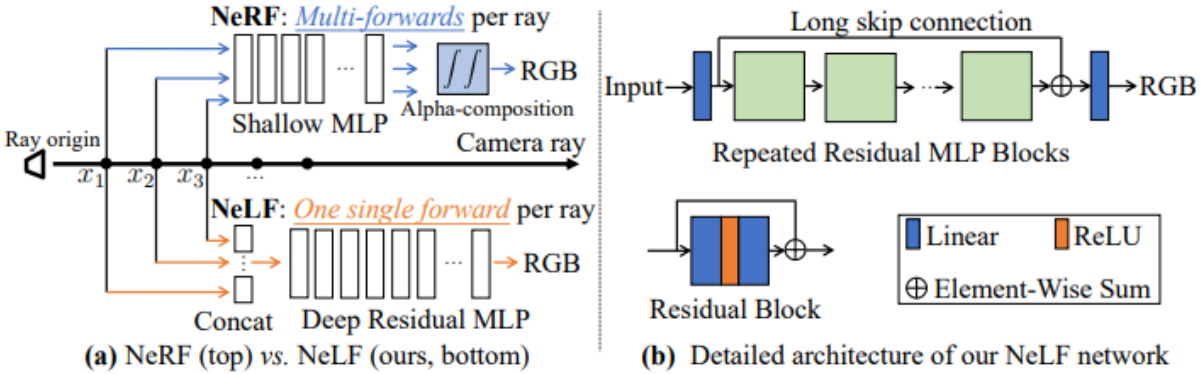
- However, NeLF representation is much **harder to learn** than NeRF
- so, R2L proposes an **88-layer deep ResMLP** (residual MLP) architecture.

Preliminaries

R2L

NeRF를 Teacher로, NeLF를 학생으로 Knowledge Distillation.

- During **training**, points are **randomly** sampled
  - Epoch1_samples = [0.3, 1.2, 3.1, …]
  - Epoch2_samples = [0.2, 1.6, 4.1, …]
- During **testing**, the points are **fixed**.
  - samples = [1, 2, 3, 4, …]



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

- R2L's Output is **RGB**, **no density**, and **no alpha-compositing.**
- → R2L much faster than NeRF in rendering.

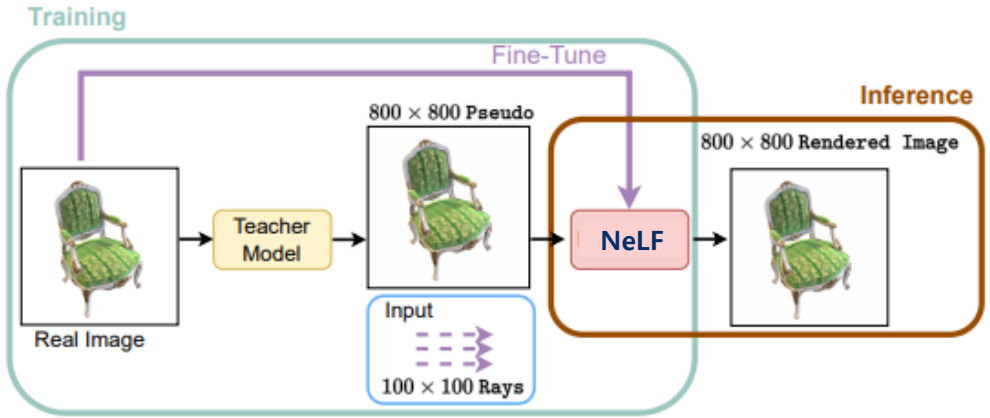
Preliminaries

R2L's Two stage of training

First stage.
- use a **pre-trained NeRF** model as a **teacher**.
- synthesize **triplets** (origin, direction, RGB) as **pseudo data**
- feed the pseudo data to train the deep ResMLP.

Second stage
- **Fine-tune** the R2L from the first stage on the **original data**

➔ can make the student model achieve **comparable performance** to the teacher NeRF model.



Preliminaries

R2L's Two stage of training

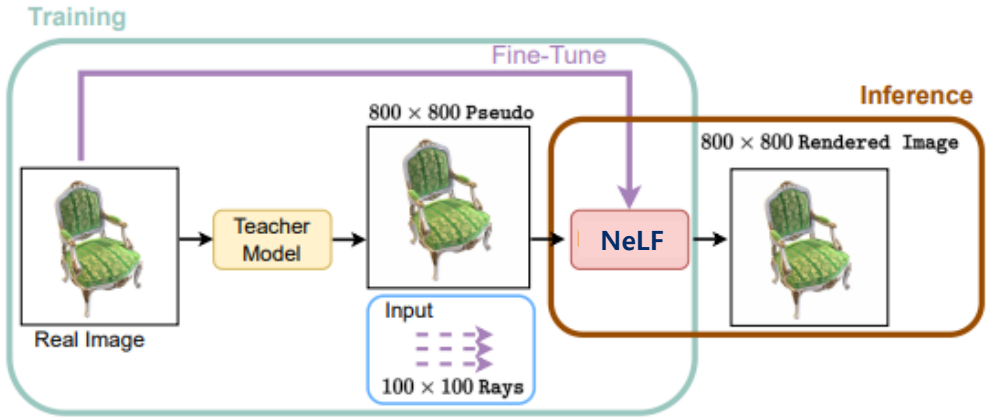
First stage.

- use a **pre-trained NeRF** model as a **teacher**.
- synthesize **triplets** (origin, direction, RGB) as **pseudo data**
  (origin, direction: NeLF 입력 / RGB: NeLF 출력)
- feed the pseudo data to train the deep ResMLP.

Second stage

- **Fine-tune** the R2L from the first stage on the **original data**

➔ can make the student model achieve **comparable performance** to the teacher NeRF model.



