Big Data Visual Analytics (CS 661)

Instructor: Soumya Dutta

Department of Computer Science and Engineering

Indian Institute of Technology Kanpur (IITK)

email: soumyad@cse.iitk.ac.in

Acknowledgements

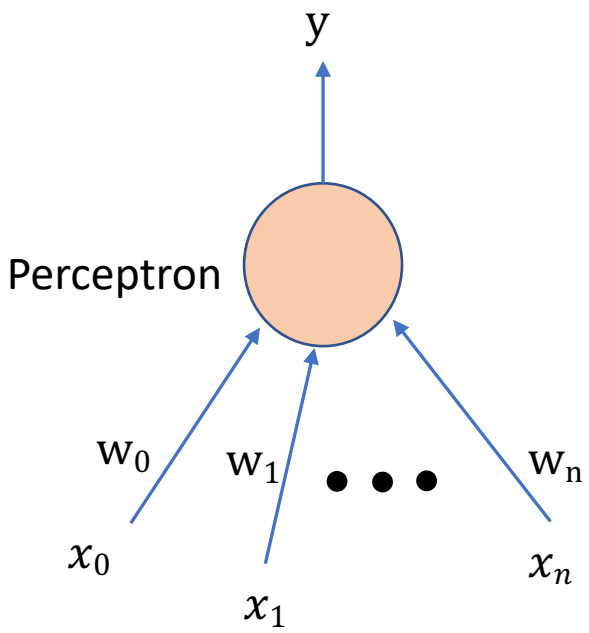
- Prof. Mitesh Khapra (IIT Madras)

Study Materials for Lecture 22

- Dive into Deep Learning (https://d2l.ai/index.html)
- Deep Learning by Prof. Mitesh Khapra, IITM, NPTEL Course
  - https://www.youtube.com/playlist?list=PLyqSpQzTE6M9gCgajvQbc68Hk_JKGBAYT
- NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis
- On the Spectral Bias of Neural Networks, Rahaman et al.
- Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains, Tancik et al.

Basics of Feed Forward Neural Network

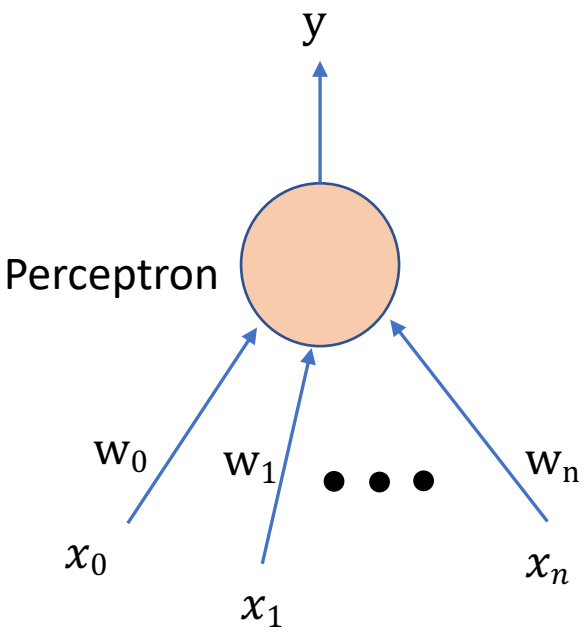
Perceptron and Sigmoid Neuron



$$y = 1 \; if \sum_{i=0}^{n} w_i * xi \geq 0$$

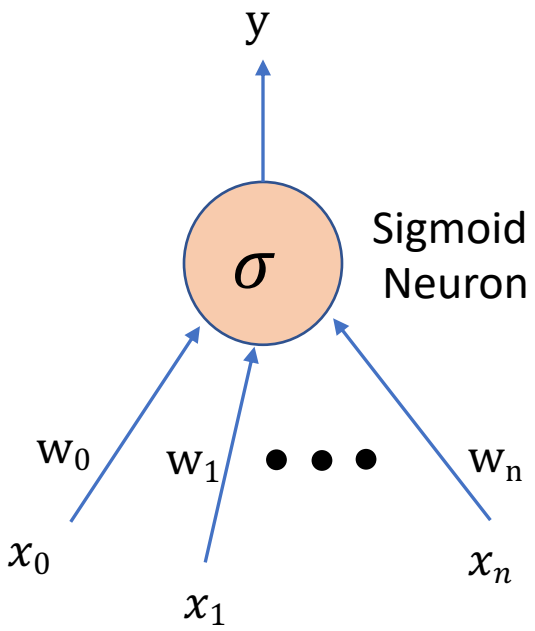
$$= 0 \; if \sum_{i=0}^{n} w_i * xi < 0$$

Perceptron and Sigmoid (Logistic) Neuron



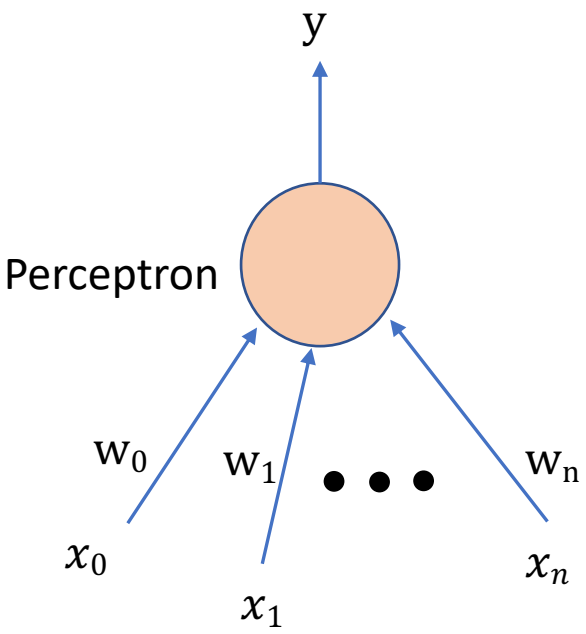
$$y = 1 \; if \sum_{i=0}^{n} w_i * x_i \geq 0$$

$$= 0 \; if \sum_{i=0}^{n} w_i * x_i < 0$$



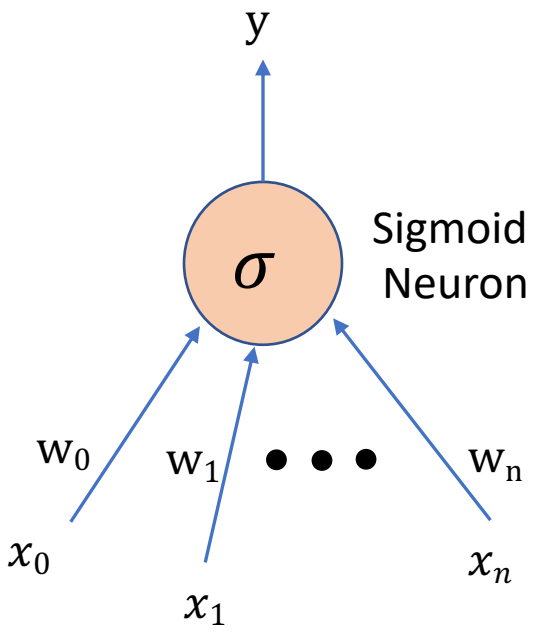
$$y = \frac{1}{1 + e^{-(\sum_{i=0}^{n} w_i * x_i)}}$$

Perceptron and Sigmoid (Logistic) Neuron

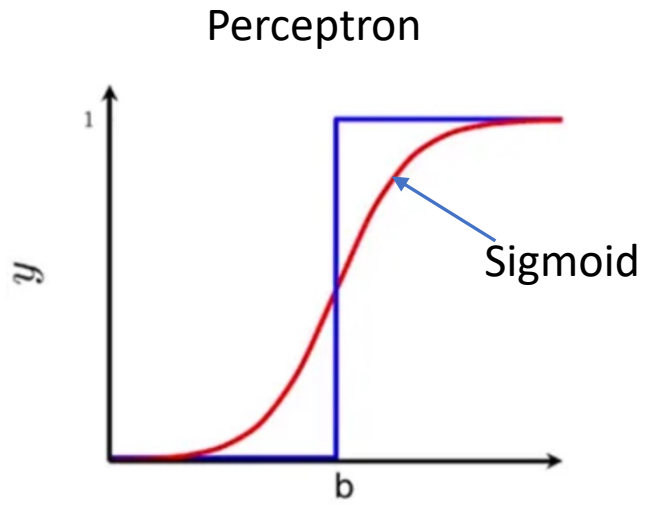


$$y = 1 \; if \sum_{i=0}^{n} w_i * xi \geq 0$$

$$= 0 \; if \sum_{i=0}^{n} w_i * xi < 0$$



$$y = \frac{1}{1 + e^{-(\sum_{i=0}^{n} w_i * xi)}}$$

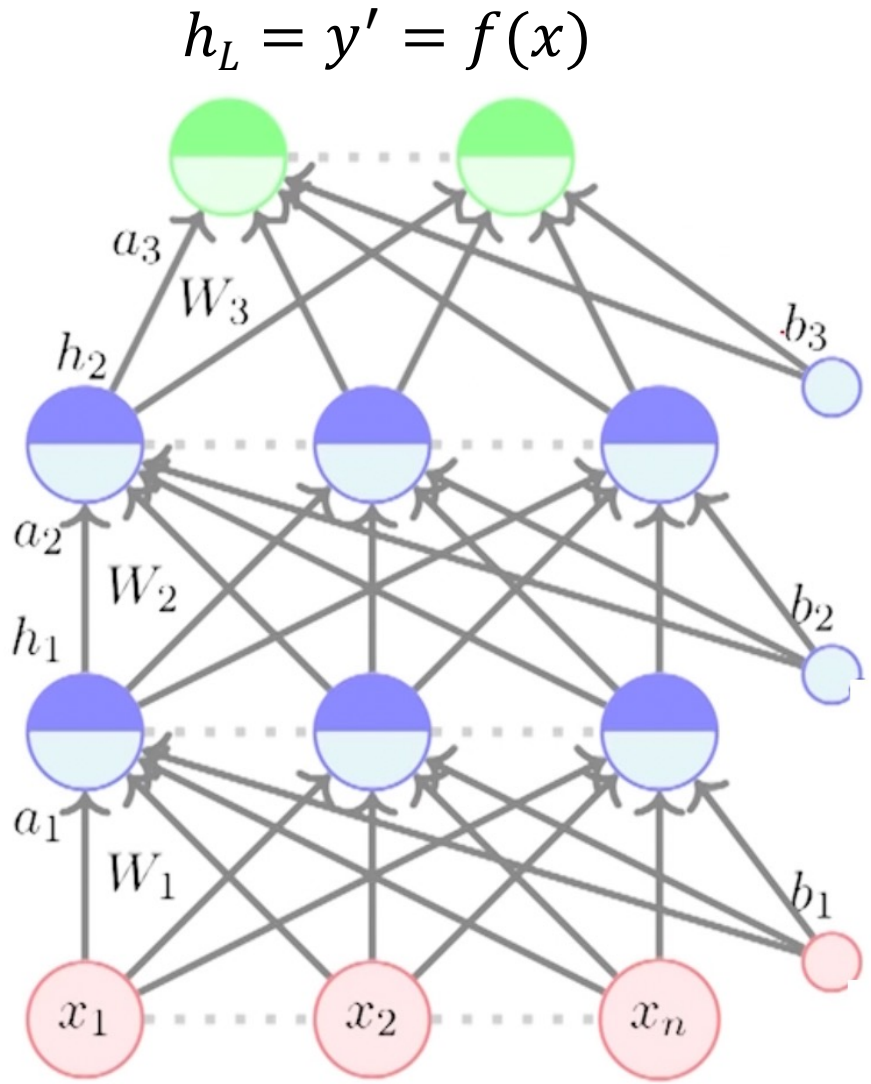


Feed Forward Neural Network

- The input to the network is an n-dimensional vector
- The network contains $L-1$ hidden layers, having n neurons in each hidden layer
- There is one output layer containing k neurons
- Each neuron in hidden layer can be thought of having two parts:
  - pre-activation and activation
- The input layer can be called 0-th layer, and the output layer can be called L-th layer

