Neural Fields

AI602: Recent Advances in Deep Learning

Lecture 8

Slide made by

Sihyun Yu, Subin Kim, Jaehyun Nam

KAIST AI

Introduction: Neural Fields

- **Neural fields** (or implicit neural representations):
  - Alternative of conventional grid-based representations (e.g., image pixels)
  - Interpret signal as a neural network of a coordinate-to-output mapping
  - New paradigm for representing a complex, continuous signal

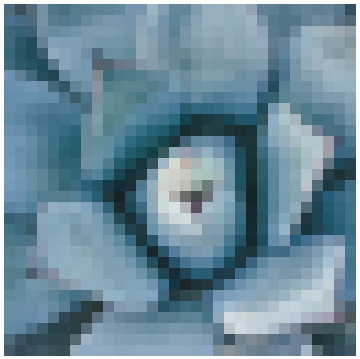
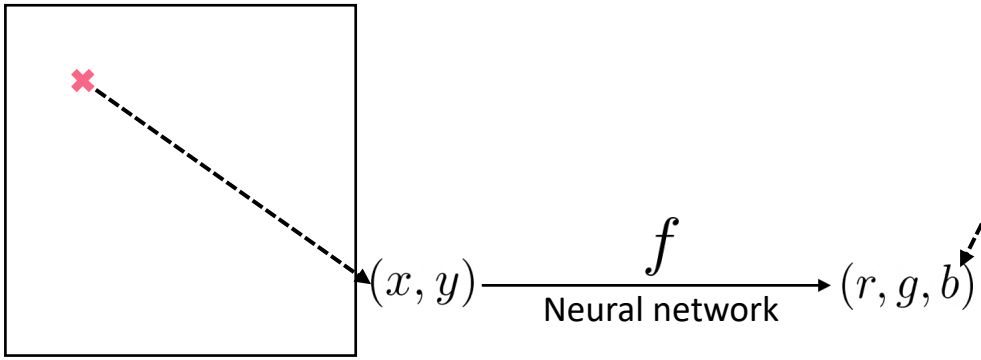


Image pixels



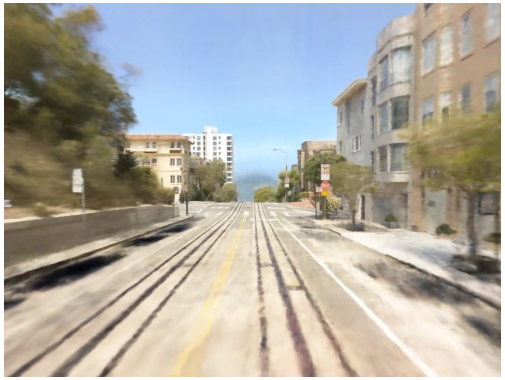


Table of Contents

Part 1. Basics

- Architectures: SIREN, FFN
- Neural radiance fields (NeRF)

Part 2. Advanced Topics

- Techniques for faster training
- Techniques for real-time inference
- Training at real-world scenarios (e.g., deformable, unconstrained)

Part 3. Beyond Neural Fields of a Single Signal

- Generalization of neural fields on multiple signals

Table of Contents

Part 1. Basics

- Architectures: SIREN, FFN
- Neural radiance fields (NeRF)

Part 2. Advanced Topics

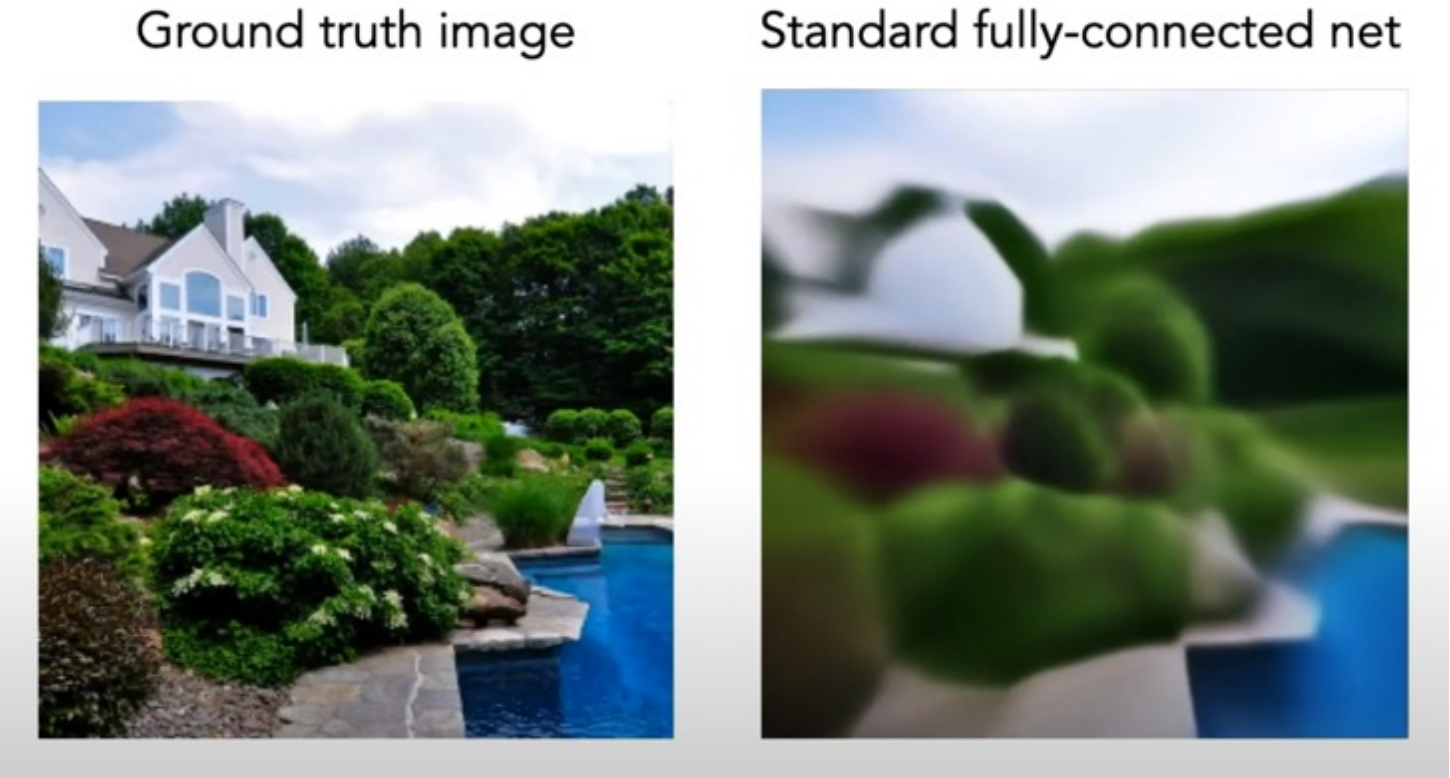
- Techniques for faster training
- Techniques for real-time inference
- Training at real-world scenarios (e.g., deformable, unconstrained)

Part 3. Beyond Neural Fields of a Single Signal

- Generalization of neural fields on multiple signals

SIREN: Implicit Neural Representations with Periodic Activations [Sitzmann et al., 2020]

- "**Spectral bias**" of neural networks [Rahaman et al., 2018]:
  - Conventional neural networks (e.g., ReLU MLP) often miss high-frequency details
  - Reconstruction often suffer from blurry artifacts
- **Question:** Is their any architectural prior that mitigates this issue?

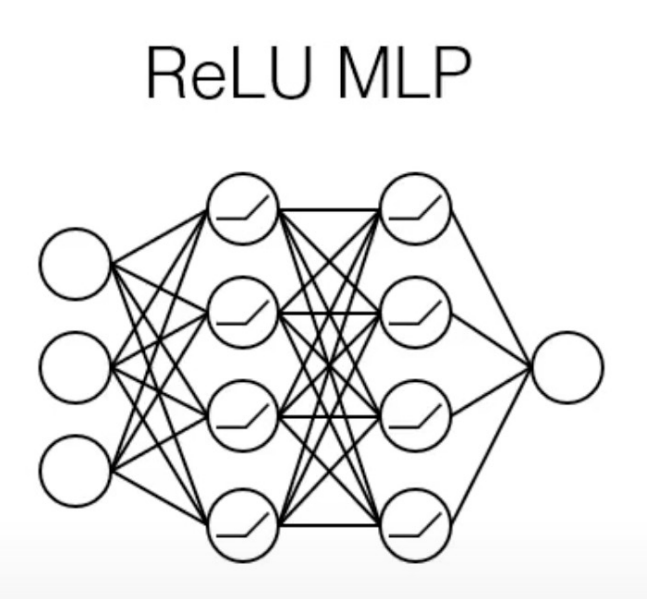


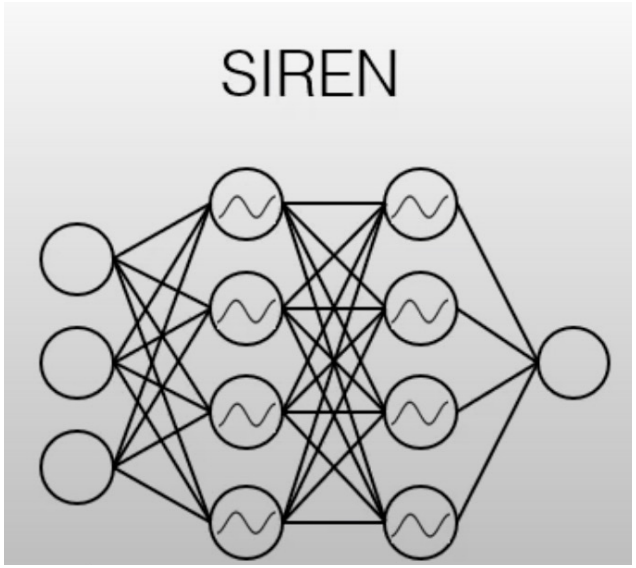
* source : Tancik et al., Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains, NeurIPS 2020

- **Question:** Is their any architectural prior that mitigates this issue?
- **SIREN:** Sinusoidal activations is one of the good priors
  - In particular, high-frequency sine function is the key

$$f_\theta(\mathbf{x}) = \mathbf{W}_n(\phi_{n-1} \circ \phi_{n-2} \circ \cdots \phi_0)(\mathbf{x}) + \mathbf{b}_n$$

$$\mathbf{x}_i \mapsto \phi_i(\mathbf{x}_i) = \sin(\omega_0(\mathbf{W}_i\mathbf{x}_i + \mathbf{b}_i)), \quad \omega_0 \gg 1$$

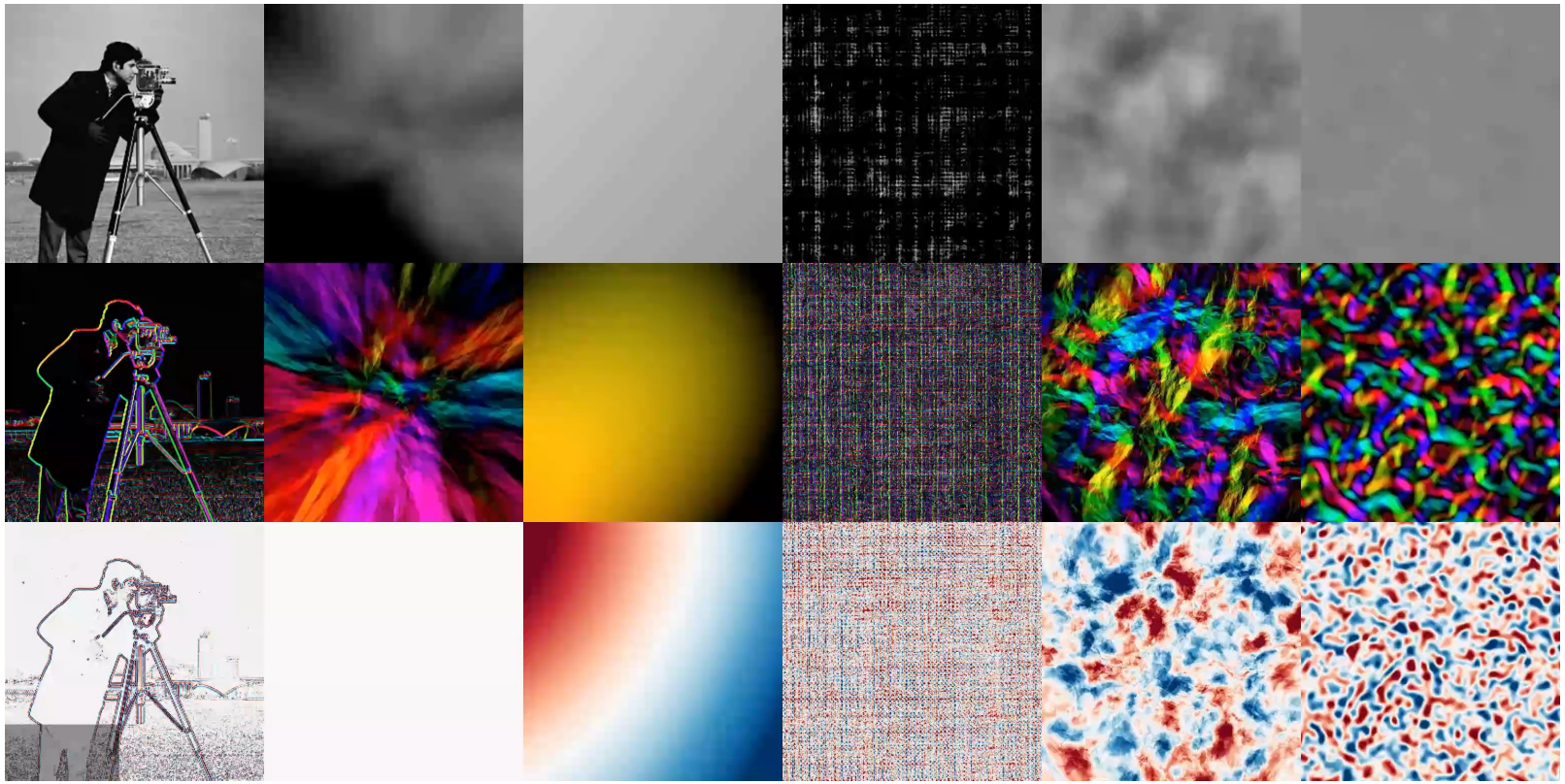




* source : Sitzmann et. al., Implicit Neural Representations with Periodic Activation Functions, NeurIPS 2020

SIREN: Implicit Neural Representations with Periodic Activations [Sitzmann et al., 2020]

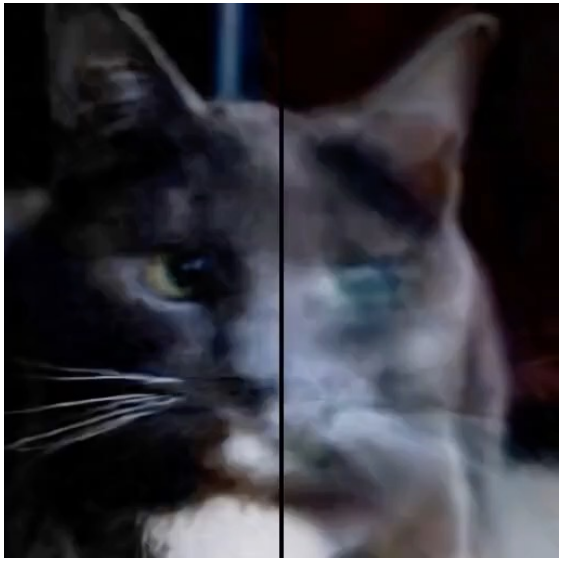
- **SIREN can fit complex signal with a high quality in a signal-agnostic manner**
  - Unlike conventional neural network architectures that suffer from spectral bias



* source : Sitzmann et. al., Implicit Neural Representations with Periodic Activation Functions, NeurIPS 2020

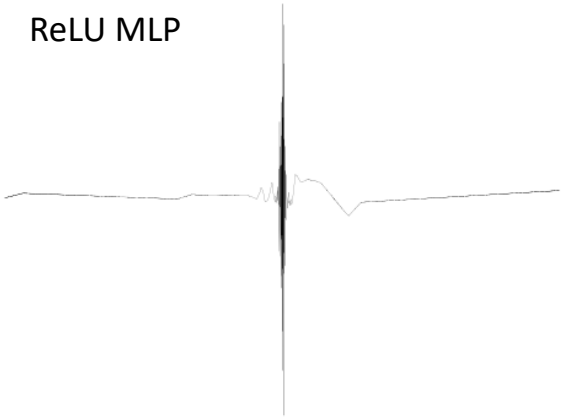
SIREN: Implicit Neural Representations with Periodic Activations [Sitzmann et al., 2020]

- **SIREN can fit complex signal with a high quality in a signal-agnostic manner**
  - Unlike conventional neural network architectures that suffer from spectral bias



Ground Truth

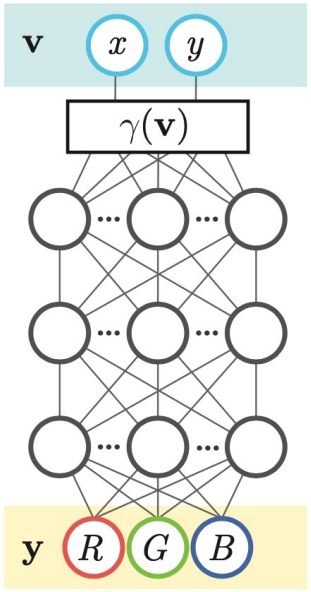
ReLU MLP



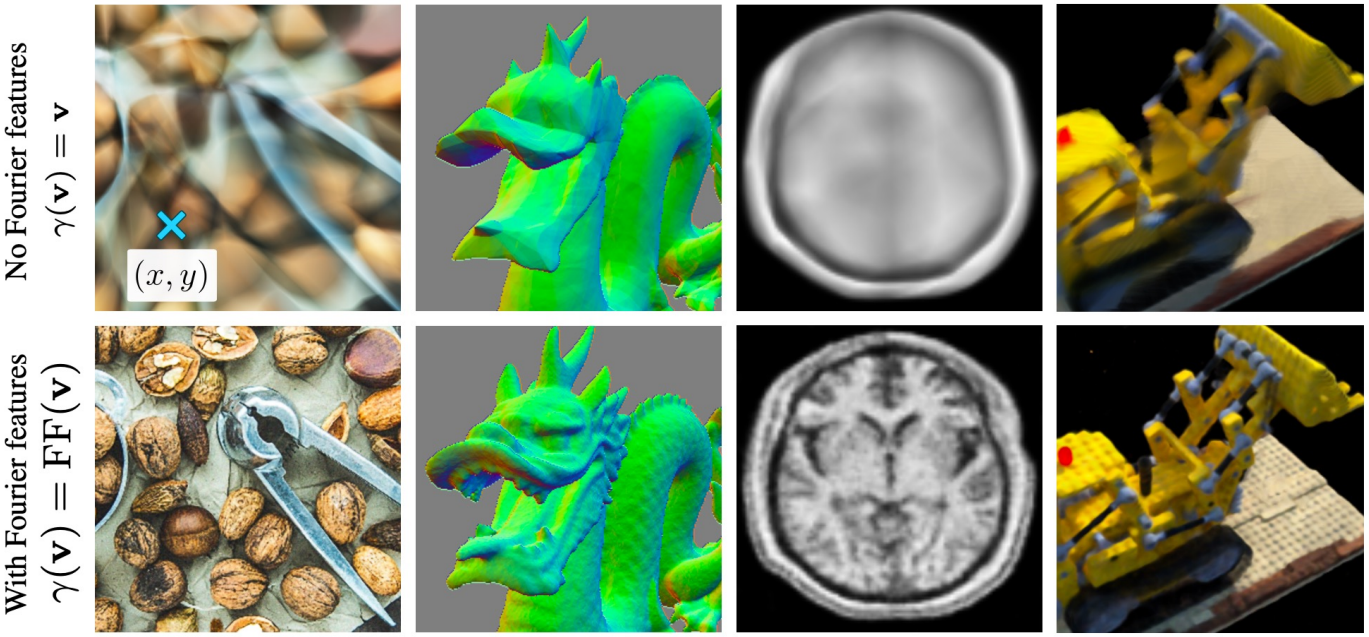
SIREN

* source : Sitzmann et. al., Implicit Neural Representations with Periodic Activation Functions, NeurIPS 2020

- **FFN: Random Fourier feature mapping before neural networks is sufficient**
  - Rather than applying sine activations to all layers
  - Feature matrix is not trained (deterministic feature mapping)



(a) Coordinate-based MLP



(b) Image regression $(x,y) \rightarrow$ RGB

(c) 3D shape regression $(x,y,z) \rightarrow$ occupancy

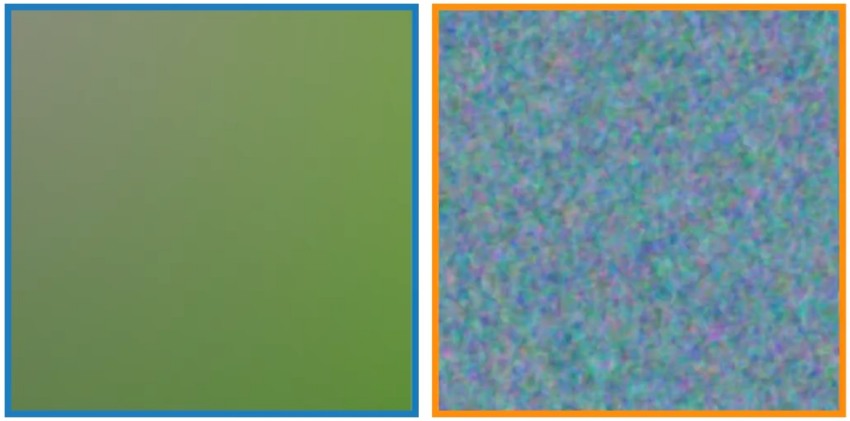
(d) MRI reconstruction $(x,y,z) \rightarrow$ density

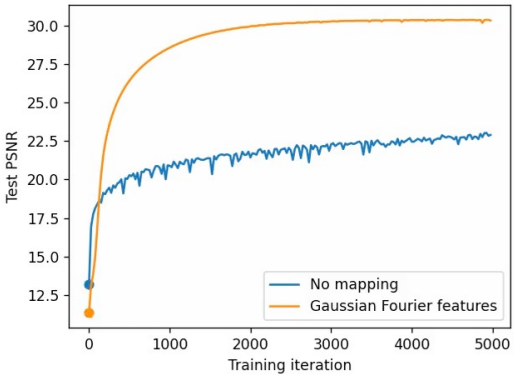
(e) Inverse rendering $(x,y,z) \rightarrow$ RGB, density

$$\gamma(\mathbf{v}) = [\cos(2\pi \mathbf{B}\mathbf{v}), \sin(2\pi \mathbf{B}\mathbf{v})]^{\mathrm{T}}, \quad \mathbf{B}_{ij} \sim \mathcal{N}(0, \sigma^2)$$

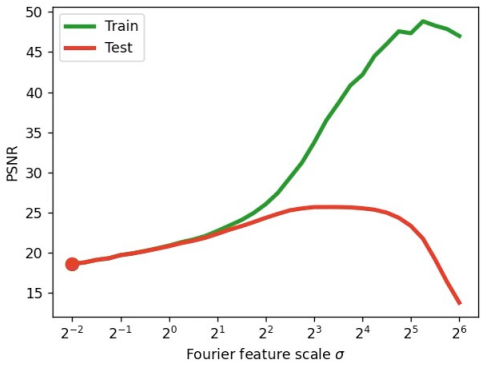
FFN: Fourier Feature Networks Learn High Frequency Functions [Tancik et al., 2020]

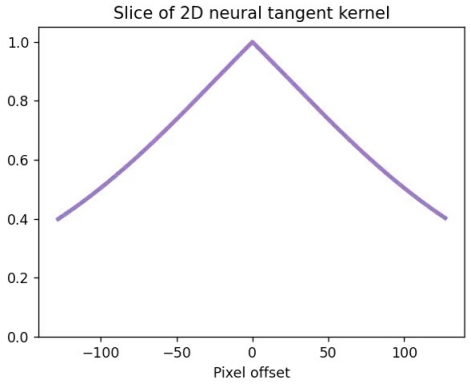
- **FFN allows for a rapid convergence**
  - Not only achieving a high-quality representations, convergence speed is also fast





- **Appropriate choice of frequencies leads to a good generalization**
  - e.g., spatial interpolation at the inference (i.e., super-resolution)





* source : Tancik et al., Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains, NeurIPS 2020

Table of Contents

Part 1. Basics

- Architectures: SIREN, FFN
- Neural radiance fields (NeRF)

Part 2. Advanced Topics

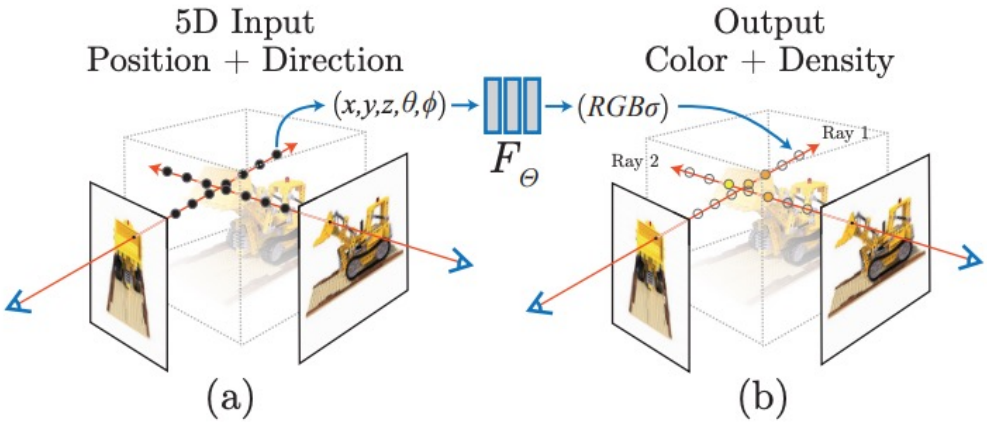
- Techniques for faster training
- Techniques for real-time inference
- Training at real-world scenarios (e.g., deformable, unconstrained)

Part 3. Beyond Neural Fields of a Single Signal

- Generalization of neural fields on multiple signals

NeRF: Neural Radiance Fields for View Synthesis [Mindelhall et al., 2020]

- **NeRF** represents a 3D scene (signal) into the representation
  - Input: (x, y, z, θ, ϕ) or expressed as $(\boldsymbol{x}, \boldsymbol{d})$ (coordinate position and camera pose)
  - Output: σ, R, G, B (density and RGB value)



σ: occupancy at coordinate (x, y, z)
= probability whether object exist at (x, y, z)

RGB: emitted color of (x, y, z) from (θ, ϕ)

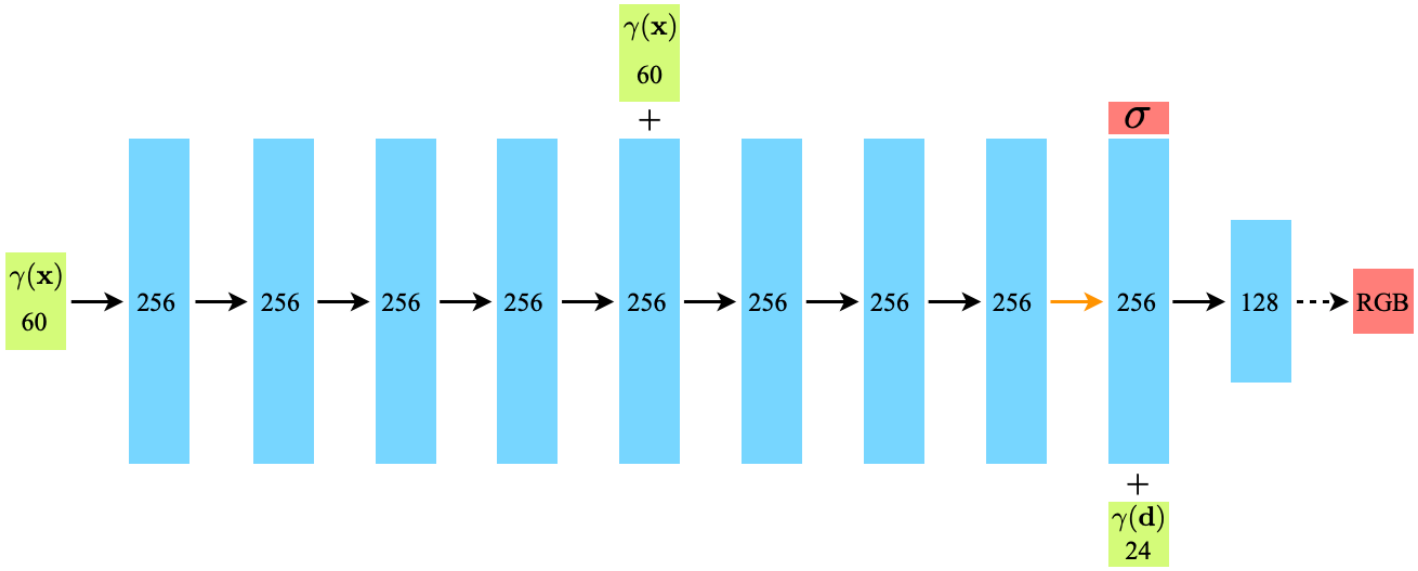
3D space coordinate

Camera ray

- Effective at novel view synthesis
  - e.g.) Synthesis realistic scene at unseen random view only with 100 different views

* source: Mindenhall et al., NeRF: Representing Scenes as Neural Raiance Fields for Novel View Synthesis, ECCV 2020

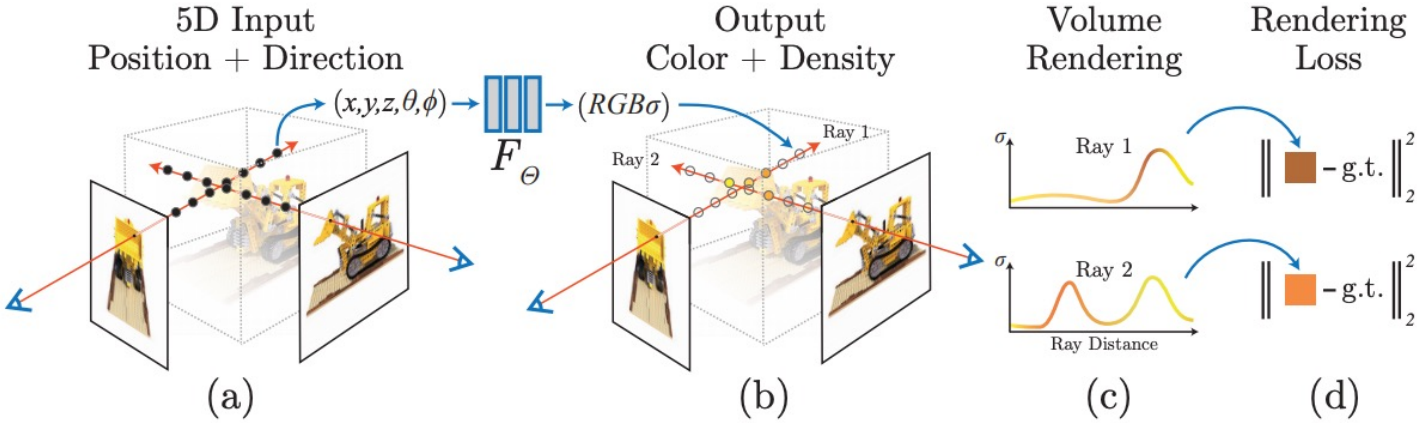
- **Architecture: follows MLP architecture**
  - $\sigma(\boldsymbol{x})$: occupancy at coordinate $\boldsymbol{x}$ (only depend on $\boldsymbol{x}$)
  - $\mathbf{c}(\boldsymbol{x}, \boldsymbol{d})$: emitted color (RGB value) of $\boldsymbol{x}$ when the view direction is $\boldsymbol{d}$



$$F_\Theta : (\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma)$$

* source : Tancik et al., Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains, NeurIPS 2020

- **Objective: follows traditional volume rendering formula**
  - Evaluated as Monte Carlo sampling



$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t),\mathbf{d})dt\,, \text{ where } T(t) = \exp\left(-\int_{t_n}^{t}\sigma(\mathbf{r}(s))ds\right)$$

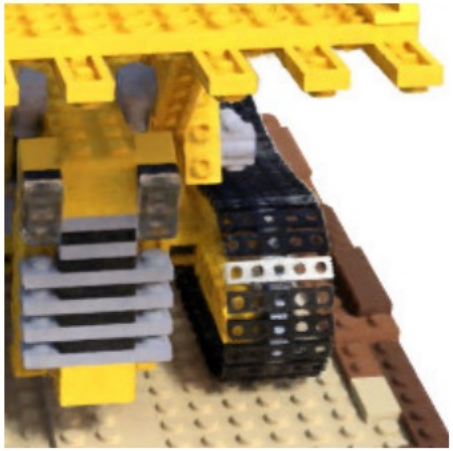
$\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$: camera ray

$$t_i \sim \mathcal{U}\left[t_n + \frac{i-1}{N}(t_f - t_n),\ t_n + \frac{i}{N}(t_f - t_n)\right]$$

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i\delta_i))\mathbf{c}_i\,, \text{ where } T_i = \exp\left(-\sum_{j=1}^{i-1}\sigma_j\delta_j\right)$$

* source : Tancik et al., Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains, NeurIPS 2020

- $\gamma(x)$: **positional embedding that includes periodic functions**
  - Serves as a similar role to random Fourier feature or sinusoidal activations
  - Enables to capture high-frequency details



$$F_\theta : (\mathrm{x}, \mathrm{d}) \rightarrow (\mathrm{c}, \sigma) \qquad F_\theta : \big(\gamma(\mathrm{x}), \gamma(\mathrm{d})\big) \rightarrow (\mathrm{c}, \sigma)$$

$$\gamma(p) = \big(\sin\big(2^0\pi p\big), \cos\big(2^0\pi p\big), \cdots, \sin\big(2^{L-1}\pi p\big), \cos\big(2^{L-1}\pi p\big)\big)$$

* source : Tancik et al., Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains, NeurIPS 2020

- **NeRF outperforms existing strong baselines on 3D rendering**
  - While capturing out high-frequency details (e.g., flour texture) well

Method	Diffuse Synthetic 360° [41] PSNR↑	SSIM↑	LPIPS↓	Realistic Synthetic 360° PSNR↑	SSIM↑	LPIPS↓	Real Forward-Facing [28] PSNR↑	SSIM↑	LPIPS↓
SRN [42]	33.20	0.963	0.073	22.26	0.846	0.170	22.84	0.668	0.378
NV [24]	29.62	0.929	0.099	26.05	0.893	0.160	-	-	-
LLFF [28]	34.38	0.985	0.048	24.88	0.911	0.114	24.13	0.798	**0.212**
Ours	**40.15**	**0.991**	**0.023**	**31.01**	**0.947**	**0.081**	**26.50**	**0.811**	0.250

* source : Tancik et al., Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains, NeurIPS 2020

Table of Contents

Part 1. Basics

- Architectures: SIREN, FFN
- Neural radiance fields (NeRF)

Part 2. Advanced Topics

- Techniques for faster training
- Techniques for real-time inference
- Training at real-world scenarios (e.g., deformable, unconstrained)

Part 3. Beyond Neural Fields of a Single Signal

- Generalization of neural fields on multiple signals

Direct Voxel Grid Optimization: fast convergence speed and high-quality [Sun et al., 2022]