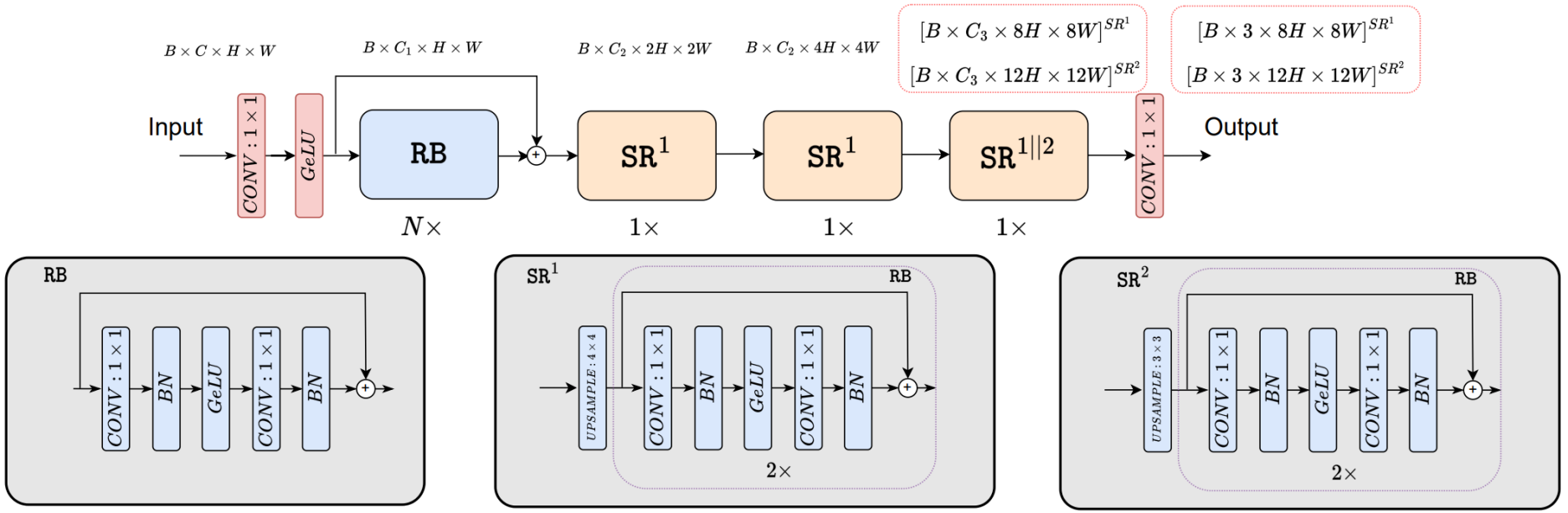
Method

Method – Network Architectures

Network Architectures

Two main part: efficient backbone (R2L) & Super-Resolution Modules

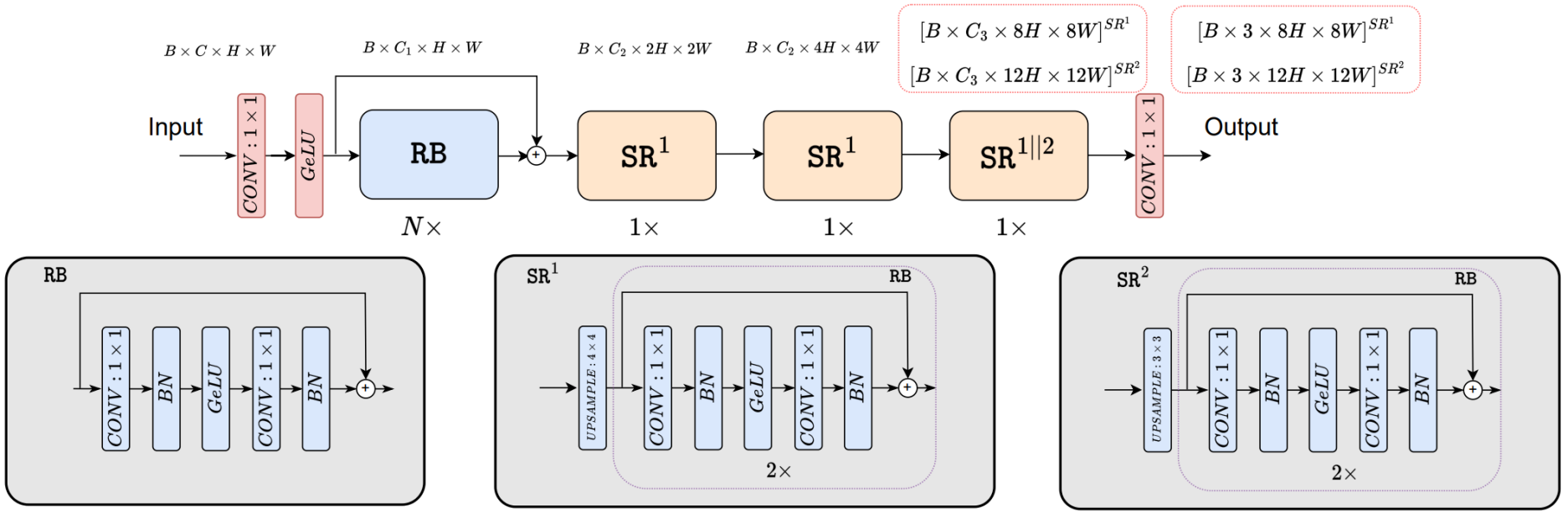
- $\mathbf{x} \in \mathbb{R}^{B,6,H,W}$: Input rays
- B : batch size
- C (=6) : concat of ray origin(x,y,z) and view direction(d_x, d_y, d_z) : 특점지점(0,360)에서 꼬이는 문제가 있어서 안정성을 위해 3채널 direction
- H, W : spatial size



Method – Efficient backbone

Efficient backbone

- The architecture of the backbone **follows** the design of **residual blocks** from **R2L.**
- Different from R2L, **MobileR2L** adopt the **CONV layer** instead of the **FC layer(MLP)** in each Residual Block.
- CONV layer has the **kernel** size and **stride** as **1**.