Feed Forward Neural Network

- This network has 2 hidden layers
- 'a' and 'h' are vectors of length n
- Dimension of h_L is k (as k neurons at output layer)
- Dimension of W_1 and W_2 are nxn and for W_3, dimension is nxk
- Dimension of b (bias) vectors are n, for output layer, dimension of b is k



Feed Forward Neural Network

- The pre-activation at layer i is given by

$$a_i(x) = b_i + w_i h_{i-1}(x)$$

- The activation at layer i is given by

$$h_i(x) = g(ai(x))$$

Here $g(.)$ can be sigmoid function, which is called the activation function

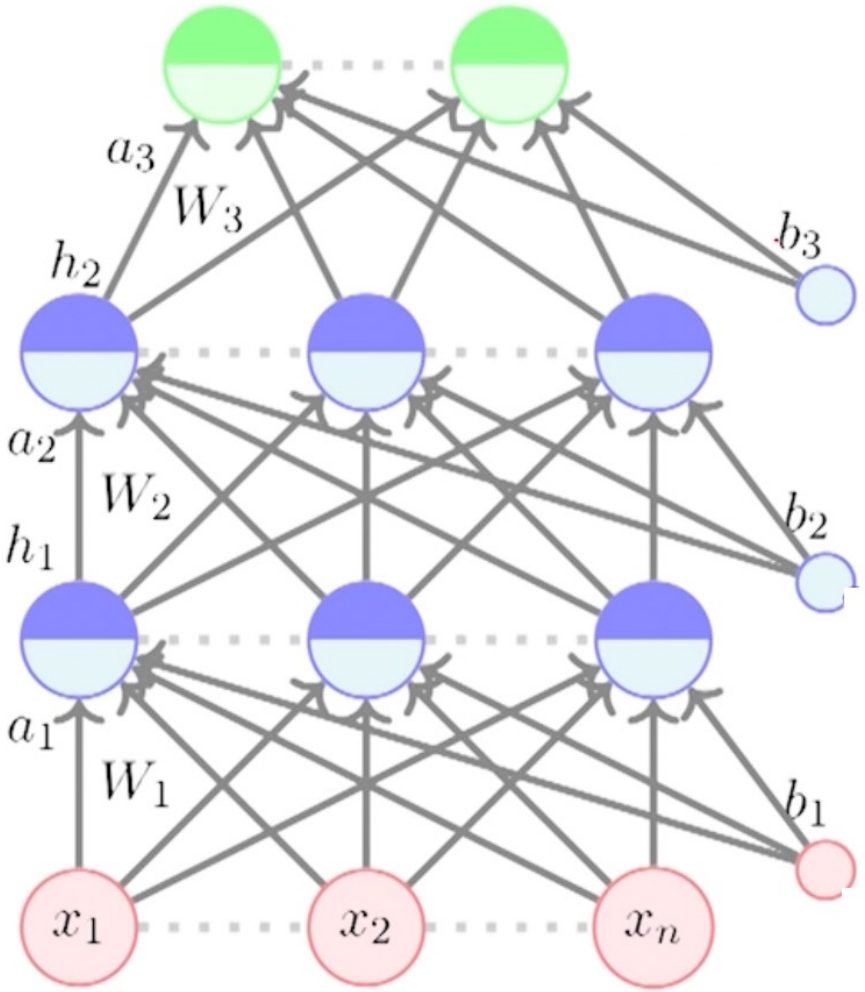
- The output at layer L is given by

$$f(x) = hL = O(aL)$$

O is the output activation function such as softmax, linear, etc.

$$y' = f(x) = O(W_3 g(W_2 g(W_1 x + b_1) + b_2 + b_3)$$

$$h_L = y' = f(x)$$



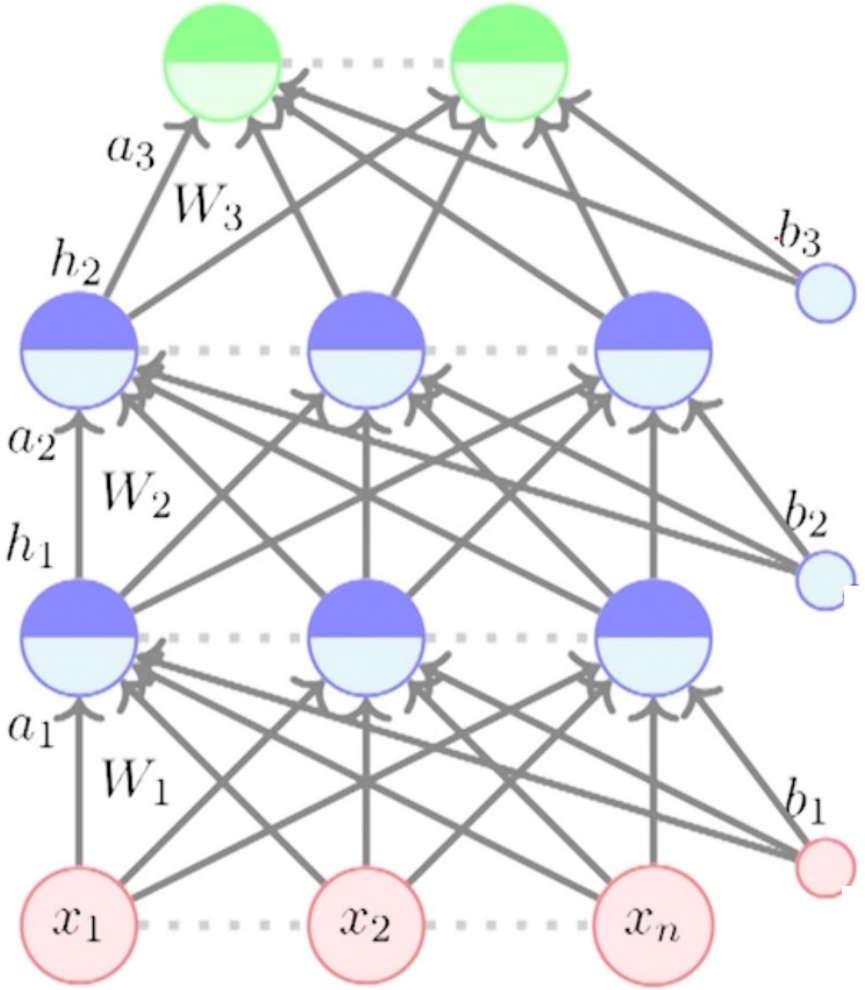
Feed Forward Neural Network

$$h_L = y' = f(x)$$

- Data: $\{x_i, yi\} \forall i \in n$
- Model:

$$y' = f(x) = O(W_3 g(W_2 g(W_1 x + b_1) + b_2 + b_3)$$

- Parameters: W and b
- Learning Algorithm: Gradient Descent with backpropagation
- Objective/Loss function: Mean squared error (MSE), Binary cross entropy, etc.



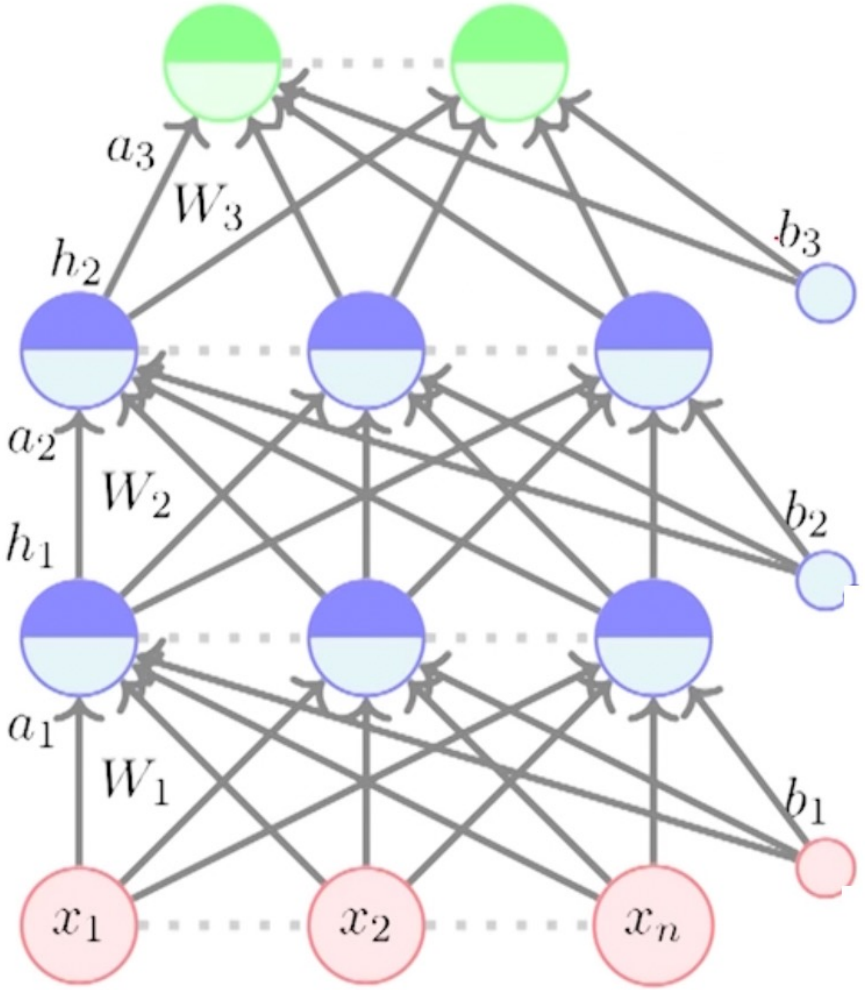
Feed Forward Neural Network

$$h_L = y' = f(x)$$

- Gradient descent algorithm:

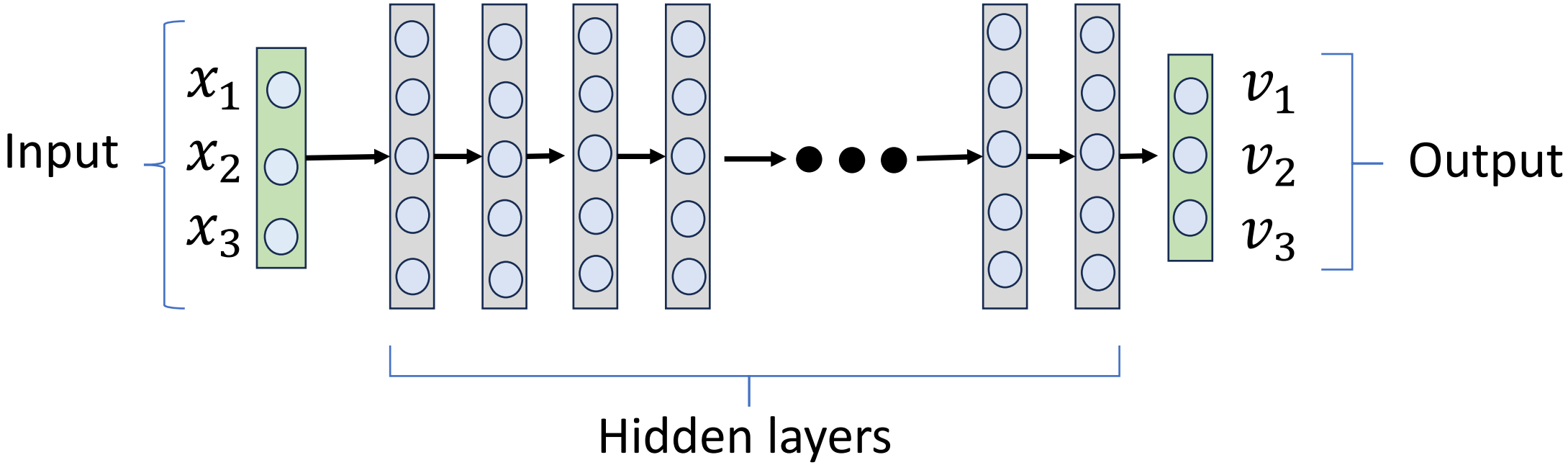
Algorithm: gradient_descent()

$t \leftarrow 0$;
$max\_iterations \leftarrow 1000$;
$Initialize \quad \theta_0 = [w_0, b_0]$;
while $t{+}{+} < max\_iterations$ **do**
  $\theta_{t+1} \leftarrow \theta_t - \eta \nabla \theta_t$;
end