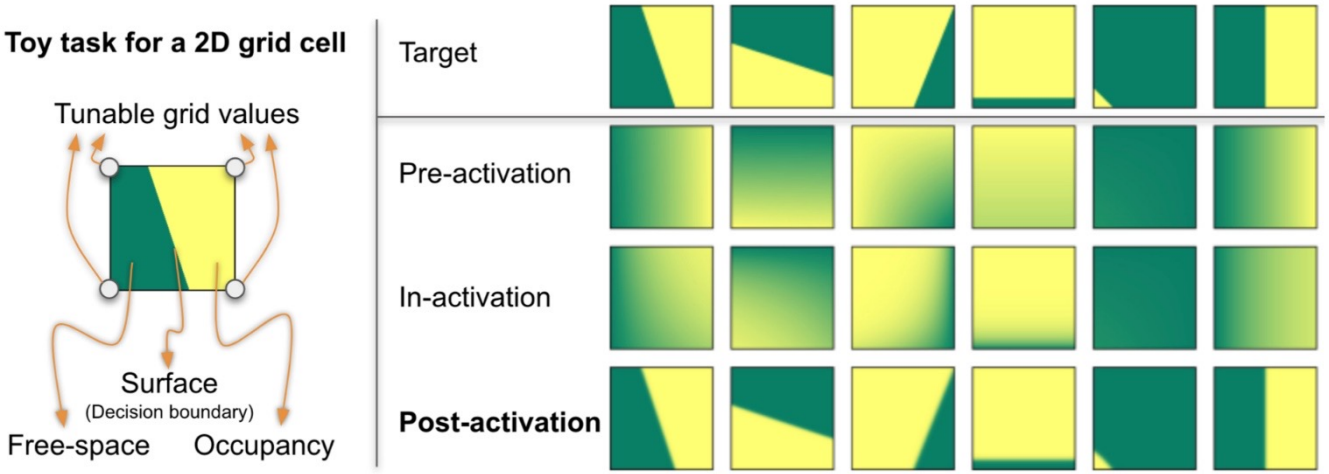
- Sun et al. (2022): Direct Voxel Grid Optimization (DVGO)
  - DVGO is an explicit-implicit hybrid representation
  - Converges rapidly in less than 15 mins on a single GPU with NeRF-comparable quality
  - Does not require depth guidance and cross-scene pre-training

- **Find the coarse geometry** first and then **reconstruct the fine** details
  1. Post-activated voxel-grid interpolation for sharp boundary modeling
  2. Two priors to avoid suboptimal geometry in direct voxel density optimization

* source : Sun et. al., Direct voxel grid optimization: super-fast convergence for radiance fields reconstruction, CVPR 2022 Oral

Direct Voxel Grid Optimization: fast convergence speed and high-quality [Sun et al., 2022]

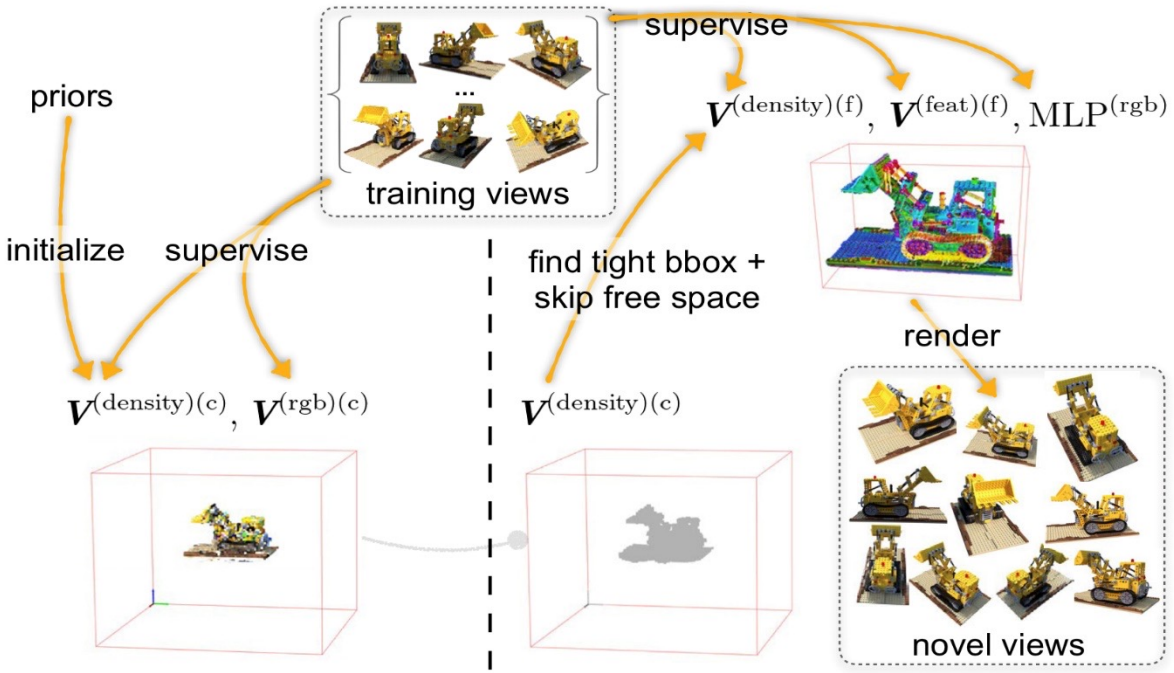
- DVGO proposed post-activated density voxel grid
  - For a queried 3D point $\boldsymbol{x}$, voxel features are trilinearly interpolated
  - In practice, voxel grid optimization can be processed as follows:
    - Pre-activation: $\alpha^{(\text{pre})} = \underline{\text{interp}}\left(\boldsymbol{x}, \underline{\text{alpha}}\left(\underline{\text{softplus}}\left(\boldsymbol{V}^{(\text{density})}\right)\right)\right)$
    - In-activation: $\alpha^{(\text{in})} = \underline{\text{alpha}}\left(\underline{\text{interp}}\left(\boldsymbol{x}, \underline{\text{softplus}}\left(\boldsymbol{V}^{(\text{density})}\right)\right)\right)$
    - Post-activation: $\alpha^{(\text{post})} = \underline{\text{alpha}}\left(\underline{\text{softplus}}\left(\underline{\text{interp}}\left(\boldsymbol{x}, \boldsymbol{V}^{(\text{density})}\right)\right)\right)$
  - DVGO applies all the non-linear activation after trilinear interpolation
  - Post-activation is capable of producing sharp surfaces with much fewer grid cells



* source : Sun et. al., Direct voxel grid optimization: super-fast convergence for radiance fields reconstruction, CVPR 2022 Oral

Direct Voxel Grid Optimization: fast convergence speed and high-quality [Sun et al., 2022]

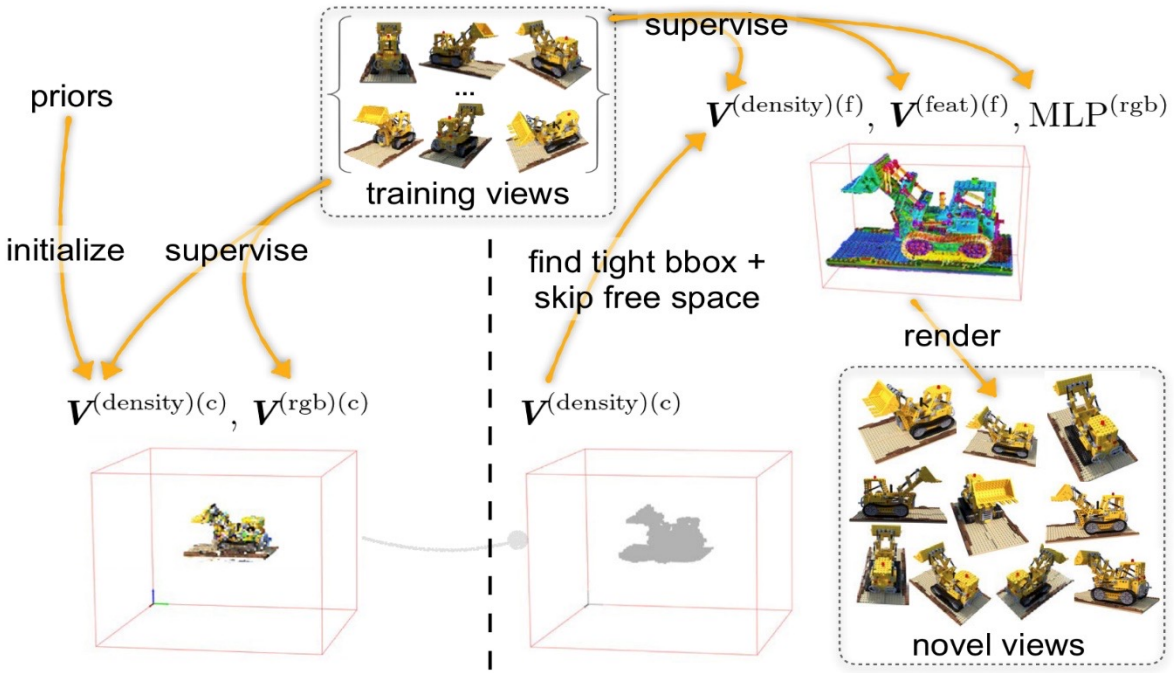
- **DVGO** finds the **coarse geometry first** and then **reconstructs the fine details**



* source : Sun et. al., Direct voxel grid optimization: super-fast convergence for radiance fields reconstruction, CVPR 2022 Oral

Direct Voxel Grid Optimization: fast convergence speed and high-quality [Sun et al., 2022]

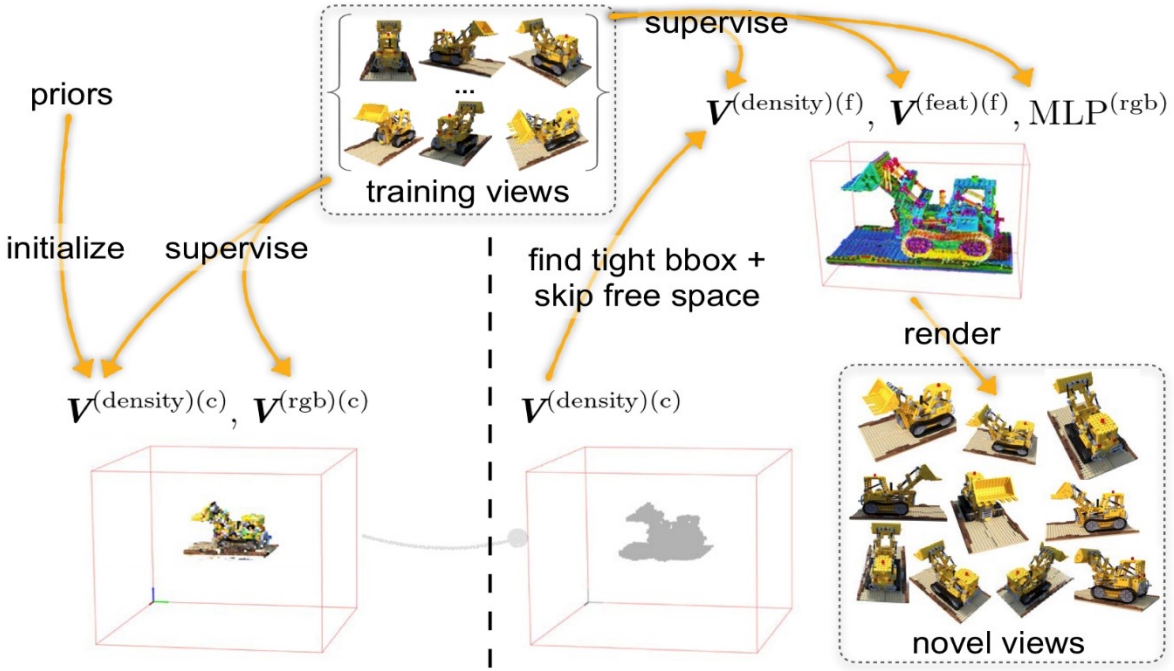
- **DVGO** finds the **coarse geometry first** and then **reconstructs the fine details**
- Prior 1: low-density initialization
  - The coarse density voxel grid could be trapped into a suboptimal *cloudy* geometry
  - Initialize $\boldsymbol{V}^{(\text{density})(\text{c})}$ more carefully to ensure that all sampled points are visible to the camera at the beginning



* source : Sun et. al., Direct voxel grid optimization: super-fast convergence for radiance fields reconstruction, CVPR 2022 Oral

Direct Voxel Grid Optimization: fast convergence speed and high-quality [Sun et al., 2022]

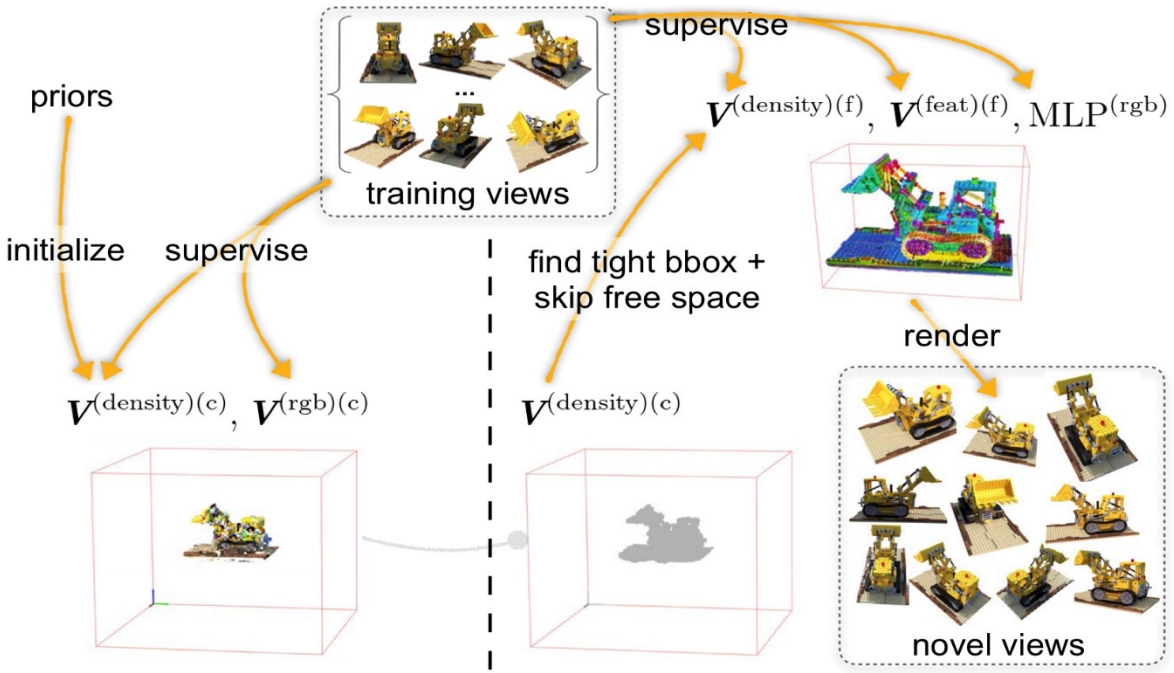
- **DVGO** finds the **coarse geometry first** and then **reconstructs the fine details**
- Prior 2: view-count-based learning rate
  - Set different learning rates for different grid points $\boldsymbol{V}^{(\text{density})(\text{c})}$
  - Scale the learning rate for each grid point *j* by n_j / n_{max}
    - n_j indicates the number of training views at point *j*
    - n_{max} indicates the maximum view count overall grid points



* source : Sun et. al., Direct voxel grid optimization: super-fast convergence for radiance fields reconstruction, CVPR 2022 Oral

Direct Voxel Grid Optimization: fast convergence speed and high-quality [Sun et al., 2022]

- **DVGO** finds the **coarse geometry first** and then **reconstructs the fine details**
  - Frozen the optimized coarse geometry $\boldsymbol{V}^{(\text{density})(\text{c})}$ in this stage
- Fine scene representation uses an **explicit-implicit hybrid** representation
  1. Explicit representation: a feature voxel grid $\boldsymbol{V}^{(\text{feat})(\text{f})}$
  2. Implicit representation: a shallow MLP parameterized by Θ



* source : Sun et. al., Direct voxel grid optimization: super-fast convergence for radiance fields reconstruction, CVPR 2022 Oral

Direct Voxel Grid Optimization: fast convergence speed and high-quality [Sun et al., 2022]

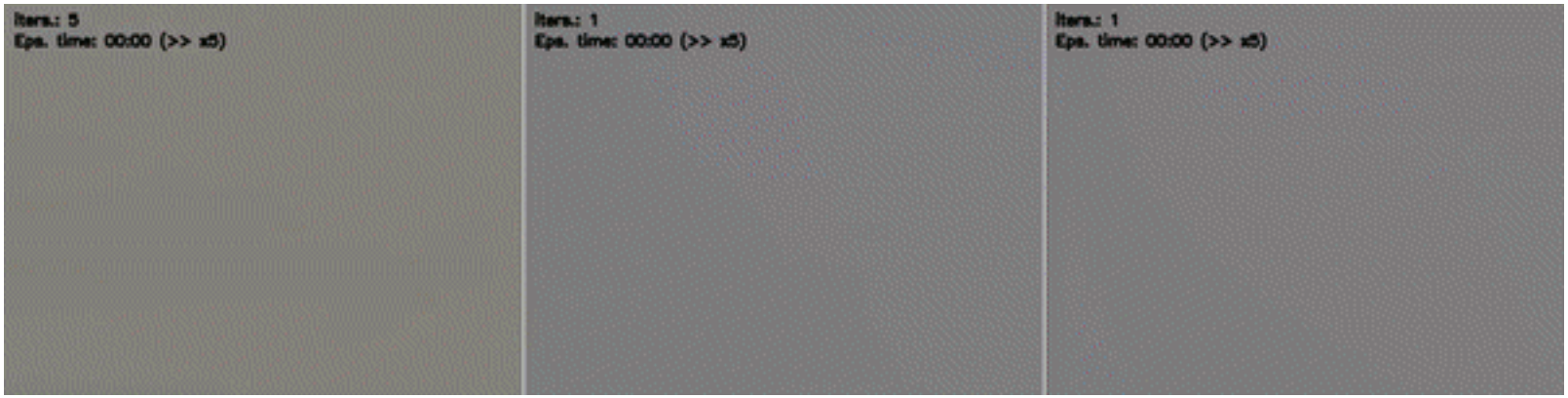
- The key merit of DVGO is the significant improvement in convergence speed

Training time comparisons

Methods	PSNR↑	generalizable pre-training	per-scene optimization
NeRF [37]	31.01	no need	1–2 days (V100)
MVSNeRF [6]	27.21	30 hrs (2080Ti)	15 mins (2080Ti)
IBRNet [60]	28.14	1 day (8xV100)	6 hrs (V100)
NeuRay [31]†	32.42	2 days (2080Ti)	23 hrs (2080Ti)
Mip-NeRF [1]‡	30.85	no need	6 hrs (2080Ti)
ours	31.95	no need	15 mins (2080Ti)

† Use external depth information.

‡ Our reproduction with early stopping on our machine.

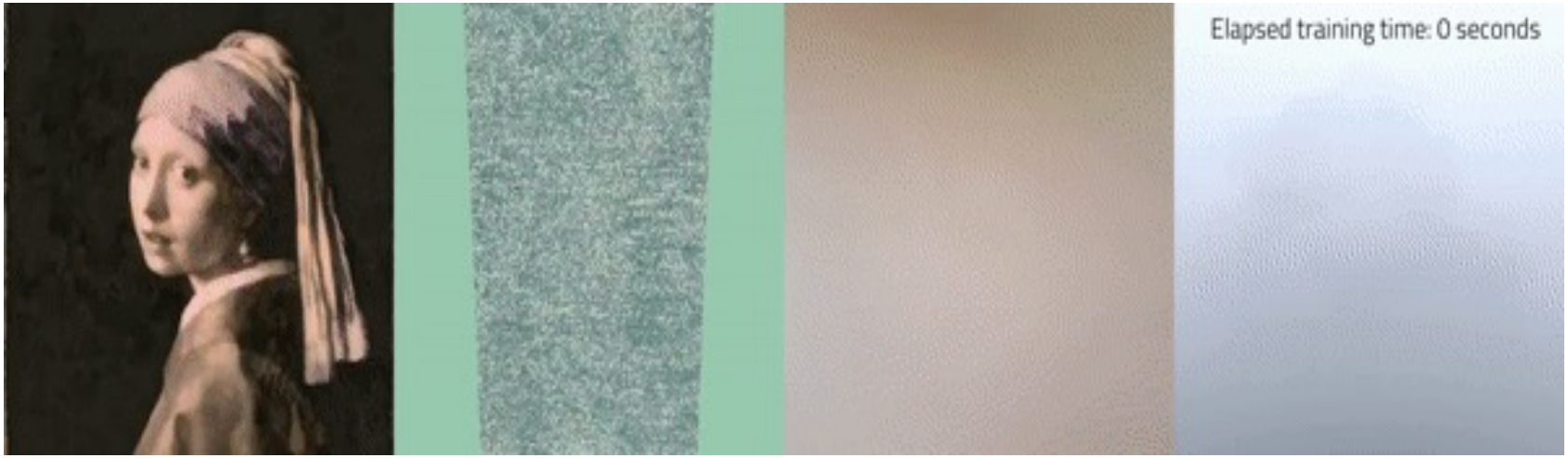


* source : Sun et. al., Direct voxel grid optimization: super-fast convergence for radiance fields reconstruction, CVPR 2022 Oral

Instant Neural Graphic Primitives: a versatile hash-based encoding [Mùller et al., 2022]

- Fully connected neural networks can be costly to train and evaluate

- Mùller et al. (2022): **Instant neural graphic primitives** (Instant-ngp)
  - Reduces this cost with a versatile hash-based input encoding
  - The hash encoding permits the use of a smaller network without sacrificing quality
  - Speeds up representative tasks by several orders of magnitude (10x - 100x speedup)



* source : Mùller et. al., Instant neural graphics primitives with a multiresolution hash encoding, SIGGRAPH 2022

Instant Neural Graphic Primitives: a versatile hash-based encoding [Mùller et al., 2022]

- Fully connected neural networks can be costly to train and evaluate

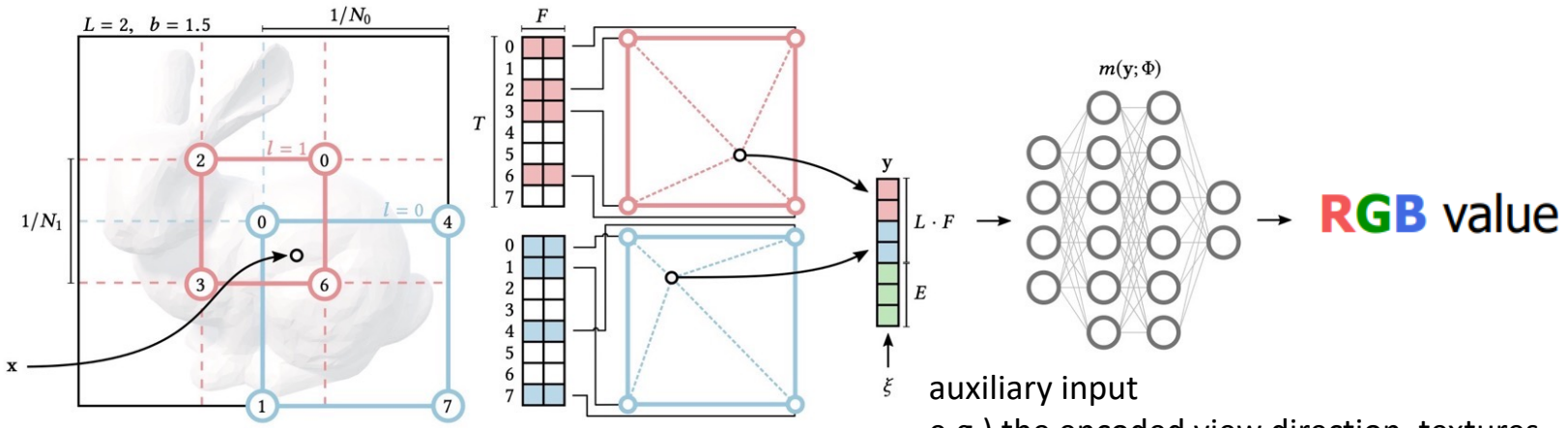
- Mùller et al. (2022): **Instant neural graphic primitives** (Instant-ngp)
  - Reduces this cost with a versatile hash-based input encoding
  - The hash encoding permits the use of a smaller network without sacrificing quality
  - Speeds up representative tasks by several orders of magnitude (10x - 100x speedup)

- Unlike prior works which target faster training,
  - No complex training process is needed
  - No limitation to data structure exists

- Despite its simplicity, it provides clear benefits:
  - Rapid training
  - High quality

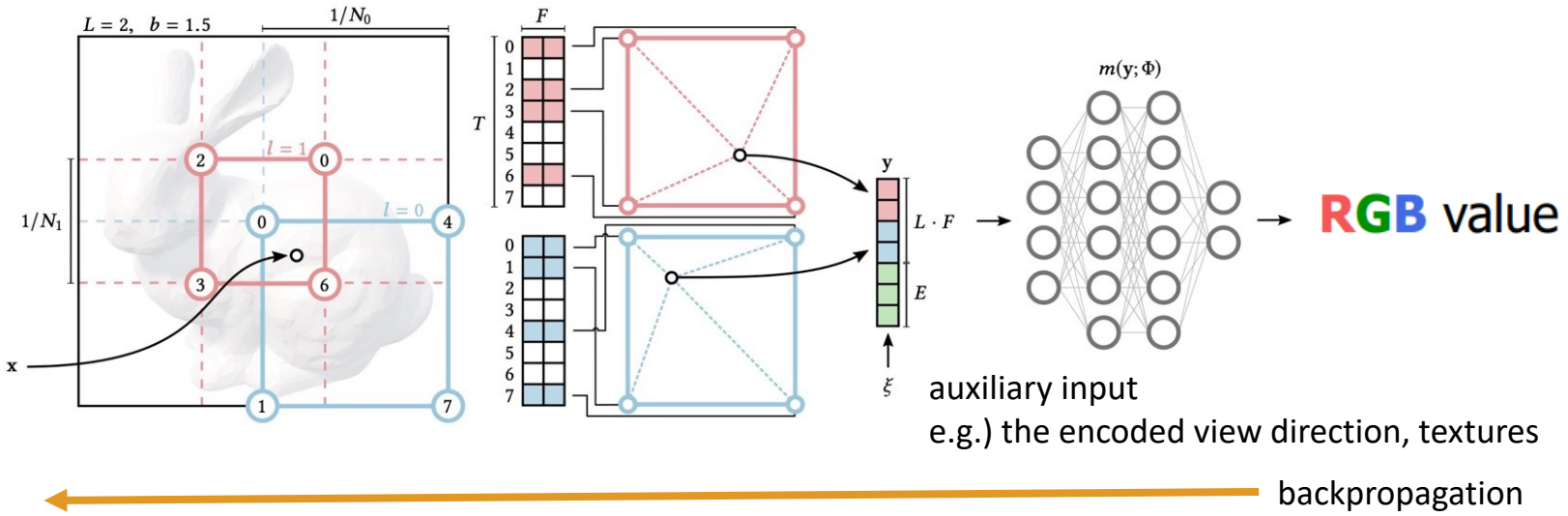
* source : Mùller et. al., Instant neural graphics primitives with a multiresolution hash encoding, SIGGRAPH 2022

- **Multiresolution hash encoding**
  - The encoding structure consists of multiple hash tables of various resolutions
  - For a queried coordinate ***x***,
    - ① find the surrounding corners at ***L*** resolution levels
    - ② assign indices to their corners by spatial hashing
    - ③ look up the corresponding F-dimensional trainable feature vectors from the hash tables θ_l
    - ④ linearly interpolate them
    - ⑤ concatenate the result of each level, and auxiliary inputs



* source : Mùller et. al., Instant neural graphics primitives with a multiresolution hash encoding, SIGGRAPH 2022

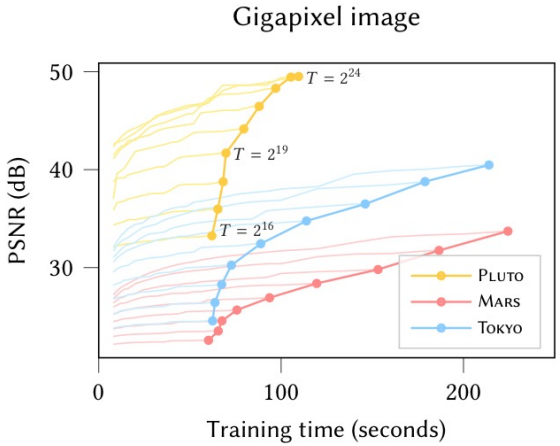
- **Implicitly handle a hash collision with MLP**
  - At coarse levels,
    - 1:1 mapping exists between entries and spatial locations; no collision
  - At fine levels,
    - disparate points can hash to the same table entry; collision
  - Do not explicitly handle collision of the hash functions
  - Rely on the MLP to learn to disambiguate hash collision itself by averaging gradient
  - The more important point dominates the collision average
  - The less important point will be corrected by other levels in the multi-resolution

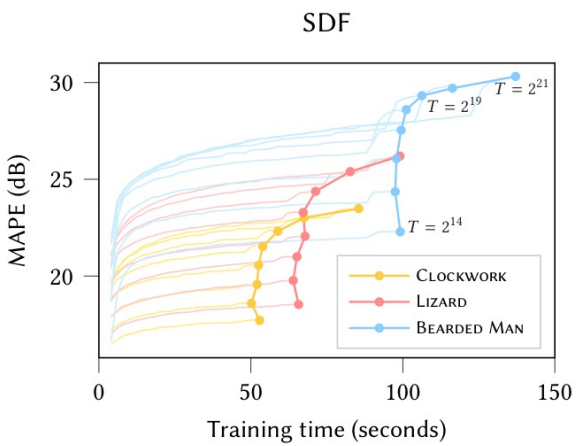


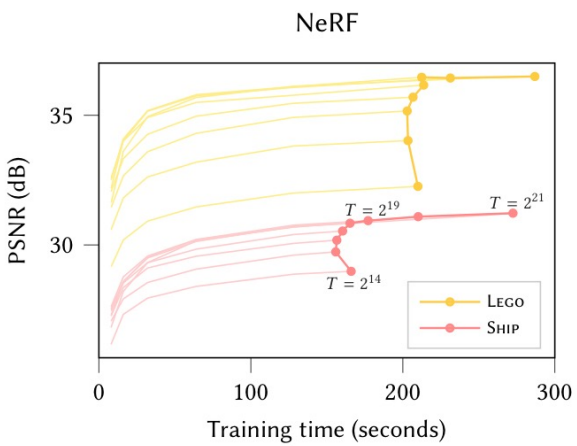
* source : Mùller et. al., Instant neural graphics primitives with a multiresolution hash encoding, SIGGRAPH 2022

Instant Neural Graphic Primitives: a versatile hash-based encoding [Mùller et al., 2022]

- **Instant-ngp represents complex signals in a matter of seconds**







(a) None
411k parameters
10:45 (mm:ss)

(b) Multiresolution grid
10k + 16.3M parameters
1:26 (mm:ss)

(c) Frequency
438k + 0 parameters
13:53 (mm:ss)

(d) Hashtable ($T=2^{14}$)
10k + 494k parameters
1:40 (mm:ss)

(e) Hashtable ($T=2^{19}$)
10k + 12.6M parameters
1:45 (mm:ss)

* source : Mùller et. al., Instant neural graphics primitives with a multiresolution hash encoding, SIGGRAPH 2022

Table of Contents

Part 1. Basics

- Architectures: SIREN, FFN
- Neural radiance fields (NeRF)

Part 2. Advanced Topics

- Techniques for faster training
- Techniques for real-time inference
- Training at real-world scenarios (e.g., deformable, unconstrained)

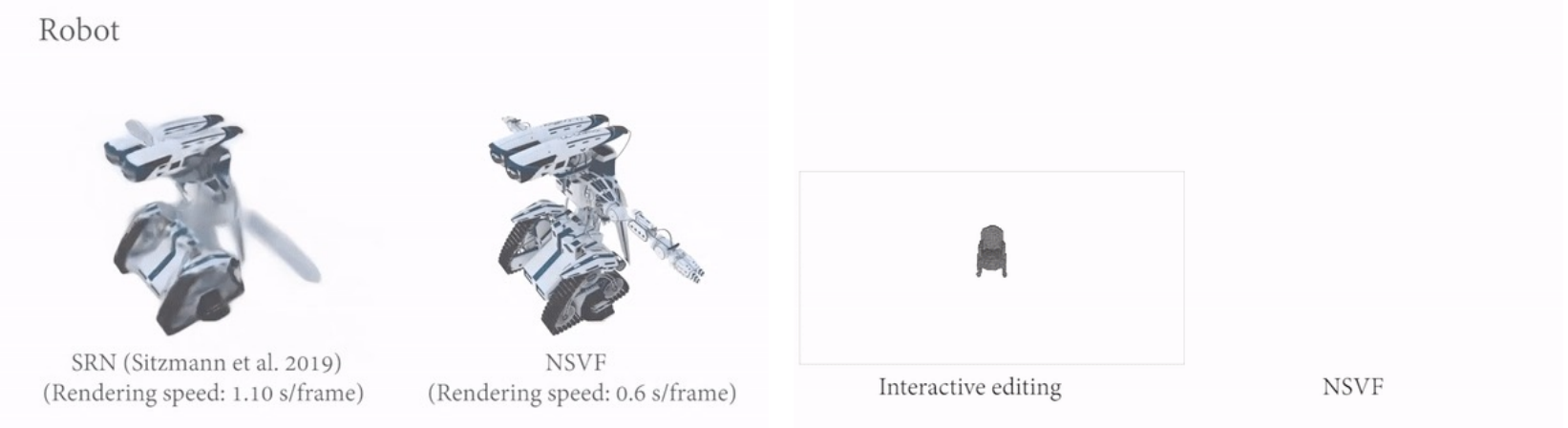
Part 3. Beyond Neural Fields of a Single Signal

- Generalization of neural fields on multiple signals

Neural Sparse Voxel Fields: Store a 3D voxel of latent codes and sparsify [Liu et al., 2021]

- Existing novel scene representations have clear and practical limitations:
  - (-) Low rendering quality
    - Blurry rendering caused by the limited network capacity
    - Difficulty in finding exact camera rays' intersections with the scene geometry
  - (-) Time-consuming
    - Time-consuming optical ray marching hinders real-time inference

- Liu et al. (2022): **Neural Sparse Voxel Fields** (NVSF)
  - Accelerates inference time >10 times than NeRF with higher quality
  - Easily applied to scene editing and scene composition



* source : Liu et. al., Neural sparse voxel fields, NeurIPS 2021

Neural Sparse Voxel Fields: Store a 3D voxel of latent codes and sparsify [Liu et al., 2021]

- Liu et al. (2022): **Neural Sparse Voxel Fields** (NVSF)
  - Defines a set of voxel-bounded implicit fields;
    - Each voxel, voxel embeddings are learned to encode local properties
  - Underlying sparse voxel structures are progressively learned with a differentiable ray marching operation

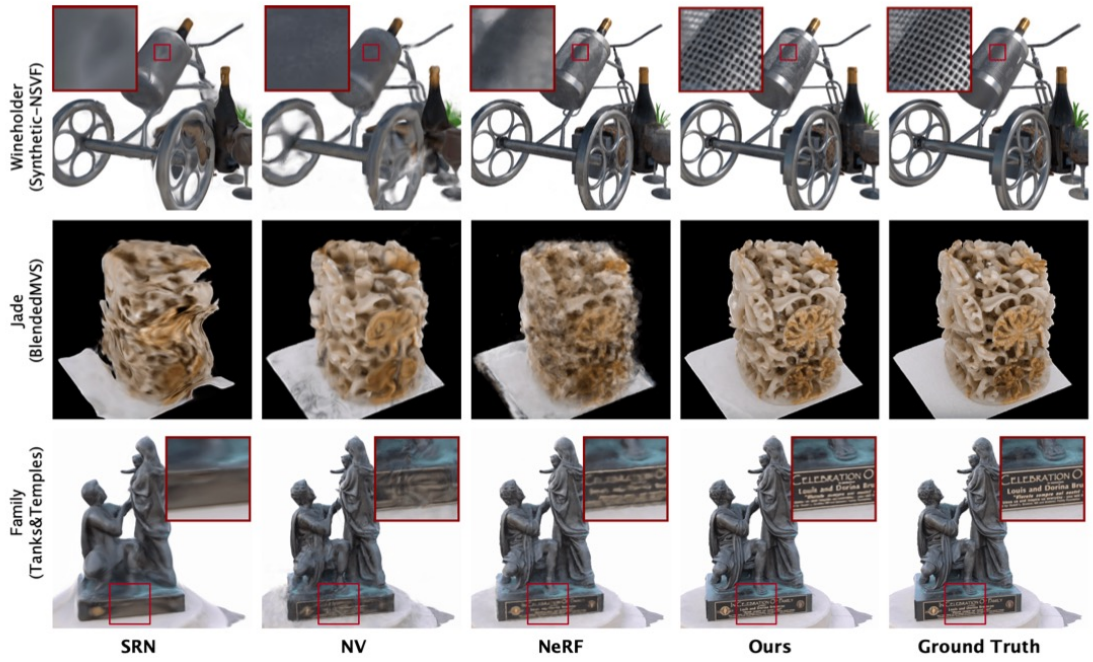


Figure 3: Comparisons on test views for scenes from the single-scene datasets. For *wineholder* and *family*, closeups are shown for clearer visual comparison.