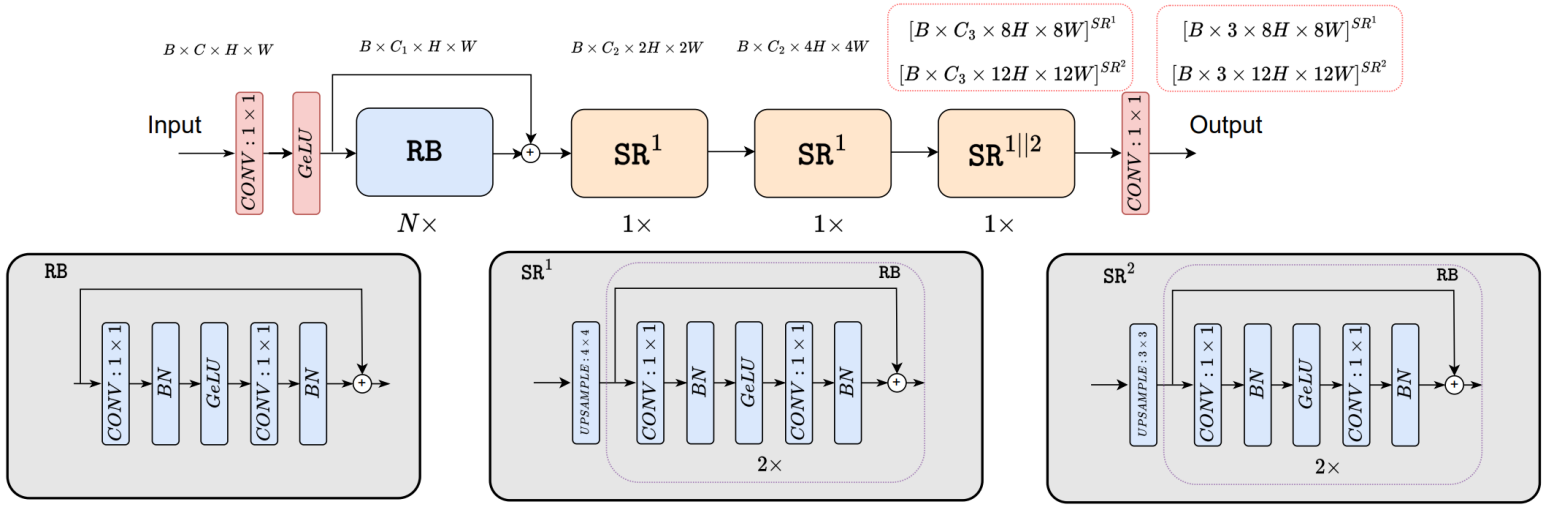


Method – Efficient backbone

Two main reasons for replacing FC with CONV

- First, the **CONV layer** is better **optimized** by **compilers** than the FC layer.
  - Under the **same** number of **parameters**, CONV 1 × 1 model **runs 27% faster** than FC layers model
  - Sliding Window 방식이라 미래 연산 예측이 가능해서 효율적이다.

- Second, **Reshape** and **Permute** operations are **required**
  - because **dimension** of output from **FC** is **not compatible** with **CONV** layer in the **super-resolution modules**.
  - Reshape or Permute operation might **not be hardware-friendly** on some **mobile devices**.



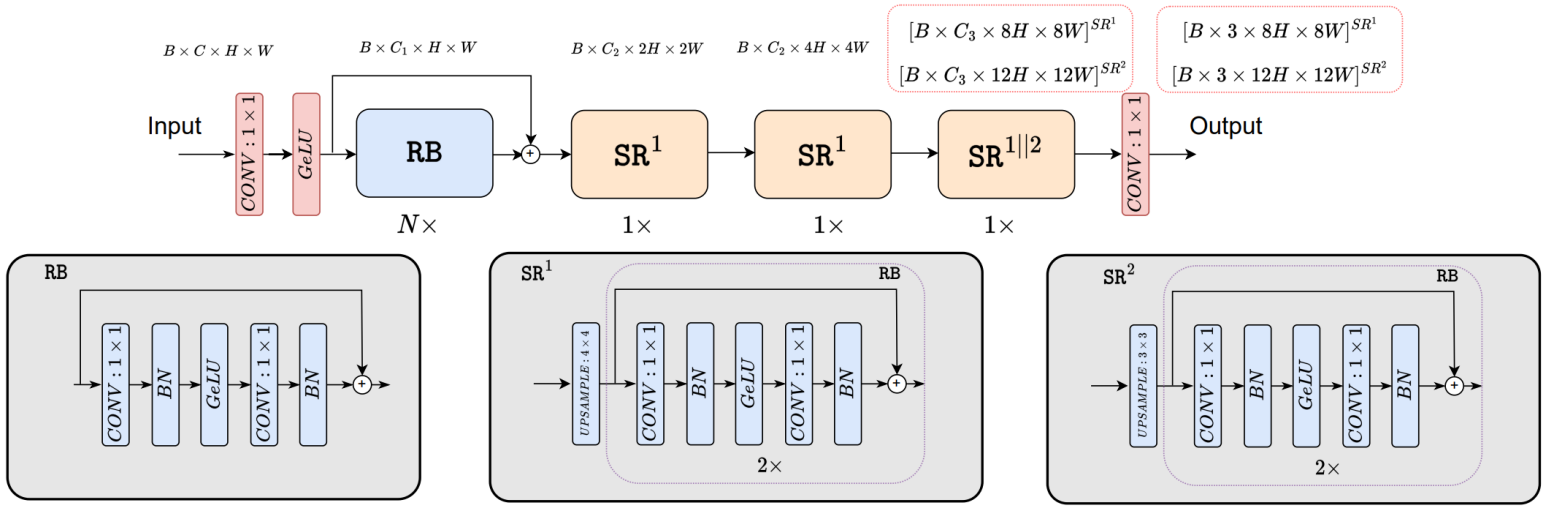
Method – Super-Resolution Modules

Super-Resolution Modules

고해상도 이미지를 rendering할 때 굉장히 큰 메모리를 필요로 한다.