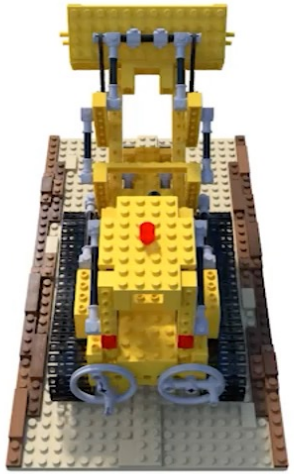


A Multilayer Perceptron (MLP)

- A type of feed forward neural network



NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

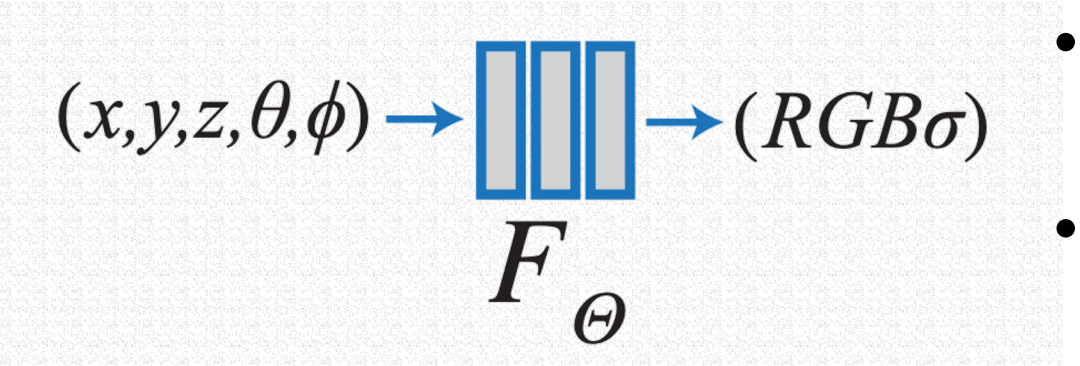





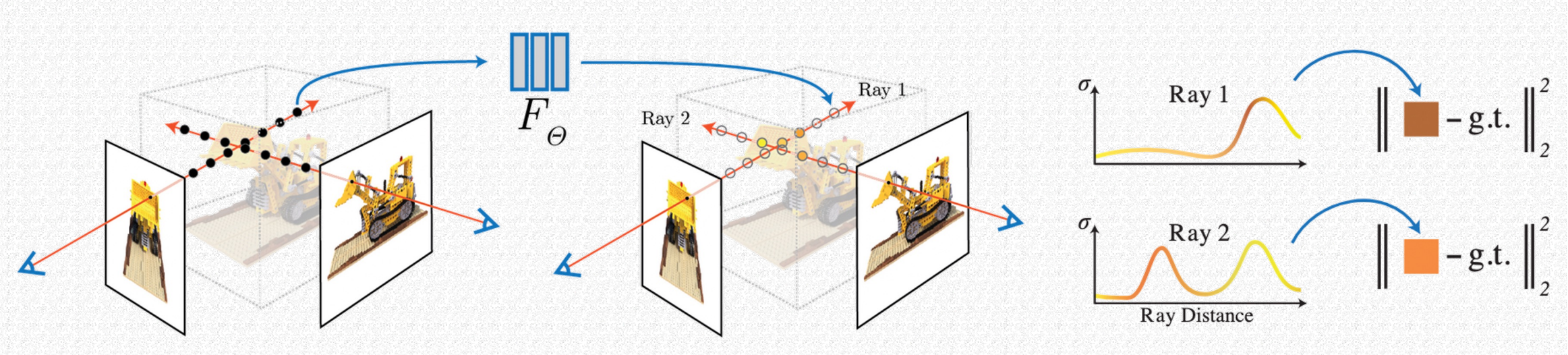


NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis



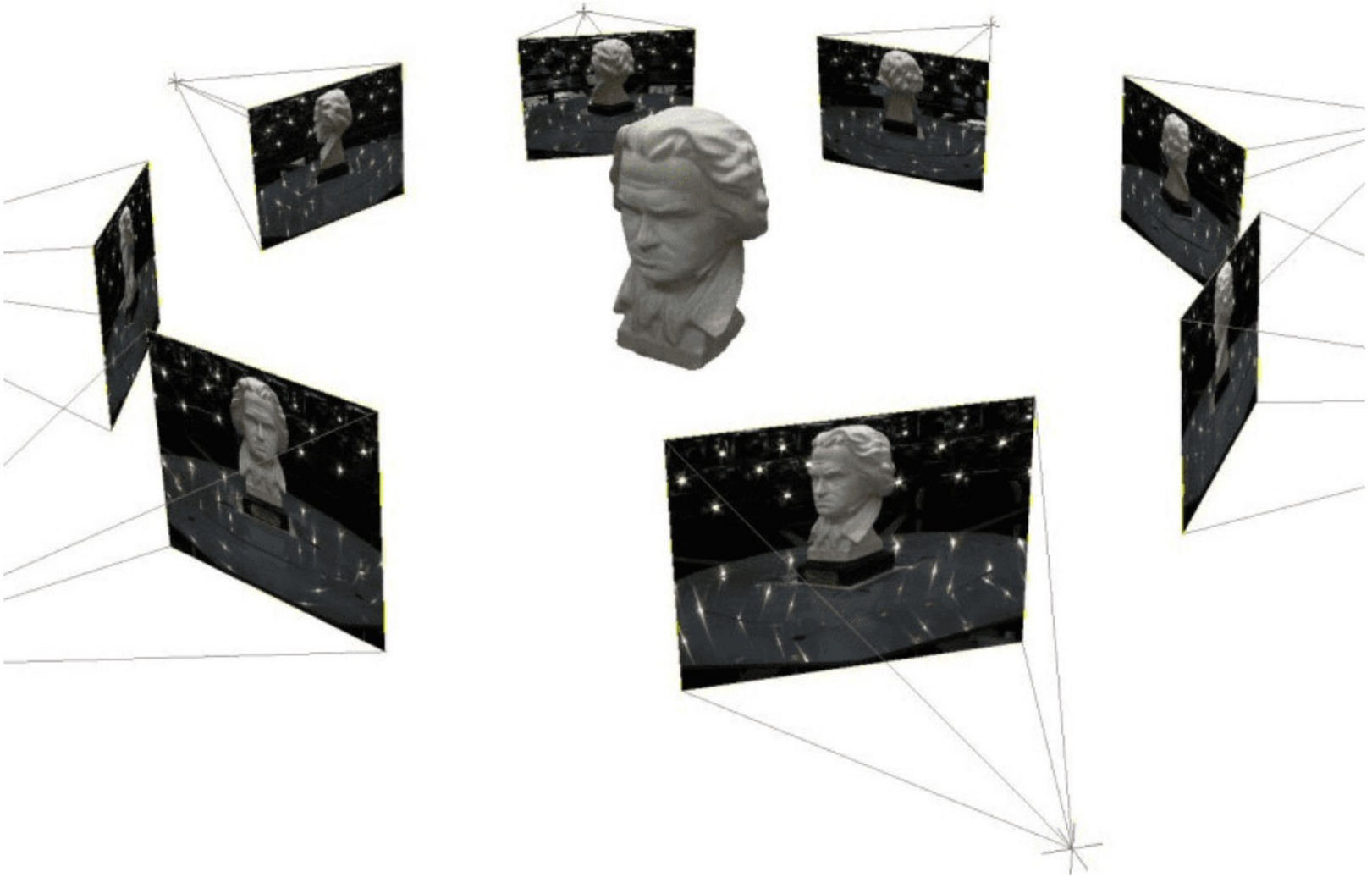
- Input: camera position and view direction
- Output: color and volume density



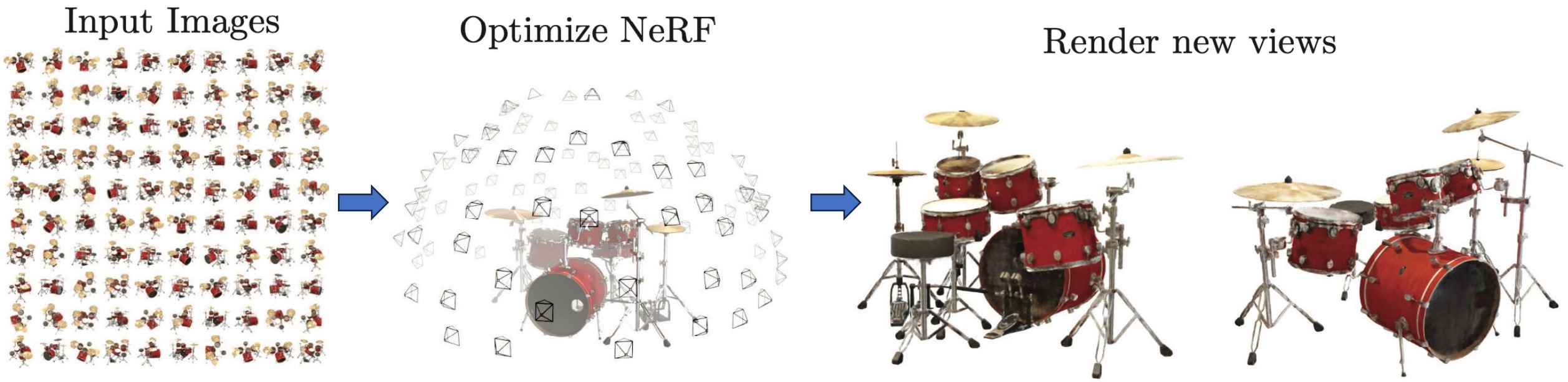
NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

- NeRF learns the radiance field in 3D volume space
  - Given a dataset containing images of a scene, their corresponding camera poses, and intrinsic parameters
  - Output: Predict the color and volume density for every viewing location and direction
- Input to the model is images of the scene from several different view directions
- After training, model can predict images for novel views with high accuracy

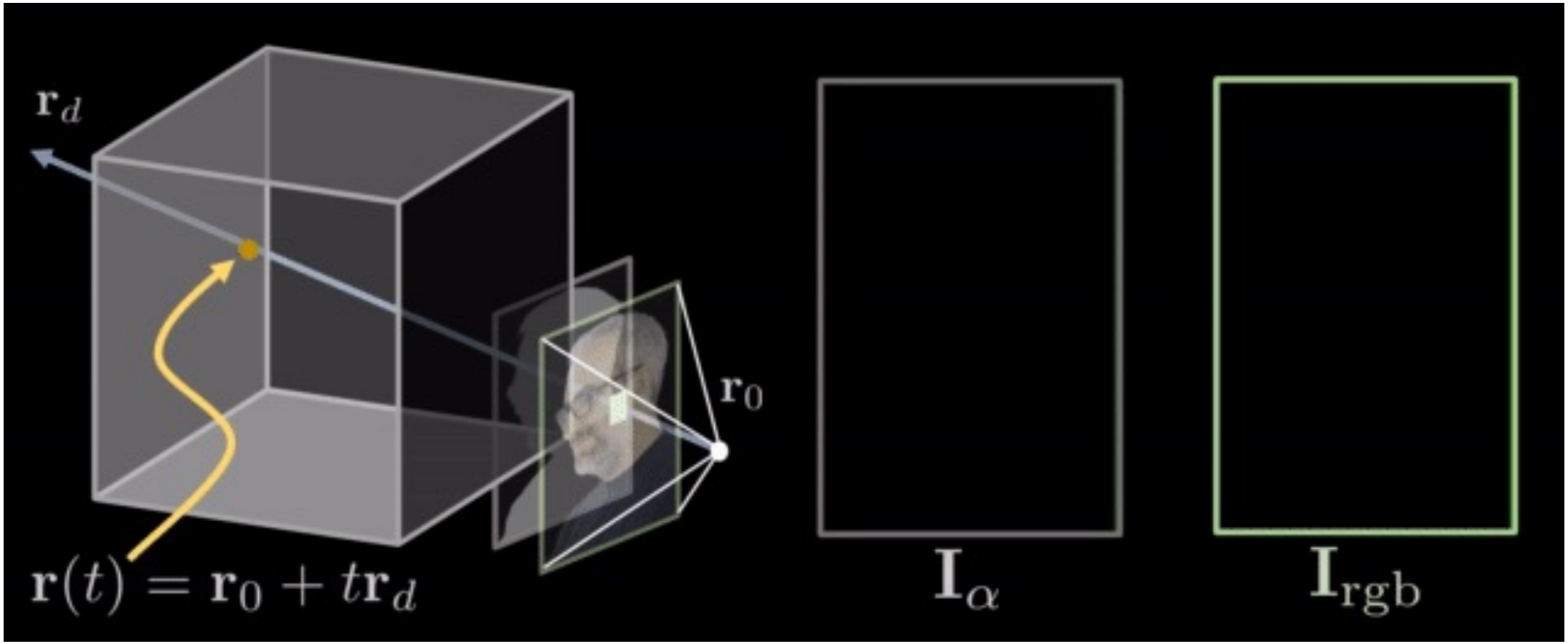
NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis



NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis



NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis



NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

- Use Volume Rendering Algorithm (Ray Casting)
- Generating a view from NeRF requires rendering all rays that pass through each pixel of the desired virtual camera

$$C(r) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i \delta_i))c_i \quad \text{where} \quad T_i = \exp(-\sum_{j=1}^{i-1} \sigma_j \delta_j)$$

Expected color of a camera ray

The relative contribution of this segment

Predicted color

Probability that nothing has blocked the ray up to this point

NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

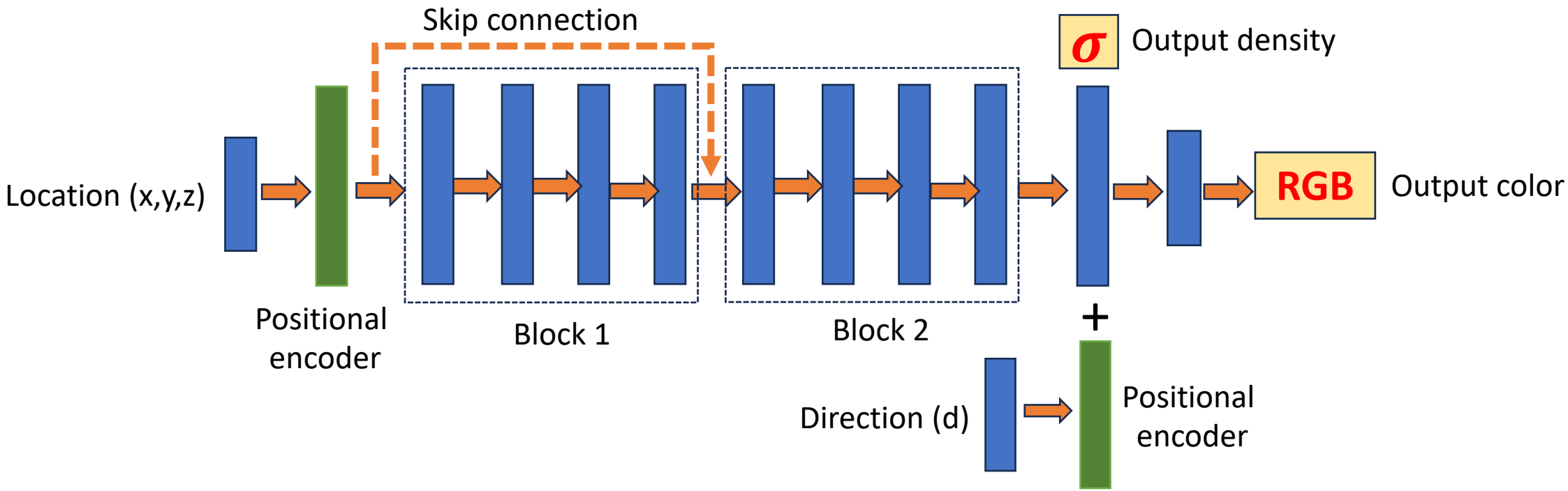
- Use Volume Rendering Algorithm (Ray Casting)
- Generating a view from NeRF requires rendering all rays that pass through each pixel of the desired virtual camera

$$C(r) = \sum_{i=1}^{N} T_i(1 - \exp(-\boldsymbol{\sigma_i}\delta_i))\boldsymbol{c_i} \quad \text{where} \quad T_i = \exp(-\sum_{j=1}^{i-1} \sigma_j\delta_j)$$

NeRF model learns to predict

NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

- Neural Network architecture is built using simple Multi Layer Perceptron (MLP)



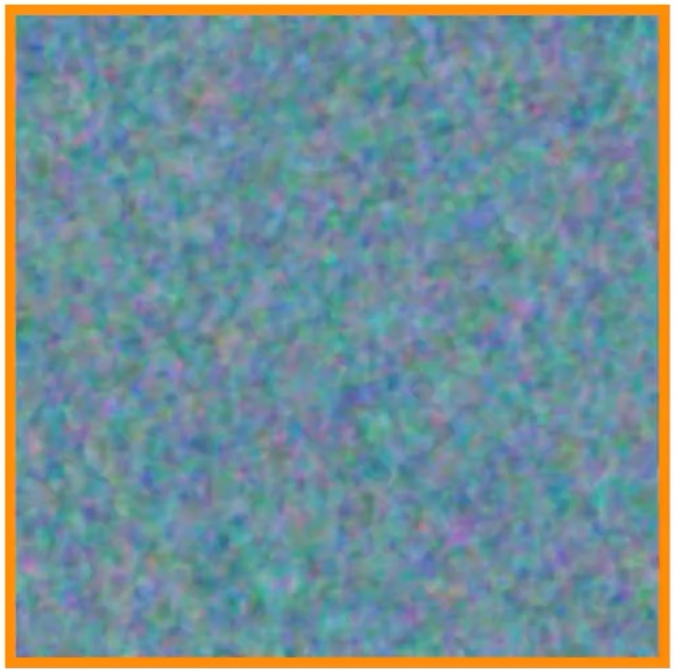
NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

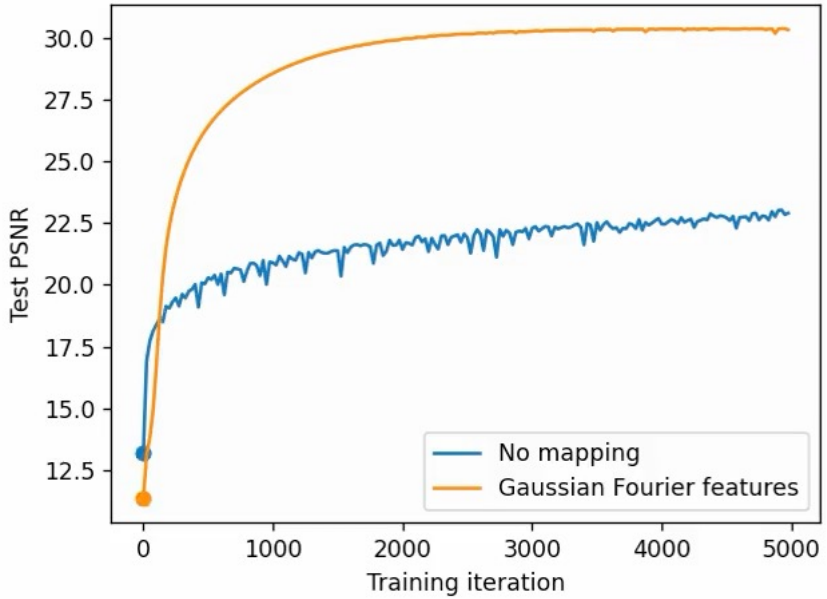
- Positional Encoding:
  - Fully-connected deep networks are biased to learn low frequencies faster
  - Positional encoding helps in learning high frequency details of the data more accurately
  - Transformer architectures use similar encoding for a different purpose

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

NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

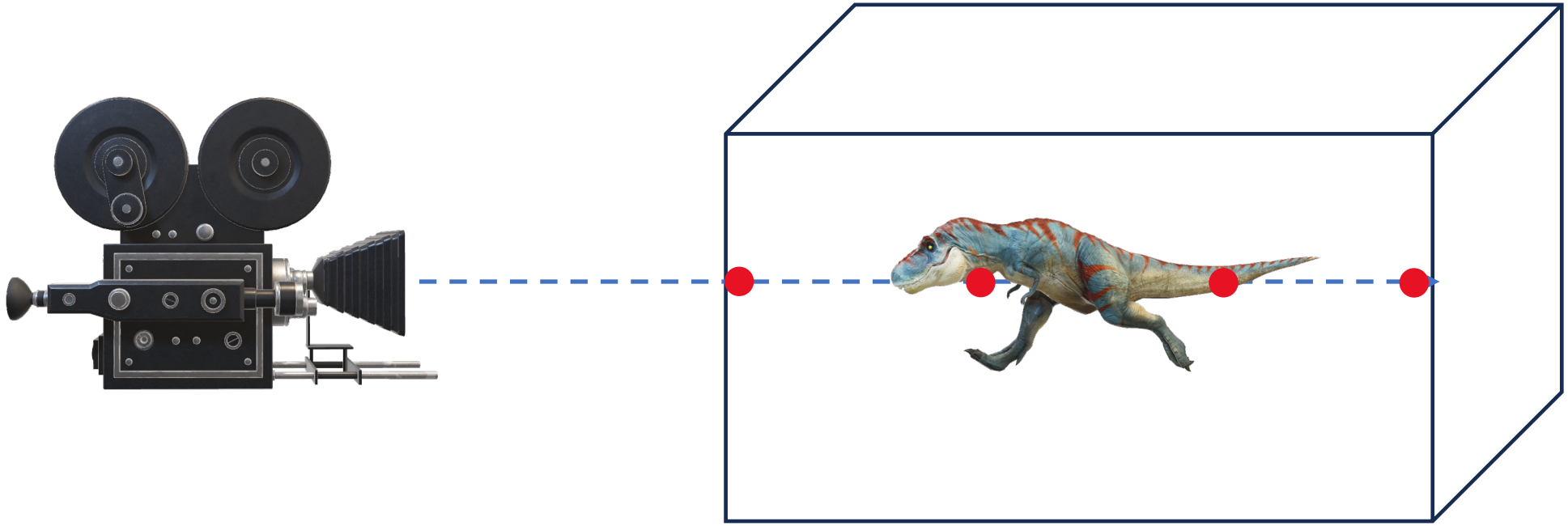
- Positional Encoding:
  - Fully-connected deep networks are biased to learn low frequencies faster
  - Positional encoding helps in learning high frequency details of the data more accurately





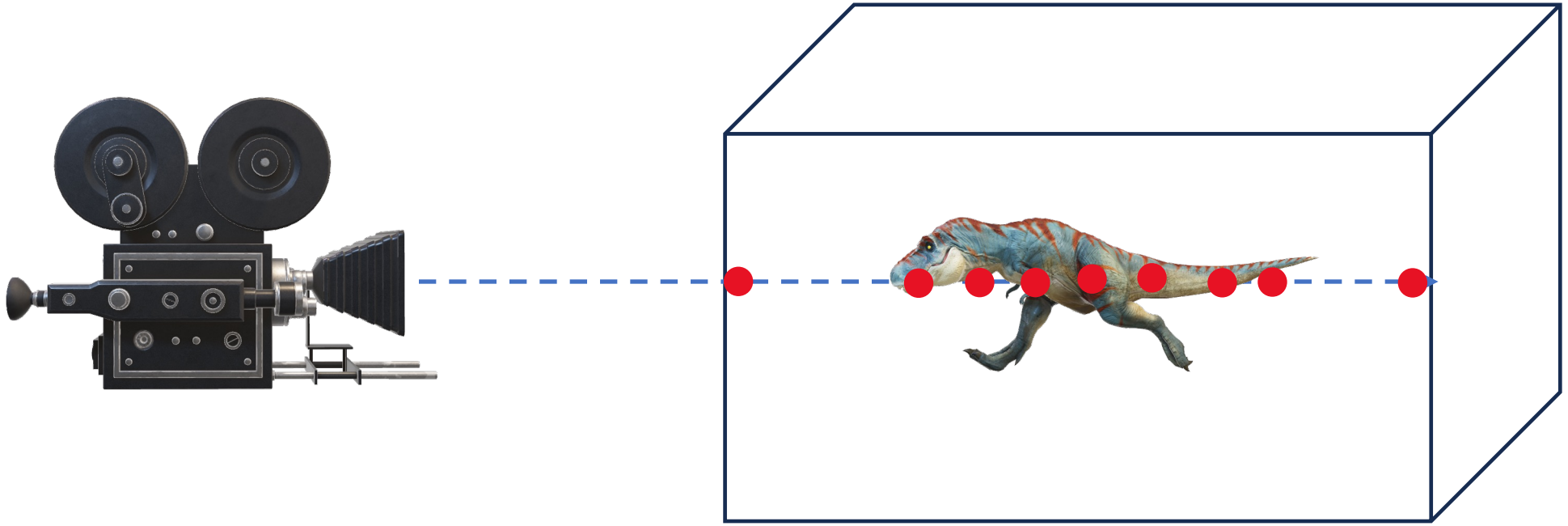
NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

