Implicit Neural Representation with Periodic Activation Functions

Presenter: Zhenyu Jiang
Slides from: Jang Hyun Cho and Marco Bueso

09/01/2021

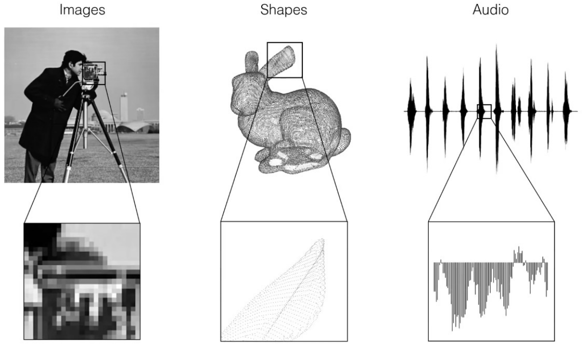
How do we represent signals ?

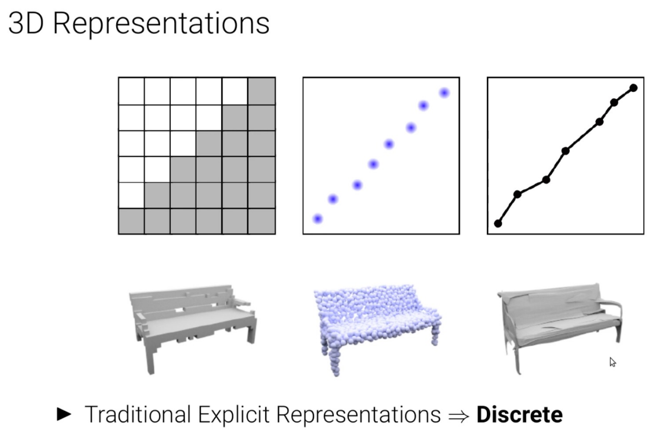
1. Image → Discrete pixels

2. 3D shape → Voxels, point clouds, meshes

3. Sound wave → Discrete samples of intensity

Lose details when representing signals in discrete manner !!

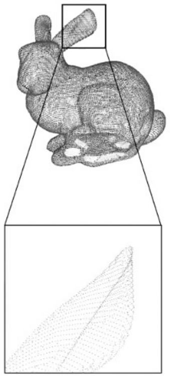




Neural Implicit Representation

Instead of representing signals in a discrete manner, new approach has been studied called **neural implicit representation.**

Shapes



Explicit representation

Spatial coordinate (x, y, z) in **N**

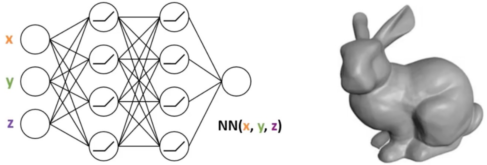
→ Occupancy

1. **Explicit way tends to lose details**
2. **Memory expensive (scales with resolution)**

Implicit representation

Spatial coordinate (x, y, z) in **R**

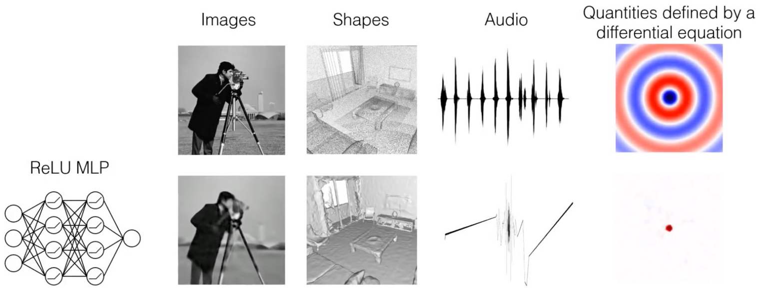
→ Occupancy



Why is it important?

Neural implicit representation is applicable to a variety of scientific fields:

1. **Image/video/audio processing**.
   → Image/video/audio reconstruction



2. **3D shape parts, objects, and scenes generation.**
   → Signed distance functions / occupancy networks.
   → Encode object appearances (from multi-view 2D images – Neural rendering)

3. Solving differential equation for inverse problems.
   → Solving the Poisson Equation.

Why is it important?