Ex) 800x800 이미지 rendering을 위해서 640,000 rays가 필요하다.

800x800를 한 번에 하지 않고, **100x100** 이미지를 rendering한 후,
800x800으로 **Upsampling**하자.

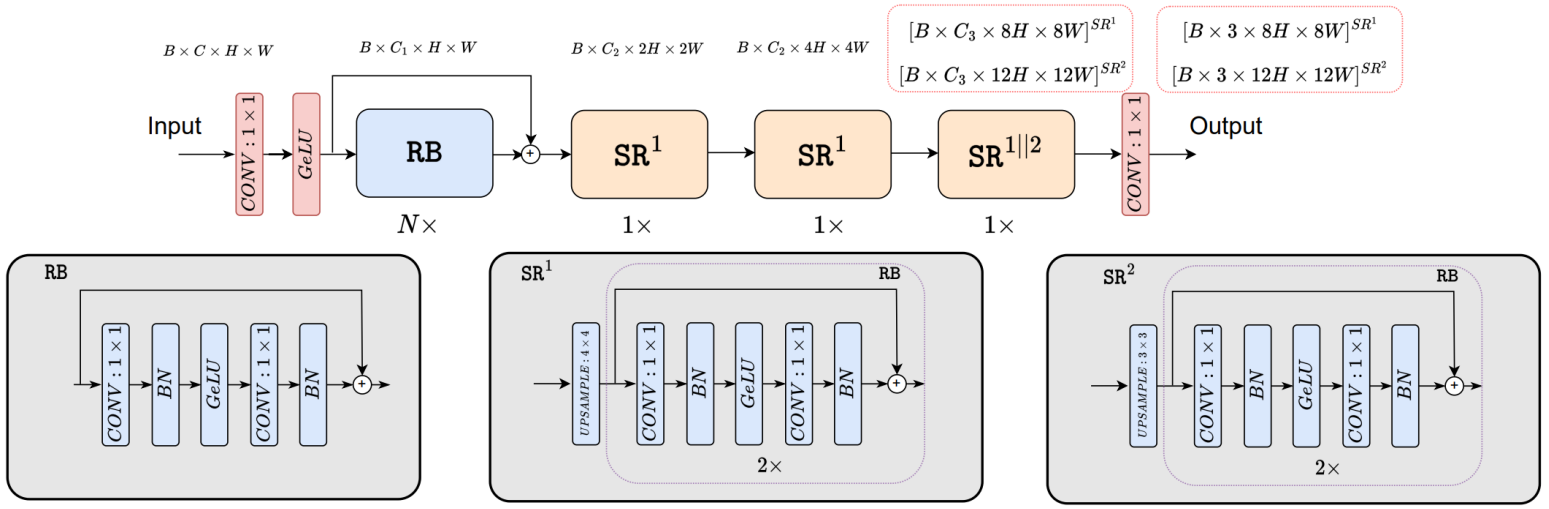


Super-Resolution Modules

SR Module includes **two** stacked **residual blocks**.

- The first block includes three CONV layers with **one as a 2D Transpose CONV layer** and **two CONV 1 × 1 layers**
- The second block includes **two CONV 1 × 1 layers**.

After the SR modules, another **CONV layer** followed by the **Sigmoid activation** to predict the final RGB color.