- Hierarchical Volume Sampling:
  - Inefficient to integrate over empty
  - Allocate samples proportionally to their expected effect on the final rendering



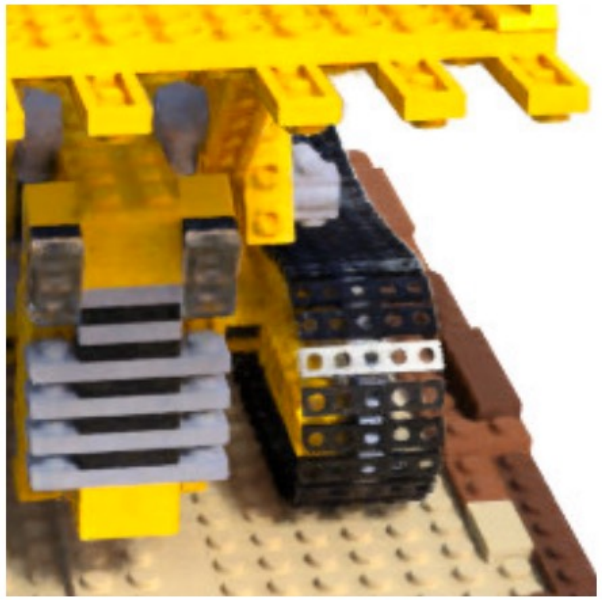
NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

- Hierarchical Volume Sampling:
  - Inefficient to integrate over empty
  - Allocate samples proportionally to their expected effect on the final rendering

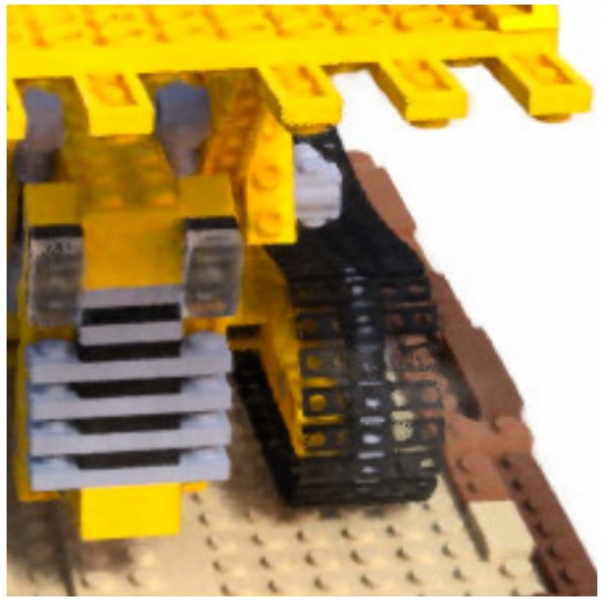


NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

Ground Truth



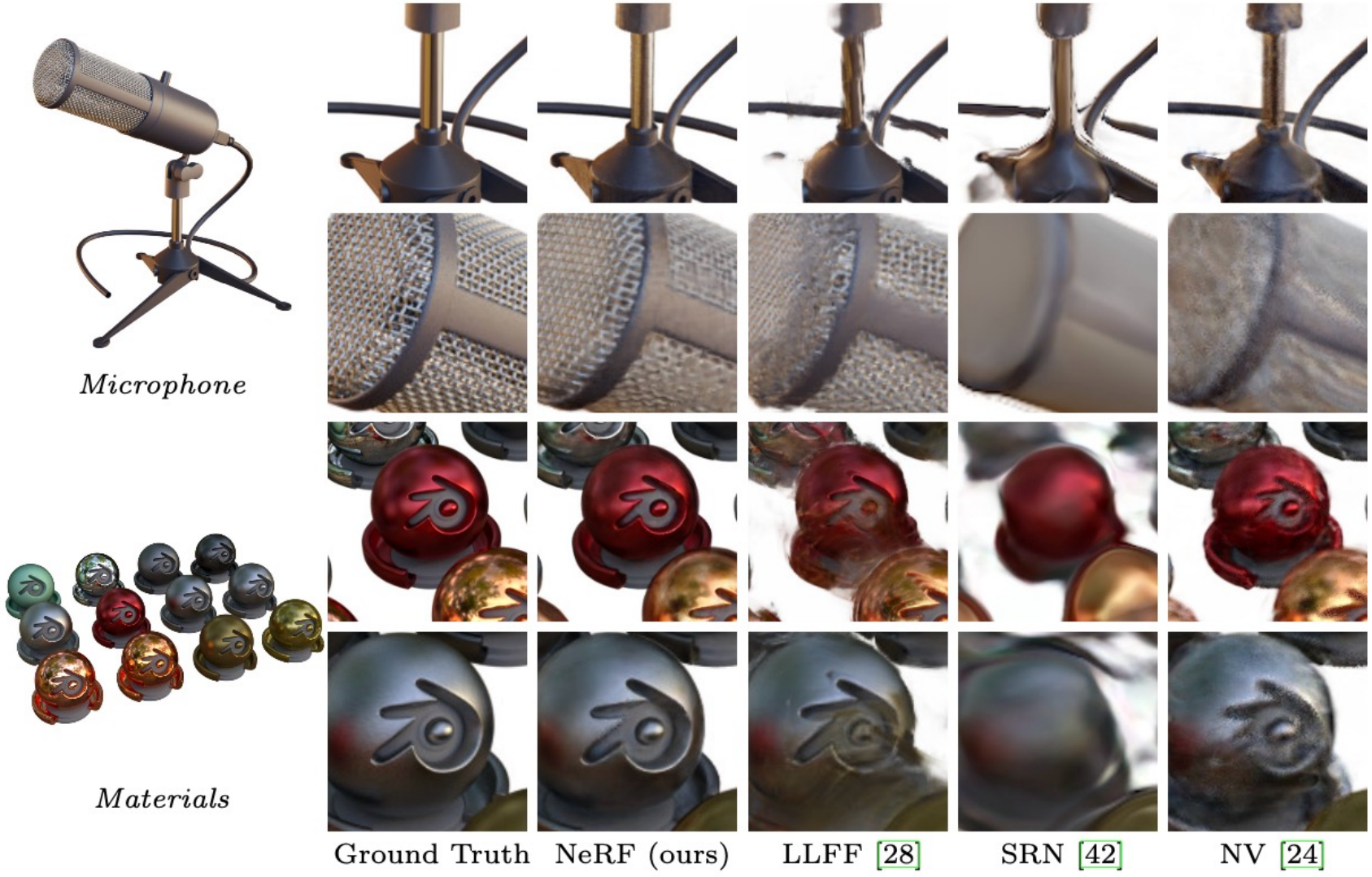
Complete Model



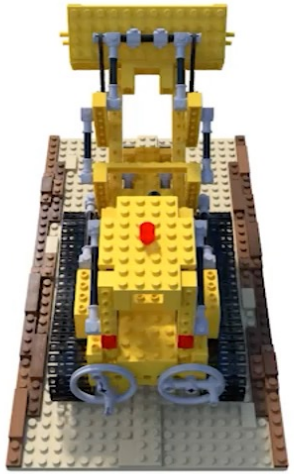
No View Dependence

No Positional Encoding

NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis



NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis



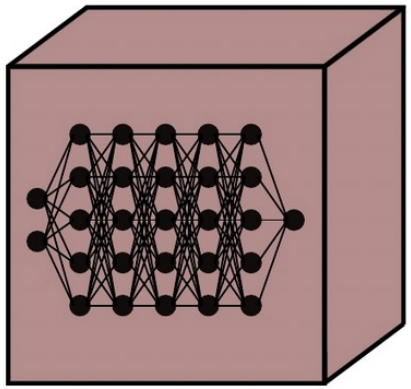


NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

	Input	#Im.	L	(N_c, N_f)	PSNR↑	SSIM↑	LPIPS↓
1) No PE, VD, H	xyz	100	-	(256, -)	26.67	0.906	0.136
2) No Pos. Encoding	$xyz\theta\phi$	100	-	(64, 128)	28.77	0.924	0.108
3) No View Dependence	xyz	100	10	(64, 128)	27.66	0.925	0.117
4) No Hierarchical	$xyz\theta\phi$	100	10	(256, -)	30.06	0.938	0.109
5) Far Fewer Images	$xyz\theta\phi$	25	10	(64, 128)	27.78	0.925	0.107
6) Fewer Images	$xyz\theta\phi$	50	10	(64, 128)	29.79	0.940	0.096
7) Fewer Frequencies	$xyz\theta\phi$	100	5	(64, 128)	30.59	0.944	0.088
8) More Frequencies	$xyz\theta\phi$	100	15	(64, 128)	30.81	0.946	0.096
9) Complete Model	$xyz\theta\phi$	100	10	(64, 128)	**31.01**	**0.947**	**0.081**

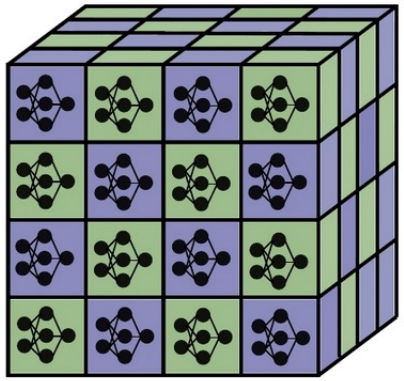
KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs

NeRF

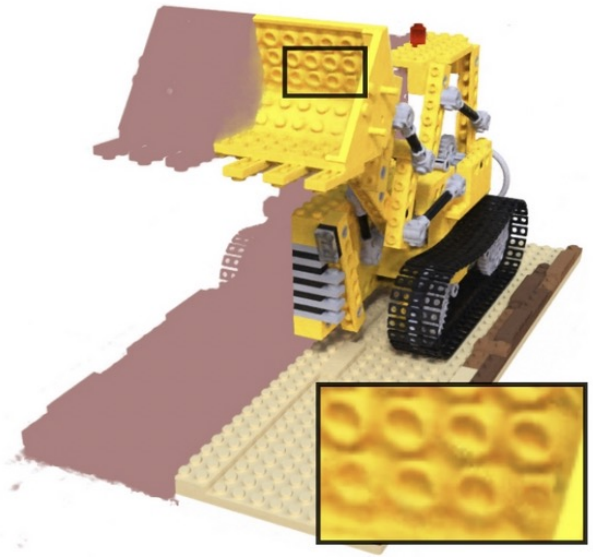


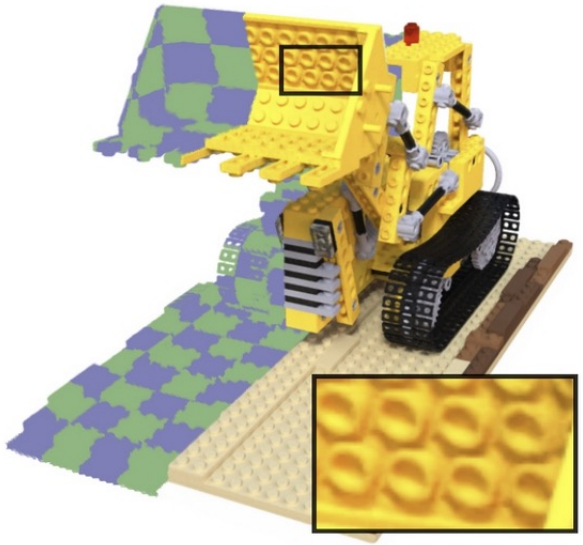
Rendering time: 56 secs

KiloNeRF

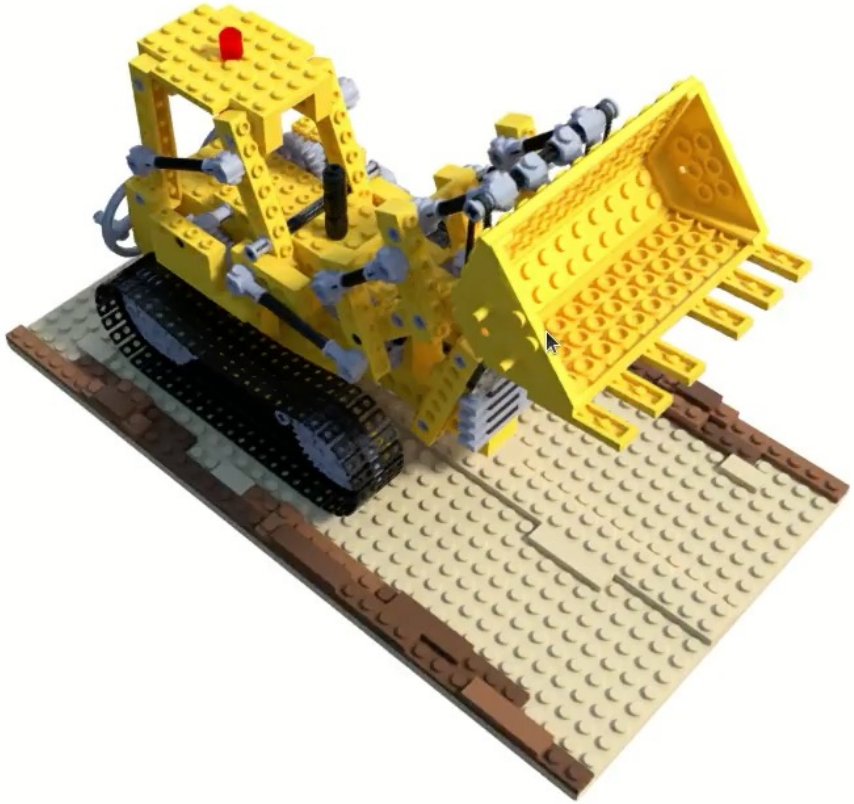


Rendering time: 0.02 secs





KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs



Other works based on Vanilla NeRF

- https://github.com/awesome-NeRF/awesome-NeRF

Papers

- NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, Mildenhall et al., ECCV 2020 | github | bibtex

▼ Faster Inference

- Learning Neural Transmittance for Efficient Rendering of Reflectance Fields, Mohammad Shafiei et al., BMVC 2021 | bibtex
- Neural Sparse Voxel Fields, Liu et al., NeurIPS 2020 | github | bibtex
- AutoInt: Automatic Integration for Fast Neural Volume Rendering, Lindell et al., CVPR 2021 | github | bibtex
- DeRF: Decomposed Radiance Fields, Rebain et al. Arxiv 2020 | bibtex
- DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks, Neff et al., CGF 2021 | github | bibtex
- FastNeRF: High-Fidelity Neural Rendering at 200FPS, Garbin et al., Arxiv 2021 | bibtex
- KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs , Reiser et al., ICCV 2021 | github | bibtex
- PlenOctrees for Real-time Rendering of Neural Radiance Fields, Yu et al., Arxiv 2021 | github | bibtex
- Mixture of Volumetric Primitives for Efficient Neural Rendering, Lombardi et al., SIGGRAPH 2021 | bibtex
- Light Field Networks: Neural Scene Representations with Single-Evaluation Rendering, Sitzmann et al., Arxiv 2021 | github | bibtex
- RT-NeRF: Real-Time On-Device Neural Radiance Fields Towards Immersive AR/VR Rendering, Li et al., ICCAD 2022 | bibtex
- ENeRF: Efficient Neural Radiance Fields for Interactive Free-viewpoint Video, Lin et al., SIGGRAPH 2022 | github | bibtex

Spectral Bias of (ReLU) MLPs

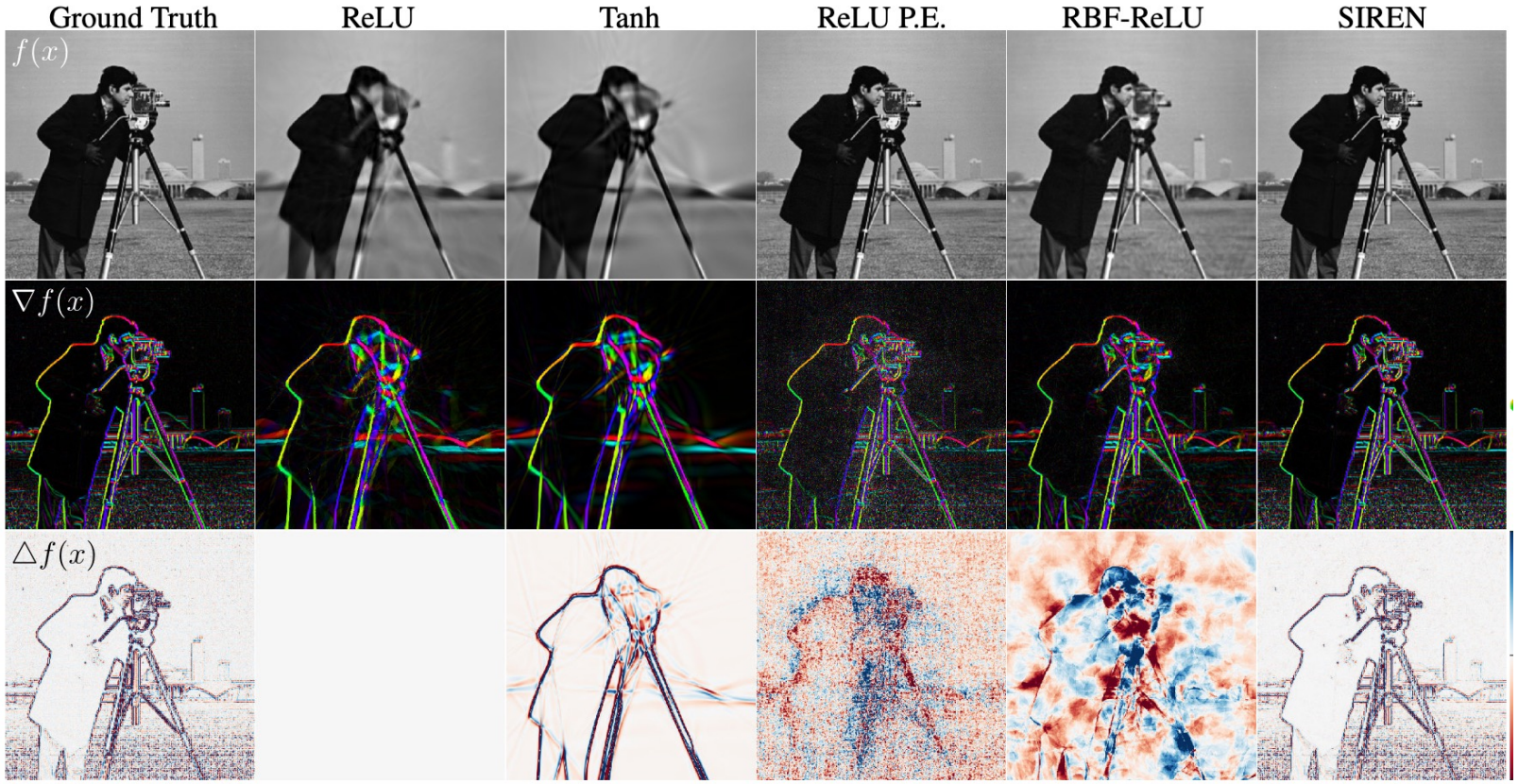
- Deep Networks prioritize learning simple patterns that usually generalize well across data samples
  - On the Spectral Bias of Neural Networks, Rahaman et al.
  - Prioritizes learning low frequency information
  - The spectral response decays quickly
  - Learning of high frequency information takes longer time and often deeper network architecture
- Solutions
  - Use Fourier feature mapping, i.e., positional encoding
    - Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains, Tancik et al.
  - Use sinusoidal activation function

MLP with Sinusoidal Activation Function

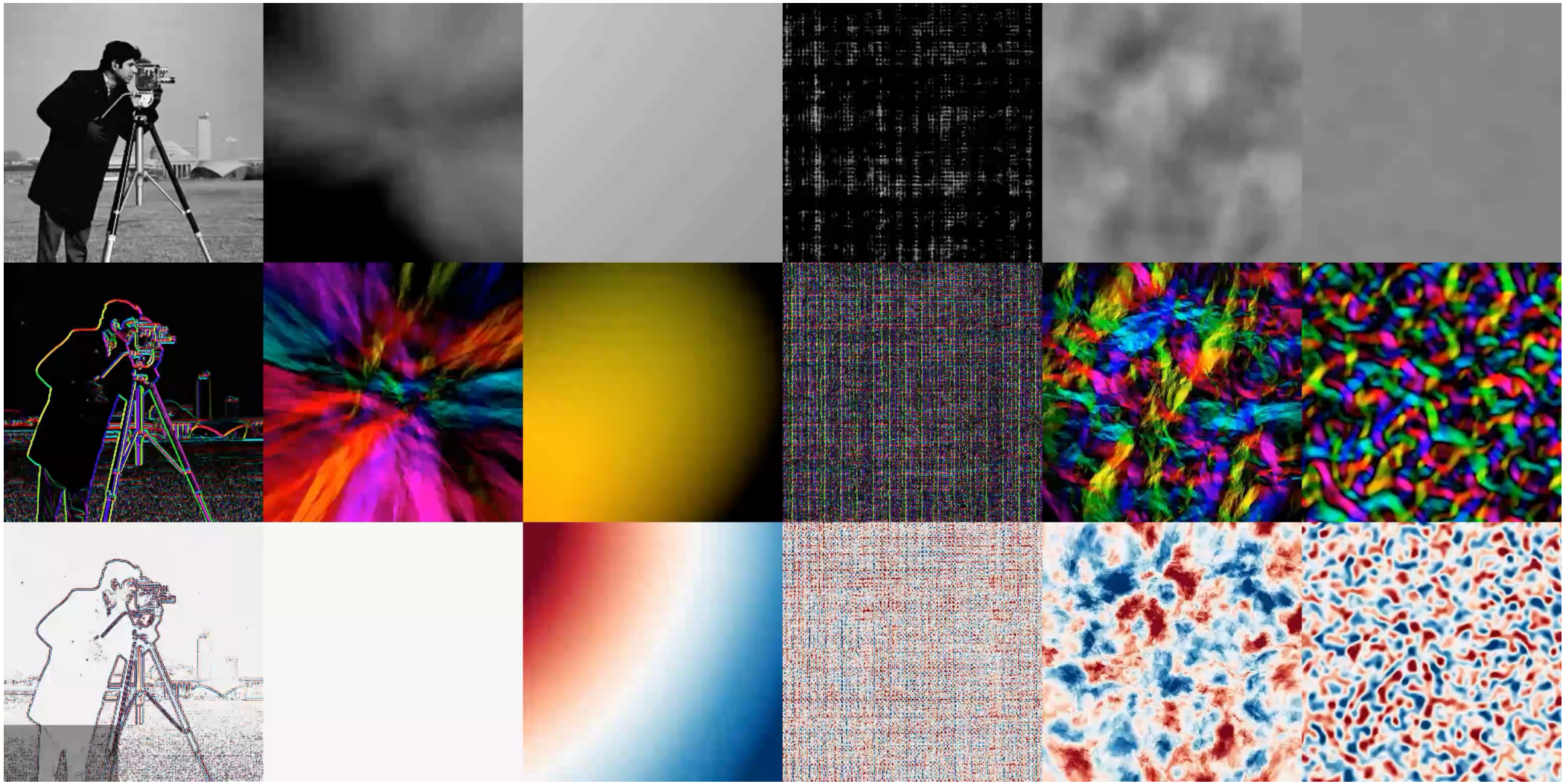
- A good fit for images, 3D field type data that can be periodic and contain information of various types of frequencies
  - Textures, edges, fine details
- Learning in a rich Fourier spectrum space
  - Model can approximate functions with high-frequency components *without* needing deep networks
- Stable Gradients
  - sine function provides stable and informative gradients
  - If initialized carefully, gradients do not vanish or explode

