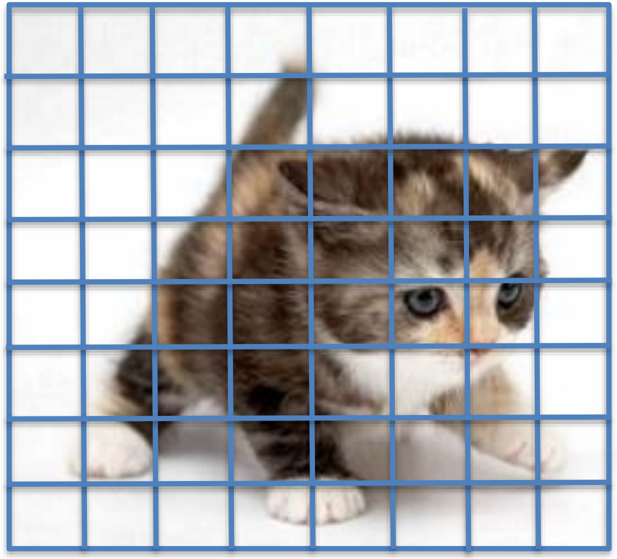
# Neural Fields

# Images

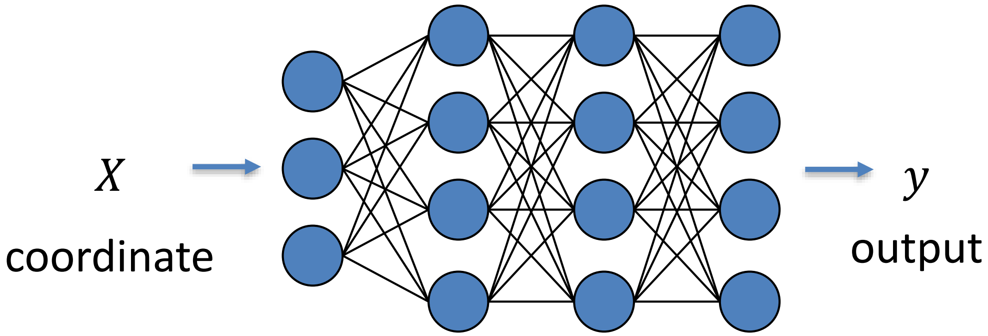
- Array of pixels
- x,y coordinates
- Maps to RGB value

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



# MLPs

- MLP(x) - > y
- X = coordinates (e.g., pixel or 3d coordinates)



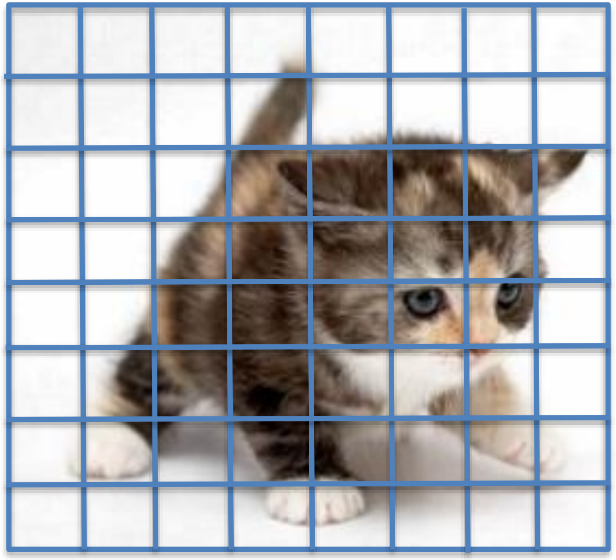
# MLP <-> Image

- MLP can fit to image

$$\text{Image}(x, y) \rightarrow \text{RGB}$$

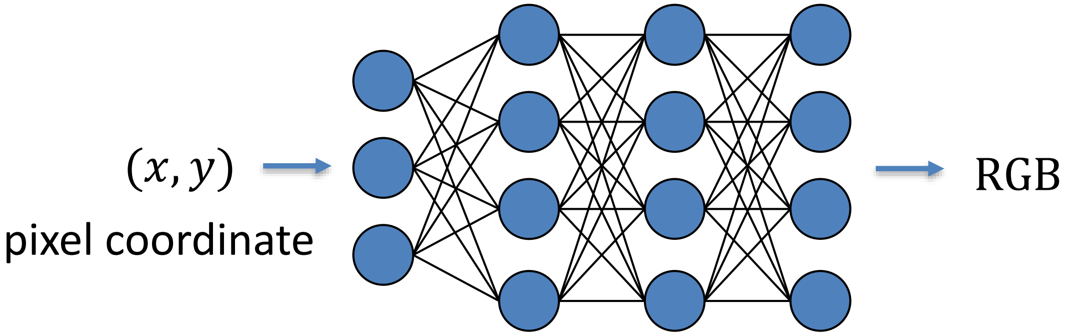
$$\text{MLP}(x, y) \rightarrow \text{RGB}$$

$$\theta^* = \underset{\forall i,j}{\text{argmin}} \, ||\text{MLP}_\theta(x_i, y_i) - \text{Image}(x, y)||$$