Covers a **family of problem setup** that can be expressed as below:

$$\text{find } \Phi(\mathbf{x}) \quad \text{subject to } \mathcal{C}_m\big(\mathbf{a}(\mathbf{x}), \Phi(\mathbf{x}), \nabla\Phi(\mathbf{x}), \ldots\big) = 0,\ \forall \mathbf{x} \in \Omega_m,\ m = 1, \ldots, M$$

Optimize for →

$$\mathcal{L} = \int_\Omega \sum_{m=1}^{M} \mathbf{1}_{\Omega_m}(\mathbf{x}) \, \|\mathcal{C}_m(\mathbf{a}(\mathbf{x}), \Phi(\mathbf{x}), \nabla\Phi(\mathbf{x}), \ldots)\| d\mathbf{x},$$

1. Image fitting
2. Gradient / Laplacian fitting
3. Signed Distance Functions (SDFs)

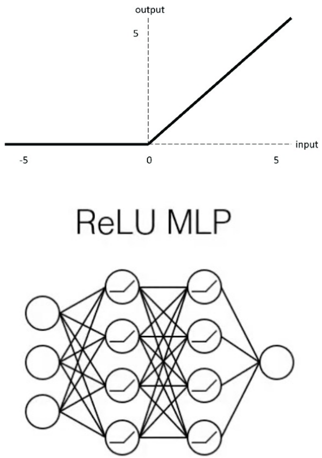
$$\mathcal{L}_{\text{img}} = \int_\Omega \|\Phi(\mathbf{x}) - f(\mathbf{x})\| \, \mathrm{d}\mathbf{x}$$

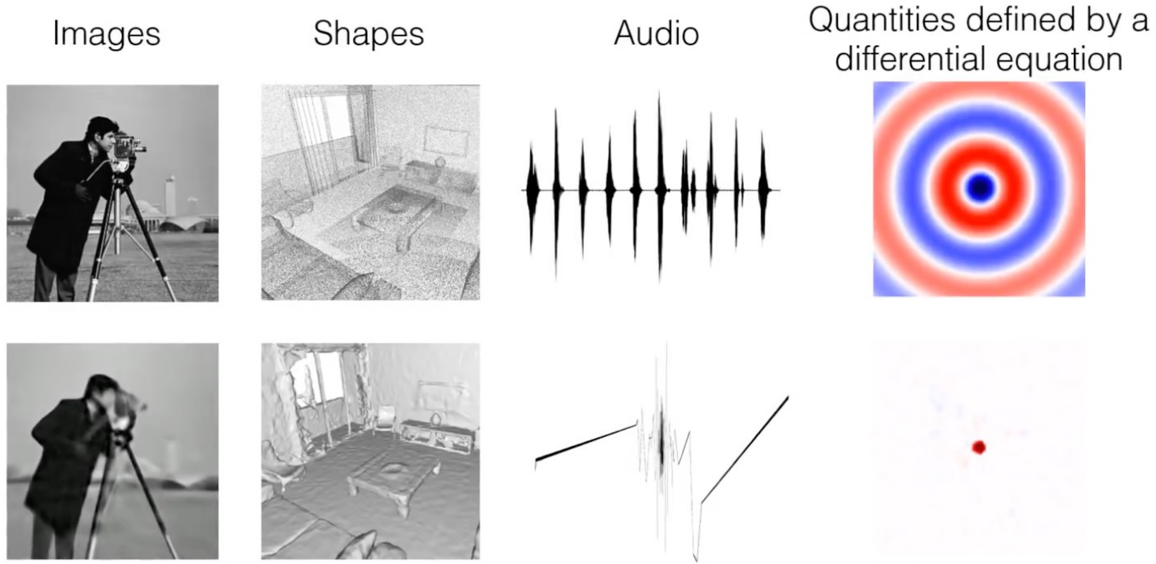
$$\mathcal{L}_{\text{grad.}} = \int_\Omega \|\nabla_{\mathbf{x}}\Phi(\mathbf{x}) - \nabla_{\mathbf{x}} f(\mathbf{x})\| \, d\mathbf{x}, \qquad \mathcal{L}_{\text{lapl.}} = \int_\Omega \|\Delta\Phi(\mathbf{x}) - \Delta f(\mathbf{x})\| \, d\mathbf{x}.$$

$$\mathcal{L}_{\text{sdf}} = \int_\Omega \big\| \, |\nabla_{\mathbf{x}}\Phi(\mathbf{x})| - 1 \big\| d\mathbf{x} + \int_{\Omega_0} \|\Phi(\mathbf{x})\| + \big(1 - \langle \nabla_{\mathbf{x}}\Phi(\mathbf{x}), \mathbf{n}(\mathbf{x}) \rangle\big) d\mathbf{x} + \int_{\Omega \setminus \Omega_0} \psi(\Phi(\mathbf{x})) d\mathbf{x},$$

What is the problem?

ReLU (one of the most common activation functions) based implicit neural representations lack the capacity to represent **fine details** in the underlying signals.