SIREN: SInusoidal REpresentation Network

- A multi-layer perceptron network with **sinusoidal activation function**
- Excellent to model coordinate-based data sets
- Can learn higher order gradients of the data

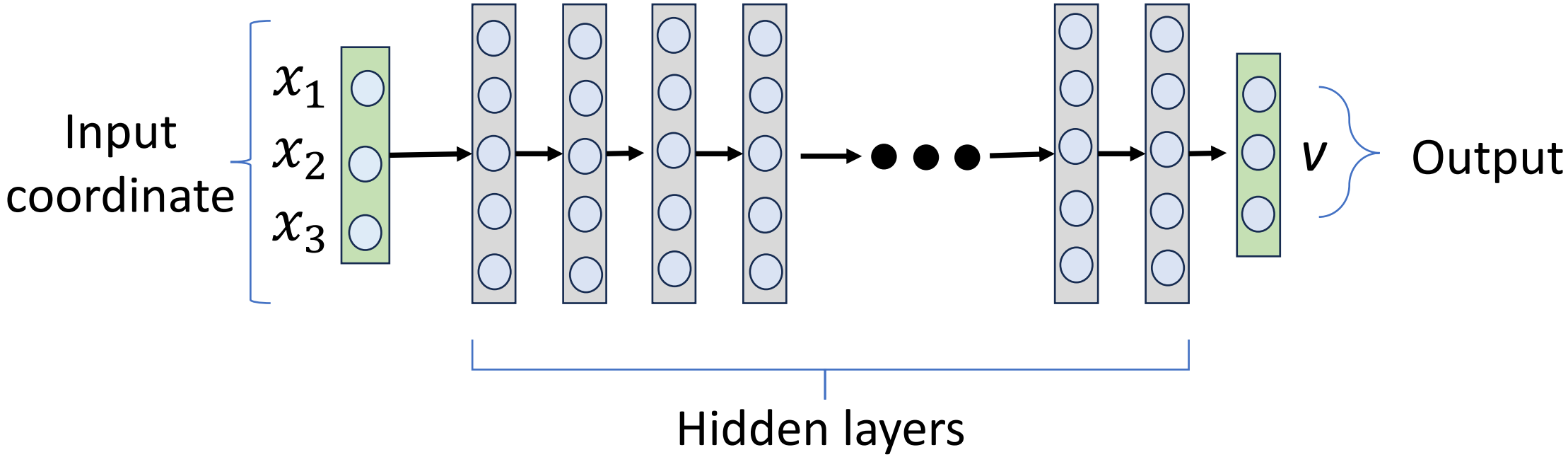


SIREN: Sinusoidal Representation Network

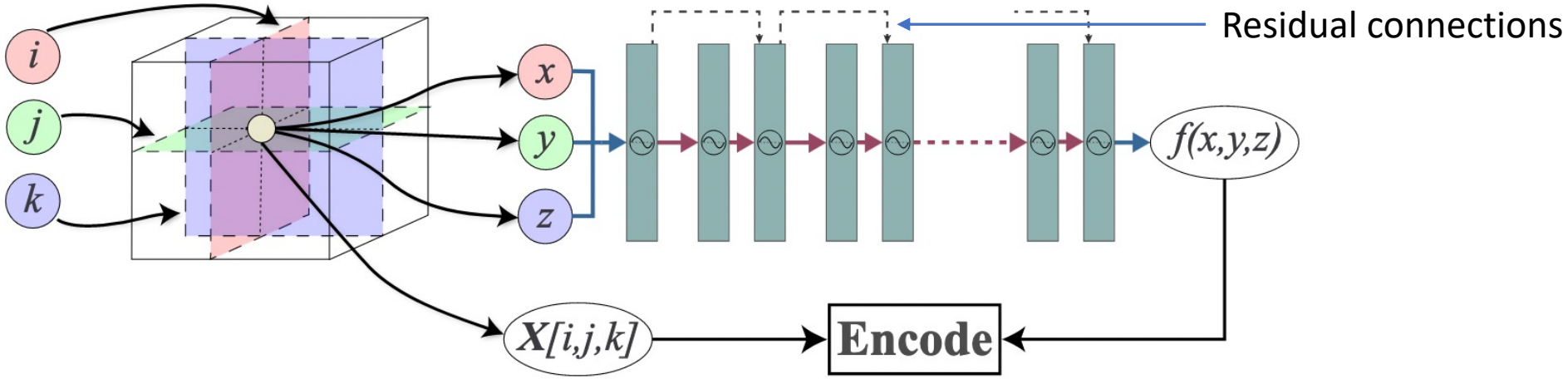


Compressive Neural Network

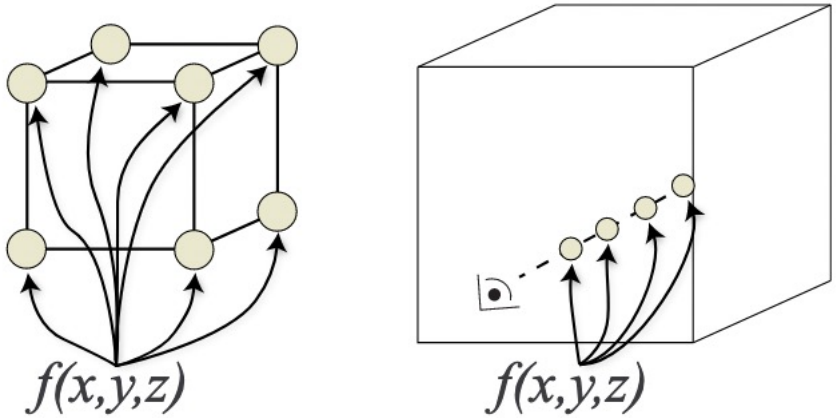
- A Multiplayer Perceptron with sine activation function



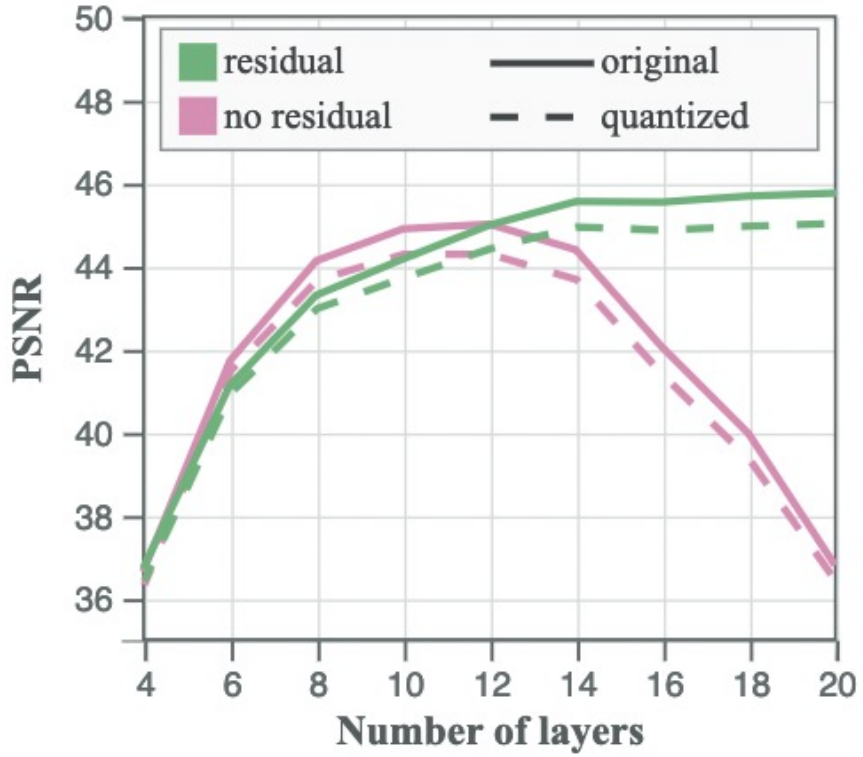
Compressive Neural Network



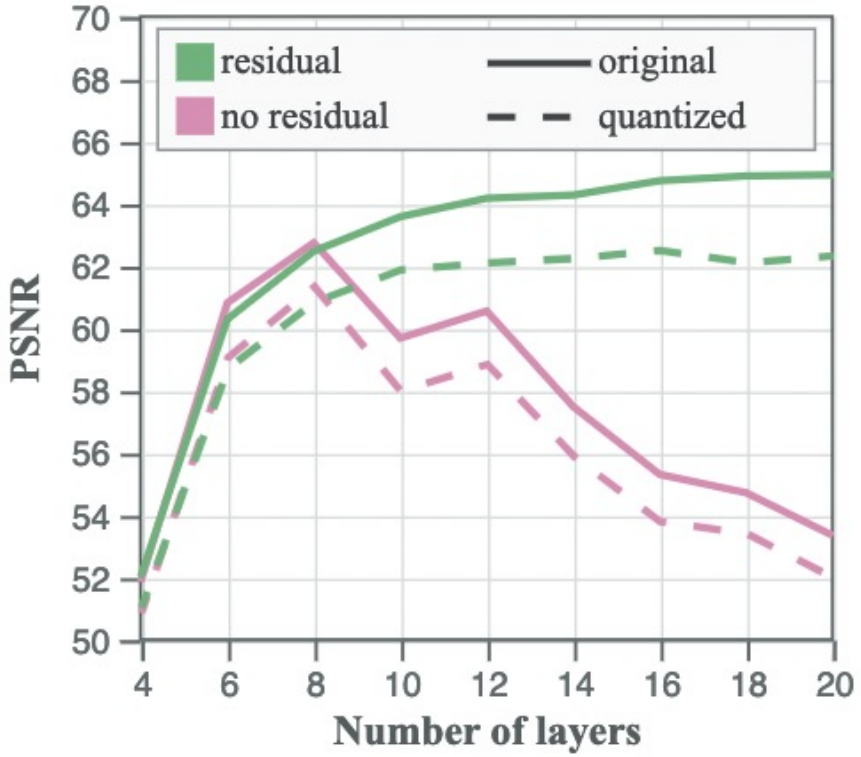
Decode



Benefits of Adding Residual Blocks



(a) Asteroid



(b) Ionization

Compressive Neural Network

- Loss function Mean squared error (MSE)

$$\min_{\Theta} \sum_{\mathbf{i}} \left(f_{\Theta}(\mathbf{p_i}) - \mathbf{X}[i_1, i_2, \cdots, i_d]\right)^2$$