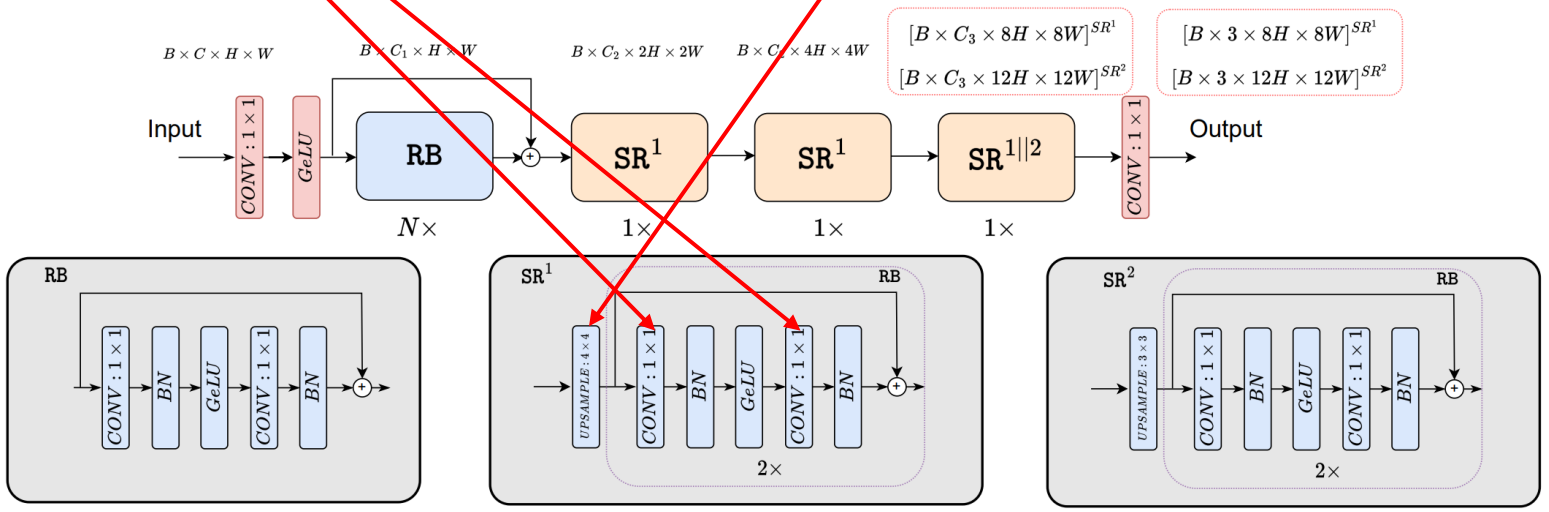
Method – Super-Resolution Modules

Super-Resolution Modules

SR Module includes **two** stacked **residual blocks**.

- The first block includes three CONV layers with **one as a 2D Transpose CONV layer** and **two CONV 1 × 1 layers**
- The second block includes **two CONV 1 × 1 layers**.

After the SR modules, another **CONV layer** followed by the **Sigmoid activation** to predict the final RGB color.



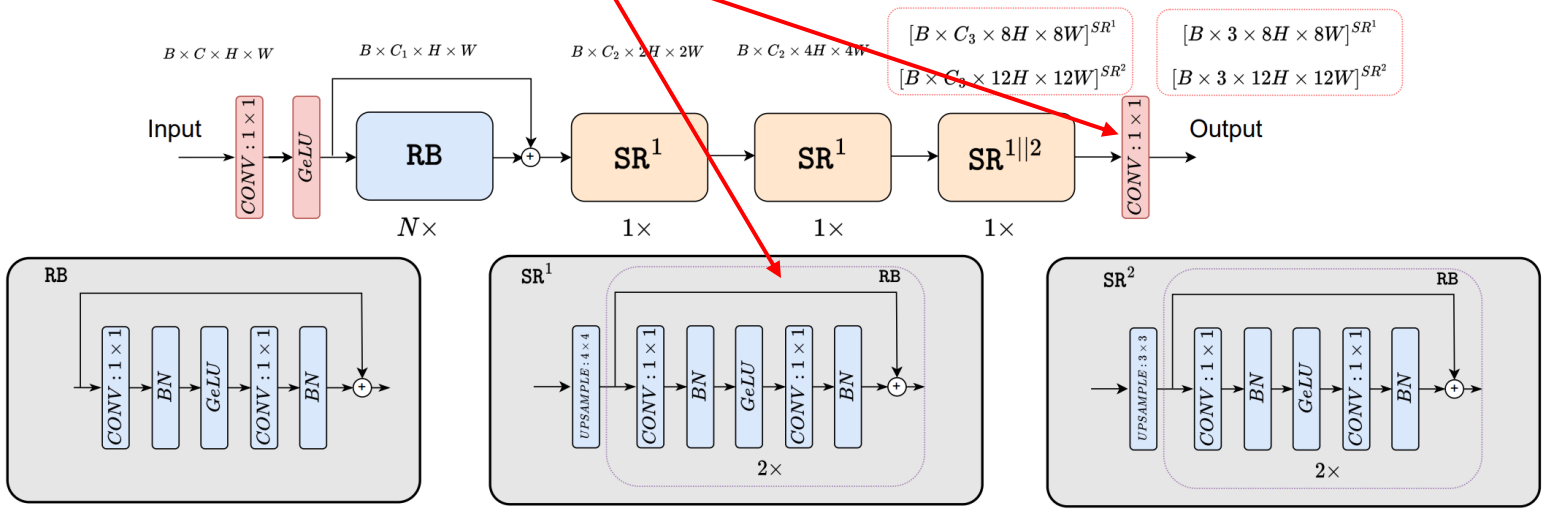
Method – Super-Resolution Modules

Super-Resolution Modules

SR Module includes **two** stacked **residual blocks**.

- The first block includes three CONV layers with **one as a 2D Transpose CONV layer** and **two CONV 1 × 1 layers**
- The second block includes **two CONV 1 × 1 layers**.

After the SR modules, another **CONV layer** followed by the **Sigmoid activation** to predict the final RGB color.