* source : Liu et. al., Neural sparse voxel fields, NeurIPS 2021

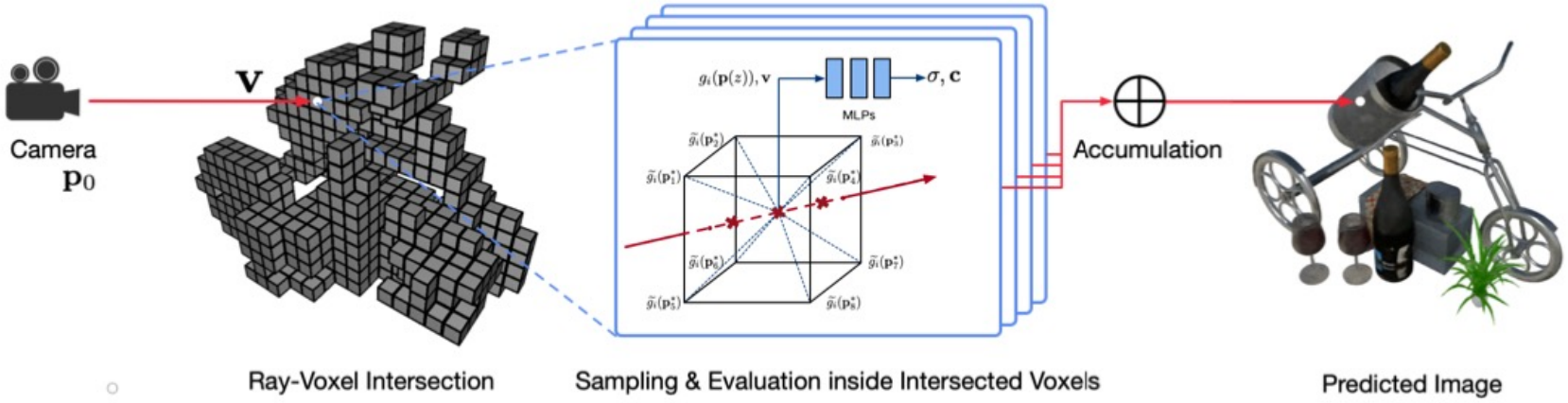
- **Prevent sampling of points in empty space without relevant scene content**

1. Voxel-bounded implicit fields
   - (Assumption) Relevant non-empty parts are contained within a set of sparse voxels:
     $$\mathcal{V} = \{V_1, \cdots, V_k\}$$
   - The scene is modeled as a set of voxel-bounded implicit functions:
     $F_\theta(\mathbf{p}, \mathbf{v}) = F_\theta^i(g_i(\mathbf{p}), \mathbf{v})$ if $\mathbf{p} \in V_i$ where $\mathbf{p}$ is the 3D position and $\mathbf{v}$ is the ray direction
   - Each implicit function is modeled as a multi-layer perceptron with shared params θ:
     $$F_\theta^i(g_i(\mathbf{p}), \mathbf{v}) \rightarrow (\mathbf{c}, \sigma), \forall \mathbf{p} \in V_i$$
   - $g_i(\mathbf{p})$ is the representation at $\mathbf{p}$ which is defined as:
     $g_i(\mathbf{p}) = \zeta(\chi(\tilde{g}_i(\mathbf{p}_1^*), \ldots, \tilde{g}_i(\mathbf{p}_8^*)))$ where χ is trilinear interpolation and is ζ positional encoding
   - Compared to previous works, feature representation $g_i(\mathbf{p})$ is aggregated by the corresponding voxels where region-specific information
     (+) Significantly eases the learning of subsequent implicit functions F_θ^i
     (+) Facilitates high-quality rendering

* source : Liu et. al., Neural sparse voxel fields, NeurIPS 2021

- **Prevent sampling of points in empty space without relevant scene content**

2. A rendering algorithm
   - Compared to rendering implicit neural representations that model the entire scenes, NVSF renders much more efficiently by preventing sampling points in empty space
   - Intersect the ray with a set of sparse voxels
   - Sampling and accumulating the color and density inside the every intersected voxel, which are predicted by neural networks



* source : Liu et. al., Neural sparse voxel fields, NeurIPS 2021

- **Prevent sampling of points in empty space without relevant scene content**

3. Progressive learning strategy
   - Self-pruning
     - Remove non-essential voxels during training based on the coarse geometry
   - Progressive training
     - Progressively adjust voxelization to the underlying scene structure
     - Adaptively allocate computational and memory resources to important regions

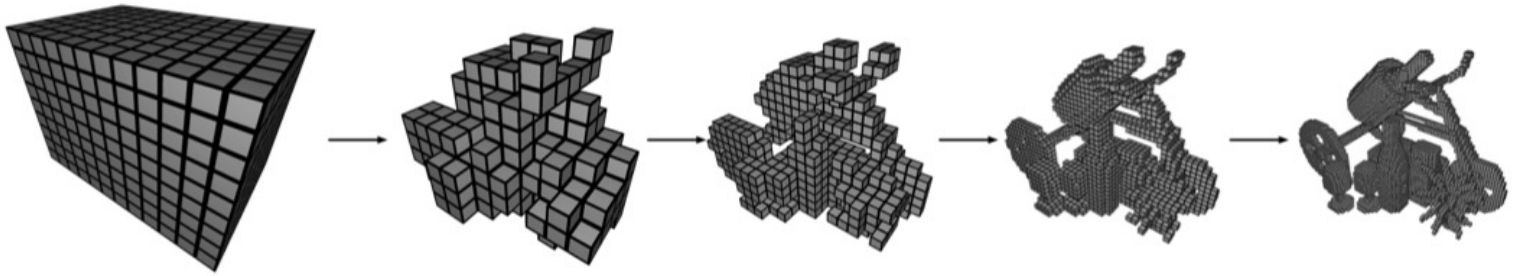


Figure 2: Illustration of self-pruning and progressive training

* source : Liu et. al., Neural sparse voxel fields, NeurIPS 2021

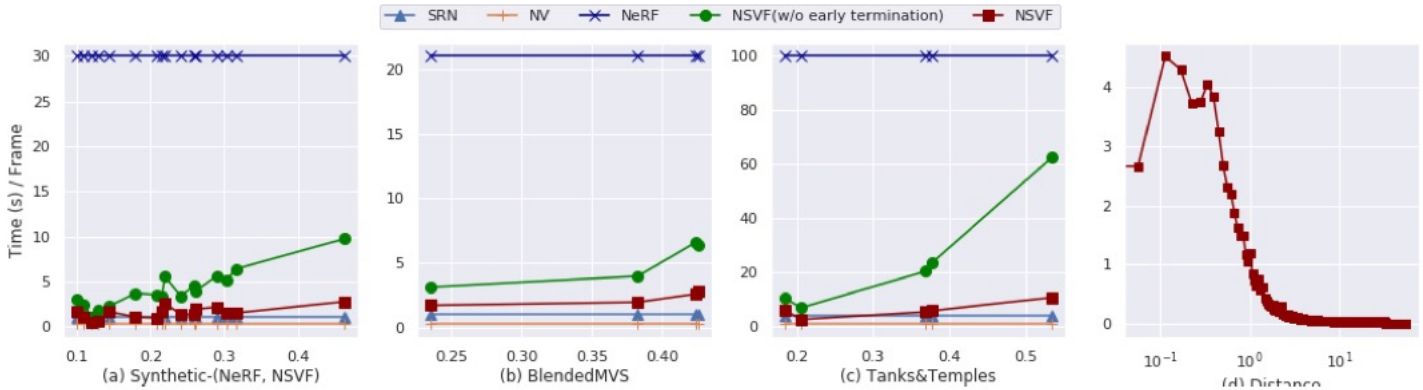
Neural Sparse Voxel Fields: Store a 3D voxel of latent codes and sparsify [Liu et al., 2021]

- **High-quality rendering**:
  NSVF significantly outperforms baselines on all datasets across all metrics

Table 1: The quantitative comparisons on test sets of four datasets. We use three metrics: PSNR (↑), SSIM (↑) and LPIPS (↓) (Zhang et al., 2018) to evaluate the rendering quality. Scores are averaged over the testing images of all scenes, and we present the per-scene breakdown results in the Appendix. By default, NSVF is executed with early termination ($\epsilon = 0.01$). We also show results without using early termination ($\epsilon = 0$) denoted as NSVF^0.

	Synthetic-NeRF			Synthetic-NSVF			BlendedMVS			Tanks and Temples		
Models	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
SRN	22.26	0.846	0.170	24.33	0.882	0.141	20.51	0.770	0.294	24.10	0.847	0.251
NV	26.05	0.893	0.160	25.83	0.892	0.124	23.03	0.793	0.243	23.70	0.834	0.260
NeRF	31.01	0.947	0.081	30.81	0.952	0.043	24.15	0.828	0.192	25.78	0.864	0.198
NSVF^0	**31.75**	**0.954**	0.048	**35.18**	**0.979**	**0.015**	26.89	**0.898**	0.114	**28.48**	**0.901**	0.155
NSVF	31.74	0.953	**0.047**	35.13	**0.979**	**0.015**	**26.90**	**0.898**	**0.113**	28.40	0.900	**0.153**

- **Fast inference:**
  NVSF is around 10 ~ 20 times faster than the SOTA method NeRF

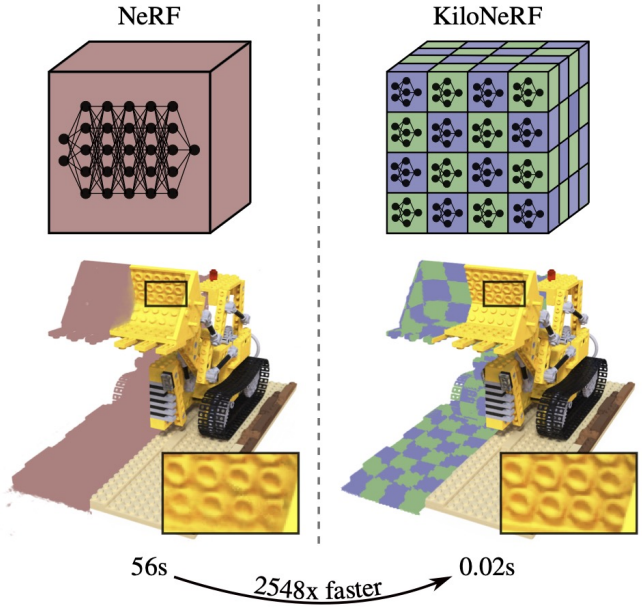


* source : Liu et. al., Neural sparse voxel fields, NeurIPS 2021

KiloNeRF: Utilize thousands of tiny MLPs in place of one single large MLP [Reiser et al., 2021]

- Using one single MLP has a trade-off between sampling time and rendering quality
  - For relatively deep and wide MLP, sampling hundreds of times is very slow
  - A reduction in network size leads to a lower-quality due to the limited capacity

- Reiser et al. (2021): Speeding up NeRF with thousands of tiny MLPs (KiloNeRF)
  - Utilize thousands of MLPs instead of one single large MLP for real-time rendering
  - Being able to synthesize novel views three orders of magnitude faster



* source : Reiser et. al., KiloNeRF: speeding up neural radiance fields with thousands of tiny MLPs, ICCV 2021

- **Spatially decompose the scene into a regular grid and assign tiny MLPs**

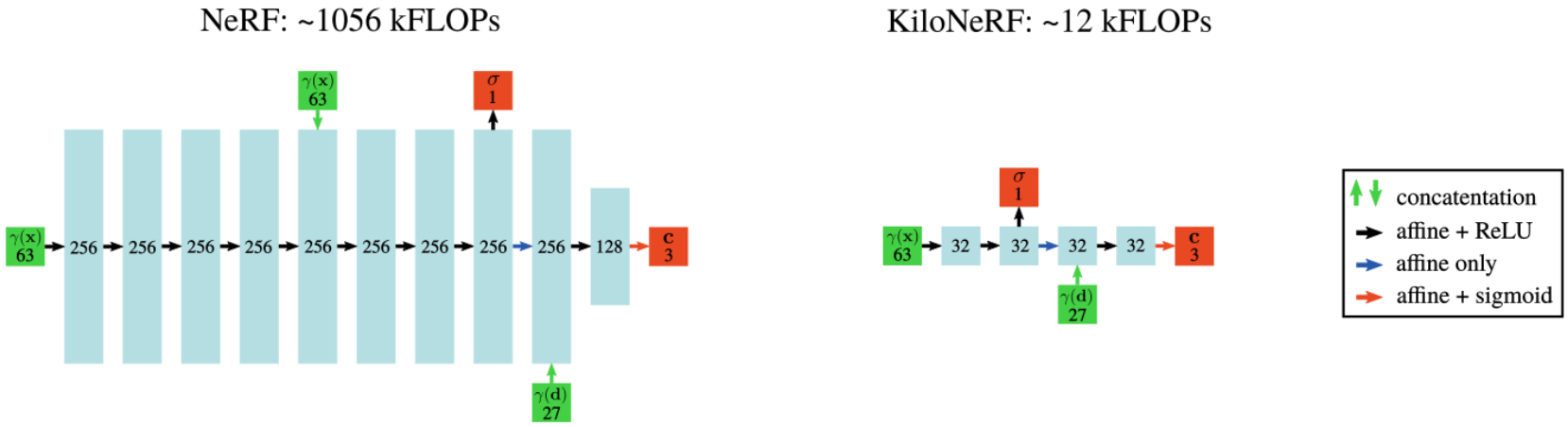
1. Train a regular NeRF as a teacher model
   - Naïvely training KiloNeRF from scratch leads to noticeable artifacts in free space
   - Distillation by imitating a pre-trained NeRF mitigates this issue

(a) Without Distillation (b) With Distillation

* source : Reiser et. al., KiloNeRF: speeding up neural radiance fields with thousands of tiny MLPs, ICCV 2021

- **Spatially decompose the scene into a regular grid and assign tiny MLPs**

2. Train a KiloNeRF model as a student model
   - Subdivide the scene into a cell that has uniform grid of resolution
   - Assign each grid cell to a tiny MLP network with an independent set of parameters θ_i
   - Train KiloNeRF to match its outputs match the outputs (color, density) of the teacher



* source : Reiser et. al., KiloNeRF: speeding up neural radiance fields with thousands of tiny MLPs, ICCV 2021

- **Spatially decompose the scene into a regular grid and assign tiny MLPs**

3. Fine-tune KiloNeRF on the training images
   - Distillation alone limits the KiloNeRF's rendering quality to be the teacher model
   - Use photometric L_2 loss between rendered images and training images
   - Distillation can be considered as a powerful weight initialization to KiloNeRF

* source : Reiser et. al., KiloNeRF: speeding up neural radiance fields with thousands of tiny MLPs, ICCV 2021

KiloNeRF: Utilize thousands of tiny MLPs in place of one single large MLP [Reiser et al., 2021]

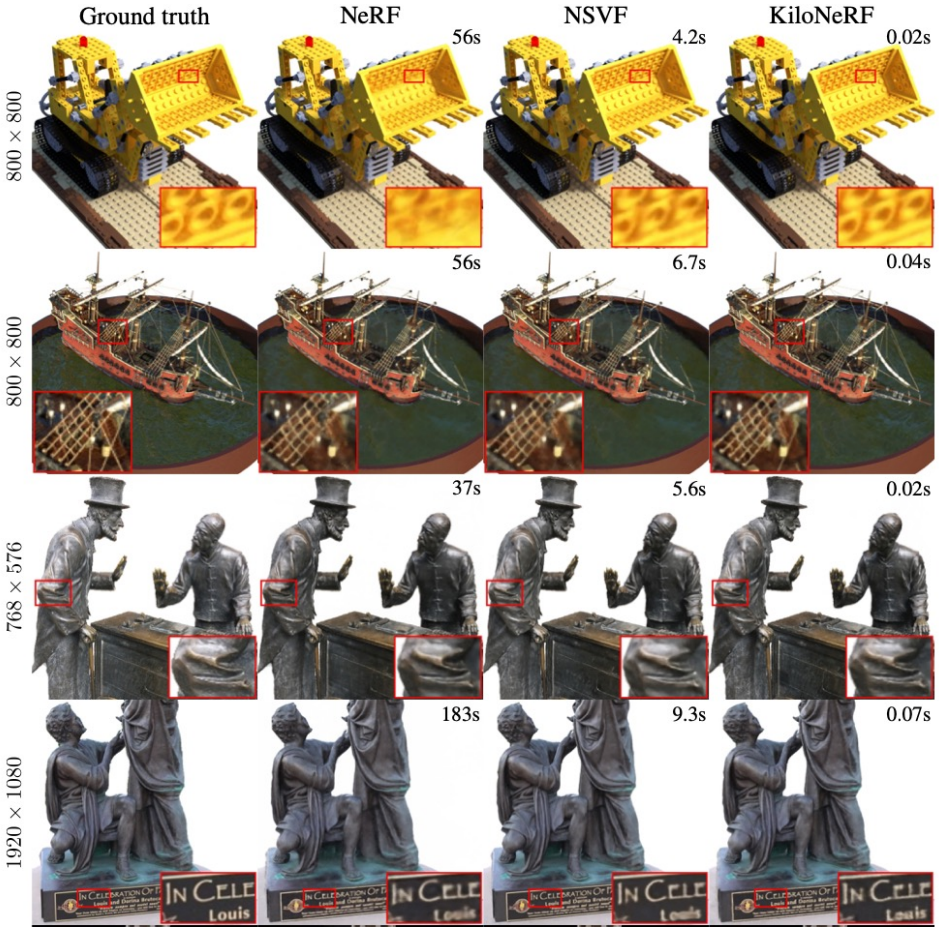
- **Fast inference:**
  - KiloNeRF achieves similar rendering quality as the baselines while being much faster
  - Compared to NeRF, KiloNeRF shows three orders of magnitude faster rendering
  - Compared to NSVF, KiloNeRF shows two orders of magnitude faster rendering

		BlendedMVS	Synthetic-NeRF	Synthetic-NSVF	Tanks & Temples
Resolution		768 × 576	800 × 800	800 × 800	1920 × 1080
PSNR ↑	NeRF	27.29	31.01	31.55	28.32
	NSVF	26.90	**31.74**	**35.13**	28.40
	KiloNeRF	**27.39**	31.00	33.37	**28.41**
SSIM ↑	NeRF	0.91	**0.95**	0.95	0.90
	NSVF	0.90	**0.95**	**0.98**	0.90
	KiloNeRF	**0.92**	**0.95**	0.97	**0.91**
LPIPS ↓	NeRF	0.07	0.08	0.04	0.11
	NSVF	0.11	0.05	**0.01**	0.15
	KiloNeRF	**0.06**	**0.03**	0.02	**0.09**
Render time (milliseconds) ↓	NeRF	37266	56185	56185	182671
	NSVF	4398	4344	10497	15697
	KiloNeRF	**30**	**26**	**26**	**91**
Speedup over NeRF ↑	NSVF	8	13	5	12
	KiloNeRF	**1258**	**2165**	**2167**	**2002**

* source : Reiser et. al., KiloNeRF: speeding up neural radiance fields with thousands of tiny MLPs, ICCV 2021

KiloNeRF: Utilize thousands of tiny MLPs in place of one single large MLP [Reiser et al., 2021]

- **Fast inference:**
  - KiloNeRF achieves similar rendering quality as the baselines while being much faster
  - The numbers in the top-right correspond to the average render time of each baseline

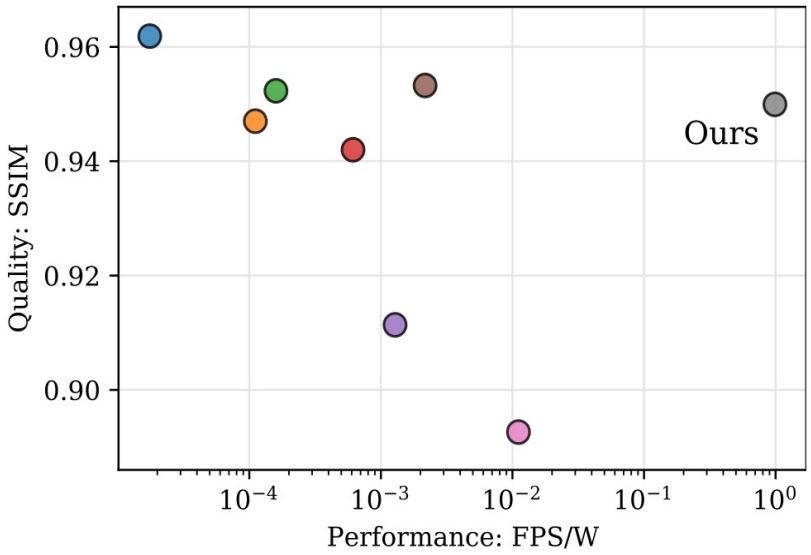


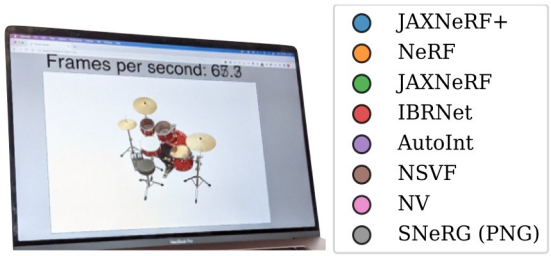
* source : Reiser et. al., KiloNeRF: speeding up neural radiance fields with thousands of tiny MLPs, ICCV 2021

Baking Neural Radiance Fields for Real-time View Synthesis [Hedman et al., 2021]

- Rendering procedure of NeRF is slow in real-time
  - e.g., rendering a frame at 800 x 800 resolution takes a minute on a modern GPU

- Henman et al. (2022): **Baking neural radiance fields** (SNeRG)
  - Address slow rendering problem by precomputing and storing in a hybrid approach
  - Rendering time of 12ms per frame on a single GPU which is 3 times faster than NeRF





* source : Henman et. al., Baking neural radiance fields for real-time view synthesis, ICCV 2021 Oral

- **A trained NeRF is baked into a sparse 3D voxel grid data structure**

1. Reformulation of NeRF's architecture (deferred NeRF)
   - Unlike NeRF, deferred NeRF limits the computation of view-dependent effects
     - NeRF: $\sigma(t), \mathbf{c}(t) = \mathrm{MLP}_\Theta(\mathbf{r}(t), \mathbf{d})$.
     - Deferred NeRF: $\sigma(t), \mathbf{c}_d(t), \mathbf{v}_s(t) = \mathrm{MLP}_\Theta(\mathbf{r}(t))$.

       where $\mathbf{c}_d$: a diffuse RGB color; $\mathbf{v}_s$: a 4-dimensional feature vector; σ : volume density
   - Rendering
     - Accumulate the diffuse colors and feature vectors along each ray

$$\hat{\mathbf{C}}_d(\mathbf{r}) = \sum_k T(t_k)\,\alpha(\sigma(t_k)\delta_k)\,\mathbf{c}_d(t_k)\,,$$

$$\mathbf{V}_s(\mathbf{r}) = \sum_k T(t_k)\,\alpha(\sigma(t_k)\delta_k)\,\mathbf{v}_s(t_k)\,,$$

     - Concatenate accumulated feature vector with ray direction
     - Pass them to a very small MLP parameter ϕ (2 layers with 16 channels each)

$$\mathrm{MLP}_\Phi(\mathbf{V}_s(\mathbf{r}), \mathbf{d})$$

     - Produce a view-dependent residual and add it with accumulated diffuse color

$$\hat{\mathbf{C}}(\mathbf{r}) = \hat{\mathbf{C}}_d(\mathbf{r}) + \mathrm{MLP}_\Phi(\mathbf{V}_s(\mathbf{r}), \mathbf{d})$$

* source : Henman et. al., Baking neural radiance fields for real-time view synthesis, ICCV 2021 Oral

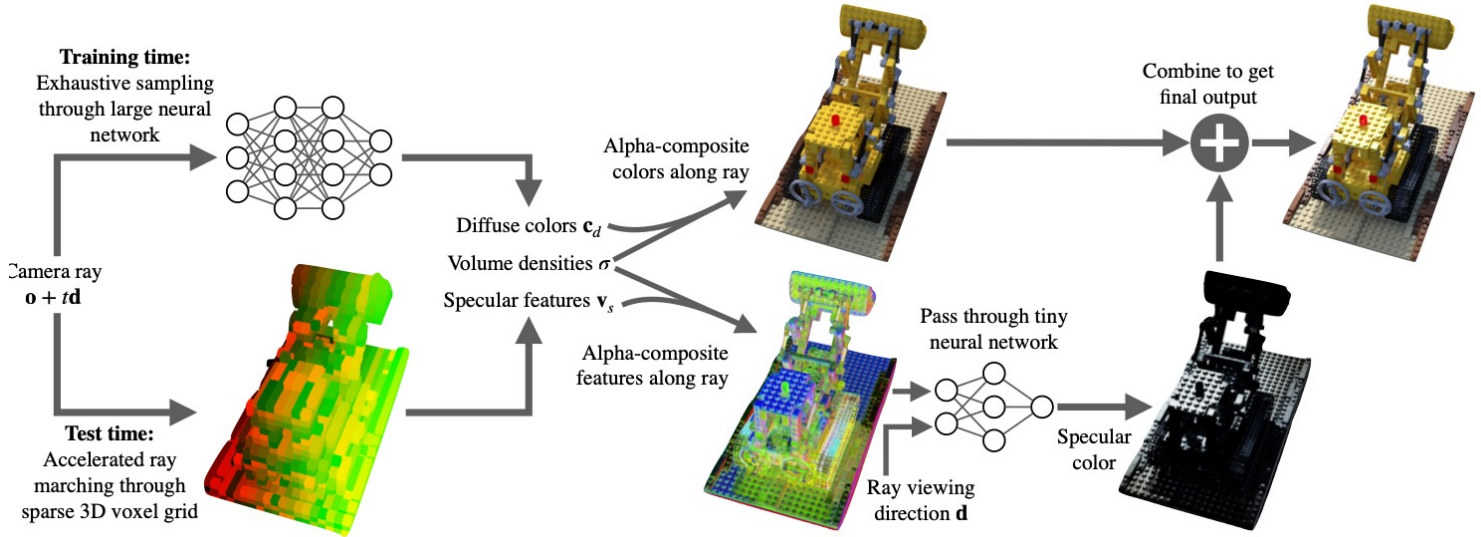
- **A trained NeRF is baked into a sparse 3D voxel grid data structure**

1. Reformulation of NeRF's architecture (deferred NeRF)
   - Unlike NeRF, deferred NeRF limits the computation of view-dependent effects
   - With this modification based on deferred rendering:
     - Precomputing and storing the diffuse colors and feature vectors are possible
     - Only need to evaluate a very small MLP_Φ once per pixel. not per ray

* source : Henman et. al., Baking neural radiance fields for real-time view synthesis, ICCV 2021 Oral

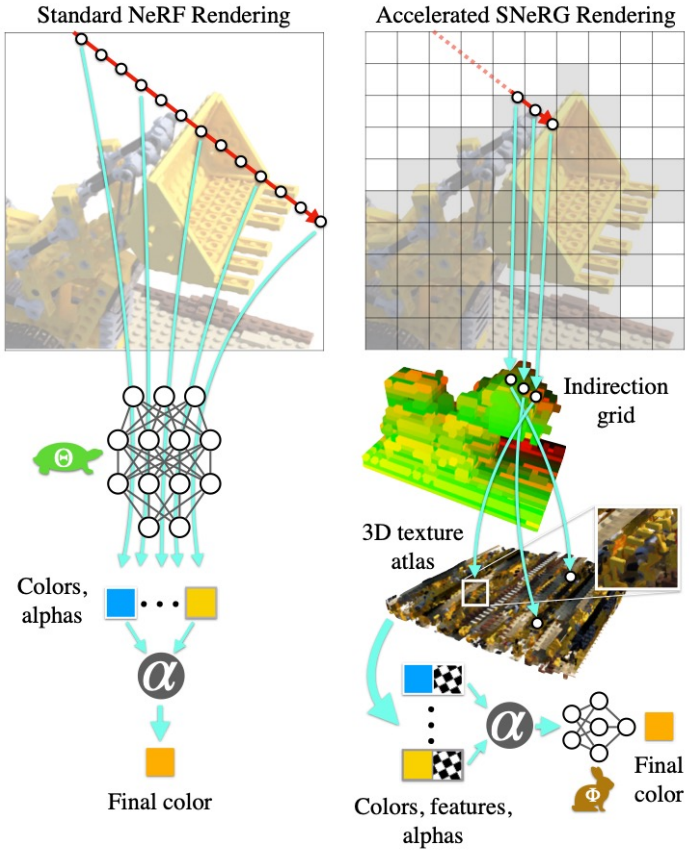
- **A trained NeRF is baked into a sparse 3D voxel grid data structure**

2. Sparse voxel grid representation; by trading computation for storage
   a. Store the diffuse colors, volume densities, and 4-dim feature vectors in a voxel grid
   b. To minimize storage cost, "bake" the voxel grid:
      - Only allocate storage for voxels in the scene that are both non-empty and visible
      - Calculate the opacity values of the voxel and sparsify by culling away empty space
   c. Quantize voxel to 8-bit and compress each slice of voxel with a set of JPEGs
   d. Fine-tune MLP_Φ to recoup all of the lost accuracies



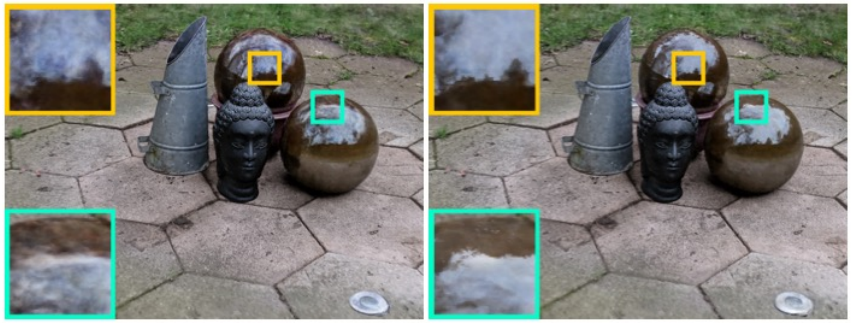
* source : Henman et. al., Baking neural radiance fields for real-time view synthesis, ICCV 2021 Oral

- **A trained NeRF is baked into a sparse 3D voxel grid data structure**
  - Significantly accelerates rendering by replacing compute-intensive MLP evaluations with lookups into a precomputed sparse 3D grid representation



* source : Henman et. al., Baking neural radiance fields for real-time view synthesis, ICCV 2021 Oral

- **A trained NeRF is baked into a sparse 3D voxel grid data structure**
  - Significantly accelerates rendering by replacing compute-intensive MLP evaluations with lookups into a precomputed sparse 3D grid representation



(a) JAXNeRF+ (23.00) (b) Deferred (22.75)

(c) SNeRG (21.45) (d) Ground Truth

Figure 7: **Real 360° scene** (PSNR in parentheses). Note how our real-time method is able to model the mirror-like reflective surface of the garden spheres.

	PSNR ↑	SSIM ↑	LPIPS ↓
JAXNeRF+	33.00	0.962	0.038
JAXNeRF+ Tinyview	31.65	0.954	0.047
JAXNeRF+ Deferred	30.55	0.952	0.049
SNeRG (PNG)	30.38	0.950	0.050
SNeRG (PNG, no $\mathcal{L}_s$)	30.22	0.949	0.050
SNeRG (PNG, no FT)	26.68	0.930	0.053
JAXNeRF+ Diffuse	27.39	0.927	0.068

Table 2: Quality ablation study on Synthetic 360° scenes.

	PSNR ↑	SSIM ↑	LPIPS ↓	MB ↓
SNeRG (Float)	30.47	0.951	0.049	6883.6
SNeRG (PNG, no $\mathcal{L}_s$)	30.22	0.949	0.050	176.0
SNeRG (PNG)	30.38	0.950	0.050	86.7
SNeRG (JPEG)	29.71	0.939	0.062	70.9
SNeRG (H264)	29.86	0.938	0.065	30.2
JAXNeRF+	33.00	0.962	0.038	18.0
JAXNeRF	31.65	0.952	0.051	4.8

Table 3: Storage ablation study on Synthetic 360° scenes.

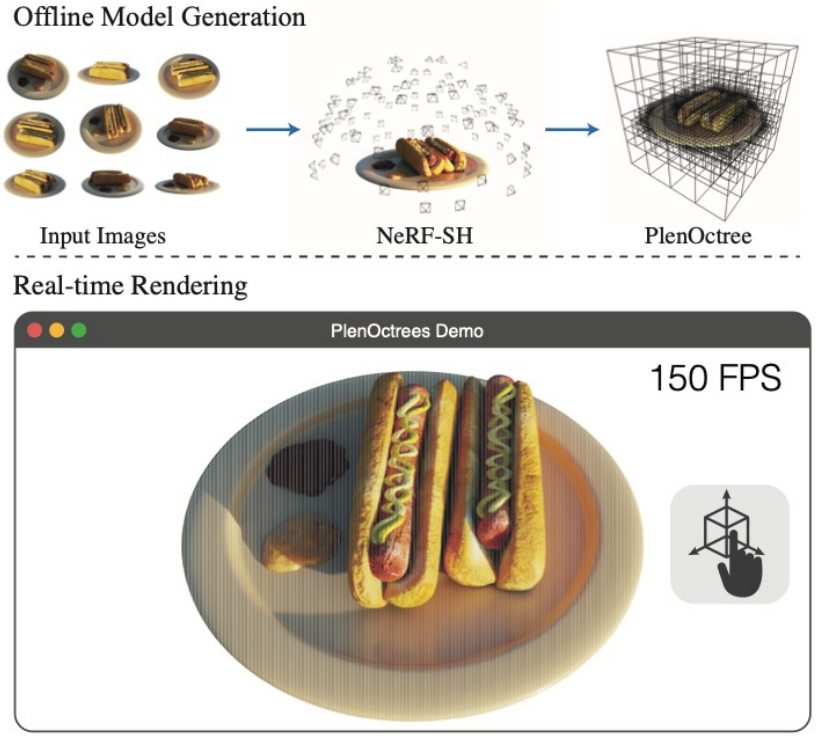
	PSNR ↑	SSIM ↑	LPIPS ↓	W ↓	FPS ↑	FPS/W ↑
JAXNeRF+ [11]	33.00	0.962	0.038	300	0.01	0.00002
NeRF [29]	31.00	0.947	0.081	300	0.03	0.00011
JAXNeRF [11]	31.65	0.952	0.051	300	0.05	0.00016
IBRNet [43]	28.14	0.942	0.072	300	0.18	0.00061
AutoInt [23]	25.55	0.911	0.170	300	0.38	0.00128
NSVF [24]	31.74	0.953	0.047	300	0.65	0.00217
NV [25]	26.05	0.893	0.160	300	3.33	0.01111
SNeRG (PNG)	30.38	0.950	0.050	85	84.06	0.98897

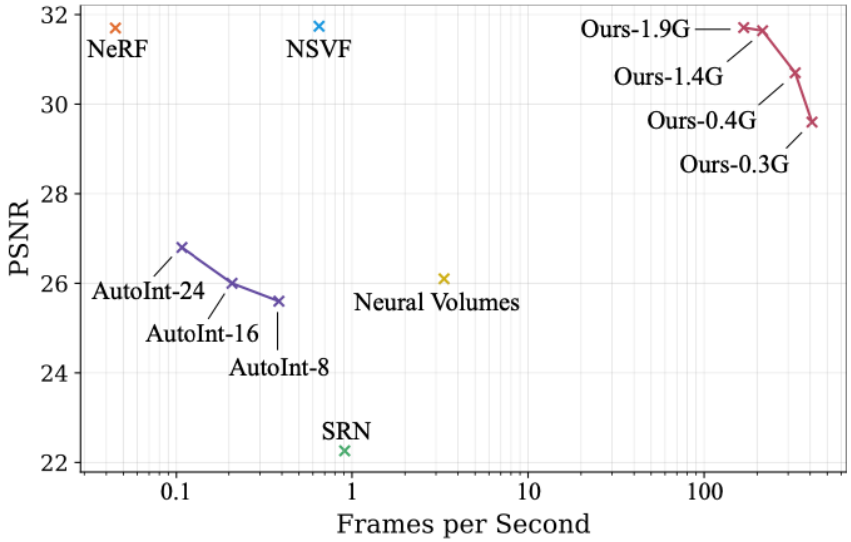
Table 4: Baseline comparisons on Synthetic 360° scenes.