Compressive Neural Network

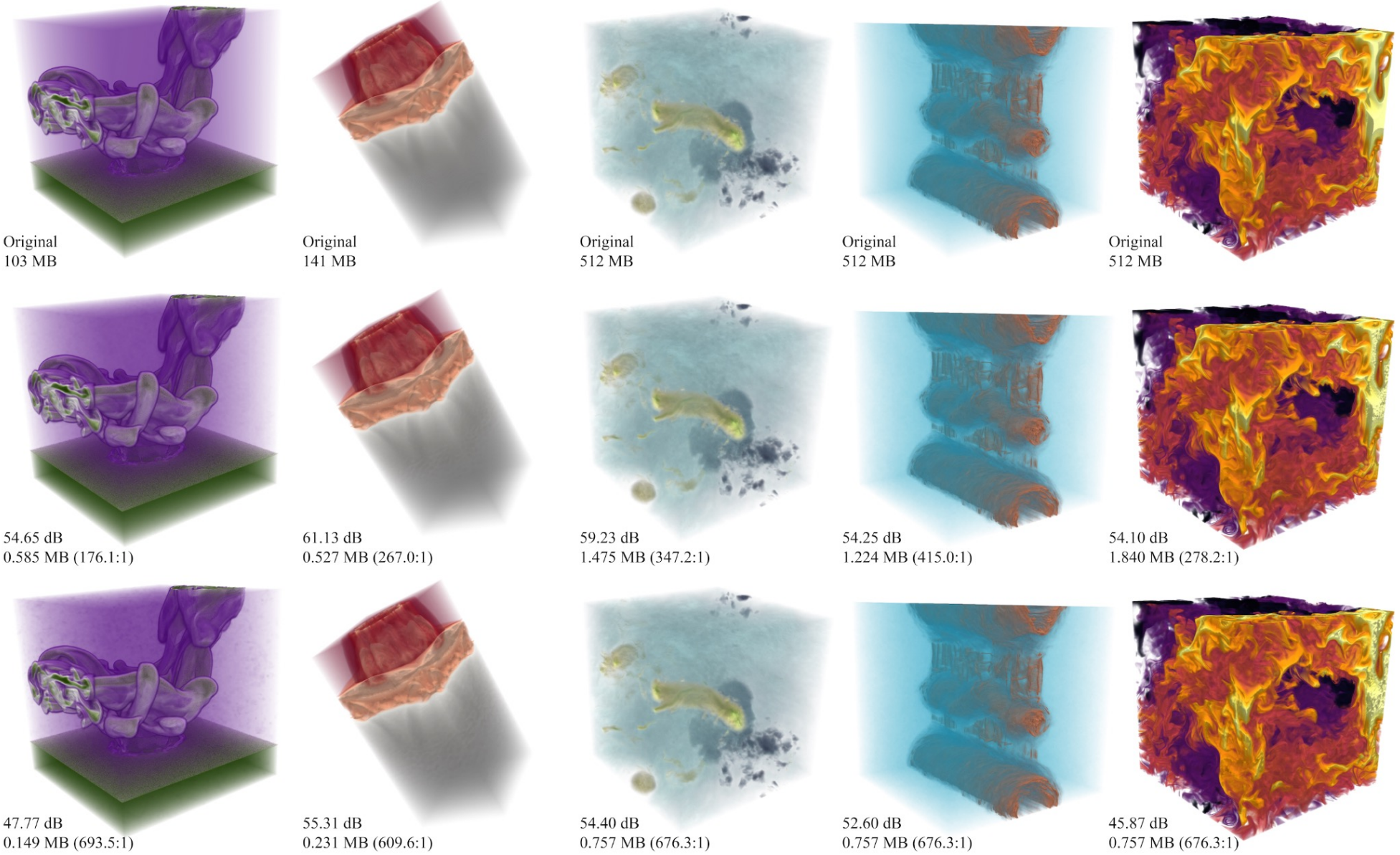
- Loss function Mean squared error (MSE) + Gradient regularization

$$\min_{\Theta} \sum_{\mathbf{i}} \left(f_{\Theta}(\mathbf{p_i}) - \mathbf{X}[i_1, i_2, \cdots, i_d]\right)^2$$

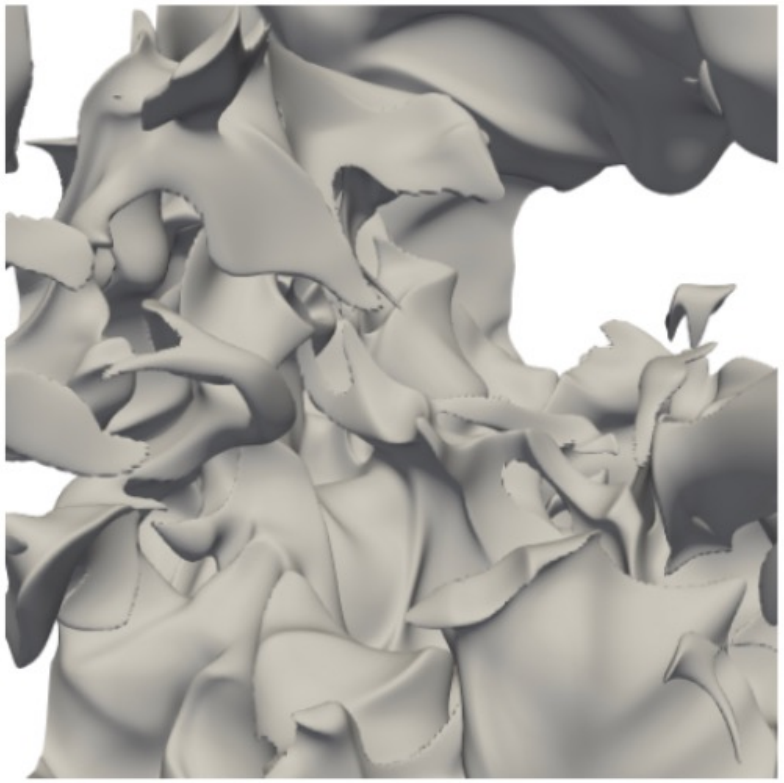
$$+$$

$$\lambda \| \nabla f_{\Theta}(\mathbf{p_i}) - \mathbf{X}^{'}[i_1, i_2, \cdots, i_d] \|_2^2$$

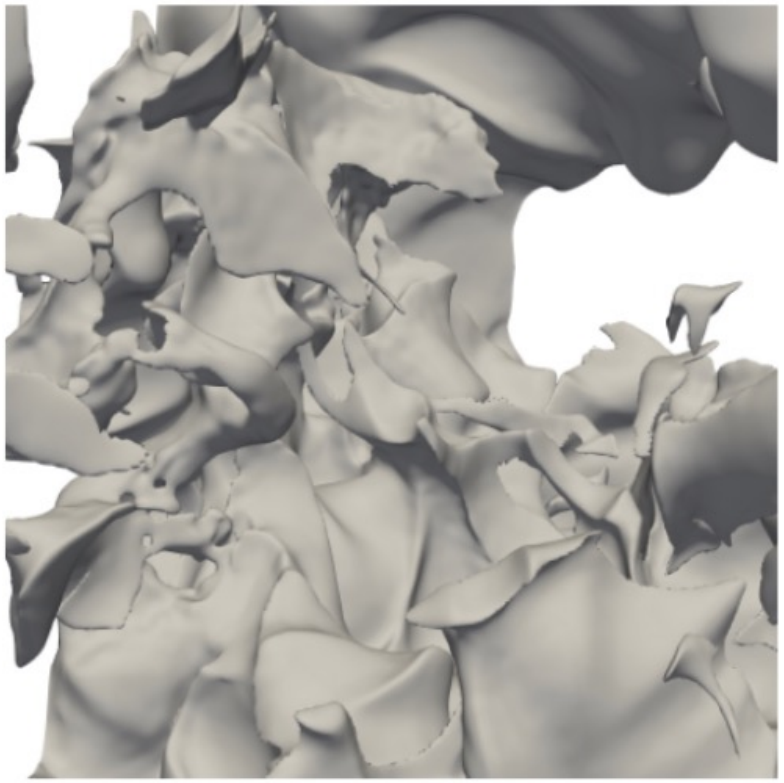
Reconstruction Results



Reconstruction Results: Isosurface

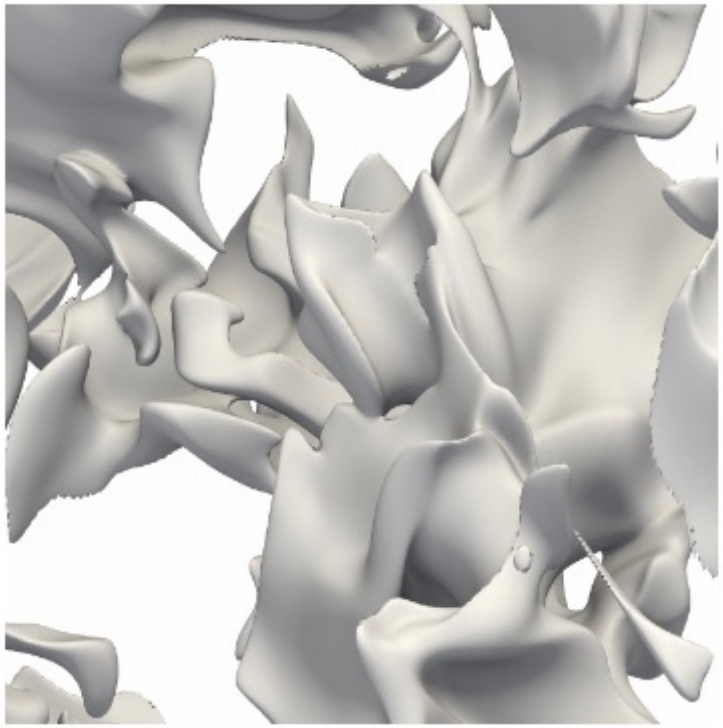


(a) ground truth

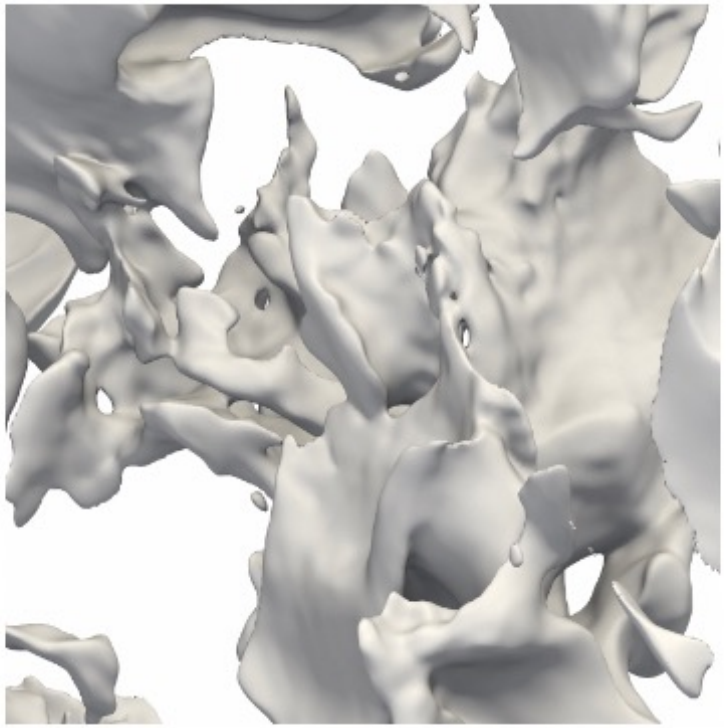


(b) neurcomp

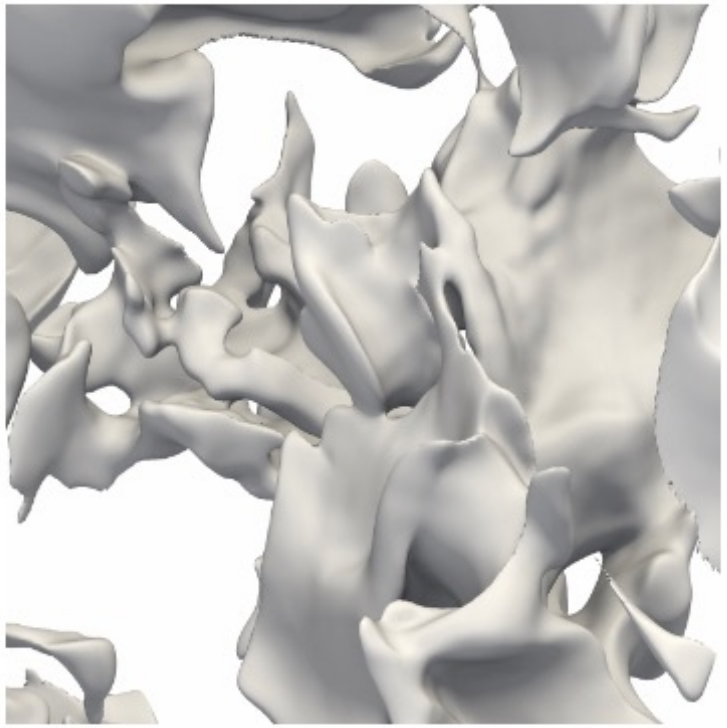
Impact of Gradient Regularization



(a) ground truth



(b) no gradient reg.



(c) gradient reg.