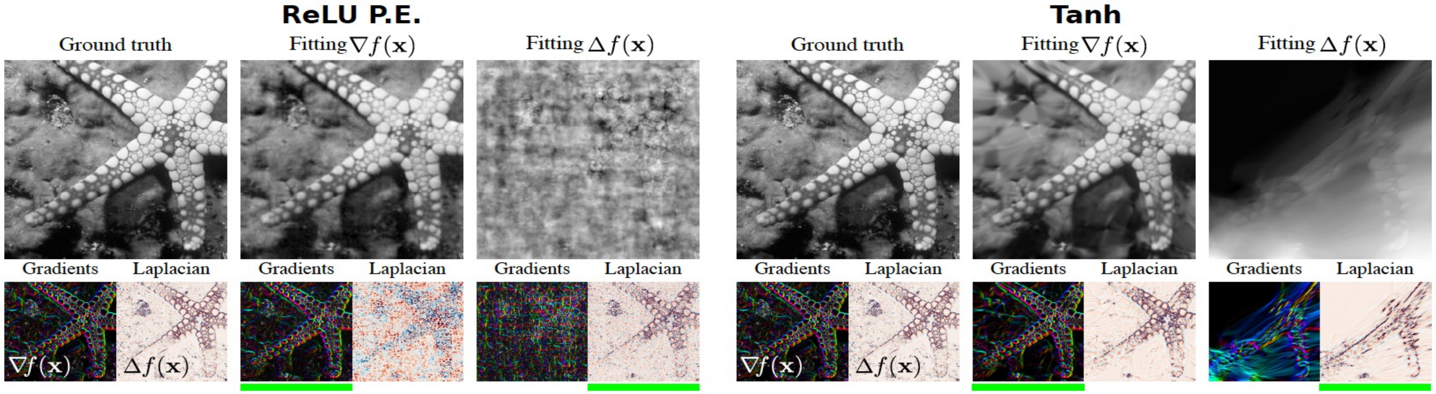
What is the problem?

ReLU-based network is incapable of representing **the derivatives** since ReLU is piecewise linear (high order derivative 0 everywhere).

Why is derivative important?

→ Solving PDE (example below: solving Poisson equation for image fitting)

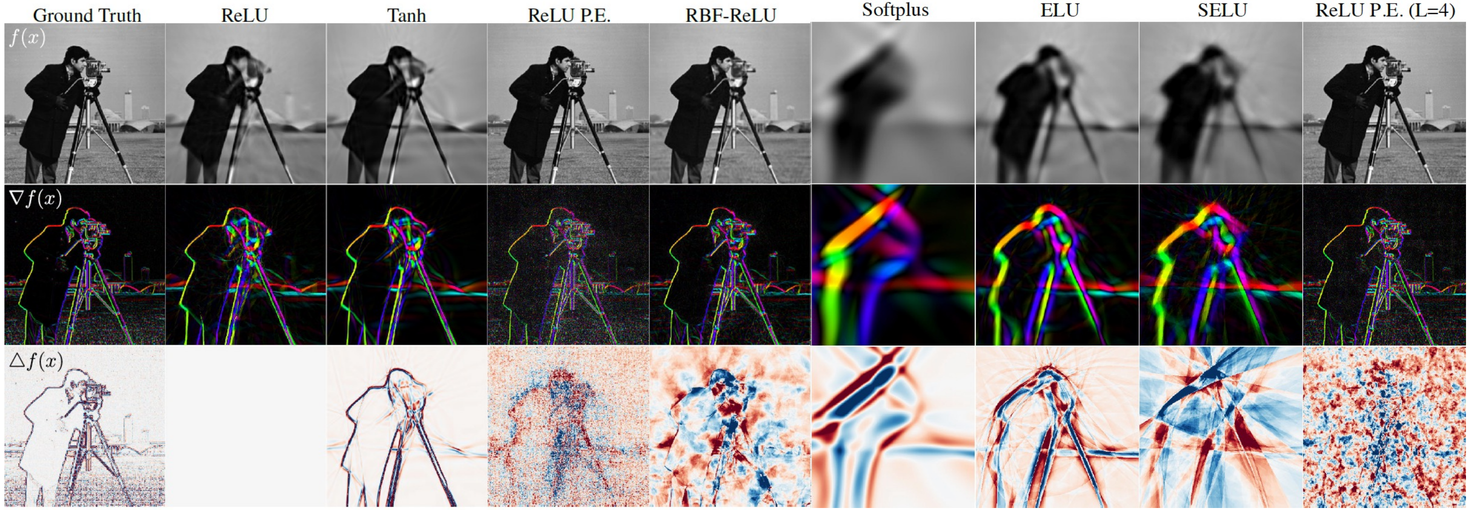
$$\mathcal{L}_{\text{grad.}} = \int_{\Omega} \|\nabla_{\mathbf{x}}\Phi(\mathbf{x}) - \nabla_{\mathbf{x}} f(\mathbf{x})\| \, d\mathbf{x}, \quad \text{or} \quad \mathcal{L}_{\text{lapl.}} = \int_{\Omega} \|\Delta\Phi(\mathbf{x}) - \Delta f(\mathbf{x})\| \, d\mathbf{x}.$$



What is the problem?