* source : Henman et. al., Baking neural radiance fields for real-time view synthesis, ICCV 2021 Oral

PlenOctrees for Real-time Rendering of Neural Radiance Fields [Yu et al., 2021]

- Yu et al. (2022): **PlenOctrees for Real-time Rendering of NeRF** (PlenOctrees)
  - Firstly achieved **real-time** rendering of NeRF with improved quality
  - Speeds up rendering performance of original NeRF by more than 3000 times
  - Trained significantly faster as it involves PlenOctree, which is not a neural network

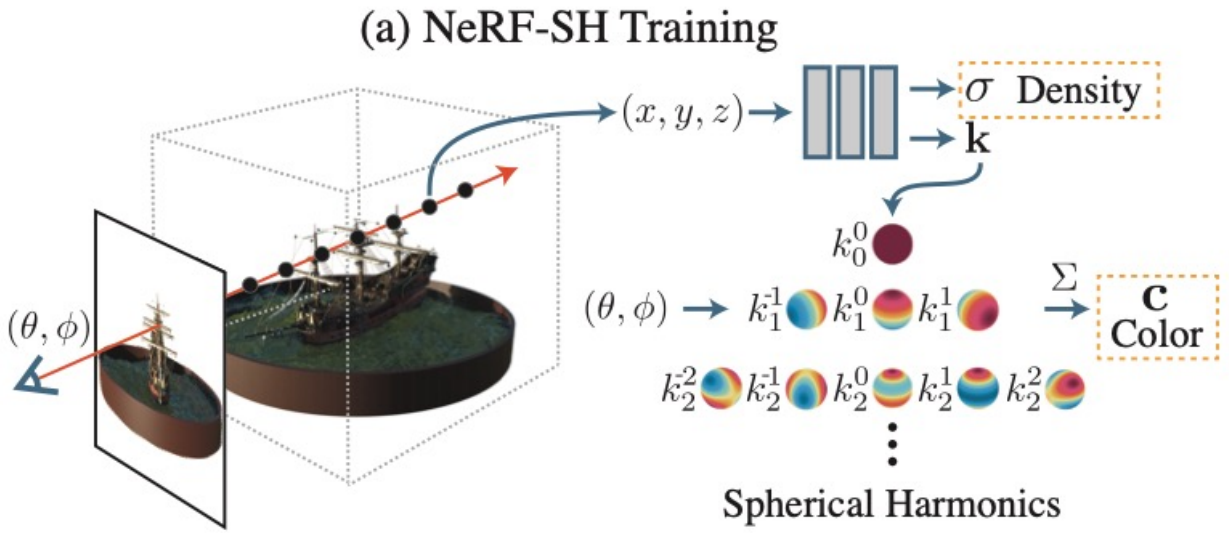




* source : Yu et. al., PlenOctrees for real-time rendering of neural radiance fields, ICCV 2021 Oral

- **Achieved real-time rendering by pre-tabulating the NeRF-SH into a PlenOctree**

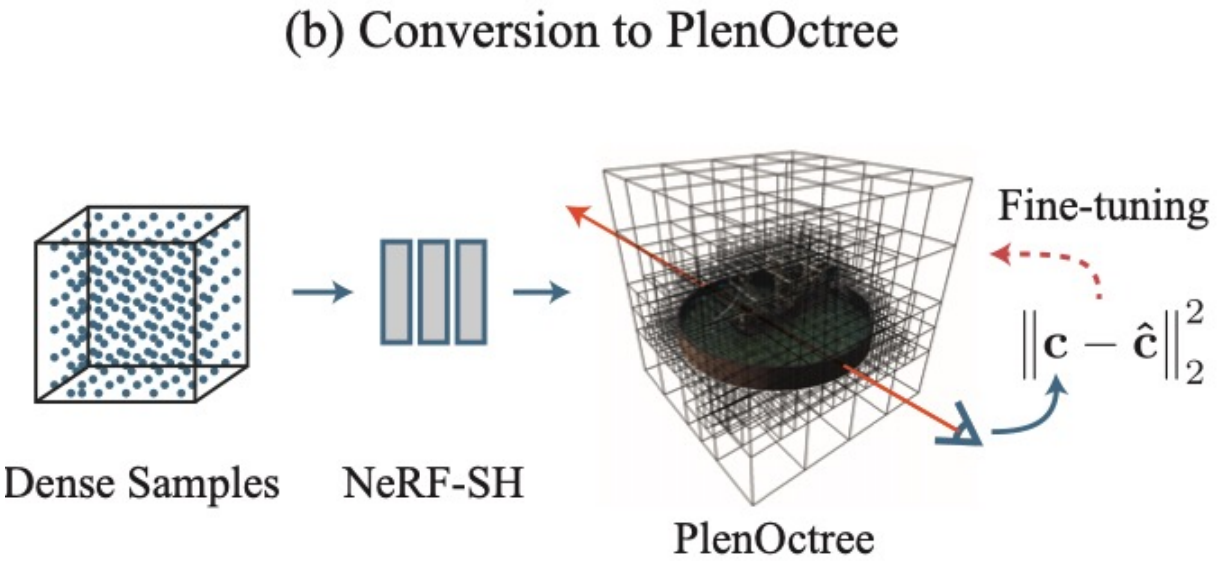
1. Train a NeRF-SH model
   - A variant of NeRF which directly outputs the spherical harmonics (SH) coefficients k
   - Color c is calculated by summing the weighted SH bases at the ray direction (θ, ϕ)
   - Eliminate the need for a view-direction input to the network
   - Instead, SH enables the representation to model view-dependent appearance
   - The other optimization procedure and volume rendering is the same as the NeRF



* source : Yu et. al., PlenOctrees for real-time rendering of neural radiance fields, ICCV 2021 Oral

- **Achieved real-time rendering by pre-tabulating the NeRF-SH into a PlenOctree**

2. Convert trained NeRF-SH to PlenOctree
   - With NeRF-SH, which eliminates view-dependency, conversion is more straightforward
   - Densely sample the NeRF-SH model and tabulate the density and SH coefficient
   - Further optimize the PlenOctree directly with the training images to improve quality



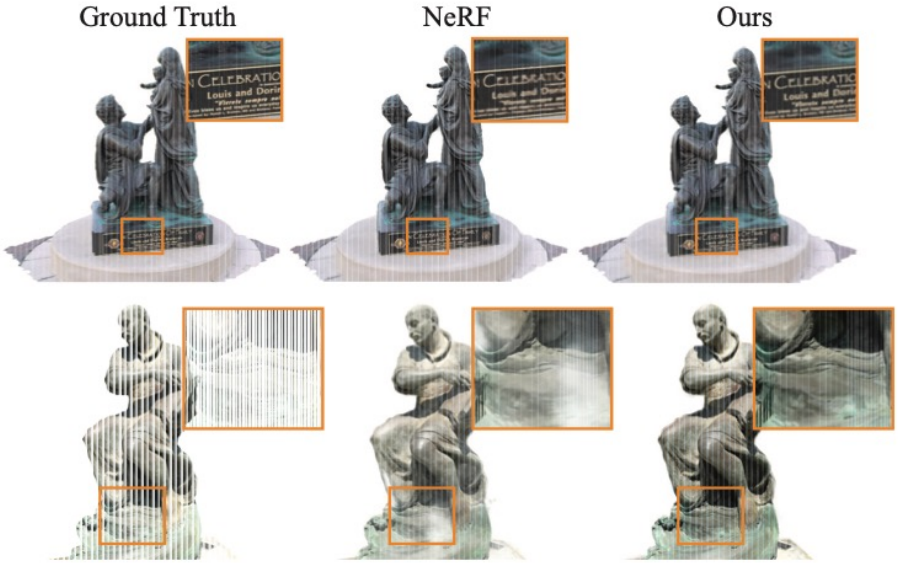
* source : Yu et. al., PlenOctrees for real-time rendering of neural radiance fields, ICCV 2021 Oral

PlenOctrees for Real-time Rendering of Neural Radiance Fields [Yu et al., 2021]

- **Achieved real-time rendering by pre-tabulating the NeRF-SH into a PlenOctree**

Synthetic NeRF Dataset — best — second-best

	PSNR ↑	SSIM ↑	LPIPS ↓	FPS ↑
NeRF (original)	31.01	0.947	0.081	0.023
NeRF	31.69	0.953	0.068	0.045
SRN	22.26	0.846	0.170	0.909
Neural Volumes	26.05	0.893	0.160	3.330
NSVF	31.75	0.953	0.047	0.815
AutoInt (8 sections)	25.55	0.911	0.170	0.380
NeRF-SH	31.57	0.952	0.063	0.051
PlenOctree from NeRF-SH	31.02	0.951	0.066	167.68
PlenOctree after fine-tuning	31.71	0.958	0.053	167.68



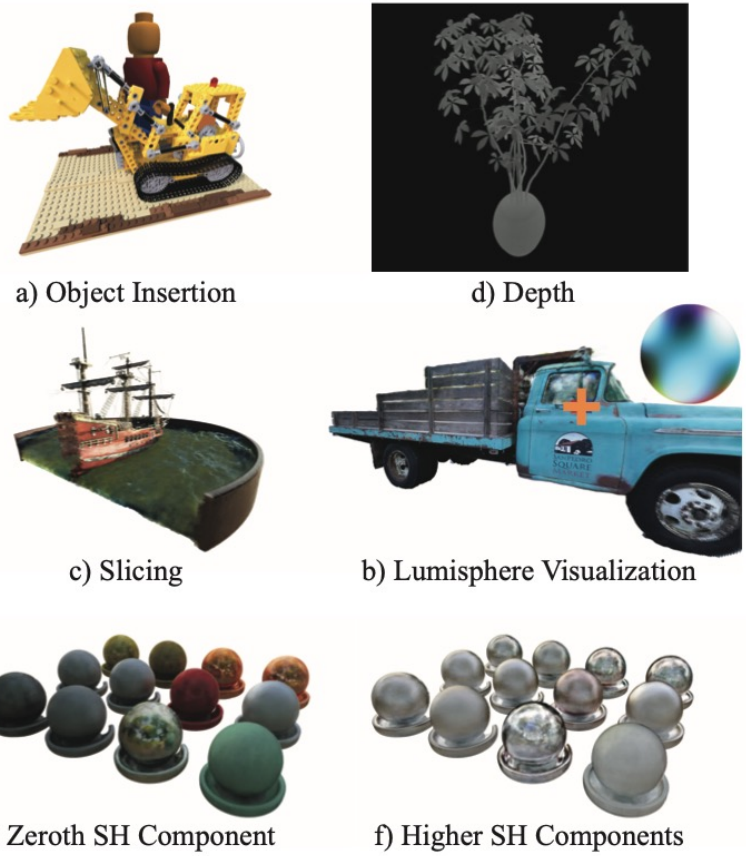


Figure 9: **Real-time interactive demos.** Set of real-time operations that can be performed on PlenOctree within our interactive viewer. This application will be released to the public.

* source : Yu et. al., PlenOctrees for real-time rendering of neural radiance fields, ICCV 2021 Oral

Table of Contents

Part 1. Basics

- Architectures: SIREN, FFN
- Neural radiance fields (NeRF)

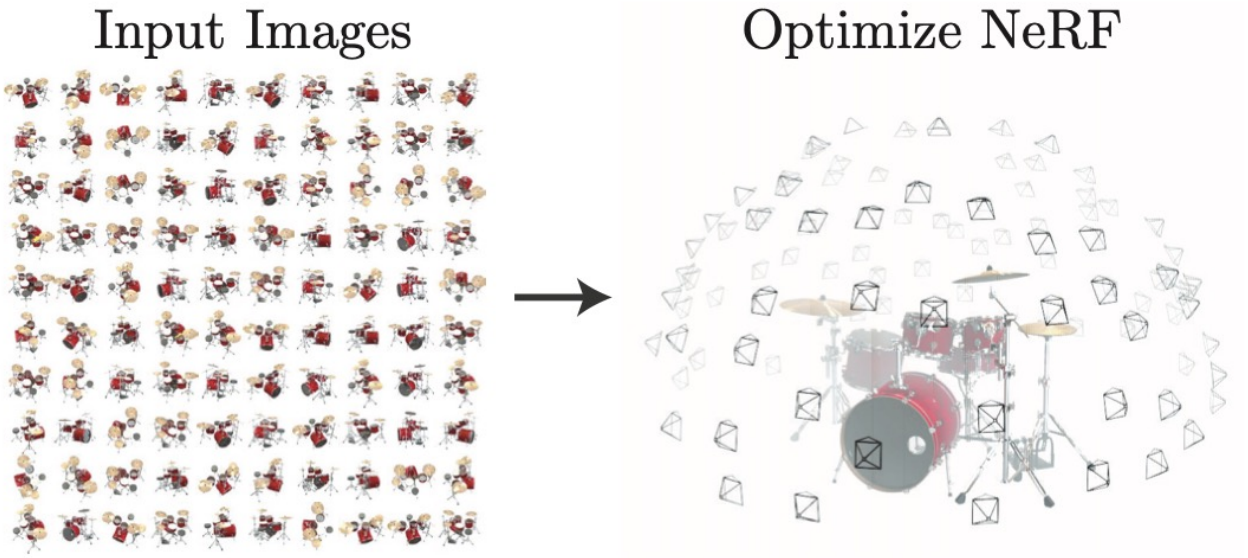
Part 2. Advanced Topics

- Techniques for faster training
- Techniques for real-time inference
- Training at real-world scenarios (e.g., deformable, unconstrained)

Part 3. Beyond Neural Fields of a Single Signal

- Generalization of neural fields on multiple signals

- **Challenge in training NeRF: practically unfeasible data collection**
  - NeRF assumes static scene (as images) from diverse and known camera poses



- Recent works have focused on training NeRF on more challenging situations:
  - (few-shot) limited number of input images
  - (unconstrained) training data from different sources (e.g., internet photo collection)
  - (deformable) training data from non-rigid, non-static scene (e.g., modeling people)

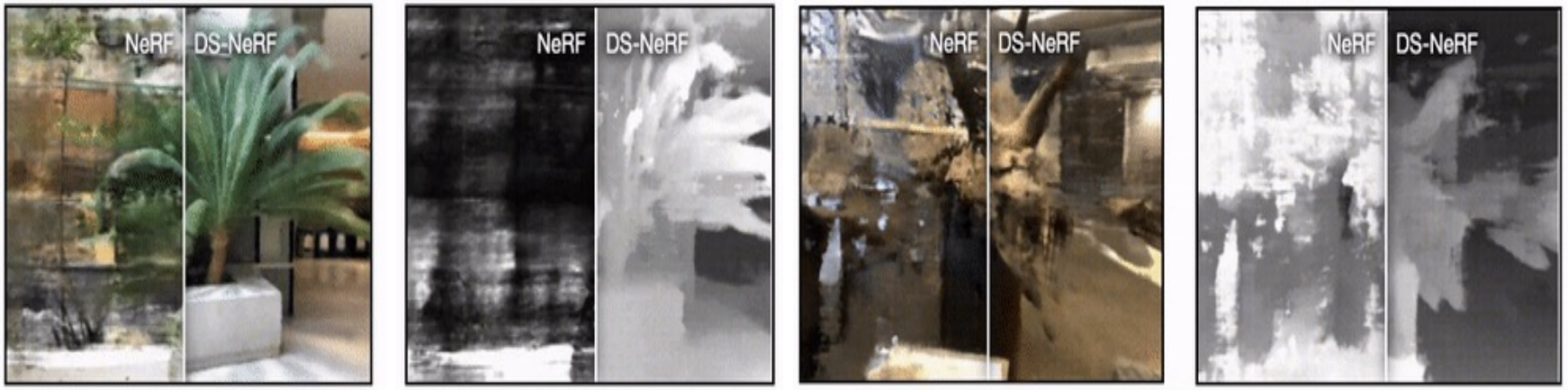
* source : Yu et. al., PlenOctrees for real-time rendering of neural radiance fields, ICCV 2021 Oral

Depth-Supervised NeRF: learned geometry regularization for few-shot [Deng et al., 2022]

- NeRF and its variants produce poor renderings of novel test views given a small number of train-views
- This overfitting is due to widely inaccurate geometry being learned

- Deng et al. (2022): **Depth-Supervised NeRF** (DS-NeRF)
  - DS-NeRF incorporates depth supervision terms to the loss function
  - Given only 2 views, depth supervision regularizes the learned geometry
  - Significantly reduces the required observation images and speeds up training

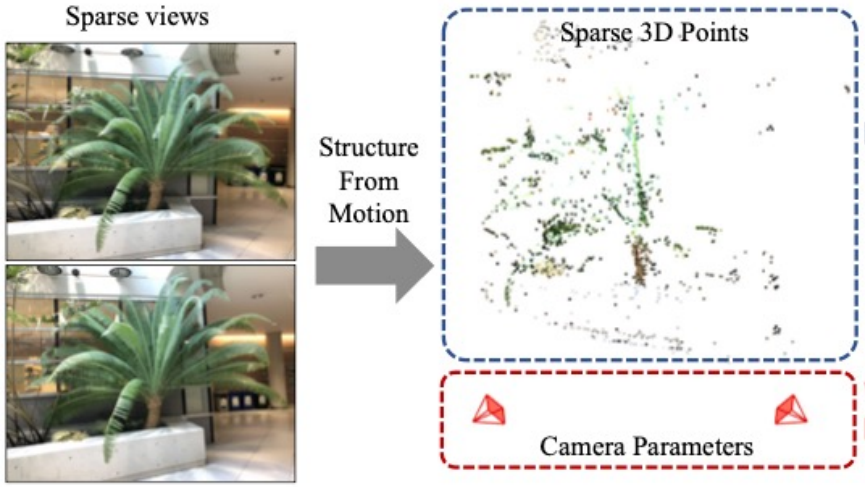
RGB Depth RGB Depth



* source : Deng, et. al., Depth-supervised NeRF: fewer views and faster training for free, CVPR 2022

Depth-Supervised NeRF: learned geometry regularization for few-shot [Deng et al., 2022]

- Depth-Supervised by 3D point clouds
  - A sparse 3D point cloud can be easily retrieved using structure-from-motion (SfM)
  - Depth derived from other sources (e.g., a depth camera, RGBD) can be used



  - Depth images from a particular pose may be rendered as the same with RGB pixels

$$\hat{\mathbf{D}}(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(t)t dt$$

  - $\hat{\mathbf{D}}(\mathbf{r})$ is the expected depth along the camera axis of the ray $\mathbf{r}$ and $T(t)$ is the occlusion checking integration

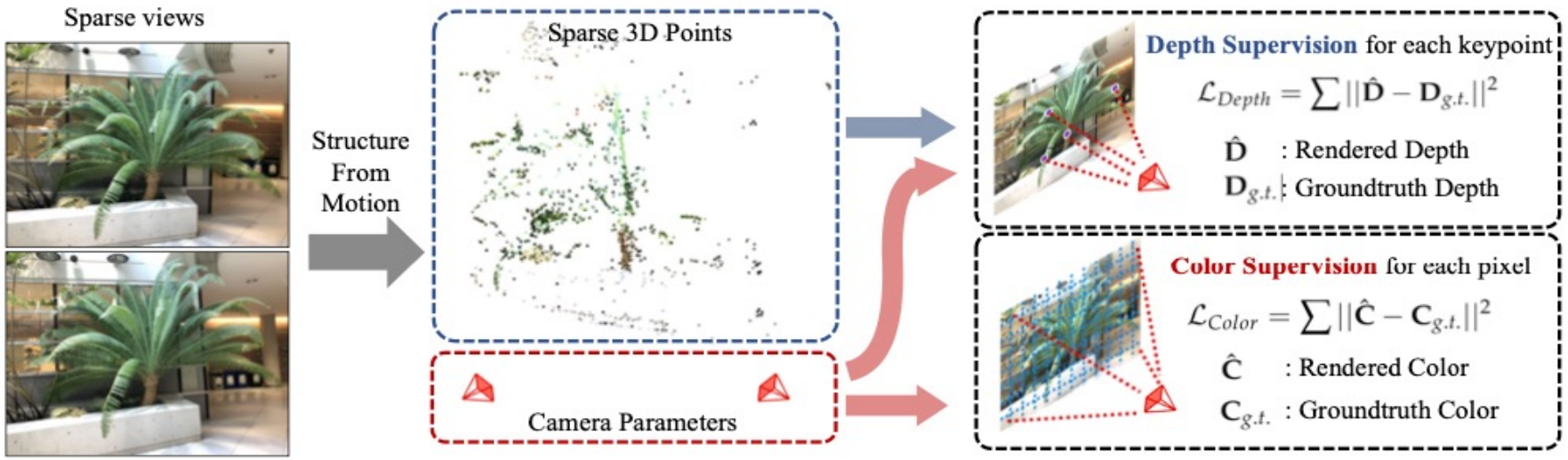
* source : Deng, et. al., Depth-supervised NeRF: fewer views and faster training for free, CVPR 2022

Depth-Supervised NeRF: learned geometry regularization for few-shot [Deng et al., 2022]

- Depth-Supervision loss
  - Implicitly convert key points $\mathbf{X}_j$ to their homogeneous representation

$$\mathcal{L}_{\text{Depth}} = \sum_{\mathbf{x}_i \in \mathbf{X}_j} w_i \left| \hat{\mathbf{D}}(\mathbf{r}_{ij}) - (\mathbf{P}_j \mathbf{x}_i) \cdot [0, 0, 1] \right|^2$$

  - $\mathbf{P}_j$ indicates estimated camera pose and $[0, 0, 1]$ indicates unit camera axis
  - To handle inaccuracy in the sparse reconstruction, the 3D points are confidence-weighted w_i according to their reprojection error
- Total training loss: $\mathcal{L} = \mathcal{L}_{\text{Color}} + \lambda_D \mathcal{L}_{\text{Depth}}$



* source : Deng, et. al., Depth-supervised NeRF: fewer views and faster training for free, CVPR 2022

Depth-Supervised NeRF: learned geometry regularization for few-shot [Deng et al., 2022]

- With the imposed depth loss, rendered depth along with rays that intersect 3D points are close to their observed depth

Depth err%↓	NeRF real-world			Redwood-3dscan		
	2-view	5-view	10-view	2-view	5-view	10-view
NeRF	20.32	15.00	12.41	25.32	24.34	21.34
metaNeRF-DTU	22.23	22.07	22.30	20.84	21.12	20.96
pixelNeRF-DTU	22.12	22.09	22.06	19.46	19.87	19.54
DS-NeRF	**10.41**	**8.61**	**8.15**	11.42	10.43	9.43
DS-NeRF w/ RGB-D	-	-	-	**5.81**	**5.31**	**4.22**

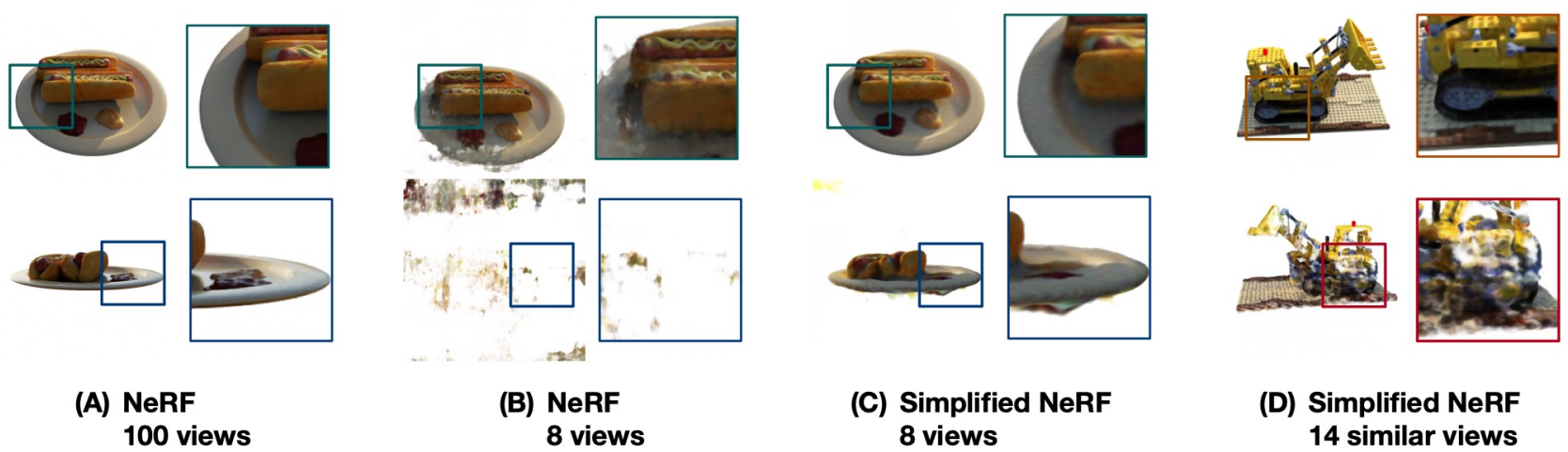
- Renders more accurate images given fewer views while training 2-6x faster

NeRF Real	PSNR↑			SSIM↑			LPIPS↓		
	2-view	5-view	10-view	2-view	5-view	10-view	2-view	5-view	10-view
LLFF	16.3	18.4	22.3	0.48	0.49	0.53	0.55	0.51	0.53
NeRF	18.2	22.6	24.2	0.50	0.57	0.58	0.52	0.50	0.52
metaNeRF-DTU	13.1	13.8	14.3	0.43	0.45	0.46	0.89	0.88	0.87
pixelNeRF-DTU	12.7	12.4	12.8	0.39	0.40	0.40	0.82	0.87	0.81
finetuned	20.2	23.1	24.1	0.56	0.59	0.63	0.53	0.53	0.41
finetuned with DS	20.9	23.1	24.4	0.54	0.61	0.66	0.55	0.47	0.42
IBRNet	18.4	22.9	24.3	0.50	0.51	0.54	0.53	0.54	0.51
finetuned with DS	22.2	24.3	**25.2**	0.63	0.66	0.68	0.39	**0.36**	0.38
DS-NeRF	**22.3**	**24.5**	**25.2**	**0.68**	**0.69**	0.69	**0.38**	0.37	0.37
w/o weighted loss	21.9	24.2	25.1	0.65	**0.69**	**0.71**	0.39	**0.36**	**0.36**

* source : Deng, et. al., Depth-supervised NeRF: fewer views and faster training for free, CVPR 2022

DietNeRF: Semantically Consistent Few-Shot View Synthesis [Jain et al., 2021]

- **Goal:** view-consistent few-shot synthesis
- **Challenge:** NeRF easily overfits at training views
  - Simple regularizations can mitigate the issue, but fail to capture fine details
  - Unobserved region of the similar object cannot be plausibly completed



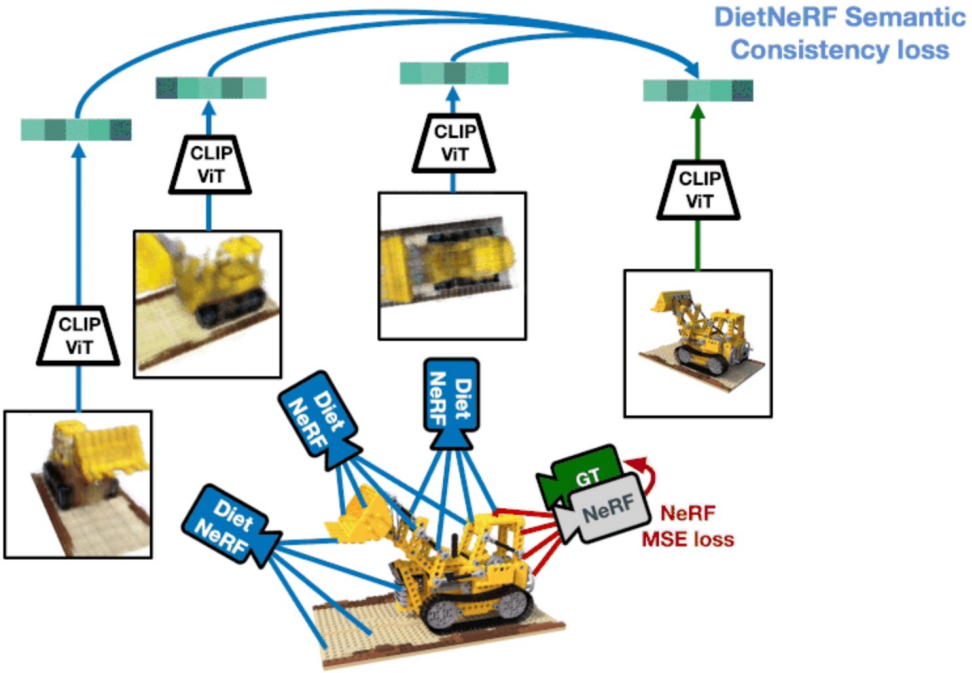
(A) NeRF 100 views | (B) NeRF 8 views | (C) Simplified NeRF 8 views | (D) Simplified NeRF 14 similar views

- **This paper:** proposes a new regularization for semantically consistent NeRF

* source : Deng, et. al., Depth-supervised NeRF: fewer views and faster training for free, CVPR 2022

DietNeRF: Semantically Consistent Few-Shot View Synthesis [Jain et al., 2021]

- **Main idea: semantic consistency regularization with large pre-trained model**
  - Mainly uses CLIP [Radford et al., 2021], a large image-text multimodal model



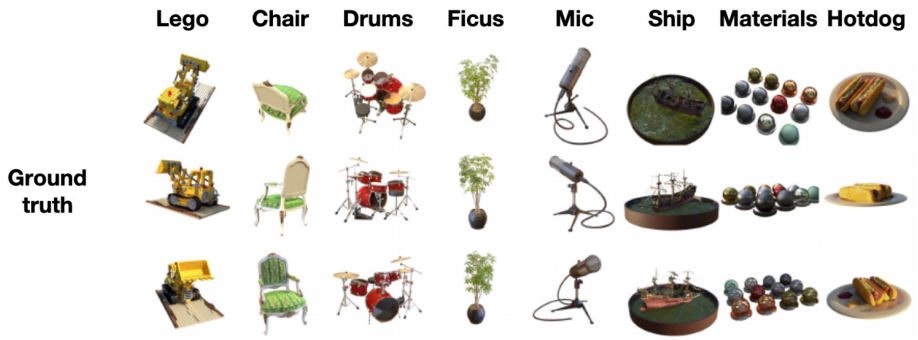
Algorithm 1: Training DietNeRF on a single scene

Data: Observed views $\mathcal{D} = \{(I, \mathbf{p})\}$, semantic embedding function $\phi(\cdot)$, pose distribution π, consistency interval K, weight λ, rendering size, batch size $|\mathcal{R}|$, lr η_{it}
Result: Trained Neural Radiance Field $f_\theta(\cdot, \cdot)$
Initialize NeRF $f_\theta(\cdot, \cdot)$;
Pre-compute target embeddings $\{\phi(I) : I \in \mathcal{D}\}$;
for *it from 1 to num_iters* **do**
  Sample ray batch $\mathcal{R}$, ground-truth colors $\mathbf{C}(\cdot)$;
  Render rays $\hat{\mathbf{C}}(\cdot)$ by (1);
  $\mathcal{L} \leftarrow \mathcal{L}_{\text{MSE}}(\mathcal{R}, \mathbf{C}, \hat{\mathbf{C}})$;
  if $it \,\%\, K = 0$ **then**
    Sample target image, pose $(I, \mathbf{p}) \sim \mathcal{D}$;
    Sample source pose $\hat{\mathbf{p}} \sim \pi$;
    Render image $\hat{I}$ from pose $\hat{\mathbf{p}}$;
    $\mathcal{L} \leftarrow \mathcal{L} + \mathcal{L}_{\text{SC}}(I, \hat{I})$;
  end
  Update parameters: $\theta \leftarrow Adam(\theta, \eta_{it}, \nabla_\theta \mathcal{L})$;
end

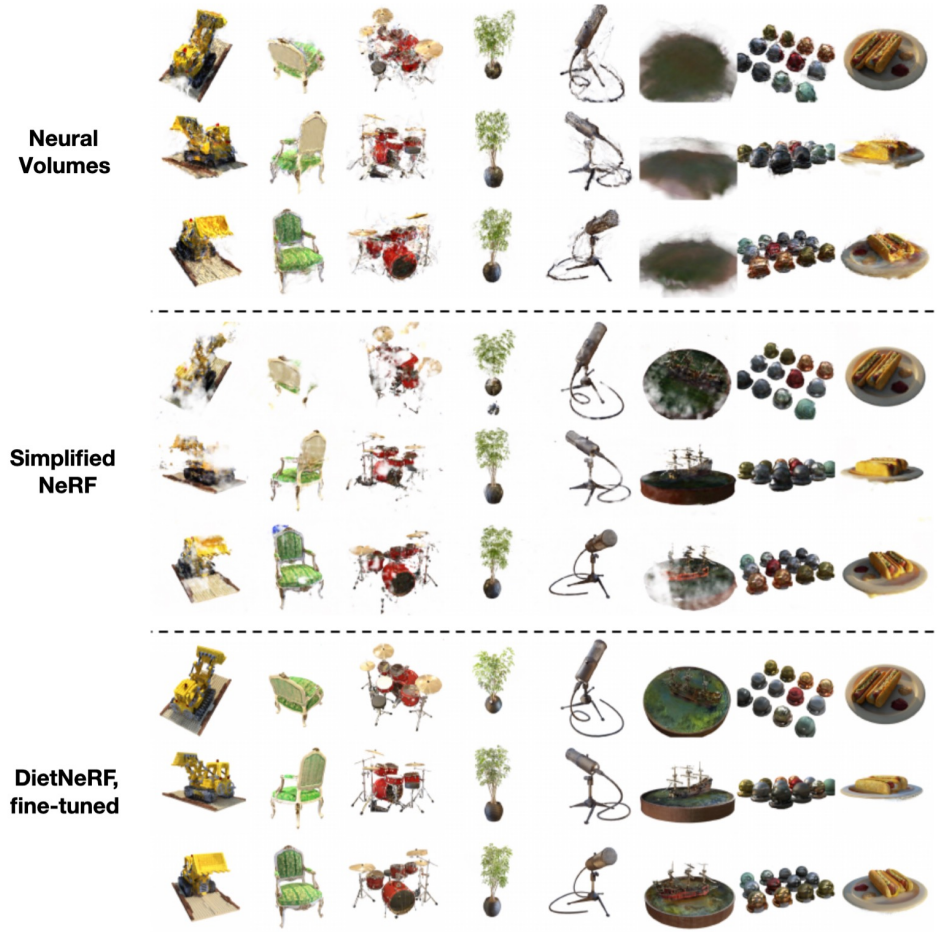
* source : Jain et al., Putting NeRF on a Diet: Semantically Consistent Few-Shot View synthesis, ICCV 2021

DietNeRF: Semantically Consistent Few-Shot View Synthesis [Jain et al., 2021]

- **DietNeRF shows reasonable reconstructions at novel views**
  - Even achieves single-view novel view synthesis for simple statics scenes