# MLP as Datastructure

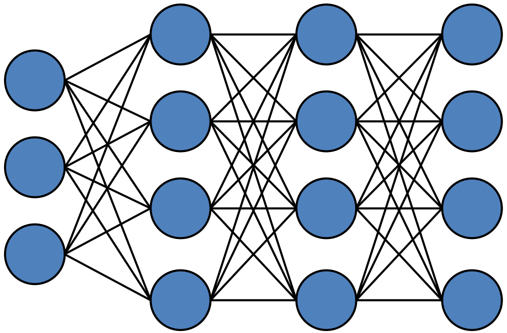
- Why do that?




- What if x,y coordinates are fractional?
- Smoothness / interpolation properties of MLP!

# MLP as Datastructure

- Works in arbitrary dimensions

$(x, y, \ldots)$ →  → $(r, g, b, \ldots)$

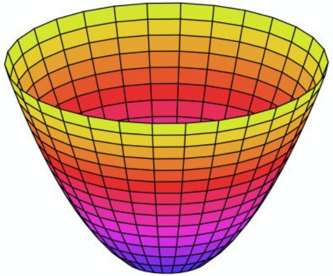
coordinate output

- Sparse encoding of signal!
- Shifts capacity where it needs it (based on optimization)

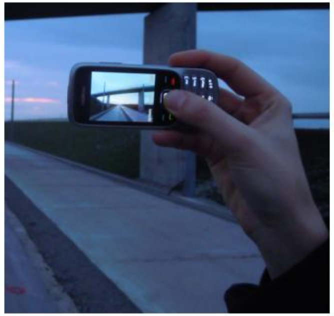
# Neural Fields

# Example of fields

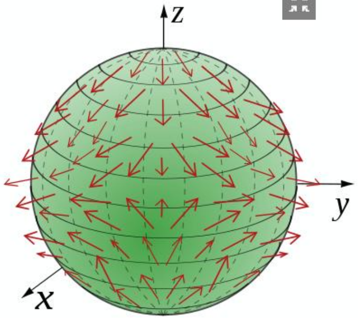
3D Signed Distance Fields
(Implicit Surface)



3D Parabola
(Explicit Surface)

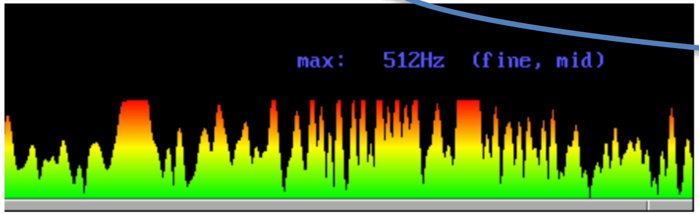


Image



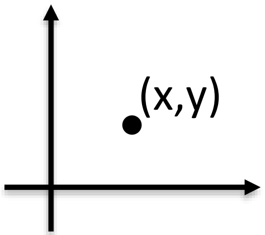

Vector Field

**Fields**

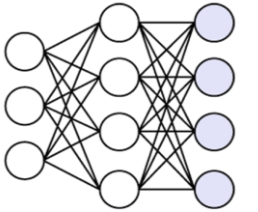



Audio

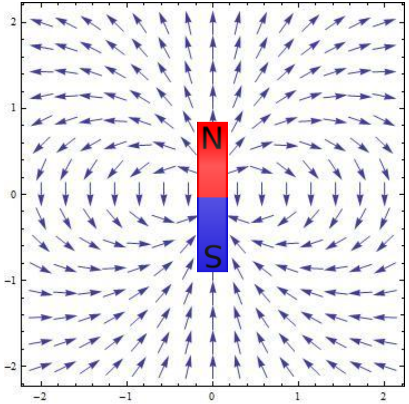
# What are neural fields?



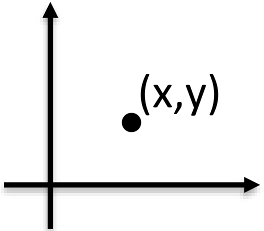

$$\Phi: \mathbb{R}^2 \rightarrow \mathbb{R}^2$$



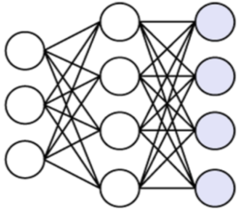
Neural Network ($\Phi$)