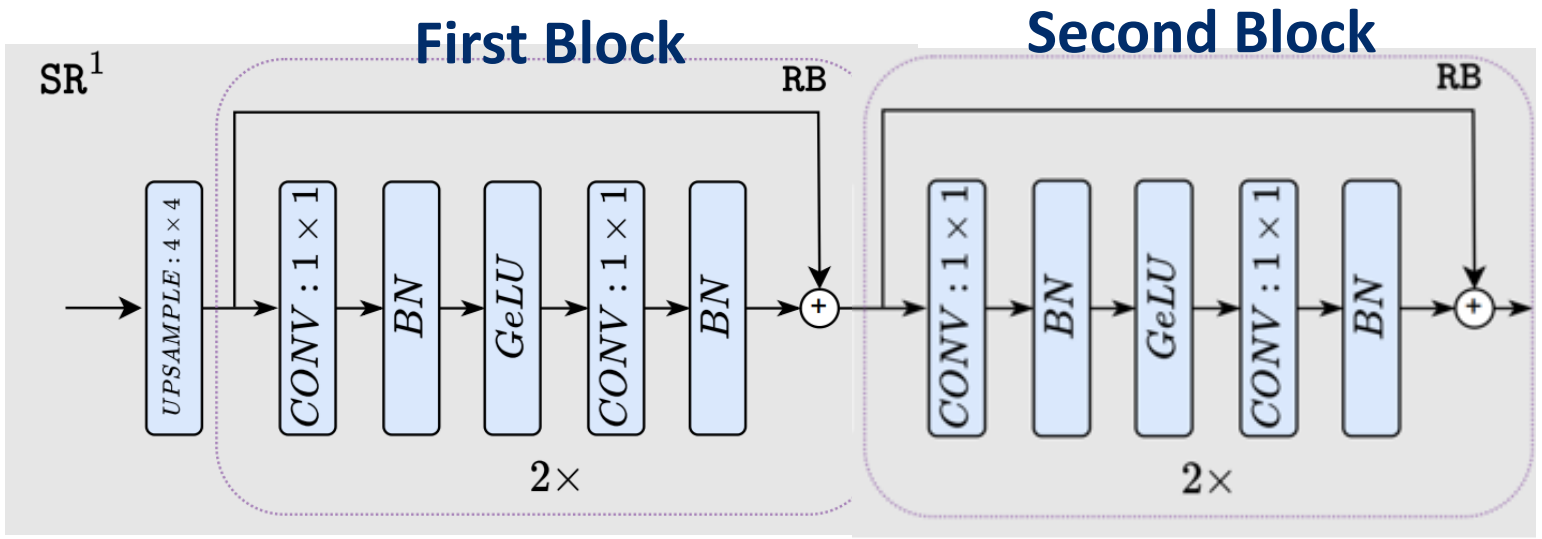


Super-Resolution Modules

SR Module includes **two** stacked **residual blocks**.

- The first block includes three CONV layers with **one as a 2D Transpose CONV layer** and **two CONV 1 × 1 layers**
- The second block includes **two CONV 1 × 1 layers**.

After the SR modules, another **CONV layer** followed by the **Sigmoid activation** to predict the final RGB color.



Experiment

Experiment

Quantitative Comparison

Table 1. **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 [33]	31.01	0.947	0.081	26.50	0.811	0.250
NeRF-Pytorch [44]	30.92	0.991	0.045	26.26	0.965	0.153
SNeRG [16]	30.38	0.950	**0.050**	25.63	0.818	**0.183**
MoibleNeRF [10]	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

성능 좋게 학습된 NeRF 선생님

Experiment

Quantitative Comparison

Table 7. Per-scene PSNR↑ comparison on the Synthetic 360° dataset between NeRF [33], NeRF-Pytorch [44], and our approach.

Method	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	Average
NeRF [33]	33.00	25.01	30.13	36.18	32.54	29.62	32.91	28.65	31.01
NeRF-Pytorch [44]	33.31	25.14	30.28	36.52	31.80	29.25	32.50	28.54	30.92
MobileR2L (Ours)	33.66	25.05	29.80	36.84	32.18	30.54	34.37	28.75	31.34

Table 8. Per-scene SSIM↑ comparison on the Synthetic 360° dataset between NeRF [33], NeRF-Pytorch [44], and our approach.

Method	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	Average
NeRF [33]	0.967	0.925	0.964	0.974	0.961	0.949	0.980	0.856	0.947
NeRF-Pytorch [44]	0.998	0.985	0.996	0.998	0.991	0.989	0.996	0.980	0.991
MobileR2L (Ours)	0.998	0.986	0.996	0.998	0.992	0.992	0.997	0.982	0.993

Table 9. Per-scene LPIPS↓ comparison on the Synthetic 360° dataset between NeRF [33], NeRF-Pytorch [44], and our approach.

Method	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	Average
NeRF [33]	0.046	0.091	0.044	0.121	0.050	0.063	0.028	0.206	0.081
NeRF-Pytorch [44]	0.025	0.066	0.023	0.022	0.029	0.035	0.021	0.144	0.045
MobileR2L (Ours)	0.027	0.083	0.025	0.026	0.043	0.029	0.012	0.162	0.051

Table 10. Per-scene PSNR↑ comparison on the Forward-facing dataset between NeRF [33], NeRF-Pytorch [44], and our approach.

Method	Room	Fern	Leaves	Fortress	Orchids	Flower	T-Rex	Horns	Average
NeRF [33]	32.70	25.17	20.92	31.16	20.36	27.40	26.80	27.45	26.50
NeRF-Pytorch [44]	32.10	24.80	20.50	31.20	20.45	27.50	26.48	27.05	26.26
MobileR2L (Ours)	32.09	24.39	20.52	30.81	20.06	27.61	26.71	27.01	26.15

Table 11. Per-scene SSIM↑ comparison on the Forward-facing dataset between NeRF [33], NeRF-Pytorch [44], and our approach.

Method	Room	Fern	Leaves	Fortress	Orchids	Flower	T-Rex	Horns	Average
NeRF [33]	0.948	0.792	0.690	0.881	0.641	0.827	0.880	0.828	0.811
NeRF-Pytorch [44]	0.989	0.976	0.921	0.995	0.920	0.968	0.972	0.983	0.965
MobileR2L (Ours)	0.995	0.973	0.923	0.995	0.916	0.971	0.973	0.982	0.966

Table 12. Per-scene LPIPS↓ comparison on the Forward-facing dataset between NeRF [33], NeRF-Pytorch [44], and our approach.