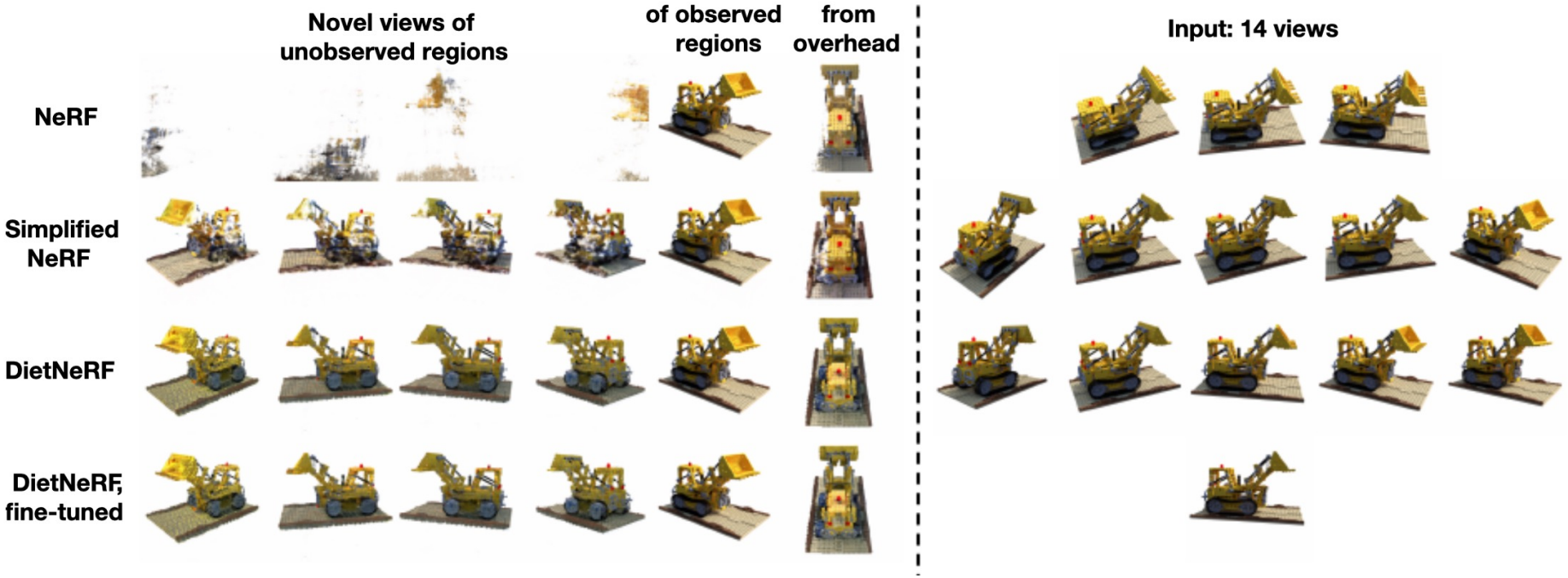
Method	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓	KID ↓
NeRF	14.934	0.687	0.318	228.1	0.076
NV	17.859	0.741	0.245	239.5	0.117
Simplified NeRF	20.092	0.822	0.179	189.2	0.047
DietNeRF (ours)	23.147	0.866	0.109	74.9	0.005
DietNeRF, $\mathcal{L}_{\text{MSE}}$ ft	**23.591**	**0.874**	**0.097**	**72.0**	**0.004**
NeRF, 100 views	**31.153**	**0.954**	**0.046**	**50.5**	**0.001**



* source : Jain et al., Putting NeRF on a Diet: Semantically Consistent Few-Shot View synthesis, ICCV 2021

DietNeRF: Semantically Consistent Few-Shot View Synthesis [Jain et al., 2021]

- **DietNeRF shows reasonable reconstructions at novel views**
  - Even achieves single-view novel view synthesis for simple statics scenes



* source : Jain et al., Putting NeRF on a Diet: Semantically Consistent Few-Shot View synthesis, ICCV 2021

- **DietNeRF shows reasonable reconstructions at novel views**
  - Even achieves single-view novel view synthesis for simple statics scenes

Method	PSNR	SSIM	LPIPS	FID	KID
NeRF	8.000	0.286	0.703	—	—
pixelNeRF	15.550	0.537	0.535	266.1	0.166
pixelNeRF, $\mathcal{L}_{\text{MSE}}$ ft	**16.048**	**0.564**	0.515	265.2	0.159
DietPixelNeRF	14.242	0.481	**0.487**	**190.7**	**0.066**

* source : Jain et al., Putting NeRF on a Diet: Semantically Consistent Few-Shot View synthesis, ICCV 2021

NeRF in the Wild: NeRF for Unconstrained Photo Collections [Matrin et al., 2021]

- **Goal:** reconstruction of 3D scenes from in-the-wild photo collections

- **Challenge:** caused by "unconstrained" data collection
  - **Photometric variation:** time of data and atmospheric condition impacts the data
  - **Transient objects:** often includes different human subjects and other pedestrians

* source: Martin et al., NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections, CVPR 2021

NeRF in the Wild: NeRF for Unconstrained Photo Collections [Matrin et al., 2021]

- **Latent appearance modeling:** for variable photometric post-processing
  - Apply Generative latent optimization to handle different lighting
  - Replace image-independent radiance to an image-"dependent" radiance $\mathbf{c}_i(t)$

$$\hat{\mathbf{C}}_i(\mathbf{r}) = \mathcal{R}(\mathbf{r}, \mathbf{c}_i, \sigma),$$

$$\mathbf{c}_i(t) = \mathrm{MLP}_{\theta_2}\left(\mathbf{z}(t), \gamma_{\mathbf{d}}(\mathbf{d}), \boldsymbol{\ell}_i^{(a)}\right)$$

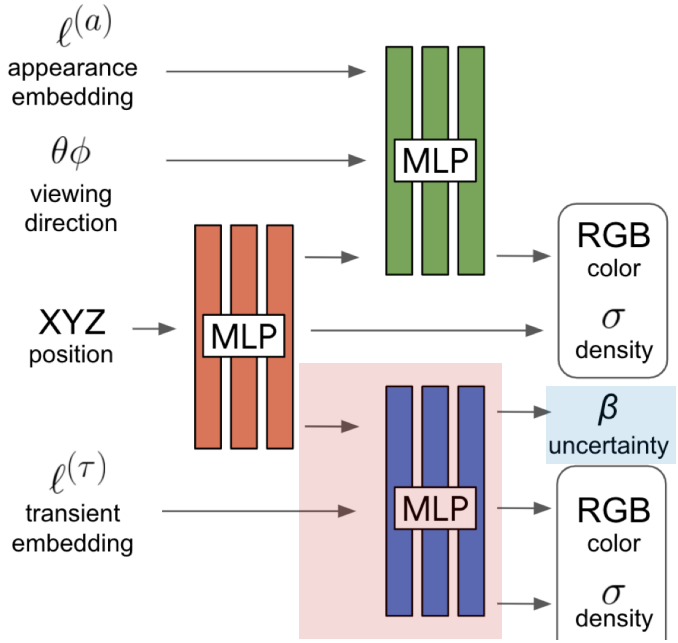
Image-dependent radiance Appearance embedding

Figure 8: Interpolations between the appearance embeddings $\boldsymbol{\ell}^{(a)}$ of two training images (left, right), which results in renderings (middle) where color and illumination are interpolated but geometry is fixed. Note that the training images contain people (left) and lights (right) that do not appear in the renderings. Photos by Flickr users mightyohm, blatez / CC BY.

* source : Martin et al., NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections, CVPR 2021

NeRF in the Wild: NeRF for Unconstrained Photo Collections [Matrin et al., 2021]

- **Transient objects:** use two different design decisions
  - Consider an additional "transient" head that the density can be vary across images
  - Transition head emits a field of uncertainty (to ignore unreliable pixels)



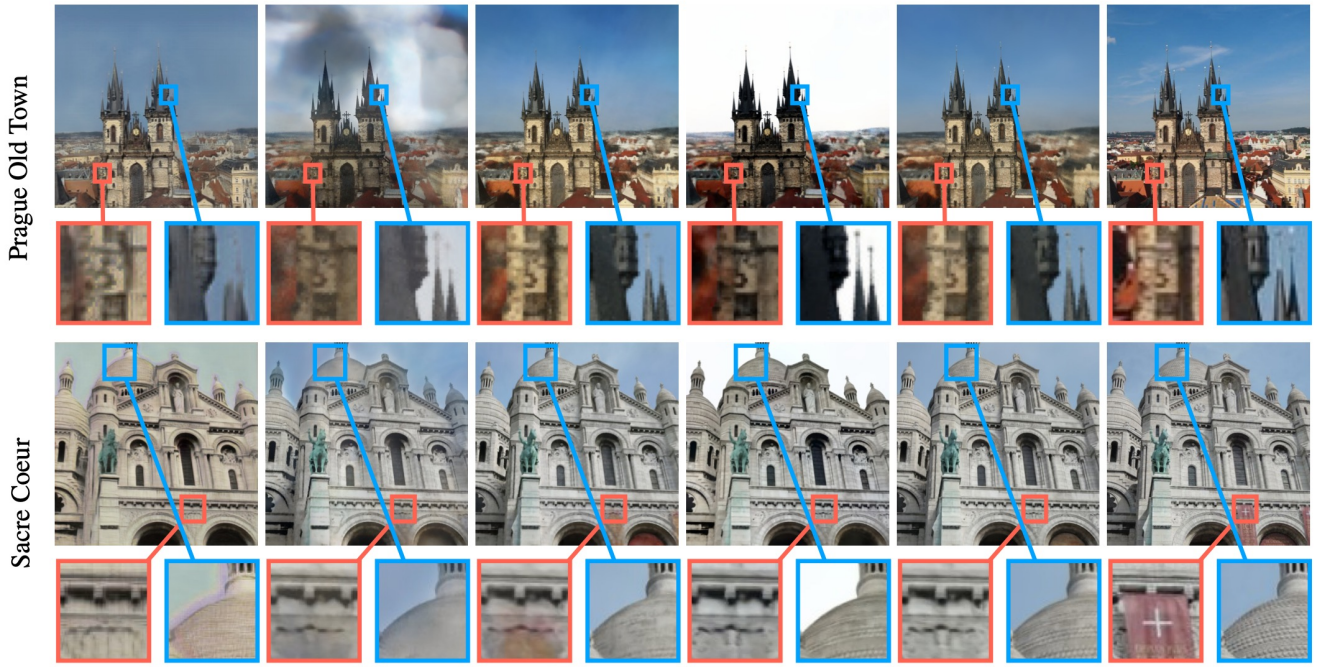
$$L_i(\mathbf{r}) = \frac{\left\|\mathbf{C}_i(\mathbf{r}) - \hat{\mathbf{C}}_i(\mathbf{r})\right\|_2^2}{2\beta_i(\mathbf{r})^2} + \frac{\log \beta_i(\mathbf{r})^2}{2} + \frac{\lambda_u}{K}\sum_{k=1}^{K}\sigma_i^{(\tau)}(t_k)$$

(a) Static (b) Transient (c) Composite (d) Image (e) Uncertainty

* source: Martin et al., NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections, CVPR 2021

NeRF in the Wild: NeRF for Unconstrained Photo Collections [Matrin et al., 2021]

- **NeRF-W shows better results on unconstrained (i.e., in the wild) situation**
  - Shows consistent background and captures details without artifacts



	Brandenburg Gate			Sacre Coeur			Trevi Fountain			Taj Mahal			Prague			Hagia Sophia		
	PSNR	MS-SSIM	LPIPS	PSNR	MS-SSIM	LPIPS	PSNR	MS-SSIM	LPIPS	PSNR	MS-SSIM	LPIPS	PSNR	MS-SSIM	LPIPS	PSNR	MS-SSIM	LPIPS
NRW [22]	23.85	0.914	0.141	19.39	0.797	0.229	20.56	0.811	0.242	21.24	0.844	**0.201**	19.89	0.803	**0.216**	20.75	0.796	**0.231**
NeRF	21.05	0.895	0.208	17.12	0.781	0.278	17.46	0.778	0.334	15.77	0.697	0.427	15.67	0.747	0.362	16.04	0.749	0.338
NeRF-A	27.96	0.941	0.145	24.43	0.923	0.174	26.24	0.924	0.211	25.99	0.893	0.225	22.52	0.870	0.244	21.83	0.820	0.276
NeRF-U	19.49	0.921	0.174	15.99	0.826	0.223	15.03	0.795	0.277	10.23	0.778	0.373	15.03	0.787	0.315	13.74	0.706	0.376
NeRF-W	**29.08**	**0.962**	**0.110**	**25.34**	**0.939**	**0.151**	**26.58**	**0.934**	**0.189**	**26.36**	**0.904**	0.207	**22.81**	**0.879**	0.227	**22.23**	**0.849**	0.250

* source: Martin et al., NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections, CVPR 2021

Nerfies: Deformable Neural Radiance Fields [Park et al., 2021]

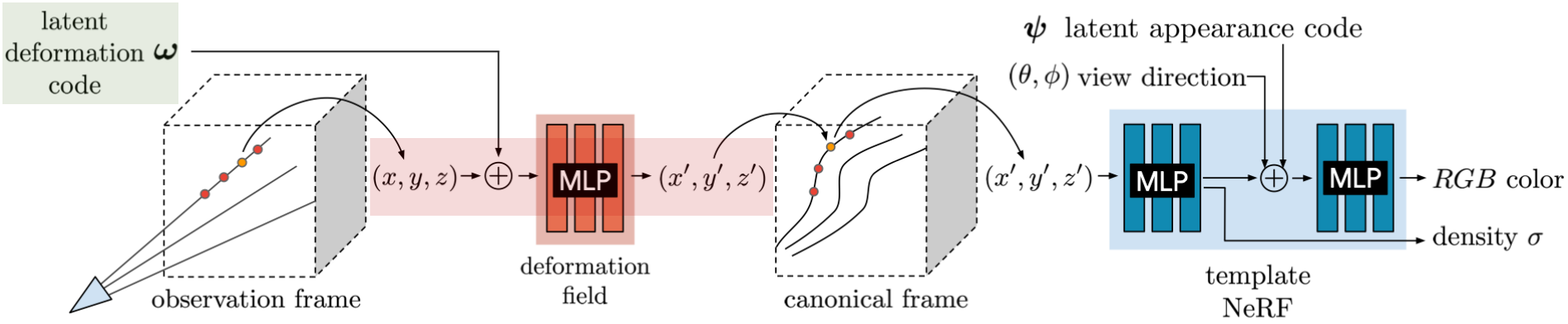
- **Goal: photo-realistic 3D scene from a user waving a mobile phone**

(a) casual capture (b) input images (c) nerfie novel views (d) novel view depth

- **Challenge: characteristic of humans from static objects (e.g., buildings)**
  - **Nonrigidity:** instability to stay perfectly still
  - **Challenging material:** violate assumptions in 3D reconstruction (e.g., glasses)

* source: Park et al., Nerfies: Deformable Neural Radiance Fields, ICCV 2021

- **Main idea: joint training of NeRF and deformation mapping**
  - Deformation mapping: transforms an observation coordinate to a canonical space
  - Associated with deformation code (per-frame learned)



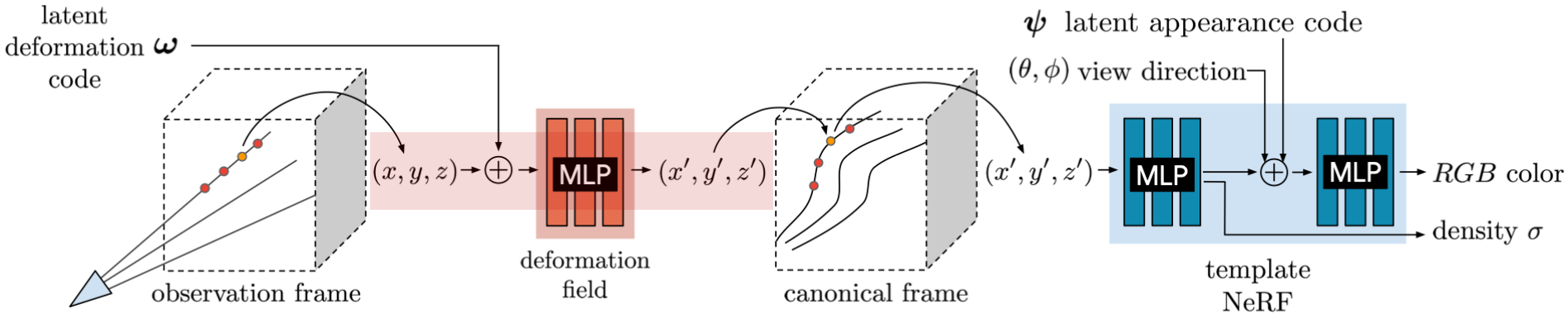
Deformation code

$$G(\mathbf{x}, \mathbf{d}, \boldsymbol{\psi}_i, \boldsymbol{\omega}_i) = F\left(T(\mathbf{x}, \boldsymbol{\omega}_i), \mathbf{d}, \boldsymbol{\psi}_i\right)$$

Appearance code (like NeRF-W)

Deformation mapping (MLP)

- **Deformation mapping T: uses a dense SE(3) field $W : (\mathbf{x}, \boldsymbol{\omega}_i) \rightarrow (\mathbf{r}, \mathbf{v})$**
  - Inputs are mapped to the rotation axis and the angle of rotation
  - Encodes rigid motion, allowing us to rotate a set of distant points



$$\mathbf{x}' = e^{\mathcal{S}}\mathbf{x} = e^{\mathbf{r}}\mathbf{x} + \mathbf{p}$$

$$e^{\mathbf{r}} \equiv e^{[\mathbf{r}]_\times} = \mathbf{I} + \frac{\sin\theta}{\theta}[\mathbf{r}]_\times + \frac{1-\cos\theta}{\theta^2}[\mathbf{r}]_\times^2$$

$$\mathbf{G} = \mathbf{I} + \frac{1-\cos\theta}{\theta^2}[\mathbf{r}]_\times + \frac{\theta-\sin\theta}{\theta^3}[\mathbf{r}]_\times^2$$

* source: Park et al., Nerfies: Deformable Neural Radiance Fields, ICCV 2021

- **Elastic regularization: helps to construct under-constrained scene better**
  - Regularizes “elastic energy”: deviation of local deformation from rigid motions
  - Promotes details (e.g., glasses) place properly

$$L_{\text{elastic}}(\mathbf{x}) = \|\log \boldsymbol{\Sigma} - \log \mathbf{I}\|_F^2 = \|\log \boldsymbol{\Sigma}\|_F^2$$

$$\mathbf{J}_T = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T,$$

Jacobian of the deformation field at the input coordinate $\mathbf{x} \in \mathbb{R}^3$

* source: Park et al., Nerfies: Deformable Neural Radiance Fields, ICCV 2021

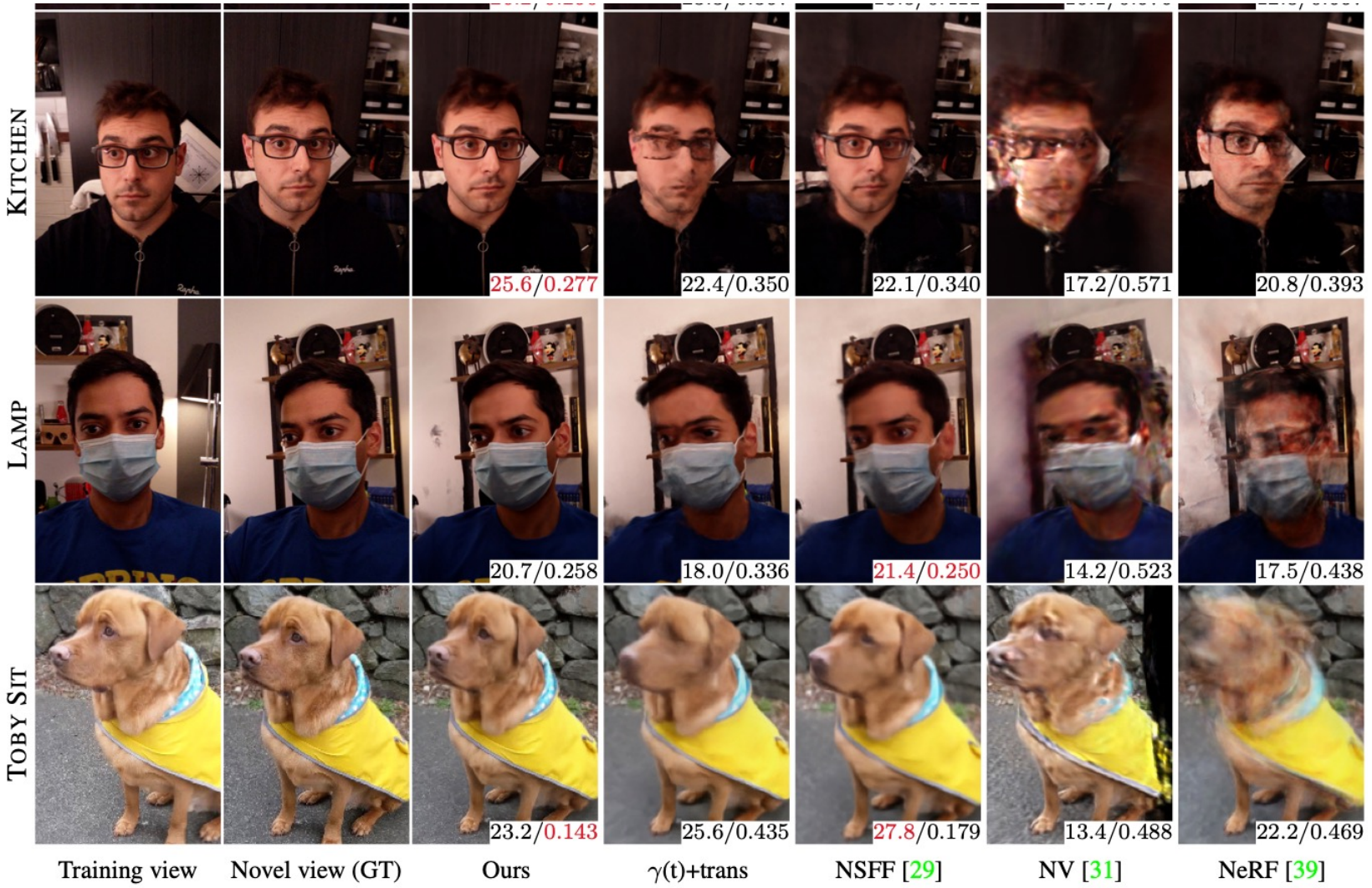
- **Background regularization: regularizes background to move a lot**
  - Since deformation field is unconstrained, everything is free to move around
  - Regularizes the deformation fields to prevent dramatic change of coordinates

$$L_{\text{bg}} = \frac{1}{K} \sum_{k=1}^{K} \left\| T(\mathbf{x}_k) - \mathbf{x}_k \right\|_2$$

* source: Park et al., Nerfies: Deformable Neural Radiance Fields, ICCV 2021

Nerfies: Deformable Neural Radiance Fields [Park et al., 2021]

- **Shows consistently improved novel view synthesis in unconstrained situation**



* source: Park et al., Nerfies: Deformable Neural Radiance Fields, ICCV 2021

Table of Contents

Part 1. Basics

- Architectures: SIREN, FFN
- Neural radiance fields (NeRF)

Part 2. Advanced Topics

- Techniques for faster training
- Techniques for real-time inference
- Training at real-world scenarios (e.g., deformable, unconstrained)

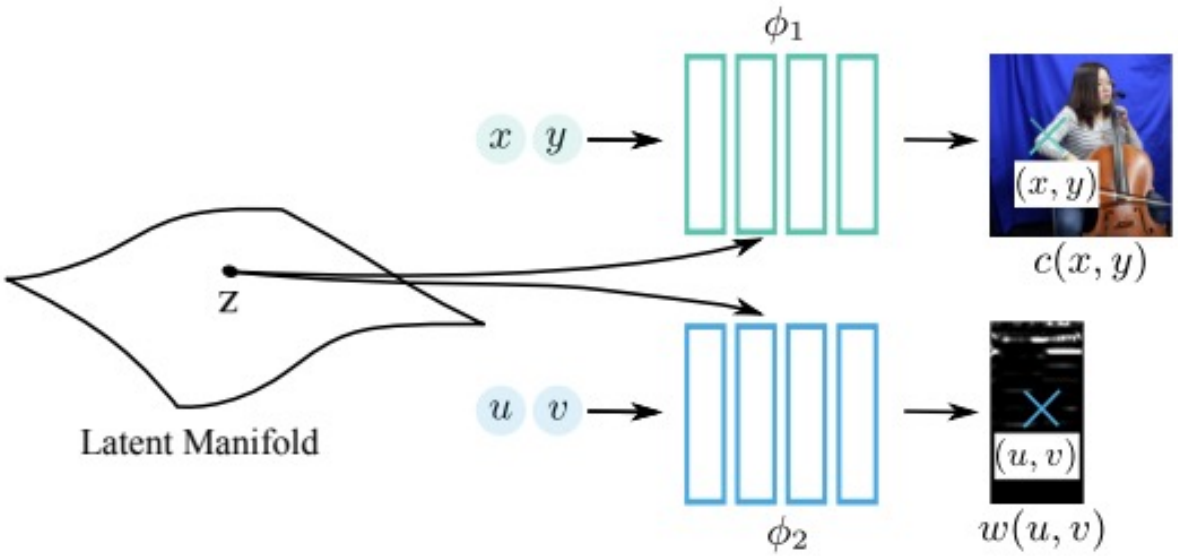
Part 3. Beyond Neural Fields of a Single Signal

- Generalization of neural fields on multiple signals

GEM: signal-agnostic manifolds of neural fields [Du et al., 2021]

- Existing models are generally modality-dependent

- Du et al. (2021): **Leverage neural fields to capture the underlying structure**
  - Image, shape, audio, and cross-modal audiovisual domains
  - Modality-independent manner



* source : Du et al., Learning Signal-Agnostic Manifolds of Neural Fields, NeurIPS 2021

- **Characteristics of Manifolds**
  - **Exhibit data coverage**: All instances are explained in the low-dimensional space
  - **Locally metric**: Enables perceptual manipulation
  - **Globally-consistent**: Similar signals should be embedded close to one another

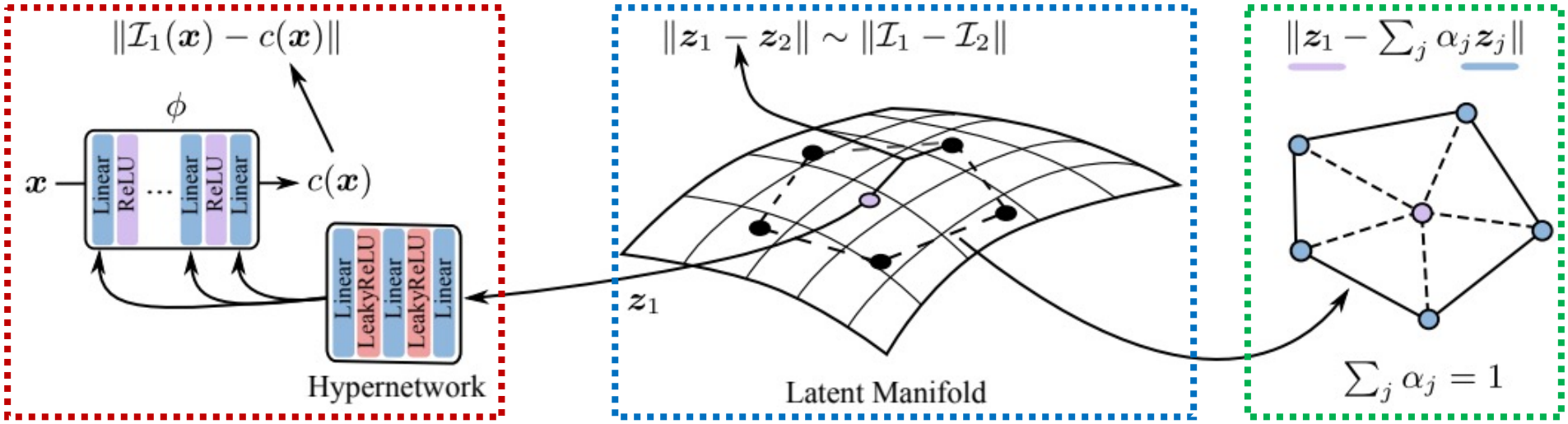


Exhibit data coverage

$$\mathcal{L}_{\text{Rec}} = \|\Psi(\boldsymbol{z}_i)(\boldsymbol{x}) - \mathcal{I}_i(\boldsymbol{x})\|^2$$

Globally-consistent

$$\mathcal{L}_{\text{Iso}} = \|\alpha * \|(\boldsymbol{z}_i - \boldsymbol{z}_j)\| - \|\mathcal{I}_i - \mathcal{I}_j\|\|$$

Locally metric

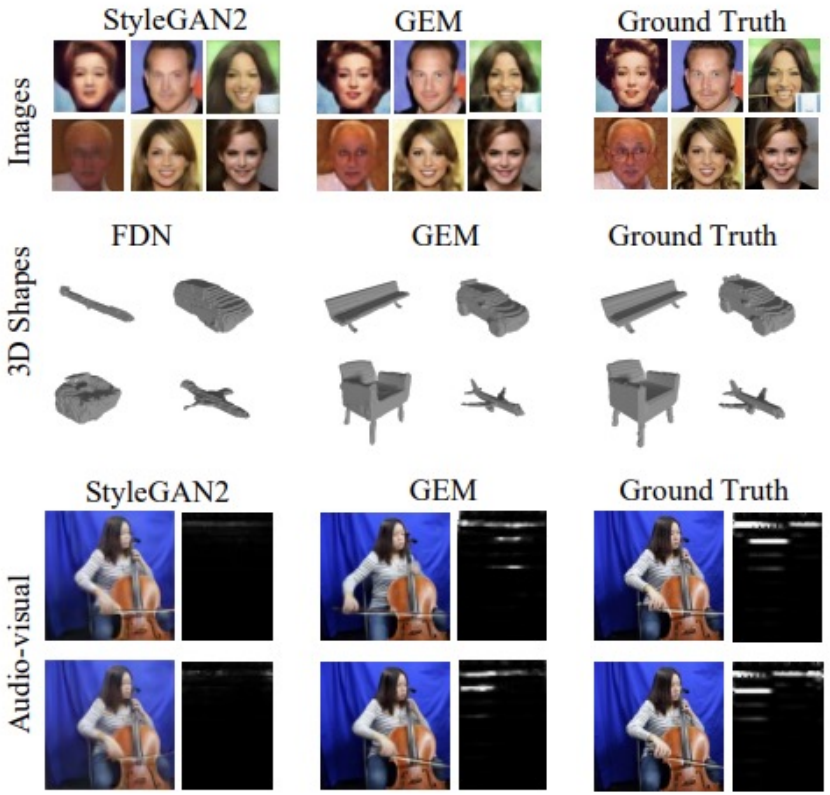
$$\mathcal{L}_{\text{LLE}} = \|\Psi(\boldsymbol{z}_i')(\boldsymbol{x}) - \mathcal{I}_i(\boldsymbol{x})\|^2$$

$$\mathcal{L}_{\text{Total}} = \mathcal{L}_{\text{Rec}} + \mathcal{L}_{\text{LLE}} + \mathcal{L}_{\text{Iso}}$$

* source : Du et al., Learning Signal-Agnostic Manifolds of Neural Fields, NeurIPS 2021

GEM: signal-agnostic manifolds of neural fields [Du et al., 2021]

- GEM obtains the sharpest samples on modality fitting experiment

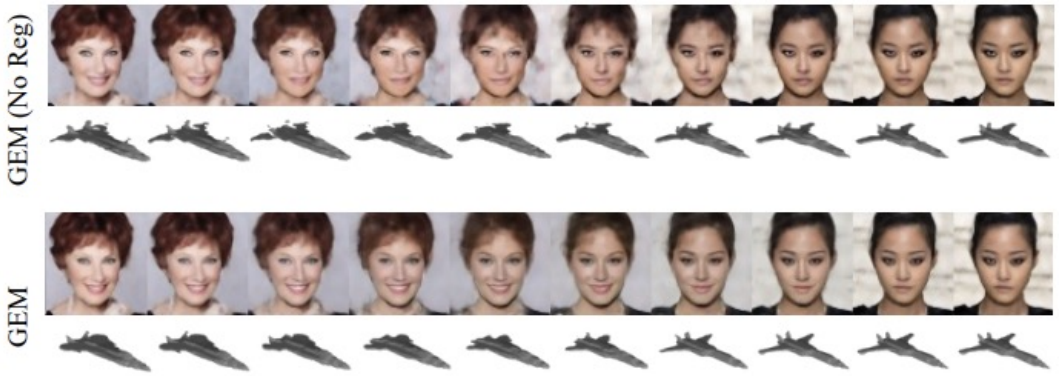


Modality	Model	MSE ↓	PSNR ↑
Images	Heat Kernel	0.0365	14.99
	VAE	0.0331	15.12
	FDN	0.0060	22.59
	StyleGAN2	0.0044	24.03
	GEM	**0.0025**	**26.53**
Audio	VAE	0.0147	18.87
	FDN	0.0050	23.68
	StyleGAN2	0.0015	28.52
	GEM	**0.0011**	**29.98**
ShapeNet	FDN	0.0296	16.52
	GEM	**0.0153**	**21.32**
Image and Audio	VAE	0.0193	17.23
	FDN	0.0663	11.78
	StyleGAN2	0.0063	22.36
	GEM	**0.0034**	**24.38**

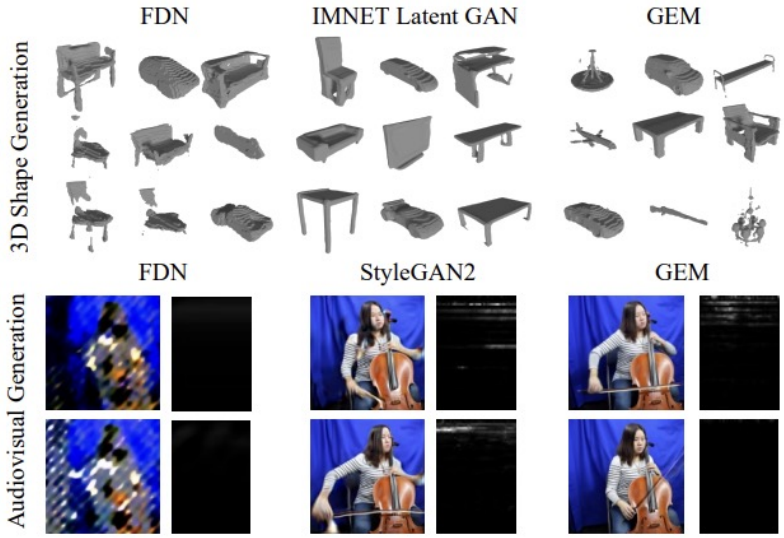
* source : Du et al., Learning Signal-Agnostic Manifolds of Neural Fields, NeurIPS 2021

GEM: signal-agnostic manifolds of neural fields [Du et al., 2021]

- GEM is able to make perceptually continuous interpolations



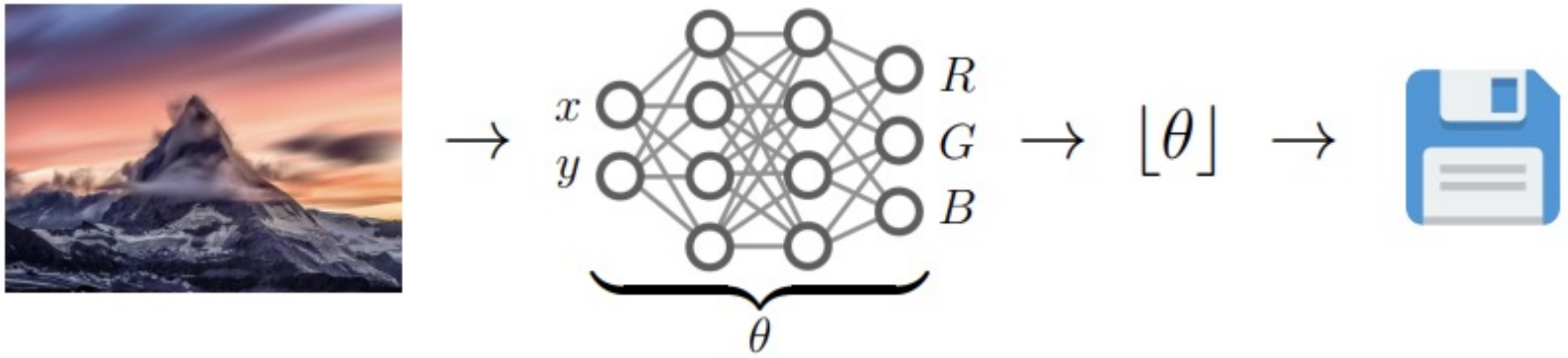
- GEM produces reasonable samples across the shape and audiovisual modalities



* source : Du et al., Learning Signal-Agnostic Manifolds of Neural Fields, NeurIPS 2021

COIN: compression with INRs [Dupont et al., 2021]

- Dupont et al. (2021): Image compression **with implicit neural representation**
  - **Encode:** Fit INR, then quantize and store the weights of INR
  - **Decode:** Evaluate INR at every pixel location



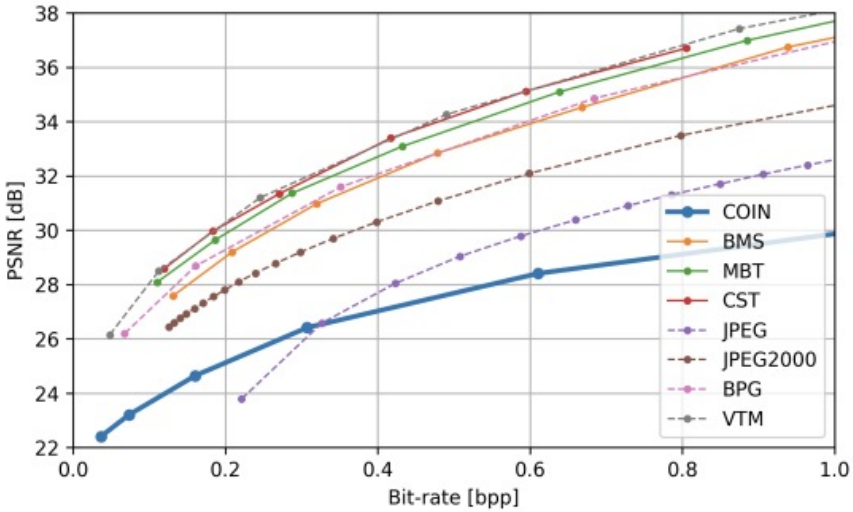


Figure 2: Rate distortion plots on the Kodak dataset.