Task: Image fitting

Input	Output supervised by	Implicit Formulation Find Φ that minimizes $\mathcal{L}$
$\mathbf{x} \in \mathbb{R}^2$	$f(\mathbf{x}) \in \mathbb{R}^3$	$\mathcal{L}_{\text{img}} = \int_\Omega \lVert \Phi(\mathbf{x}) - f(\mathbf{x}) \rVert \, d\mathbf{x}$
spatial coords.	*RGB values*	



Not just ReLU, but no activation functions can preserve output / first / second order derivatives closely to the ground truth.

Solution: Use periodic activations !

$$\Phi(\mathbf{x}) = \mathbf{W}_n \left(\phi_{n-1} \circ \phi_{n-2} \circ \ldots \circ \phi_0\right)(\mathbf{x}) + \mathbf{b}_n, \quad \mathbf{x}_i \mapsto \phi_i(\mathbf{x}_i) = \sin\left(\mathbf{W}_i \mathbf{x}_i + \mathbf{b}_i\right).$$

Sine function has a unique property: **any derivative of a sine function is sine function itself** (a phase-shifted). → **SIREN** (Sinusoidal representation network).

We can supervise any derivative of SIREN with “complicated” signals, which is crucial in solving boundary value problems.

Simple, but …

We need a particular initialization that **preserves the distribution of activations through its layers.**

→ **Keep input and the output distribution for each layer to be the same.**

→ If we don't preserve the distribution, the output of each layer **diffuses.**

→ Due to the periodicity of the activation, the output mixes together.

Initialization Scheme

Nitty-gritty details …

Initialize weight uniformly from …

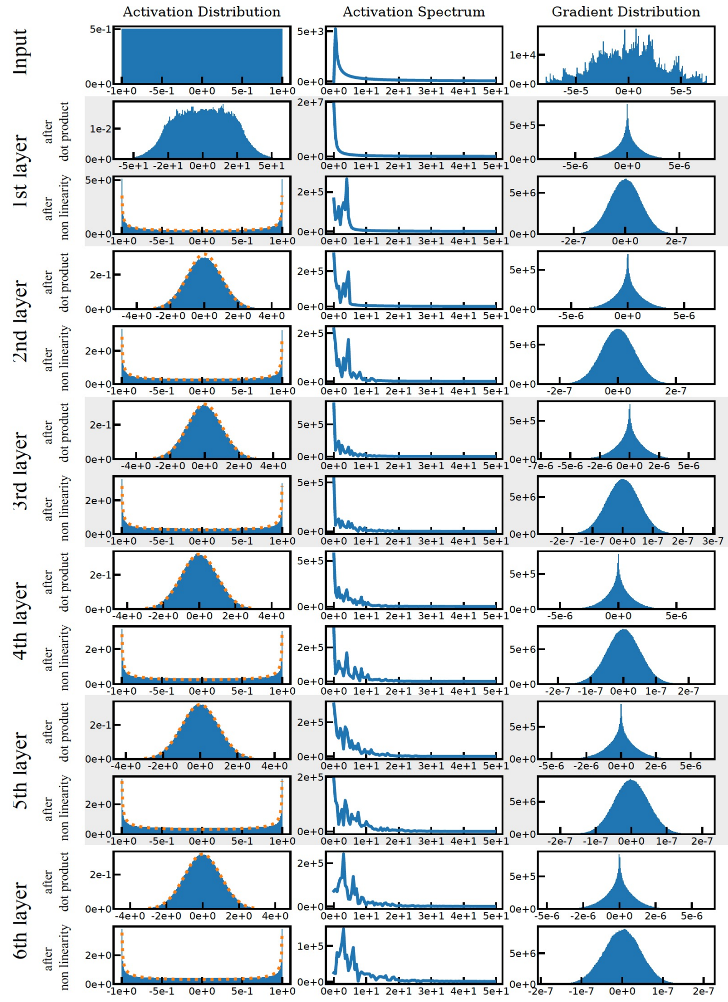
$$w_i \sim \mathcal{U}(-c/\sqrt{n}, c/\sqrt{n}), c \in \mathbb{R},$$

this lets $\mathbf{w}^T\mathbf{x} \sim \mathcal{N}(0, c^2/6)$ as n grows.