Method	Room	Fern	Leaves	Fortress	Orchids	Flower	T-Rex	Horns	Average
NeRF [33]	0.178	0.280	0.316	0.171	0.321	0.219	0.249	0.268	0.250
NeRF-Pytorch [44]	0.089	0.210	0.921	0.995	0.920	0.968	0.972	0.983	0.153
MobileR2L (Ours)	0.088	0.239	0.280	0.103	0.296	0.150	0.121	0.217	0.187

NeRF와 비교했을 때 성능이 준수하게 나온다

Experiment

Qualitative Comparison

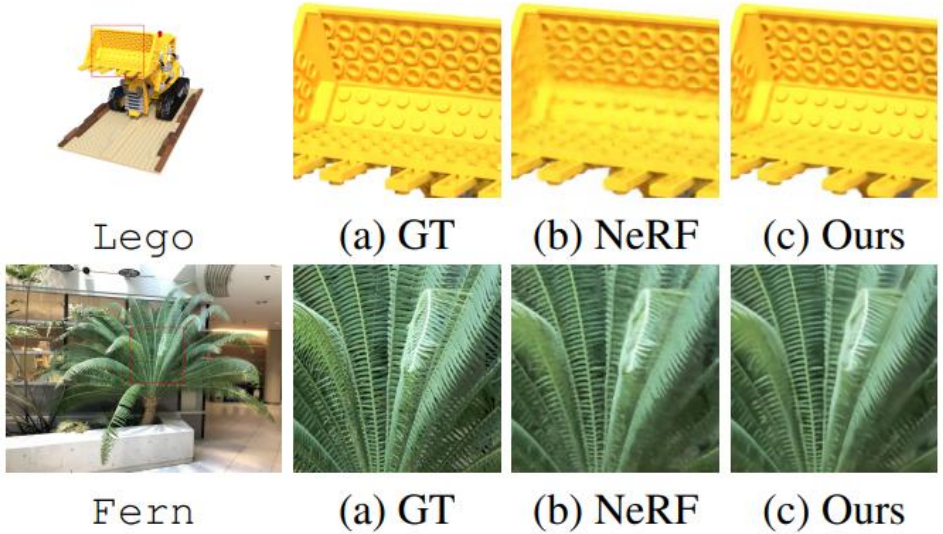


Figure 4. Visual comparison between our method and NeRF [33] (trained via NeRF-Pytorch [44]) on the synthetic 360° `Lego` (size: 800×800×3) and real-world forward-facing scene `Fern` (size: 1008×756×3). *Best viewed in color*. Please refer to our webpage for more visual comparison results.

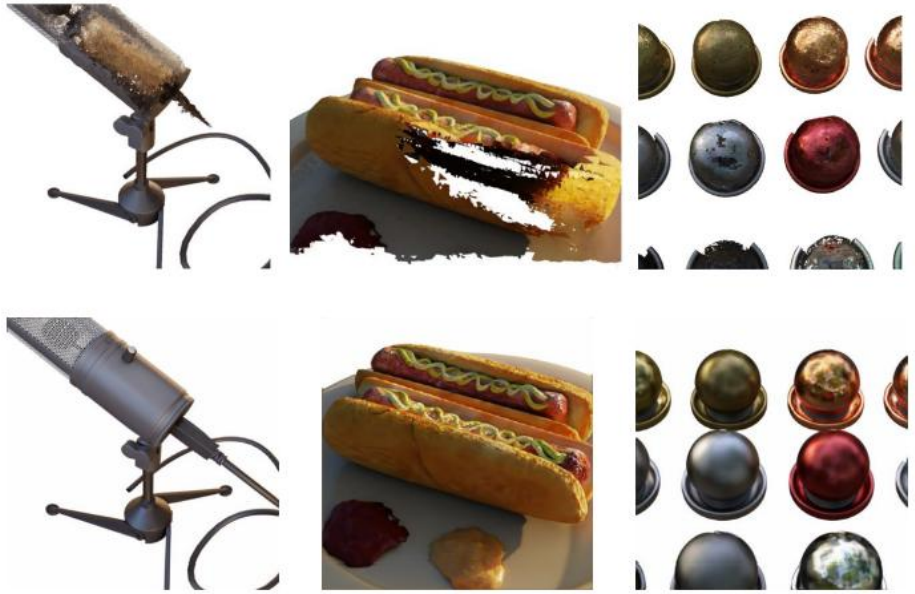


Figure 5. Zoom-in comparisons. *Top row*: MobileNeRF [10]. Results are obtained from the code and demo released by the authors. *Bottom row*: MobileR2L. Our approach renders high-quality images even for zoom-in views.