* source : Dupont et al., COIN: Compression with Implicit Neural representations, ICLR Workshop 2021

COIN: compression with INRs [Dupont et al., 2021]

- Does not require a decoder at test time
- COIN outperforms JPEG

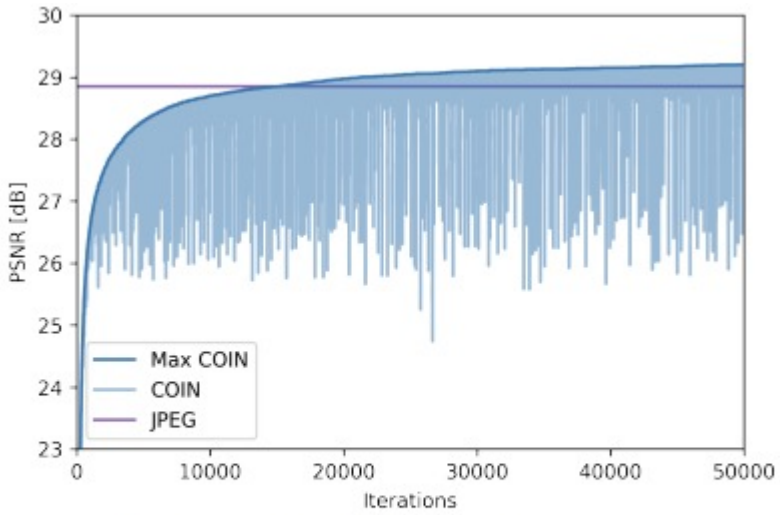


Figure 4: Model training on image 15 in the Kodak dataset. Max COIN represents the max PSNR achieved at any point during training.

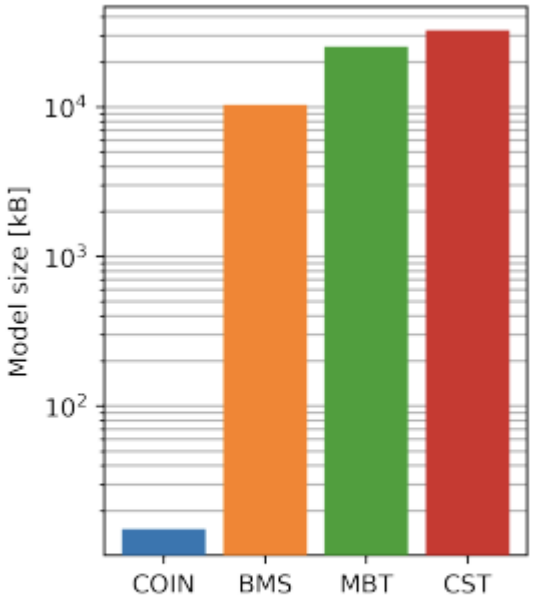


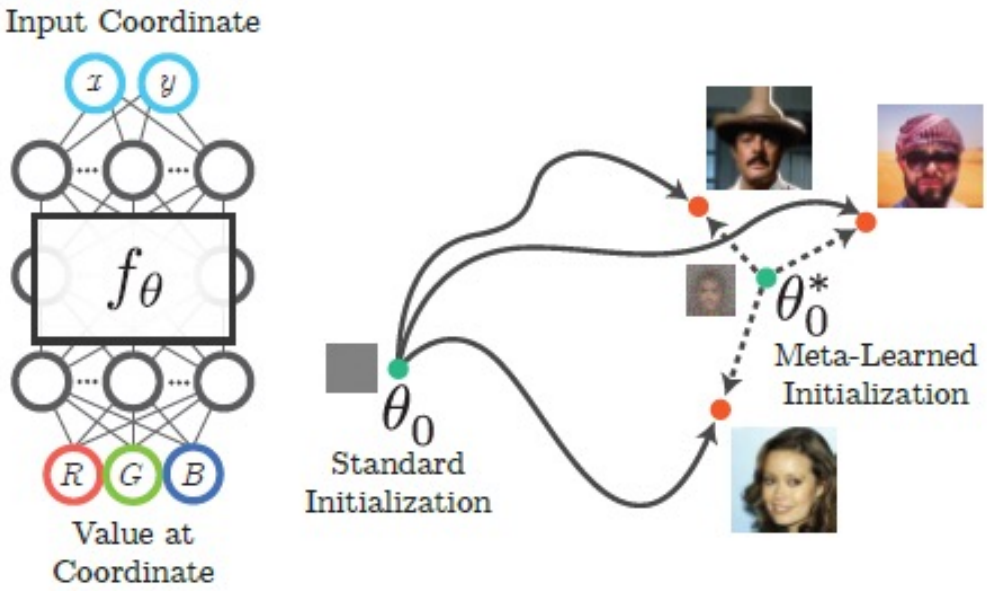
Figure 3: Model sizes at 0.3bpp.

* source : Dupont et al., COIN: Compression with Implicit Neural representations, ICLR Workshop 2021

Meta-Initialization: learned initializations for INRs [Tancik et al., 2021]

- INRs have shown significant promise as an alternative to discrete
  - However, optimizing an INR from randomly initialized weights is inefficient

- Tancik et al. (2021): **Use meta-learning to learn the good initial weight**
  - Faster convergence
  - Better generalization



* source : Tancik et al., Learned Initializations for Optimizing Coordinate-Based Neural Representations, CVPR 2021

Meta-Initialization: learned initializations for INRs [Tancik et al., 2021]

- **Meta-learning**

  optimal initial weight $\theta_0^* = \arg\min_{\theta_0} E_{T\sim\mathcal{T}}[L(\theta_m(\theta_0, T))]$

  weight after m steps

- **MAML [Finn et al., 2017]**
  - Each outer loop samples a signal T_j and applies the update rule:

$$\theta_0^{j+1} = \theta_0^j - \beta\nabla_\theta L(\theta_m(\theta, T_j))|_{\theta=\theta_0^j}$$

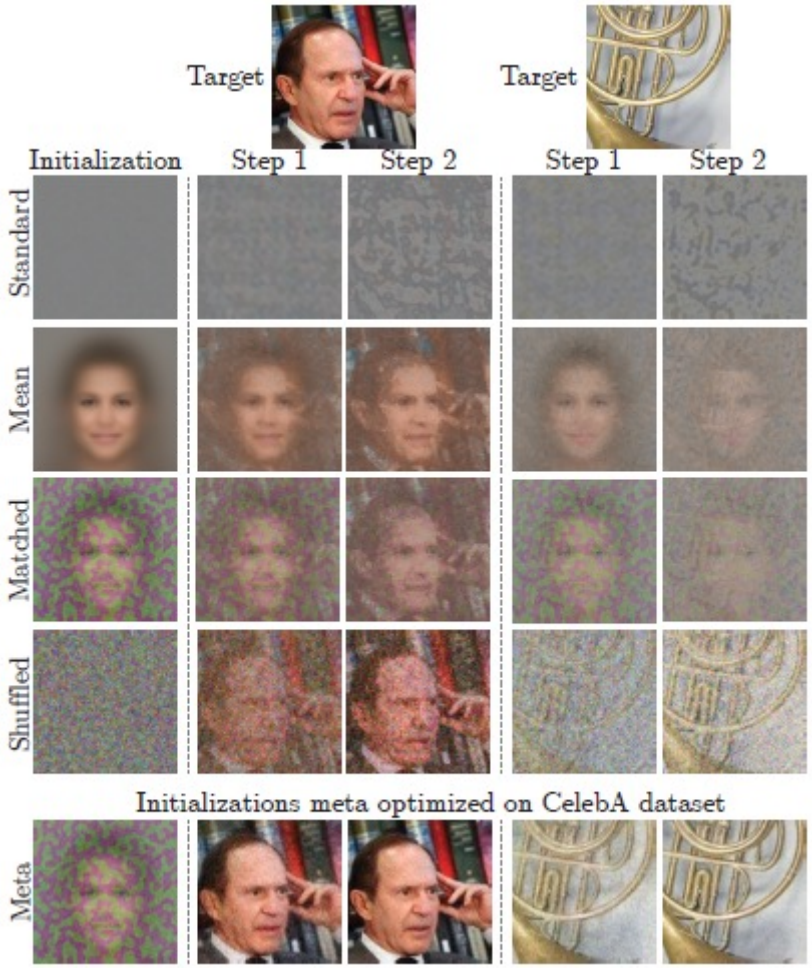
- **Reptile [Nichol et al., 2017]**
  - Same meta-learning setup as MAML
  - Applies an update rule that does not require calculating second-order gradients

$$\theta_0^{j+1} = \theta_0^j - \beta(\theta_m(\theta_0^j, T_j) - \theta_0^j)$$

* source : Tancik et al., Learned Initializations for Optimizing Coordinate-Based Neural Representations, CVPR 2021

Meta-Initialization: learned initializations for INRs [Tancik et al., 2021]

- Faster convergence on image regression experiment

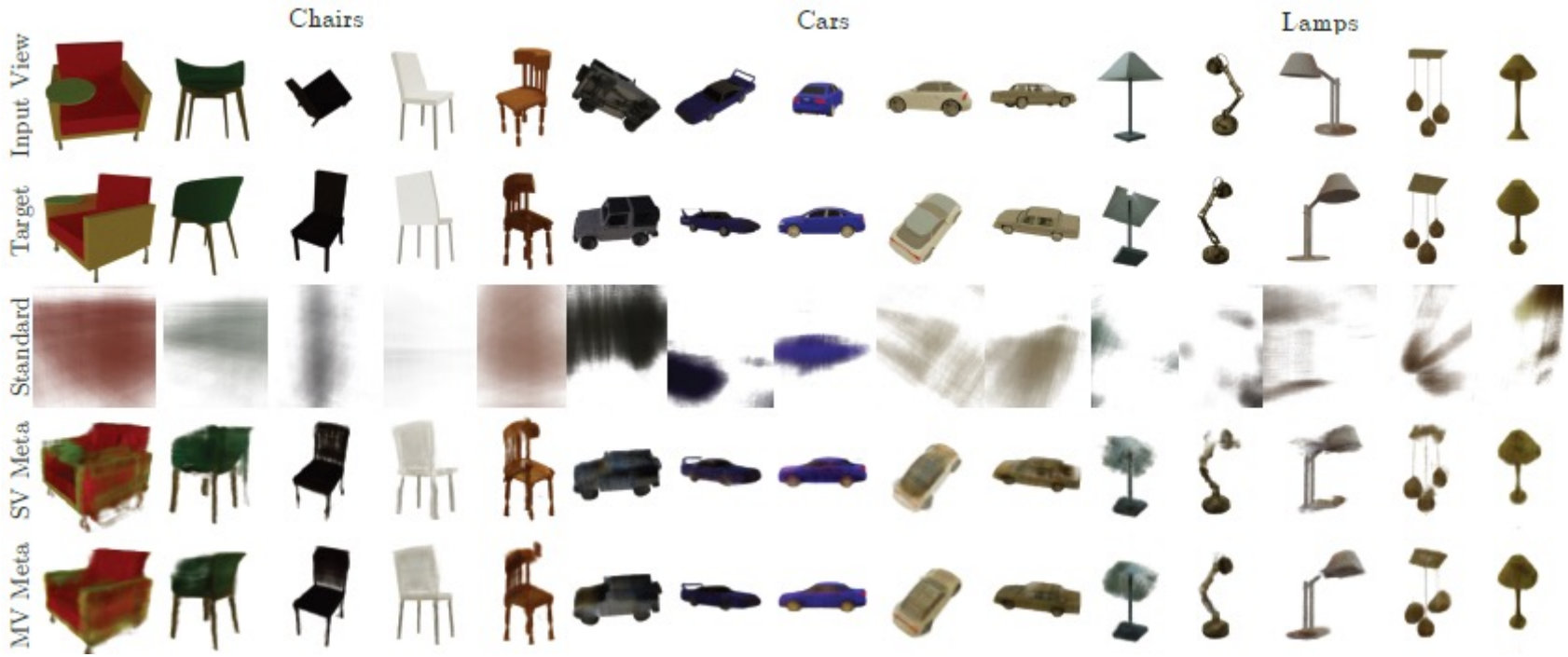


Init. Method	2 Step PSNR ↑	# of iters to match ↓
Standard	10.88	37.92 ± 6.31
Mean	14.48	25.59 ± 4.57
Matched	13.73	26.32 ± 4.17
Shuffled	16.29	25.80 ± 4.02
Meta	**30.37**	-

* source : Tancik et al., Learned Initializations for Optimizing Coordinate-Based Neural Representations, CVPR 2021

Meta-Initialization: learned initializations for INRs [Tancik et al., 2021]

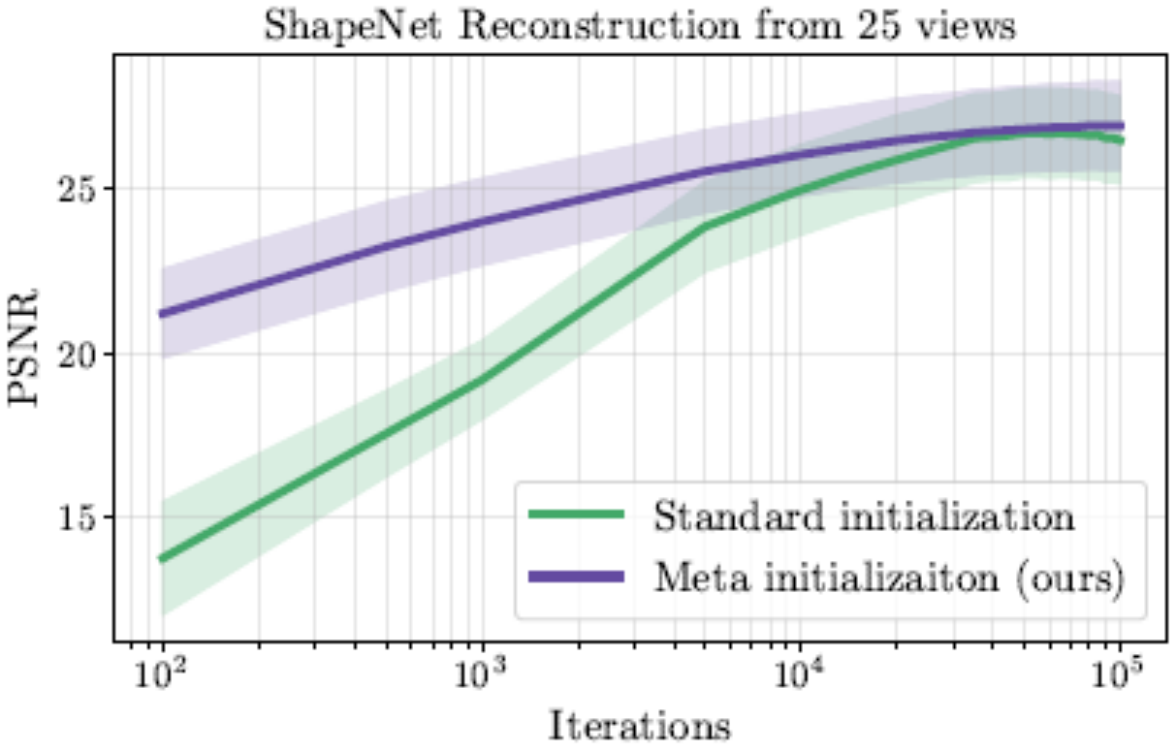
- Single view reconstructions of ShapeNet objects
  - With only a single view, NeRF is unable to recover any meaningful shape
  - However, if it is trained starting from meta-init, it is able to recover 3D geometry



* source : Tancik et al., Learned Initializations for Optimizing Coordinate-Based Neural Representations, CVPR 2021

Meta-Initialization: learned initializations for INRs [Tancik et al., 2021]

- View synthesis for ShapeNet objects
  - Reconstruct ShapeNet chairs from a set of 25 reference images

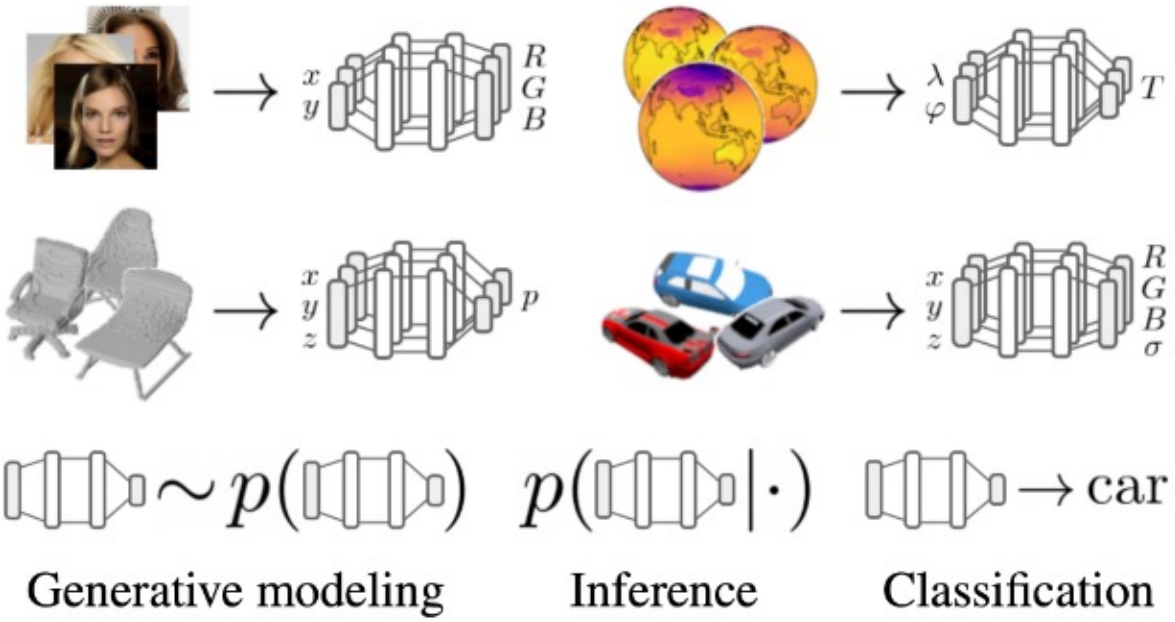


* source : Tancik et al., Learned Initializations for Optimizing Coordinate-Based Neural Representations, CVPR 2021

Functa: treating INRs as data [Dupont et al., 2022]

- It is common to represent a measurement of the world on a discrete gird
  - However, the underlying signal is often continuous

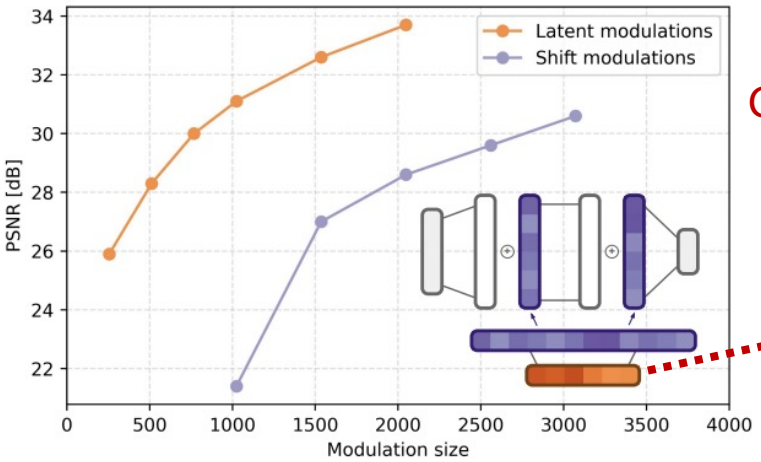
- Dupont et al. (2022): **Use implicit neural representation as a data point**
  - **Step 1**: Fit data points into INRs
  - **Step 2**: Run downstream tasks, e.g., classification



* source : Dupont et al., From data to functa: Your data point is a function and you should treat it like one, ICML 2022

- **Two main challenges**
  - how to fit INRs in a (dimensionally) compact manner: Use modulation
  - how to fit INRs in a (time and memory) scalable manner: Use meta-learning

- Modulation for INRs
  - Modulation denotes scaling and shifting the internal activation of the network
  - For INRs, author found that shifting modulation s is enough



$$\sin\left(\omega_0\left(\mathbf{W}^{(i)}\mathbf{x}+\mathbf{b}^{(i)}+\mathbf{s}^{(i)}\right)\right)$$

Globally shared parameters Shifting modulation

$$\mathbf{s}=\mathbf{W}'\phi+\mathbf{b}'$$

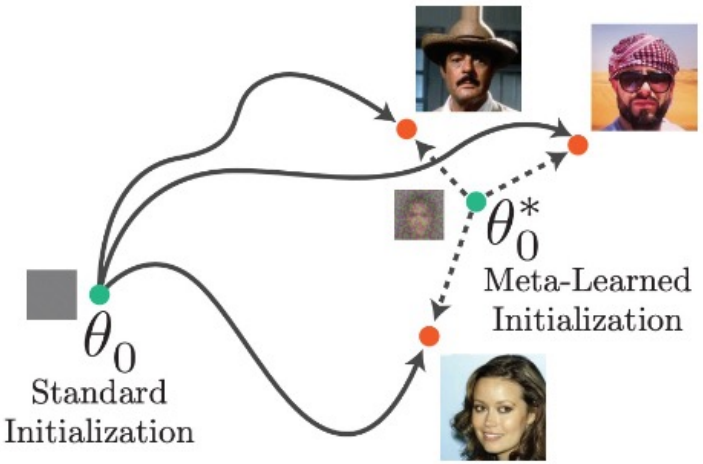
Reduce more dimension with shared linear layer

* source : Dupont et al., From data to functa: Your data point is a function and you should treat it like one, ICML 2022

- Meta-learning for INRs
  - Meta-learn a weight initialization
  - Author utilized MAML-like approach

Algorithm 1 Meta-learning functa

1: Randomly initialize shared base network θ
2: **while** not done **do**
3: Sample batch $\mathcal{B}$ of data $\{\{\mathbf{x}_i^{(j)}, \mathbf{f}_i^{(j)}\}_{i\in\mathcal{I}}\}_{j\in\mathcal{B}}$
4: Set batch modulations to zero $\phi_j \leftarrow 0\ \forall j \in \mathcal{B}$
5: **for all** step $\in \{1, \ldots, N_{inner}\}$ and $j \in \mathcal{B}$ **do**
6: $\phi_j \leftarrow \phi_j - \epsilon\nabla_\phi \mathcal{L}(f_{\theta,\phi}, \{\mathbf{x}_i^{(j)}, \mathbf{f}_i^{(j)}\}_{i\in\mathcal{I}})|_{\phi=\phi_j}$
7: **end for**
8: $\theta \leftarrow \theta - \epsilon' \frac{1}{|\mathcal{B}|}\sum_{j\in\mathcal{B}} \nabla_\theta \mathcal{L}(f_{\theta,\phi}, \{\mathbf{x}_i^{(j)}, \mathbf{f}_i^{(j)}\}_{i\in\mathcal{I}})|_{\phi=\phi_j}$
9: **end while**

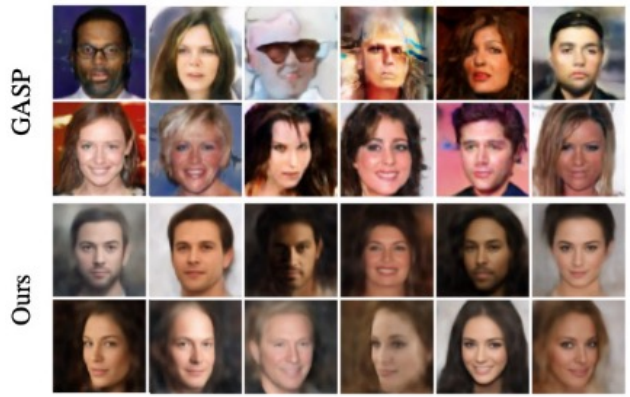


- Inner loop: Fit individual signal by optimizing modulation ϕ_j
- Outer loop: Fit overall signals by optimizing shared parameter θ

* source : Dupont et al., From data to functa: Your data point is a function and you should treat it like one, ICML 2022

Functa: treating INRs as data [Dupont et al., 2022]

- Experiments on generation



- Experiments on data imputation

- Functa achieves high accuracy on classification task of ShapeNet dataset

CLASSIFIER	TEST ACCURACY	n_{PARAMS}
MLP ON FUNCTA	$93.6 \pm 0.1\%$	83K
3D CNN	$93.3 \pm 0.3\%$	550K

* source : Dupont et al., From data to functa: Your data point is a function and you should treat it like one, ICML 2022

GRAF: Generative radiance fields [Schwarz et al., 2020]

- 2D GANs lack an understanding of the 3D and the image formation process

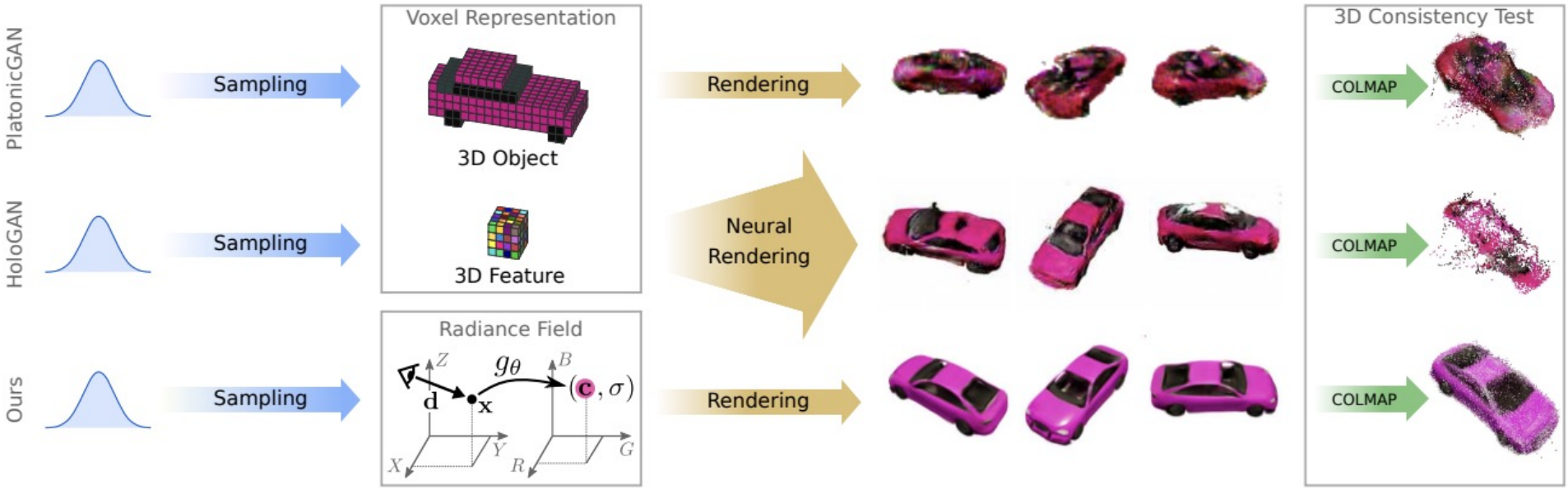
- Schwarz et al. (2020): **Generative model for radiance fields**
  - Synthesis of high-resolution images while training from unposed 2D images alone
  - Allows to modify shape and appearance of the generated objects

- **Key idea**: Patch-based discriminator
  - Samples the image at multiple scales

* source : Schwarz et al., GRAF: Generative Radiance Fields for 3D-Aware Image Synthesis, NuerIPS 2020

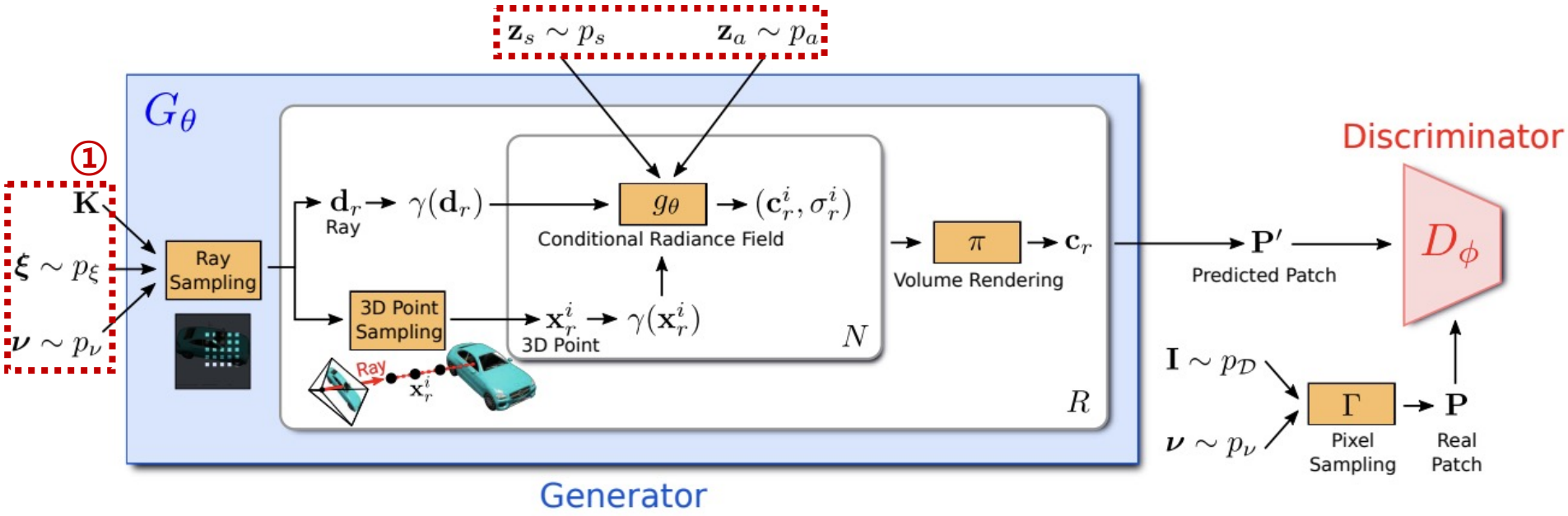
GRAF: Generative radiance fields [Schwarz et al., 2020]

- **Motivation**
  - Voxel-based approaches for 3D-aware image synthesis lead to discretization artifacts or degrades view-consistency.
  - Use neural radiance fields to represent scene as a continuous function.