… and normal distributed input for sine function

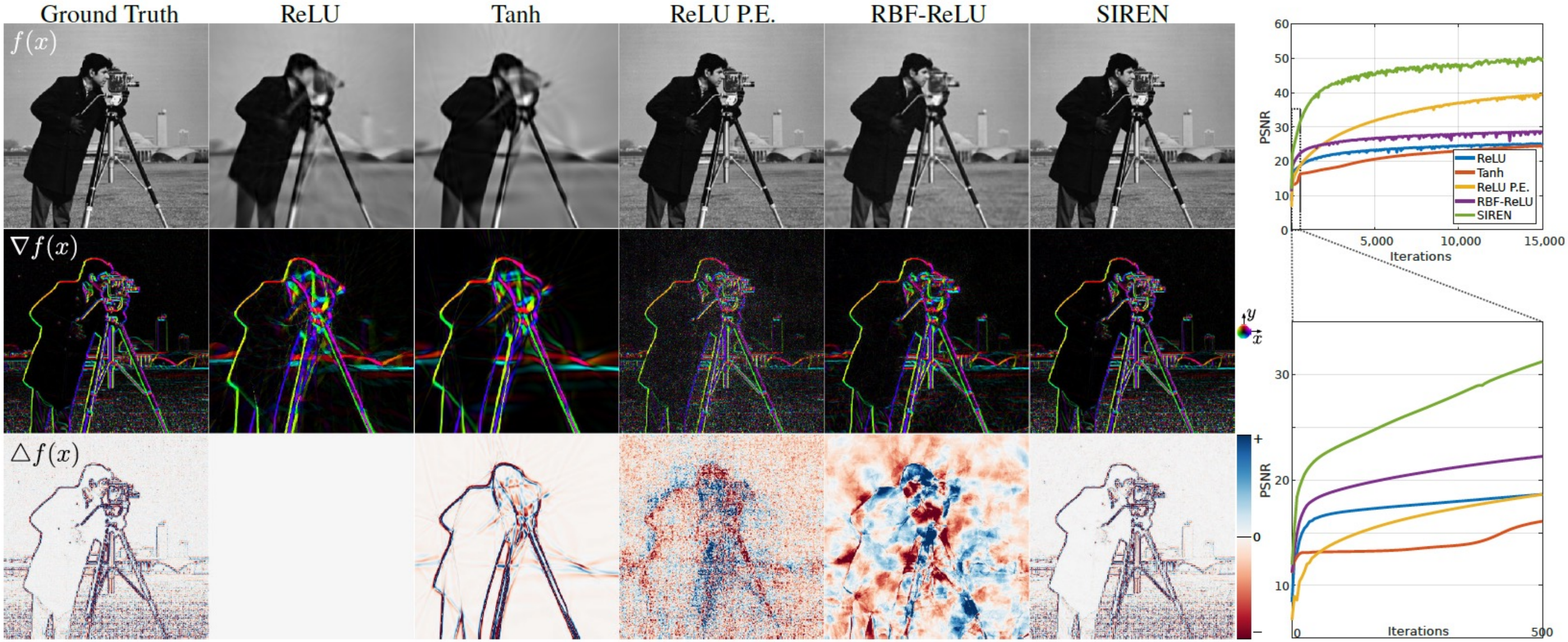
- → *Arcsine distributed for any* $c > \sqrt{6}$.
- → ***With c = sqrt(6), every layer has output before activation normally distributed with std = 1.***

(more details in supp.)



Experiment 1: Image Fitting

$$\mathcal{L}_{\text{img}} = \int_\Omega \|\Phi(\mathbf{x}) - f(\mathbf{x})\| \, d\mathbf{x}$$

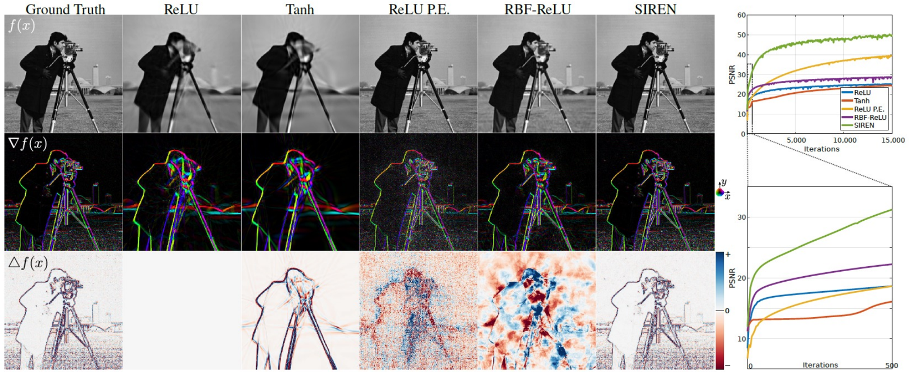


Experiment 1: Image Fitting

Setup:

Input	Output supervised by	Implicit Formulation Find Φ that minimizes $\mathcal{L}$
$\mathbf{x} \in \mathbb{R}^2$	$f(\mathbf{x}) \in \mathbb{R}^3$	$\mathcal{L}_{\text{img}} = \int_\Omega \lVert \Phi(\mathbf{x}) - f(\mathbf{x}) \rVert \, d\mathbf{x}$
spatial coords.	*RGB values*	

Input (x, y) coordinate, and output RGB / BW value for a particular image X.
e.g.) f((1, 1)) = u where u is the value of pixel 1, 1.



1. No activation function is capable of outputting all 3.
2. SIREN (sinusoidal representation network) converges a lot faster
3. For each one of output/ first/ second order derivatives, SIREN produces the closest to ground truth.

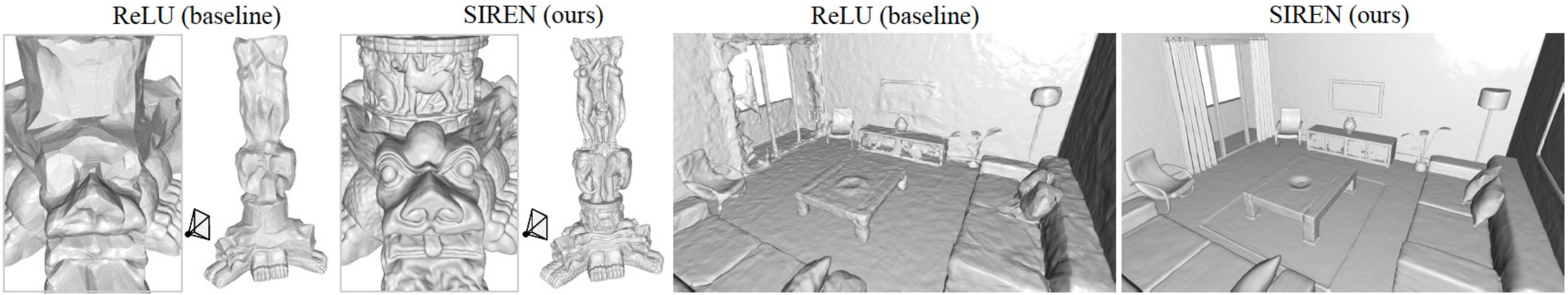
- **PSNR**: Peak Signal-to-Noise Ratio

$$MSE = \frac{1}{m\,n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2$$

$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right)$$

where I is original input, K is the network output, and MAX is the max possible value in I. (e.g., 255 in 8bits).

Experiment 2: Representing Shapes



Input	Output supervised by	Implicit Formulation Find Φ that minimizes $\mathcal{L}$
$\mathbf{x} \in \mathbb{R}^3$ *spatial coord.*	$f(\mathbf{x}) \in \mathbb{R}$ *signed distance*	$\mathcal{L}_{\text{Eikonal}} = \int_{\Omega_0} \lvert\Phi(\mathbf{x})\rvert + (1 - \langle \nabla\Phi(\mathbf{x}), \nabla f(\mathbf{x})\rangle)\, d\mathbf{x} + \int_{\Omega} \lVert \lvert\nabla\Phi(\mathbf{x})\rvert - 1 \rVert \, d\mathbf{x}$

Experiment 2: Representing Shapes

Signed distance function (SDF):

- How we model the 3D shape.
- Measures the distance from a point to a shape surface (+ inside, - outside).

Just like we train a neural network to model a metric space that represents visual/semantic similarity, we learn our neural network to model **a specific kind of metric space** where the distance is defined as **SDF**.

Why?
Think about the "**boundary**" of a set in this metric space as the **surface**. Then, if an arbitrary (x,y,z) point lies on a surface, then the distance is **minimal**. The surface has to be **smooth**, so restrict the local change. Finally, the **orientation** should match.