Experiment

Ablation Study

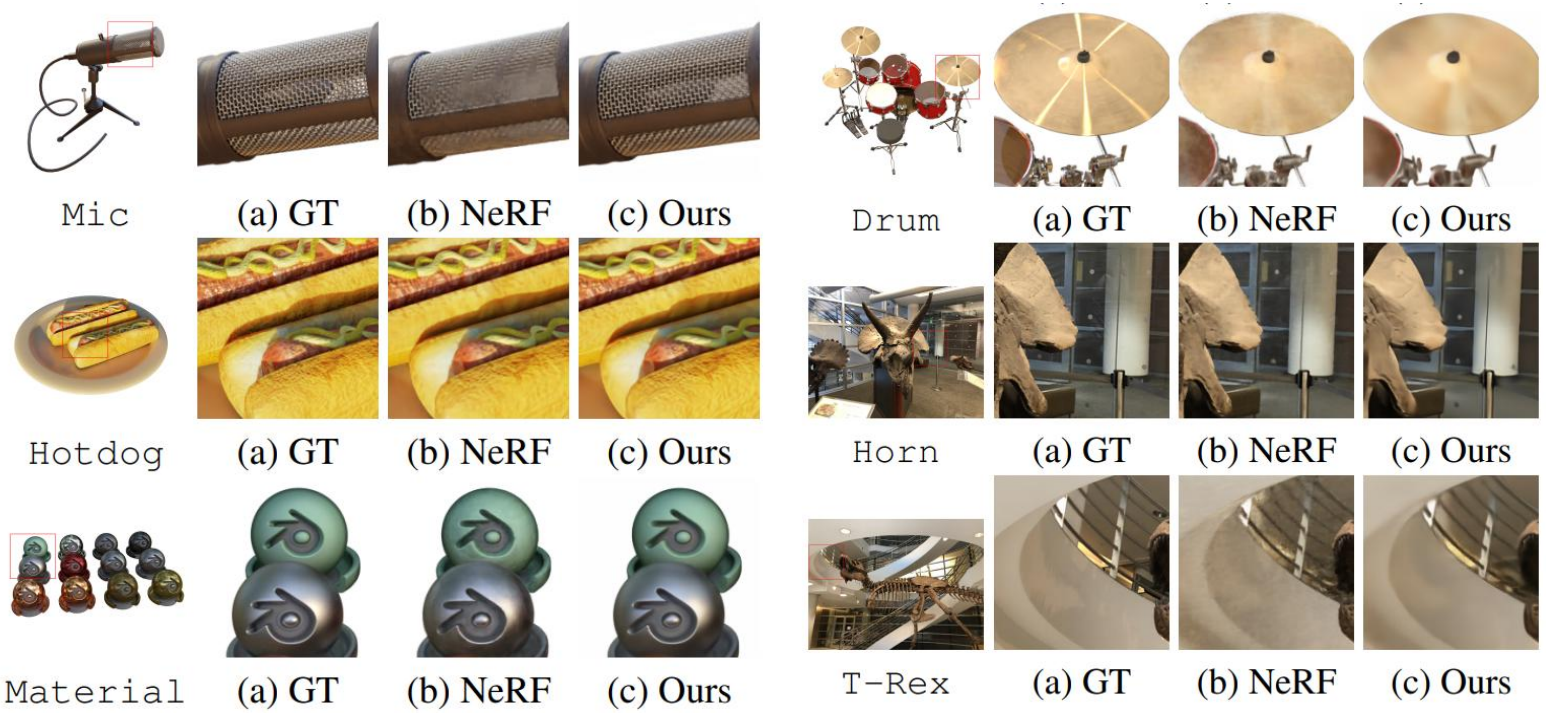


Figure 8. More visual comparisons between our method and NeRF [33] (trained via NeRF-Pytorch [44]) on the synthetic 360° (size: $800 \times 800 \times 3$) and real-world forward-facing scenes (size: $1008 \times 756 \times 3$). *Best viewed in color and zoomed in.*

Experiment

Table 2. **Analysis of Storage (MB)** required for different rendering methods. Our method has a clear advantage over existing works with much less storage required.

	Synthetic 360°			Forward-facing		
	MoibleNeRF [10]	SNeRG [16]	Ours	MobileNeRF [10]	SNeRG [16]	Ours
Disk storage	125.8	86.8	**8.3**	201.5	337.3	**8.3**

Table 3. **Analysis of Inference Speed.** Latency (ms) is obtained on iPhone with iOS 16. Follwing MoibleNeRF [10], 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 can not render `Leaves` and `Orchids` in Forward-facing.

	Synthetic 360°		Forward-facing	
	MobileNeRF [10]	Ours	MobileNeRF [10]	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

8개의 장면 중에서 2개는 렌더링에 실패
이유: 용량 문제로 실행 불가 (컴파일 에러)

...

Experiment

Ablation Study

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	19.57
CONV2D	19.16	0.9759	0.6301	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

Table 5. **Analysis of the spatial size of the input, usage of teacher model, and ray presentation**. Besides image quality metrics, we show the number of parameters for each model and the latency when running on iPhone 13.

	Params	PSNR↑	SSIM↑	LPIPS↓	Latency↓
50×50 - NeRF Teacher	3.9M	30.40	0.9965	0.0686	13.04
100×100 - NeRF Teacher	3.9M	30.65	0.9966	0.0668	26.21
200×200 - NeRF Teacher	3.9M	-	-	-	73.76
800×800 - w/o SR	3.9M	-	-	-	Error
100×100 - MipNeRF Teacher	3.9M	30.83	0.0997	0.0564	26.21
100×100 - MipNeRF Teacher, $K16$, $L10$	4.1M	30.90	0.9968	0.0583	31.05
100×100 - MipNeRF Teacher, $K16$, $L10$, D100	6.8M	31.37	0.9972	0.0470	44.52

Experiment

Figure 7. 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.

Fail to synthesize details

Experiment

Power usage

Table 13. Power usage on the Synthetic 360° dataset and the Forward-facing dataset between MobileNeRF [10] and our approach.

Synthetic 360°	Chair	Drums	Ficus	Hotdog	Lego	Material	Mic	Ship	Avg↓
MobileNeRF	1.7W	1.6W	1.4W	4.3W	2.6W	2.1W	1.2W	7.3W	2.8W
Ours	2.5W	2.5W	2.5W	2.5W	2.5W	2.5W	2.5W	2.5W	2.5W
Forward-facing	**Fern**	**Flower**	**Fortress**	**Horn**	**Leaves**	**Orchids**	**Room**	**Trex**	**Avg↓**
MobileNeRF	12.3W	13.0W	12.4W	12.8W	15.1W	14.5W	12.8W	12.9W	13.2W
Ours	1.7W	1.7W	1.7W	1.7W	1.7W	1.7W	1.7W	1.7W	1.7W

→Complexity에 따라 Polygo
수가 다르다.

→일정한 Power

Conclusion

Conclusion

Limitation

- Uses 10K pseudo images generated by the teacher NeRF model
  to train the student model.
  (**more images** used to train the **teacher** NeRF, resulting in a **longer training time**)

- Fails to generate some high-frequency details in the images.

Conclusion

Contribution

- Propose a efficient Mobile-Friendly Architecture that introduces 1x1 convolution layers and Super-Resolution modules.
- Achieves real-time rendering on mobile devices (e.g., ~18ms on iPhone 13), e nabling interactive 3D applications.
- Achieve a significant storage reduction, compressing the model size to approx imately 8.3MB, which is much smaller than MobileNeRF.