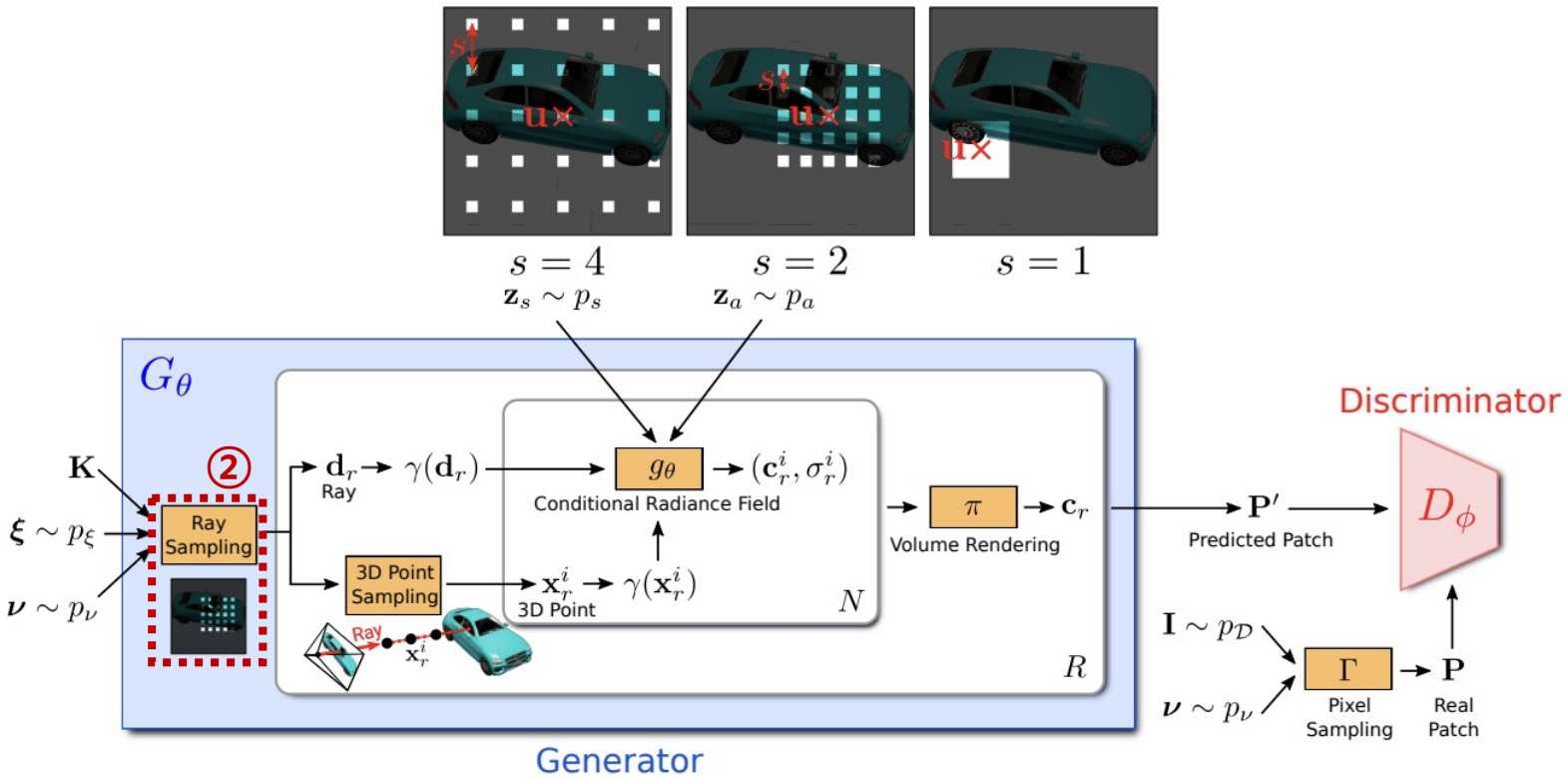
* source : Schwarz et al., GRAF: Generative Radiance Fields for 3D-Aware Image Synthesis, NeurIPS 2020

- **Method (① Input)**
  - Camera matrix (intrinsics) $\boldsymbol{K}$
  - Camera pose $\boldsymbol{\xi}$
  - 2D sampling pattern $\boldsymbol{\nu} = (\boldsymbol{u}, s)$
  - Shape & Appearance codes $\boldsymbol{z_s}, \boldsymbol{z_a}$



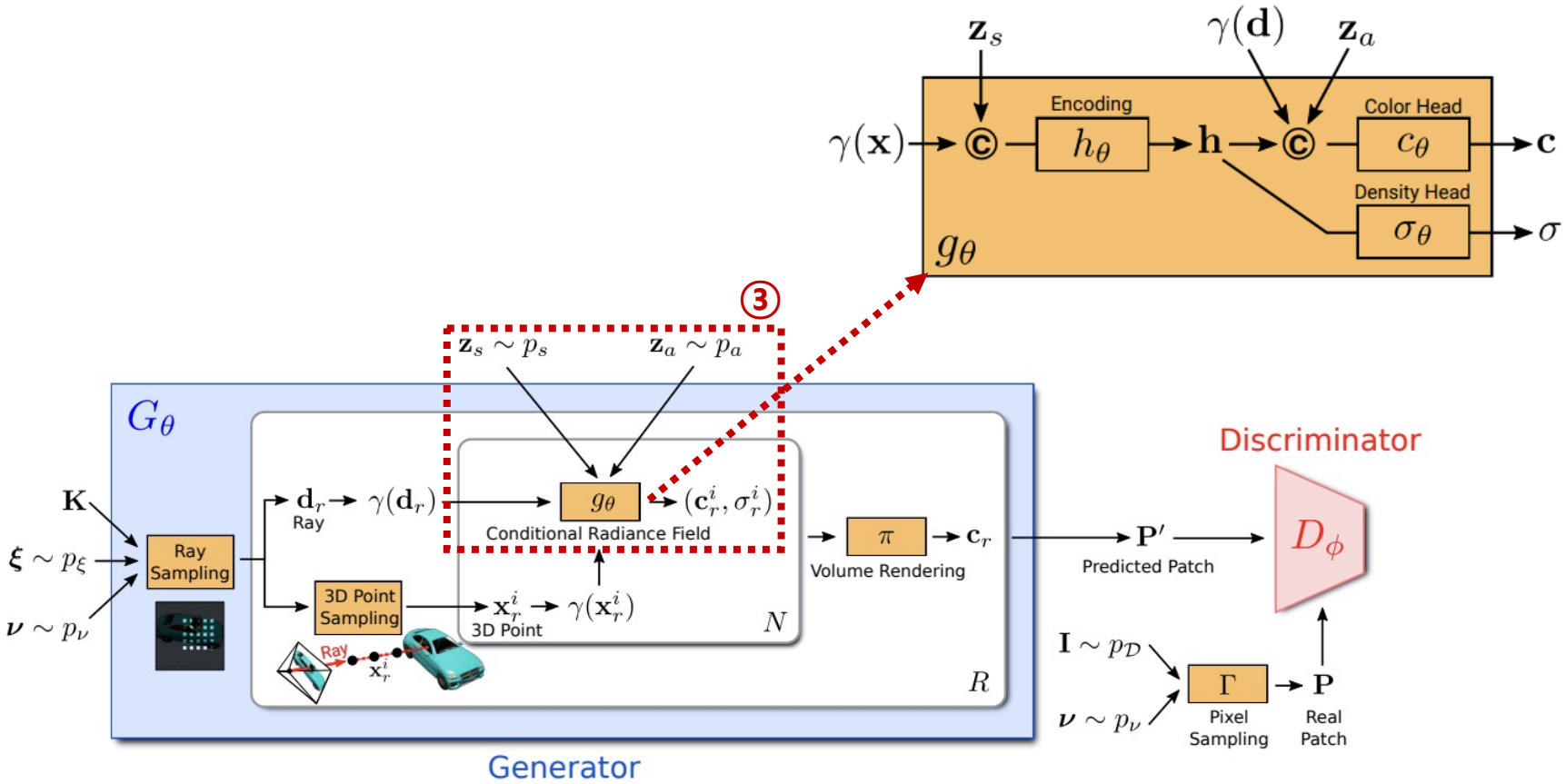
* source : Schwarz et al., GRAF: Generative Radiance Fields for 3D-Aware Image Synthesis, NeurIPS 2020

- **Method (② Ray Sampling)**
  - Given camera pose $\boldsymbol{\xi}$, sample rays according to $\boldsymbol{\nu} = (\boldsymbol{u}, s)$
  - At training time, use **2D sampling patterns of size** $K \times K$
    - Compared to original neural radiance fields, it is computation & memory efficient
    - Enable to use a convolutional discriminator independent to image resolution



* source : Schwarz et al., GRAF: Generative Radiance Fields for 3D-Aware Image Synthesis, NeurIPS 2020

- **Method (③ Conditional radiance field)**
  - $\boldsymbol{\sigma}$: Depends solely on the 3D point $\boldsymbol{x}$ and the shape code $\boldsymbol{z_s}$
  - $\boldsymbol{c}$: Additionally depends on the viewing direction $\boldsymbol{d}$ and the appearance code $\boldsymbol{z_a}$



* source : Schwarz et al., GRAF: Generative Radiance Fields for 3D-Aware Image Synthesis, NeurIPS 2020

GRAF: Generative radiance fields [Schwarz et al., 2020]

- FID at image resolution 64x64 pixels

	Chairs	Birds	Cars	Cats	Faces
2D GAN [34]	59	24	66	18	15
PLATONICGAN [19]	199	179	169	318	321
HoloGAN [39]	59	78	134	27	25
Ours	34	47	30	26	25

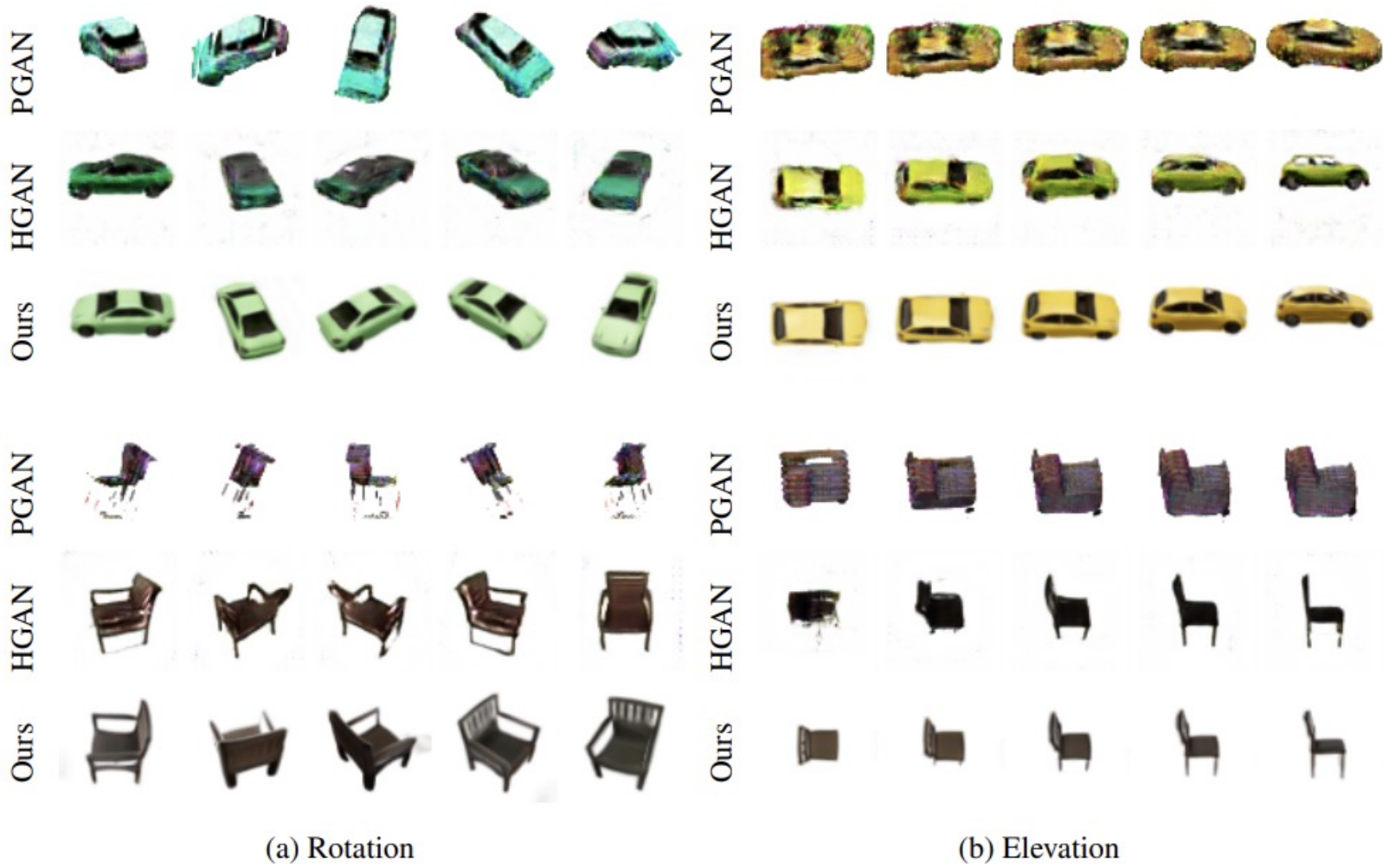
- FID at image resolution 128x128 – 512x512

	Cars			Faces		
	128	256	512	128	256	512
HoloGAN [39]	211	230	–	39	61	–
w/o 3D Conv	180	189	251	31	33	51
Ours	41	71	84	35	49	49
upsampled	–	91	128	–	63	77
sampled	–	74	104	–	50	56

* source : Schwarz et al., GRAF: Generative Radiance Fields for 3D-Aware Image Synthesis, NeurIPS 2020

GRAF: Generative radiance fields [Schwarz et al., 2020]

- Camera Pose Interpolations

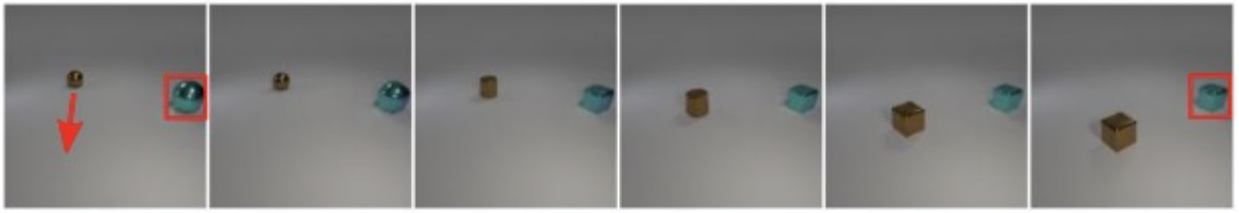


* source : Schwarz et al., GRAF: Generative Radiance Fields for 3D-Aware Image Synthesis, NeurIPS 2020

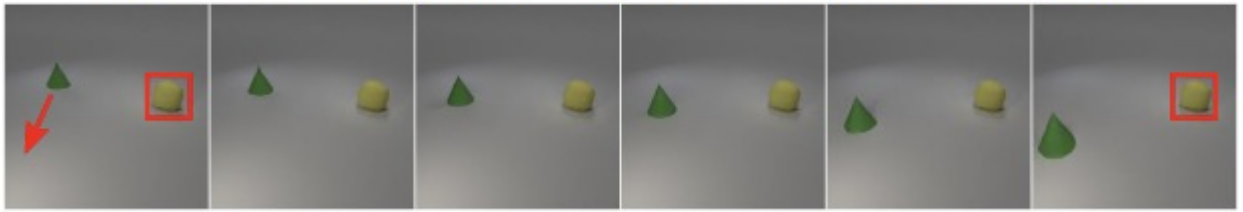
GIRAFFE: Compositional generative neural feature fields [Niemeyer et al., 2021]

- Disentanglement of underlying factors of variations in the data
  - Most of works operate in 2D

- Niemeyer et al. (2021): **Compositional generative neural feature fields**
  - Incorporating a compositional 3D scene representation into the generative model leads to more controllable image synthesis
  - Allows to disentangle multiple objects from the background



(a) Translation of Left Object (2D-based Method [71])

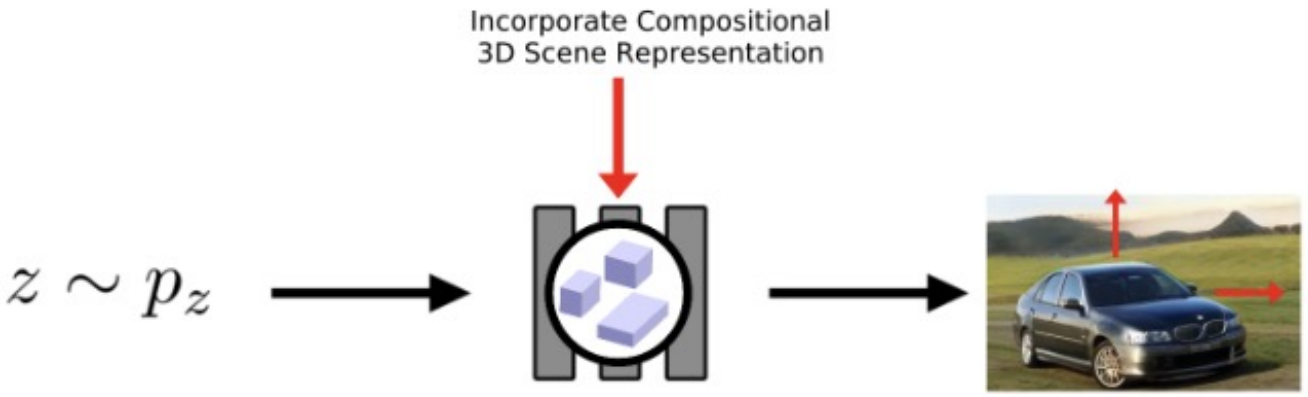


(b) Translation of Left Object (Ours)

- Translating one object might change the other when operating in 2D

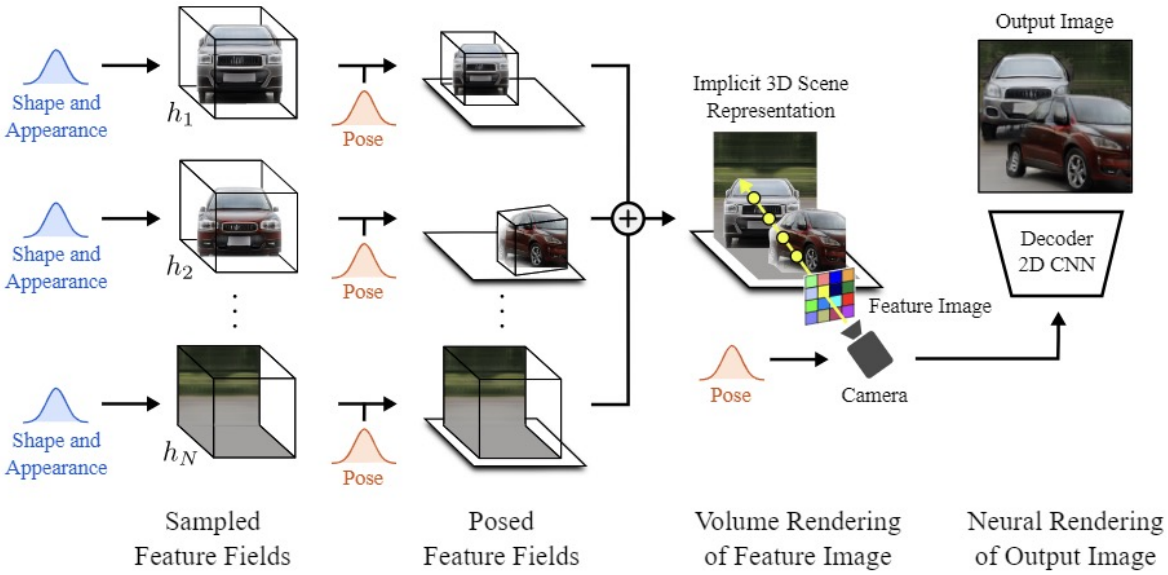
* source : Niemeyer et al., GIRAFFE: Representing Scenes as Compositional Generative Neural Feature Fields, CVPR 2021

- **Method (Main idea)**
  - Incorporate compositional 3D scene structure into the generative model
  - Disentangle individual objects from the background without explicit supervision.



* source : Niemeyer et al., GIRAFFE: Representing Scenes as Compositional Generative Neural Feature Fields, CVPR 2021

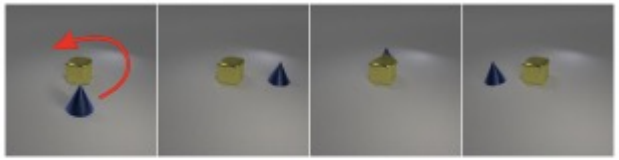
- **Method (Forward pass)**
  - Sample individual latent codes for the objects as well as the background.
    - This gives individual feature fields in a canonical space.
    - A feature field maps a point and viewing direction to a density and a feature vector.
  - Then sample a pose for each object.
    - Now can composite them into a single scene.
  - Sample a camera pose and volume render a feature image of the scene.
  - Neural renderer upsamples the feature image and outputs the final rendering.



* source : Niemeyer et al., GIRAFFE: Representing Scenes as Compositional Generative Neural Feature Fields, CVPR 2021

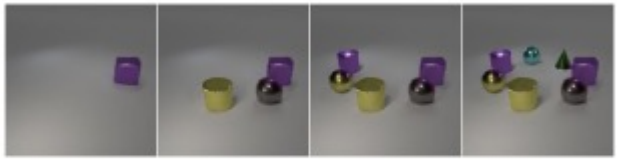
GIRAFFE: Compositional generative neural feature fields [Niemeyer et al., 2021]

- GIRAFFE can do complex operations
- Circular translation



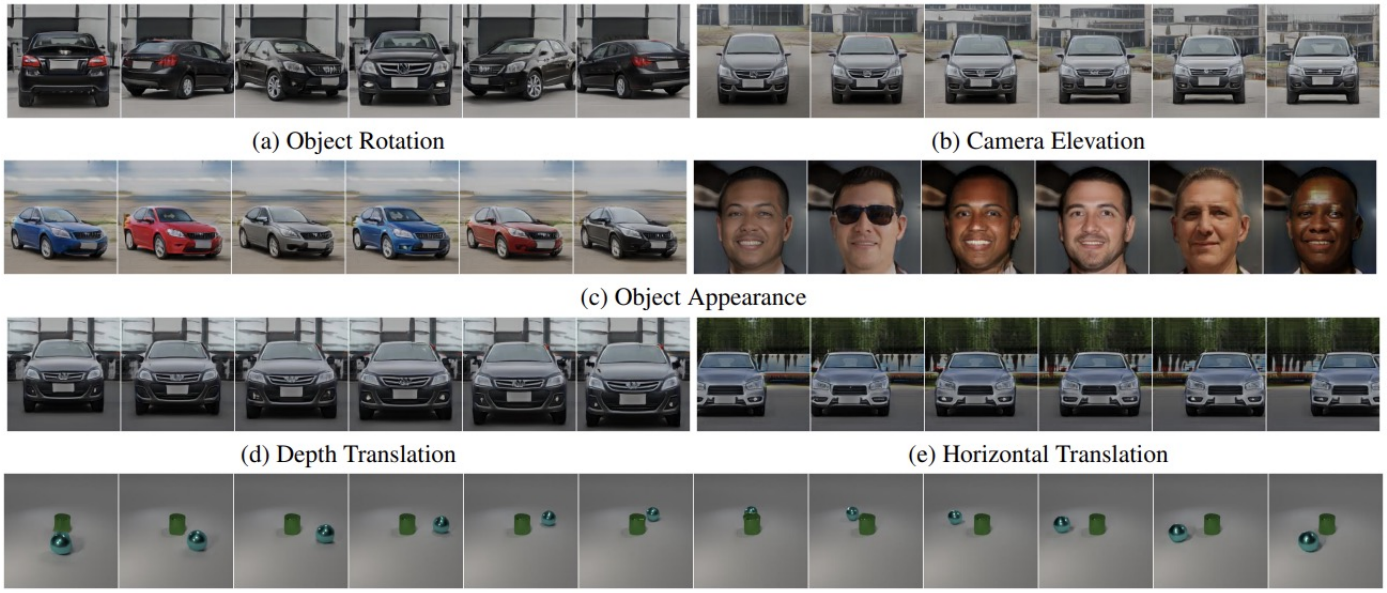
(c) Circular Translation (Ours)

- Add objects



(d) Add Objects (Ours)

- Controllable scene generation



(a) Object Rotation

(b) Camera Elevation

(c) Object Appearance

(d) Depth Translation

(e) Horizontal Translation

(f) Circular Translation of One Object Around Another Object

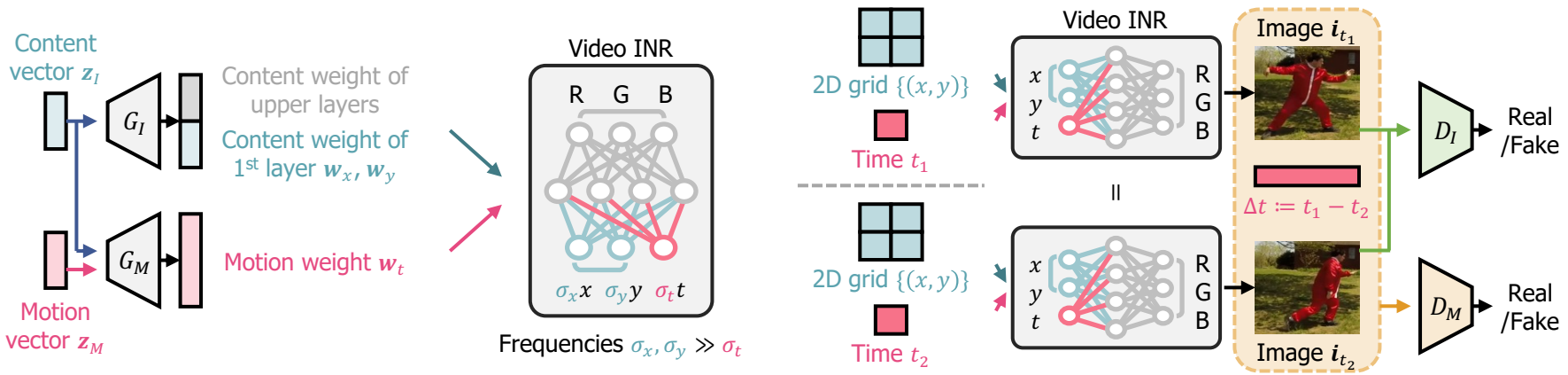
* source : Niemeyer et al., GIRAFFE: Representing Scenes as Compositional Generative Neural Feature Fields, CVPR 2021

DIGAN: Dynamics-aware implicit GANs [Yu et al., 2022]

- How can we interpret video signals for efficient, scalable video synthesis?

- Yu et al. (2022): **Leverage implicit neural representations**
  - Videos are continuous signals, where frames are correlated with temporal dynamics

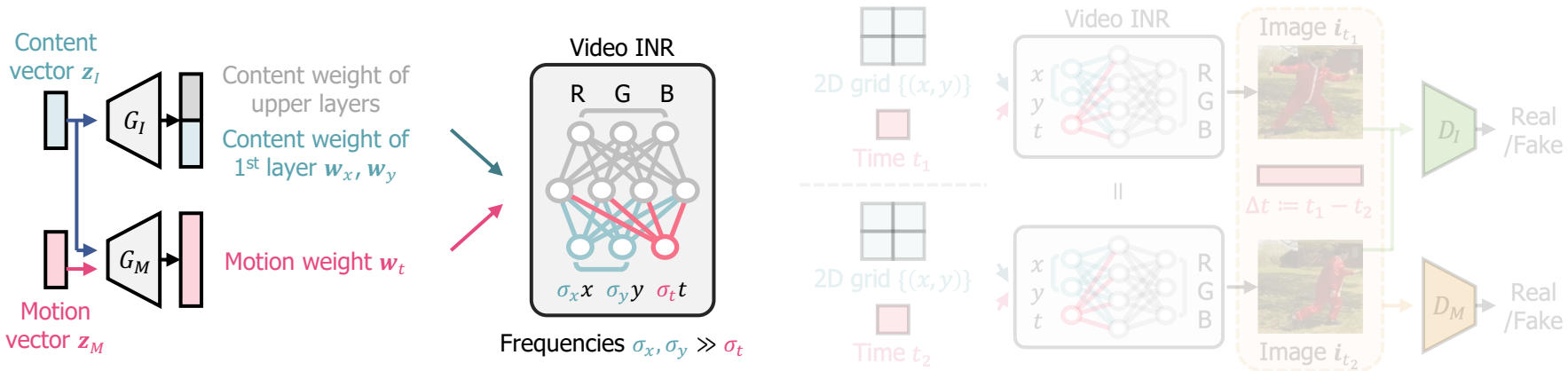
- **DIGAN**: Implicit GAN which generates weights of video INRS
  - Extends implicit GANs on image synthesis which generates weights of image INRs
  - Focus on temporal aspects of videos to design implicit GANs for video synthesis



* source : Yu et al., Generating Videos with Dynamics-aware Implicit Generative Adversarial Networks, ICLR 2022

DIGAN: Dynamics-aware implicit GANs [Yu et al., 2022]

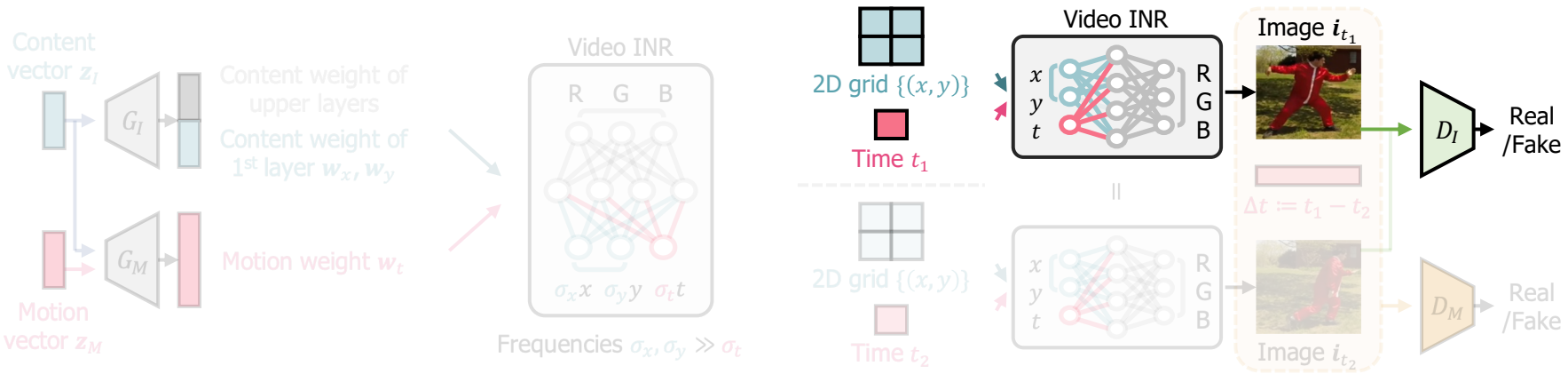
- Generators synthesize the weights of video INRs from a content vector and a motion vector
  - $\boldsymbol{z_I}$ determines overall spatial contents (or style) of generated videos
  - $\boldsymbol{z_M}$ determines the temporal motion of generated videos
  - Smaller time-frequency σ_t than space-frequencies σ_x, σ_y
  - Since frames change relatively slowly over time compared to spatial variations



* source : Yu et al., Generating Videos with Dynamics-aware Implicit Generative Adversarial Networks, ICLR 2022

DIGAN: Dynamics-aware implicit GANs [Yu et al., 2022]

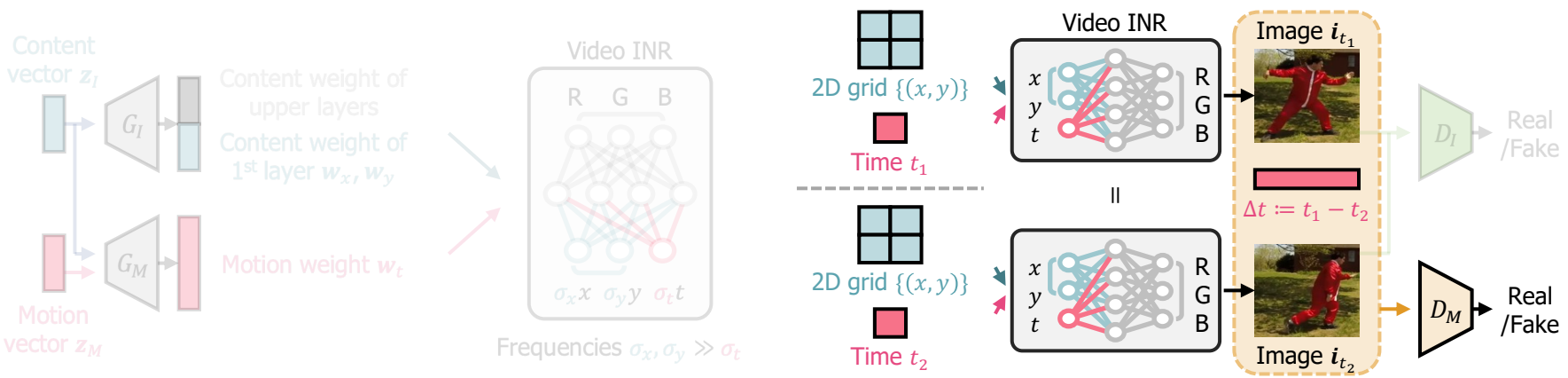
- DIGAN utilizes two discriminators:
- An image discriminator D_I and a motion discriminator D_M
  - D_I determines real/fake of the generated frame at arbitrary time $t \in [0,1]$



* source : Yu et al., Generating Videos with Dynamics-aware Implicit Generative Adversarial Networks, ICLR 2022

DIGAN: Dynamics-aware implicit GANs [Yu et al., 2022]

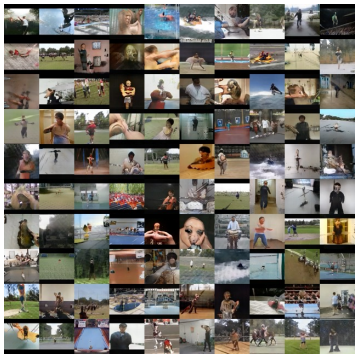
- DIGAN utilizes two discriminators:
- An image discriminator D_I and a motion discriminator D_M
  - $\boldsymbol{D_M}$ determines real/fake of the triplet $(\boldsymbol{i}_{t_1}, \boldsymbol{i}_{t_2}, \Delta t)$
    - Consists of a pair of images and their time difference
  - $\boldsymbol{D_M}$ is not a 3D convolutional discriminator → same architecture as $D_{/i}$
- Generate video INR → efficiently produce arbitrary time frames



* source : Yu et al., Generating Videos with Dynamics-aware Implicit Generative Adversarial Networks, ICLR 2022

DIGAN: Dynamics-aware implicit GANs [Yu et al., 2022]

- DIGAN significantly outperforms prior video synthesis methods
  - e.g.) UCF-101: Shows 30.7% improvement on UCF-101 (measured with FVD)



TaiChi

Kinetics-foo

Table 1: IS, FVD, and KVD values of video generation models on (a) UCF-101, (b) Sky, (c) TaiChi, and (d) Kinetics-food datasets. ↑ and ↓ imply higher and lower values are better, respectively. Subscripts denote standard deviations, and bolds indicate the best results. "Train split" and "Train+test split" denote whether the model is trained with the train split (following the setup in Saito et al. (2020)) or with the full dataset (following the setup in Tian et al. (2021)), respectively.

(a) UCF-101

Method	IS (↑)	FVD (↓)
Train split		
VGAN	8.31±.09	-
TGAN	11.85±.07	-
MoCoGAN	12.42±.07	-
ProgressiveVGAN	14.56±.05	-
LDVD-GAN	22.91±.19	-
VideoGPT	24.69±.30	-
TGANv2	28.87±.67	1209±28
DIGAN (ours)	**29.71**±.53	**655**±22
Train+test split		
DVD-GAN	27.38±.53	-
MoCoGAN-HD	32.36	838
DIGAN (ours)	**32.70**±.35	**577**±21

(b) Sky

Method	FVD (↓)	KVD (↓)
MoCoGAN-HD	183.6±5.2	13.9±0.7
DIGAN (ours)	**114.6**±4.3	**6.8**±0.5

(c) TaiChi

Method	FVD (↓)	KVD (↓)
MoCoGAN-HD	144.7±6.0	25.4±1.9
DIGAN (ours)	**128.1**±4.9	**20.6**±1.1

(d) Kinetics-food

Method	FVD (↓)	KVD (↓)
MoCoGAN-HD	430.4±29.9	276.0±50.7
DIGAN (ours)	**313.3**±36.9	**183.0**±40.3

* source : Yu et al., Generating Videos with Dynamics-aware Implicit Generative Adversarial Networks, ICLR 2022

105

DIGAN: Dynamics-aware implicit GANs [Yu et al., 2022]

- DIGAN can generate long videos (up to 256 frames)
  - It is 5.3x longer than prior state-of-the-art method [Clark et al., 2019]
  - Training time and resources are not demanding (4.4 days with 4 NVIDIA V100 GPUs)

→ Follow the arrow direction, and move to the next line at the end

* source : Yu et al., Generating Videos with Dynamics-aware Implicit Generative Adversarial Networks, ICLR 2022

DIGAN: Dynamics-aware implicit GANs [Yu et al., 2022]

- DIGAN achieves successful "spatiotemporal" inter-/extra-polation
  - **Recall:** We interpret videos as spatiotemporally continuous signals

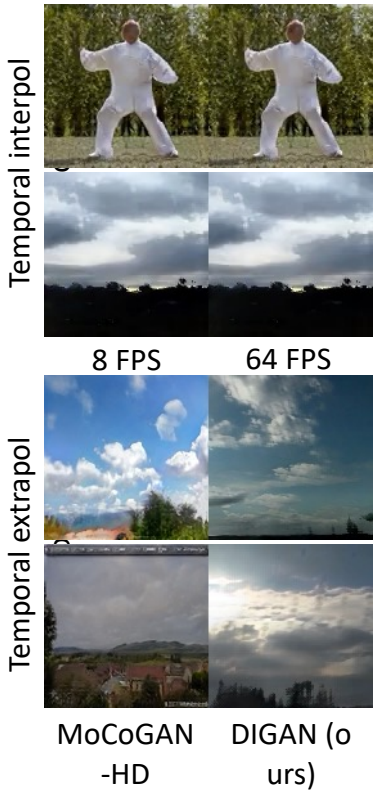
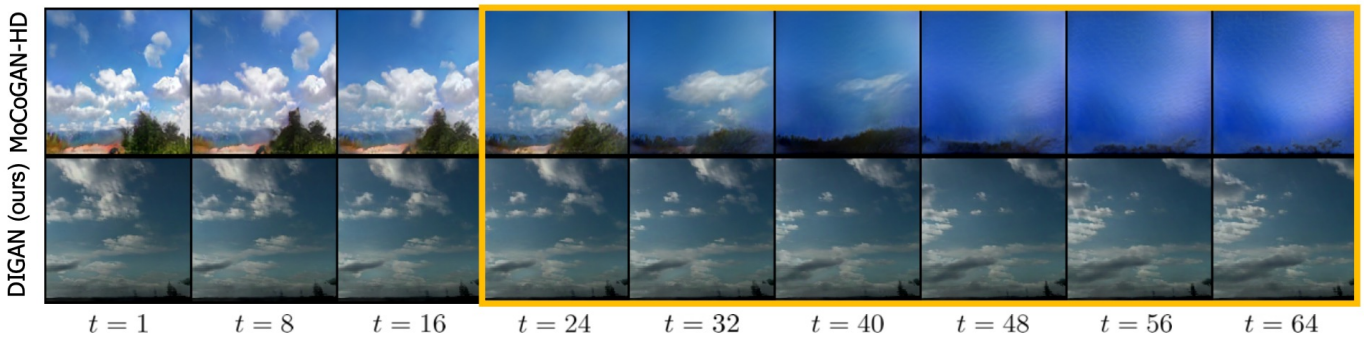


Table 2: FVD values of generated videos inter- and extra-polated over time. All models are trained on 16 frame videos of 128×128 resolution. The videos are interpolated to 64 frames (*i.e.*, 4× finer) and extrapolated 16 more frames. We measure FVD with 512 samples for Sky, since the test data size becomes less than 2,048.