Definition [edit]

If *Ω* is a subset of a metric space, *X*, with metric, *d*, then the *signed distance function*, *f*, is defined by

$$f(x) = \begin{cases} d(x, \partial\Omega) & \text{if } x \in \Omega \\ -d(x, \partial\Omega) & \text{if } x \in \Omega^c \end{cases}$$

where $\partial\Omega$ denotes the boundary of Ω. For any $x \in X$,

$$d(x, \partial\Omega) := \inf_{y \in \partial\Omega} d(x, y)$$

where *inf* denotes the infimum.

Properties in Euclidean space [edit]

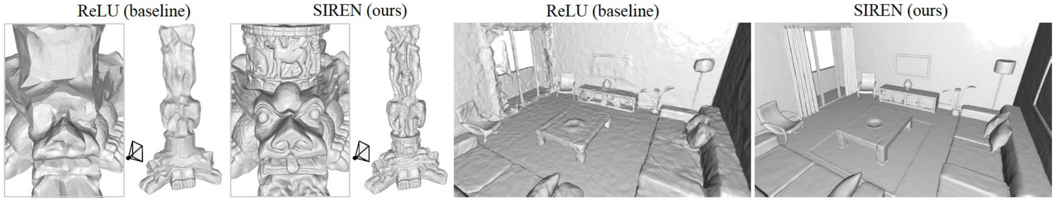
If *Ω* is a subset of the Euclidean space $\mathbf{R}^n$ with piecewise smooth boundary, then the signed distance function is differentiable almost everywhere, and its gradient satisfies the eikonal equation

$$|\nabla f| = 1.$$

If the boundary of *Ω* is C^k for $k \geq 2$ (see differentiability classes) then *d* is C^k on points sufficiently close to the boundary of *Ω*.[3] In particular, ***on*** the boundary *f* satisfies

$$\nabla f(x) = N(x),$$

Experiment 2: Representing Shapes



We represent shape with differentiable **signed distance functions** (SDFs).

Signed distance function of a set X determines the distance of a given point x from the boundary of the set X.

$$\mathcal{L}_{\mathrm{sdf}} = \int_{\Omega} \big\| |\nabla_{\mathbf{x}} \Phi(\mathbf{x})| - 1 \big\| d\mathbf{x} + \int_{\Omega_0} \|\Phi(\mathbf{x})\| + \big(1 - \langle \nabla_{\mathbf{x}} \Phi(\mathbf{x}), \mathbf{n}(\mathbf{x}) \rangle\big) d\mathbf{x} + \int_{\Omega \setminus \Omega_0} \psi\big(\Phi(\mathbf{x})\big) d\mathbf{x},$$

"Eikonal term" to make the function SDF (*smooth* surface)

Points on the shape has distance = 0 by the definition of SDF and zero-level set.

Also constrains the gradient to be equal to the *normal* of the shape (gradient at any point is perpendicular to the level set).

Constrains off-surface points.

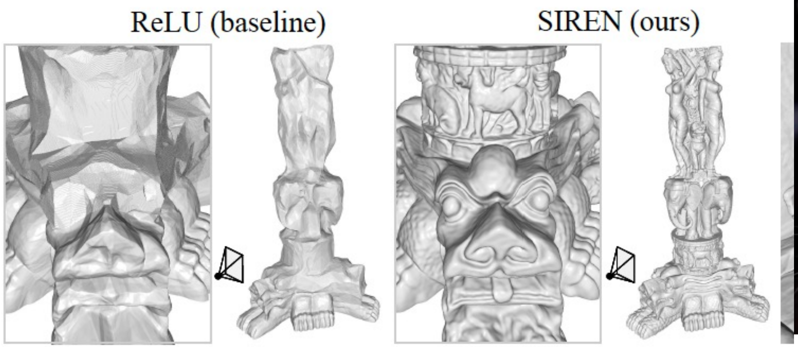
* $\psi(\mathbf{x}) = \exp(-\alpha \cdot |\Phi(\mathbf{x})|),$