	Interpolation			Extrapolation		
Method	Sky	TaiChi	Kinetics-food	Sky	TaiChi	Kinetics-food
MoCoGAN-HD	402.2±18.9	249.0±12.7	1029.8±28.4	303.2±4.3	337.8±3.7	877.8±22.6
DIGAN (ours)	**324.2**±20.5	**241.6**±7.5	**722.2**±20.1	**224.3**±6.2	**289.3**±15.6	**693.7**±14.1



* source : Yu et al., Generating Videos with Dynamics-aware Implicit Generative Adversarial Networks, ICLR 2022

StyleNeRF: Style-based 3D-aware generator [Gu et al., 2021]

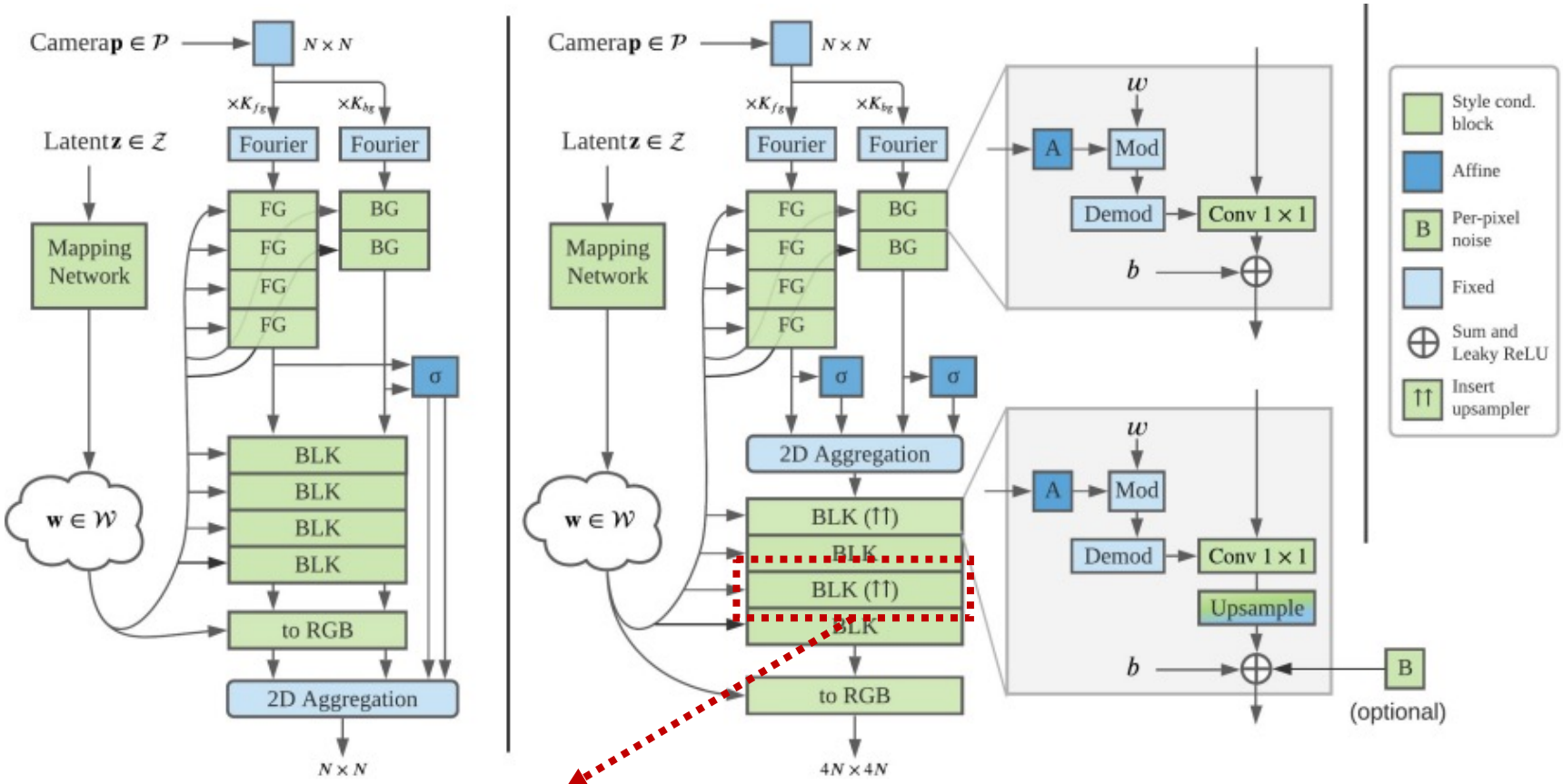
- How can we synthesize high-resolution images with fine details?
- How can we yield noticeable 3D-consistent artifacts?
- How can we control over style attributes and explicit camera poses?

- Gu et al. (2021): **3D-aware generative model**
  - Can synthesize high-resolution images with high multi-view consistency

- **StyleNeRF**: Integrates the NeRF into a style-based generator
  - Volume render only to produce a low-resolution feature map
  - Progressively apply upsampling in 2D
  - **Better upsampler & new regularization loss**: mitigates the inconsistencies caused by 2D upsampling

* source : Gu et al., StyleNeRF: A Style-based 3D-Aware Generator for High-resolution Image synthesis, ICLR 2021

- Architecture
  - **Mapping Network**: Follows StyleGAN2
  - **Synthesis Network**: Uses NeRF++ as backbone



key component: upsampler

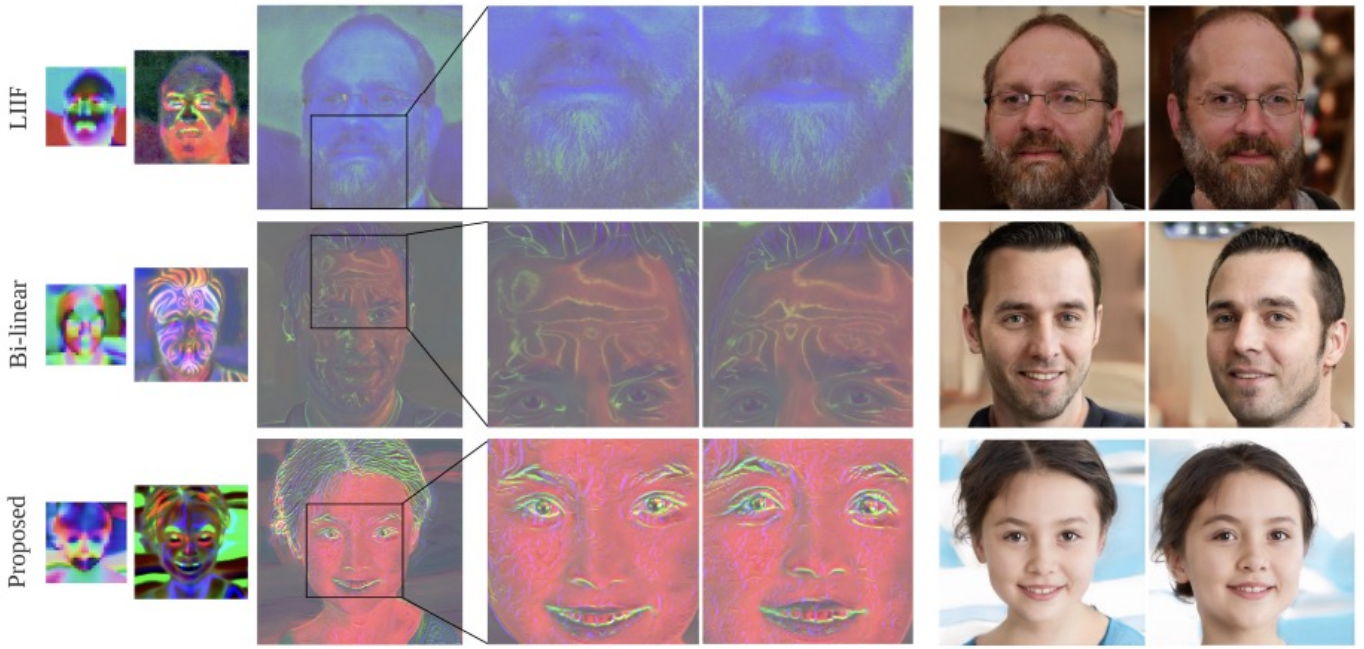
Figure 3: Example architecture. (Left) Original NeRF path; (right) Main path of StyleNeRF.

* source : Gu et al., StyleNeRF: A Style-based 3D-Aware Generator for High-resolution Image synthesis, ICLR 2021

- Upsampler
  - **Pixel-wise learnable upsamplers**: chessboard or texture sticking artifacts
  - **Non-learnable upsamplers**: non-removable bubble artifacts
  - **Combine two approaches**:
    - achieve the balance between consistency and image quality

$$\texttt{Upsample}(X) = \texttt{Conv2d}(\texttt{Pixelshuffle}(\texttt{Repeat}(X, 4) + \psi_\theta(X), 2), K)$$

learnable 2-layer MLP fixed blur kernel



* source : Gu et al., StyleNeRF: A Style-based 3D-Aware Generator for High-resolution Image synthesis, ICLR 2021

- **NeRF path regularization**
  - Enforces 3D consistency: regularizes the model output to match the original path
  - Final outputs be closer to the NeRF results, which have multi-view consistency
  - Implemented by sub-subsampling pixels on the output and comparing them against those generated by NeRF

$$\mathcal{L}_{\text{NeRF-path}} = \frac{1}{|S|} \sum_{(i,j)\in S} \left(I_{\boldsymbol{w}}^{\text{Approx}}(R_{\text{in}})[i,j] - I_{\boldsymbol{w}}^{\text{NeRF}}(R_{\text{out}}[i,j]) \right)^2$$

* source : Gu et al., StyleNeRF: A Style-based 3D-Aware Generator for High-resolution Image synthesis, ICLR 2021

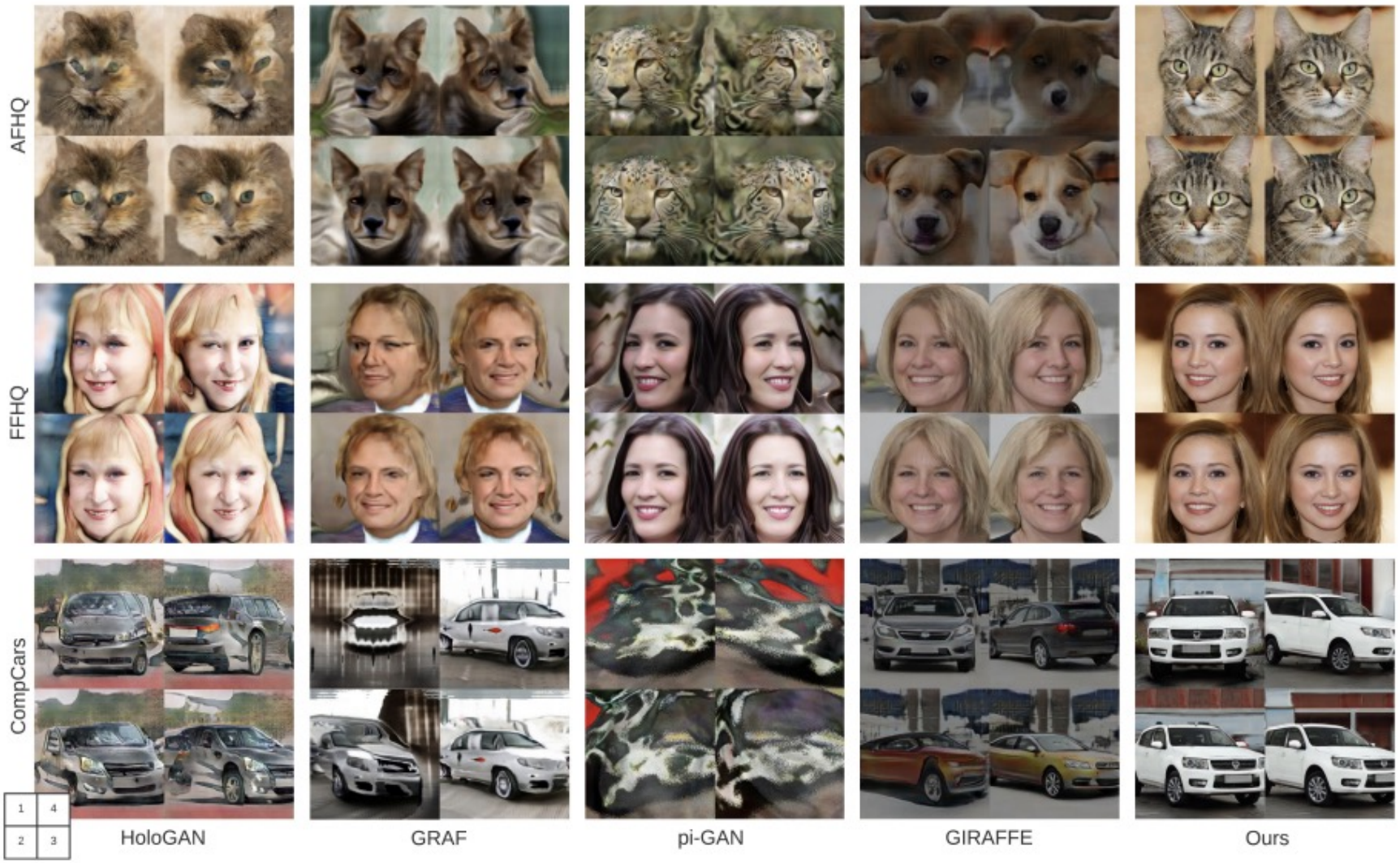
StyleNeRF: Style-based 3D-aware generator [Gu et al., 2021]

- StyleNeRF can generate photo-realistic high-resolution images from novel views

* source : Gu et al., StyleNeRF: A Style-based 3D-Aware Generator for High-resolution Image synthesis, ICLR 2021

StyleNeRF: Style-based 3D-aware generator [Gu et al., 2021]

- StyleNeRF achieves the best quality and 3D consistency



* source : Gu et al., StyleNeRF: A Style-based 3D-Aware Generator for High-resolution Image synthesis, ICLR 2021

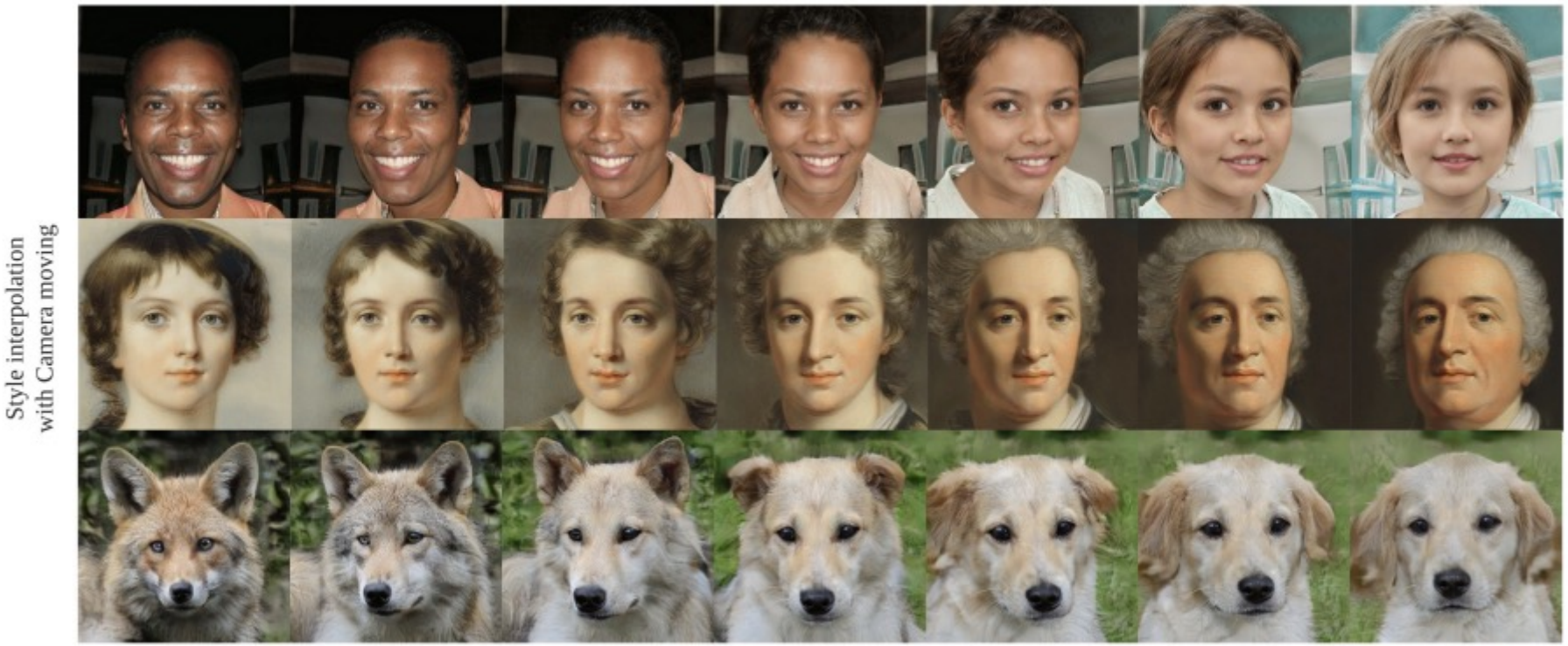
StyleNeRF: Style-based 3D-aware generator [Gu et al., 2021]

- Experiments on style mixing
  - Images were generated by copying the specified styles from source B to source A
  - All images are rendered from the same camera pose

* source : Gu et al., StyleNeRF: A Style-based 3D-Aware Generator for High-resolution Image synthesis, ICLR 2021

StyleNeRF: Style-based 3D-aware generator [Gu et al., 2021]

- Experiments on style interpolation
  - Linearly interpolated two sets of style vectors while rotating the camera



* source : Gu et al., StyleNeRF: A Style-based 3D-Aware Generator for High-resolution Image synthesis, ICLR 2021

CIPS-3D: Generator based on conditionally-independent pixel synthesis [Zhou et al., 2021]

- Zhou et al. (2021): **A style-based 3D-aware generative model**
  - Composed of a shallow NeRF network & a deep INR network
  - Generator synthesizes each pixel value independently without spatial convolution or upsampling operation
  - Diagnose the problem of mirror symmetry

 * source : Zhou et al., CIPS-3D: A 3D-Aware Generator of GANs Based on Conditionally-Independent Pixel Synthesis, arXiv 2021

CIPS-3D: Generator based on conditionally-independent pixel synthesis [Zhou et al., 2021]

- **Previous**: Using a pure NeRF network as the generator
  - Suffers from a low capacity of the generator
  - Synthesized images are blurred and lack sharp details

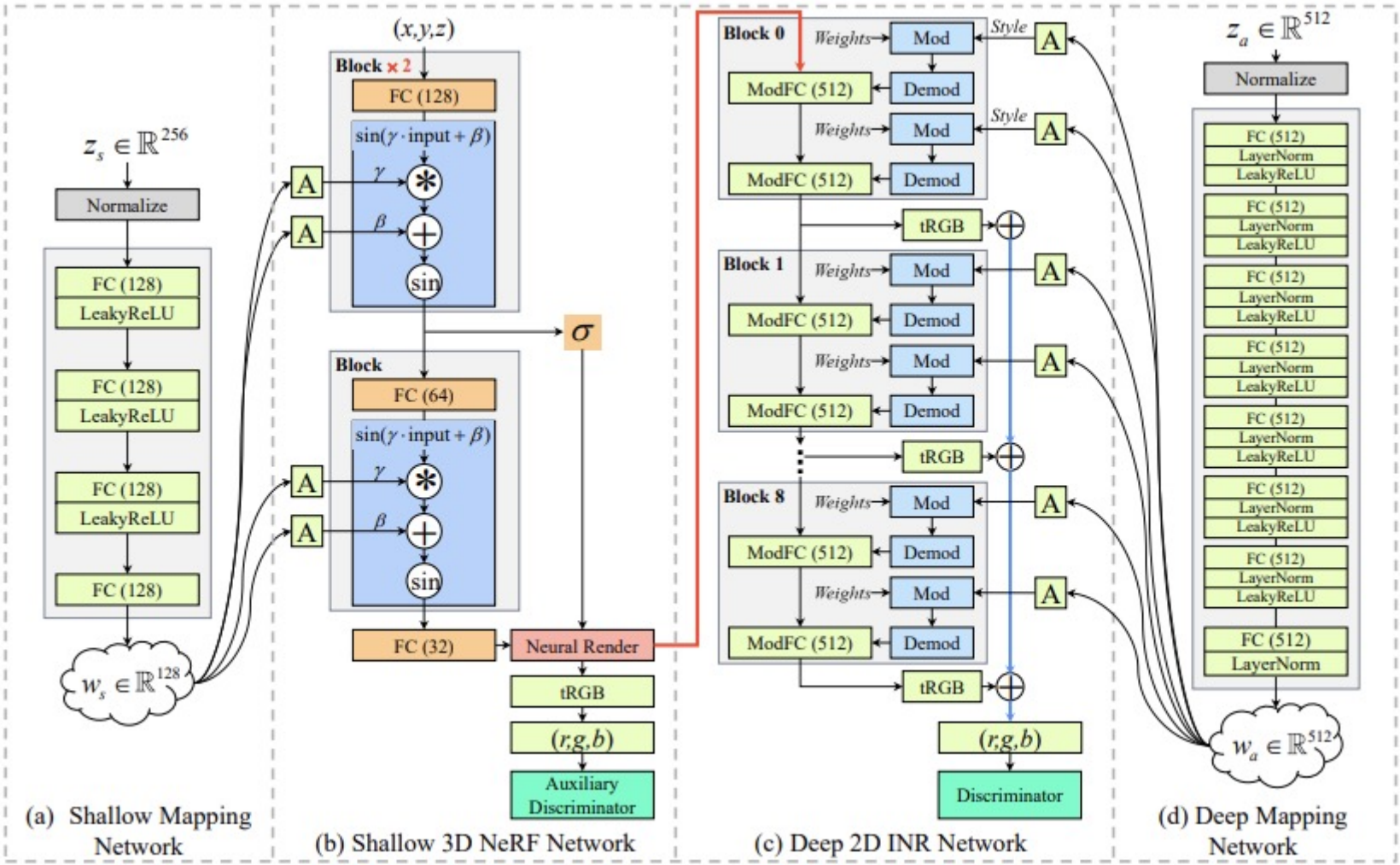
- **Previous**: Generating low-resolution feature maps with a NeRF network and then upsampling with a 2D CNN decoder
  - Susceptible to aliasing due to non-ideal upsampling filter
  - Generator synthesizes each pixel value independently without spatial convolution or upsampling operation

- **CIPS-3D**: An approach to synthesize each pixel value independently
  - **Shallow 3D NeRF**: Alleviate memory complexity
  - **Deep 2D INR network**: Enlarge the capacity of the generator
  - **Auxiliary discriminator**: Solve the mirror symmetry problem

 * source : Zhou et al., CIPS-3D: A 3D-Aware Generator of GANs Based on Conditionally-Independent Pixel Synthesis, arXiv 2021

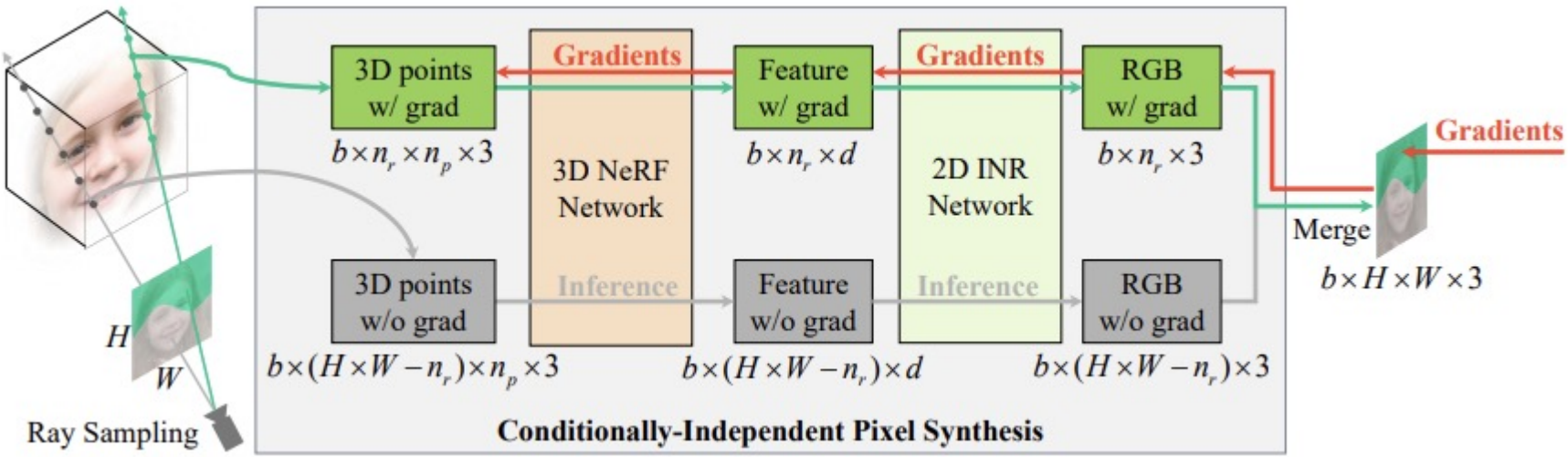
CIPS-3D: Generator based on conditionally-independent pixel synthesis [Zhou et al., 2021]

- Architecture



 * source : Zhou et al., CIPS-3D: A 3D-Aware Generator of GANs Based on Conditionally-Independent Pixel Synthesis, arXiv 2021

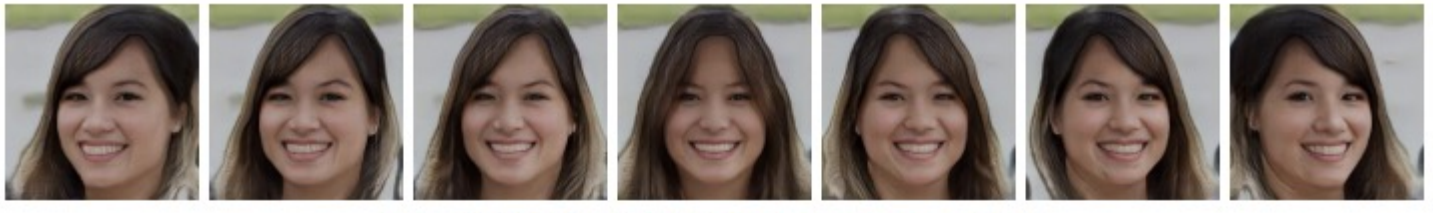
- **Partial Gradient Backpropagation**
  - Directly training generator is challenging due to limited GPU memory
  - Gradient calculation is turned on only for the green rays sampled randomly



 * source : Zhou et al., CIPS-3D: A 3D-Aware Generator of GANs Based on Conditionally-Independent Pixel Synthesis, arXiv 2021

CIPS-3D: Generator based on conditionally-independent pixel synthesis [Zhou et al., 2021]

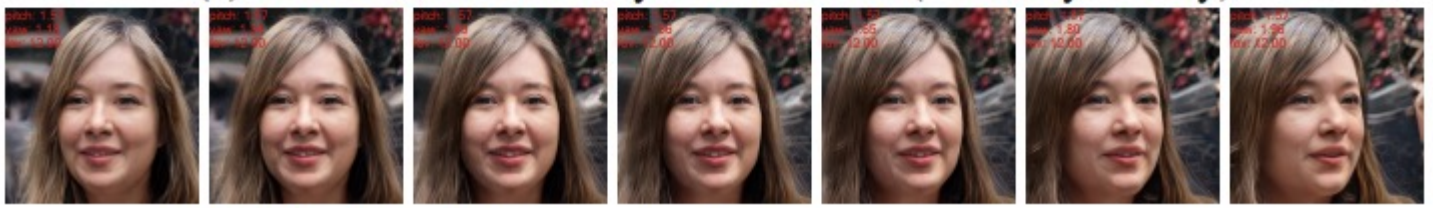
- The auxiliary discriminator helps to overcome the mirror symmetry



(a) GIRAFFE [42] (mirror symmetry)

(b) pi-GAN [11] (no mirror symmetry)

(c) CIPS-3D w/o auxiliary discriminator (mirror symmetry)



(d) CIPS-3D w/ auxiliary discriminator (no mirror symmetry)

 * source : Zhou et al., CIPS-3D: A 3D-Aware Generator of GANs Based on Conditionally-Independent Pixel Synthesis, arXiv 2021

CIPS-3D: Generator based on conditionally-independent pixel synthesis [Zhou et al., 2021]

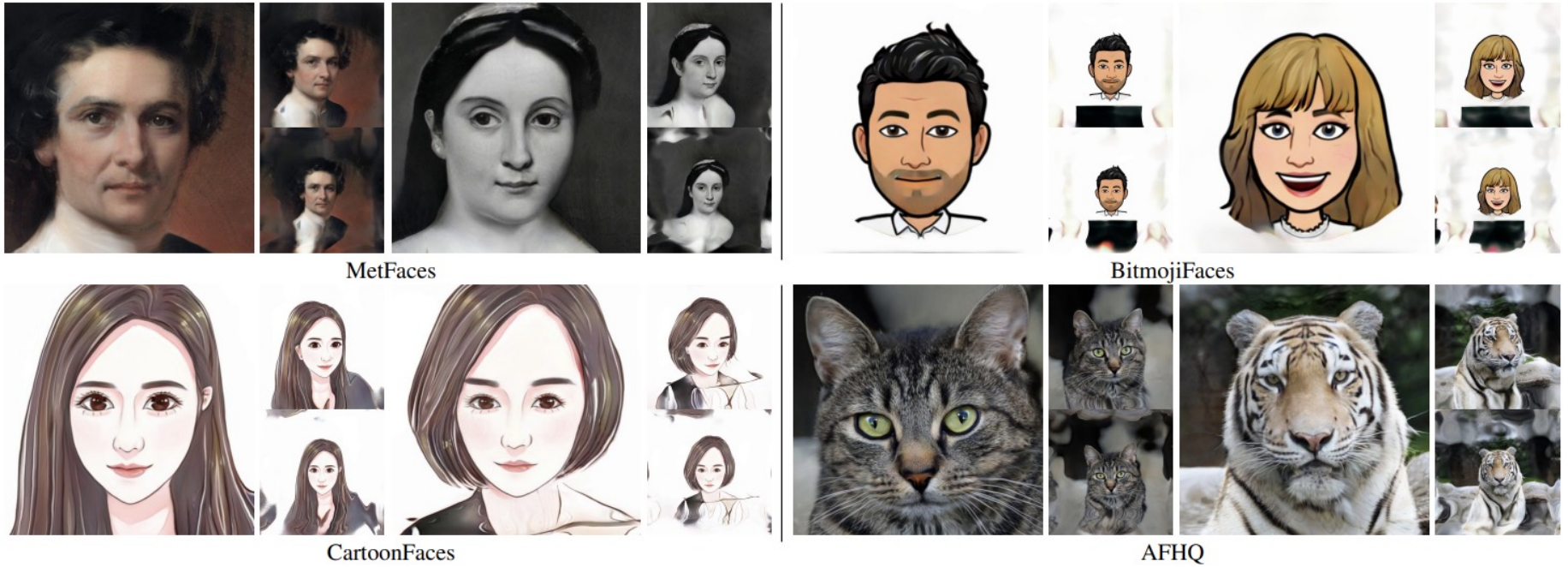
- Comparison on FFHQ

	Methods	256 × 256		1024 × 1024	
		FID ↓	KID ↓	FID ↓	KID ↓
2D	StyleGAN2 [33]	**4.30**	**1.07**	**2.86**	**0.53**
	CIPS [4]	23.06	23.04	10.03	4.79
3D	GIRAFFE [42]	63.33	50.94	-	-
	pi-GAN [11]	34.56	26.58	35.97	28.09
	StyleNeRF [5]	8.00†	3.70†	-	-
	CIPS-3D (ours)	**6.97**	**2.87**	**12.26**	**7.74**

 * source : Zhou et al., CIPS-3D: A 3D-Aware Generator of GANs Based on Conditionally-Independent Pixel Synthesis, ArXiv 2021

CIPS-3D: Generator based on conditionally-independent pixel synthesis [Zhou et al., 2021]

- Images synthesized by transferred model
  - CIPS-3D enables us to manipulate the pose of the generated faces explicitly



 * source : Zhou et al., CIPS-3D: A 3D-Aware Generator of GANs Based on Conditionally-Independent Pixel Synthesis, ArXiv 2021

References

[Rahaman, et. al., 2019] On the Spectral Bias of Neural Networks, ICML 2019
link: https://arxiv.org/pdf/1806.08734.pdf

[Sitzmann, et. al.,2020] Implicit Neural Representations with Periodic Activation Functions, NeurIPS 2020
link: https://arxiv.org/pdf/2006.09661.pdf

[Tancik, et. al.,2020] Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains
link: https://arxiv.org/pdf/2006.10739.pdf

[Mindenhall, et. al., 2020] NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis
link: https://arxiv.org/pdf/2006.10739.pdf

[Sun, et. al., 2022] Direct voxel grid optimization: super-fast convergence for radiance fields reconstruction, CVPR 2022
link: https://arxiv.org/pdf/2111.11215.pdf

[Mùller et. al., 2022] Instant neural graphics primitives with a multiresolution hash encoding, SIGGRAPH 2022
link: https://arxiv.org/pdf/2201.05989.pdf

[Liu et. al., 2021] Neural sparse voxel fields, NeurIPS 2021
link: https://arxiv.org/pdf/2007.11571.pdf

[Reiser et. al., 2021] KiloNeRF: speeding up neural radiance fields with thousands of tiny MLPs, ICCV 2021
link: https://arxiv.org/pdf/2103.13744.pdf

[Henman et. al., 2021] Baking neural radiance fields for real-time view synthesis, ICCV 2021
link: https://arxiv.org/pdf/2103.14645.pdf

[Yu et. al., 2021] PlenOctrees for real-time rendering of neural radiance fields, ICCV 2021
link: https://arxiv.org/pdf/2103.14024.pdf

References

[Deng, et. al., 2022] Depth-supervised NeRF: fewer views and faster training for free, CVPR 2022
link: https://arxiv.org/pdf/2107.02791.pdf

[Jain et al., 2021] Putting NeRF on a Diet: Semantically Consistent Few-Shot View synthesis, ICCV 2021
link: https://arxiv.org/pdf/2104.00677.pdf

[Martin et al., 2021] NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections, CVPR 2021
link: https://arxiv.org/pdf/2008.02268.pdf

[Park et al., 2021] Nerfies: Deformable Neural Radiance Fields, ICCV 2021
link: https://arxiv.org/pdf/2011.12948.pdf

[Schwarz et al., 2020] GRAF: Generative Radiance Fields for 3D-Aware Image Synthesis, NeurIPS 2020
link: https://arxiv.org/pdf/2007.02442.pdf

[Niemeyer & Geiger, 2021] GIRAFFE: Representing Scenes as Compositional Generative Neural Feature Fields, CVPR 20 21
link: https://arxiv.org/pdf/2011.12100.pdf

[Yu et al., 2022] Generating Videos with Dynamics-aware Implicit Generative Adversarial Networks, ICLR 2022
link: https://arxiv.org/pdf/2202.10571.pdf

[Gu et al., 2022] StyleNeRF: A Style-based 3D-Aware Generator for High-resolution Image Synthesis, ICLR 2022
link: https://arxiv.org/pdf/2110.08985.pdf

[Zhou et al., 2021] CIPS-3D: A 3D-Aware Generator of GANs Based on Conditionally-Independent Pixel Synthesis, arXiv 2021
link: https://arxiv.org/pdf/2110.09788.pdf

References

[Du et al., 2021] Learning Signal-Agnostic Manifolds of Neural Fields, NeurIPS 2021
link: https://arxiv.org/pdf/2111.06387.pdf

[Dupont et al., 2022] From data to functa: Your data point is a function and you can treat it like one, ICML 2022
link: https://arxiv.org/pdf/2201.12204.pdf

[Dupont et al., 2021] COIN: COmpression with Implicit Neural representations, ICLRW 2021
link: https://arxiv.org/pdf/2103.03123.pdf

[Tancik et al., 2021] Learned Initializations for Optimizing Coordinate-Based Neural Representations, CVPR 2021
link: https://arxiv.org/pdf/2012.02189.pdf