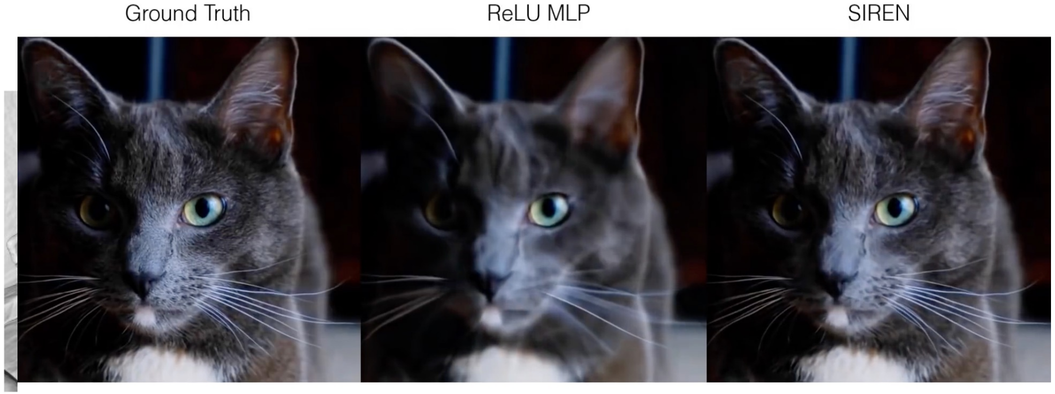
- Ω_0 Zero-level set.

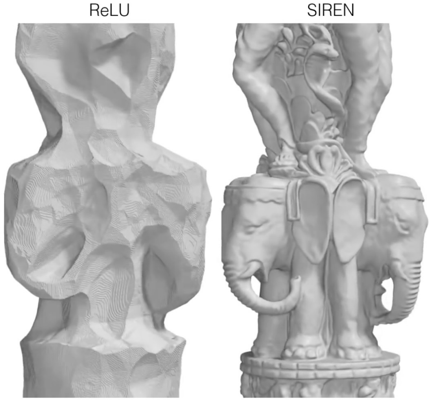
$\Gamma = \{(x, y) \mid \varphi(x, y) = 0\},$

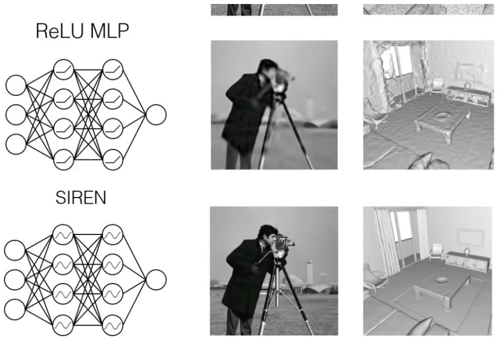
… where phi is SDF.

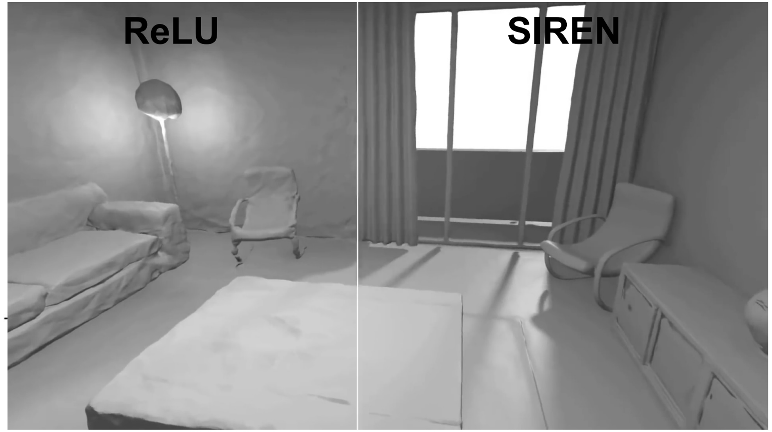
TL;DR: Accurate Representation for Reconstruction











Discussions & Critiques

1. (+) Simple but effective that can impact a variety of different scientific fields (more than what I showed in this ppt).
2. (+) Well-analyzed experiments and justifications with abundant qualitative evaluations.

3. (-) Lack of quantitative evaluations, especially for the shape estimation.
4. (-) Despite the extensive experiments for validation, the fact that the model will be inherently dependent on initialization (e.g., condition dependent on input dimension n) may limit its practicality for the future research.

Future Work & Extended Readings

<u>Related concurrent works:</u>

Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains

(https://arxiv.org/abs/2006.10739)

→ TL;DR: ”The” Fourier Mapping is $\gamma(\mathbf{v}) = [\cos(2\pi\mathbf{B}\mathbf{v}), \sin(2\pi\mathbf{B}\mathbf{v})]^{\mathrm{T}}$ where B is Gaussian sampled with tuned std.

→ Inspired by positional encoding (“ReLU + PE” in this paper for an example).

→ Only requires this transformation for the input.

Both works are for learning better neural implicit representations.

NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

(https://arxiv.org/abs/2003.08934)

→ A really cool image-based novel view synthesis paper using neural implicit representation.

→ SIREN could be used together here.

Summary

1. Implicit neural representations with periodic activation functions.

2. Initialization scheme for training these representations.

3. Applications to multiple problems.

Question?

Thank you !