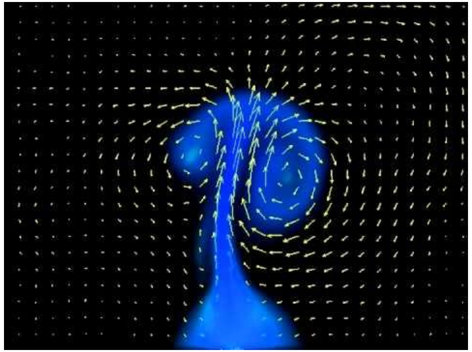

Magnetic Field




$$\Phi: \mathbb{R}^2 \rightarrow \mathbb{R}^2$$



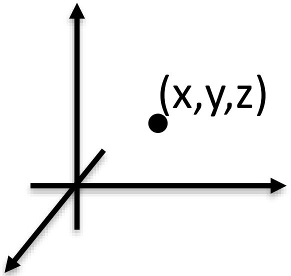
Neural Network ($\Phi$)



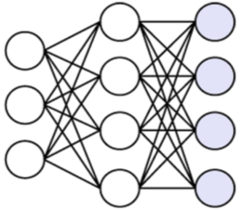
Eulerian Flow Field of a Fluid

[Koldora CC]

# What are neural fields?

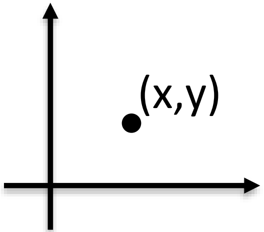



$\Phi: \mathbb{R}^n \rightarrow \mathbb{R}$

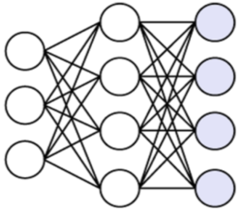


Neural Network
$(\Phi)$

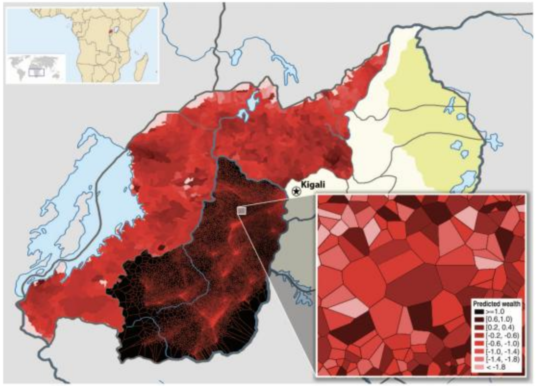
Signed Distance Function (SDF)




$\Phi: \mathbb{R}^2 \rightarrow \mathbb{R}^n$



Neural Network
$(\Phi)$




Geospatial Data
[Blumenstock et al. 2015]

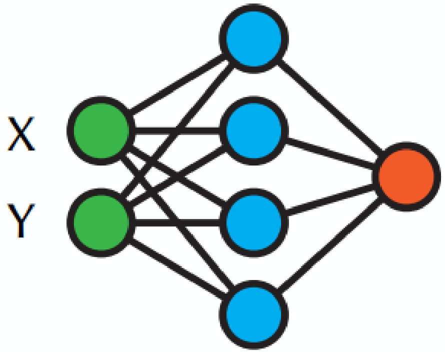
# Definitions

Definition 1: A field is a quantity defined for all spatial and / or temporal coordinates.

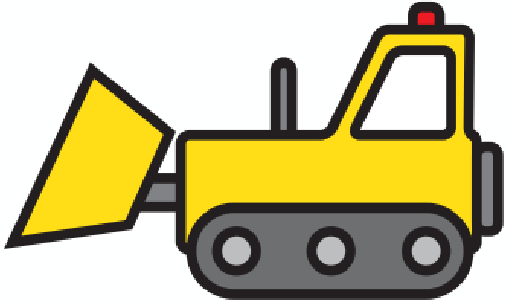
Definition 2: A *neural field* is a field that is parameterized fully or in part by a neural network.

# Related Terminology & Misnomers


Implicit Neural Representations




Coordinate-based Neural Networks



NeRFs

= Neural **Radiance** Fields

Neural **Radiance** Field is a type of neural field (see a detailed NeRF lecture in the next course!)

# Implicit vs Explicit

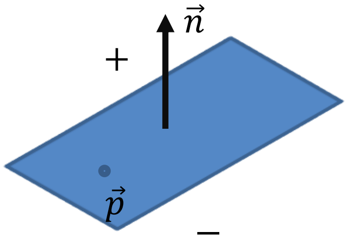
- Remember, mathematically:

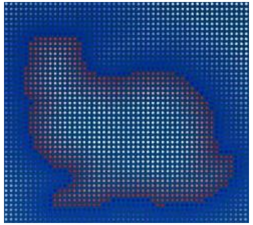
  - Explicit function: $f(x) = y$
  - Implicit function: $x^2 + y^2 - 1 = 0$

# Implicit (Surfaces)

- Implicit form: $$f(x, y, z)\colon \mathbb{R}^3 \to \mathbb{R}$$
- Surface is defined by the level set of the tri-variate scalar function $$f(x, y, z) = c$$
- Example: Hesse normal form

$$f(x, y, z) = \left( \begin{pmatrix} x \\ y \\ z \end{pmatrix} - \vec{p} \right) \cdot \vec{n} = 0$$



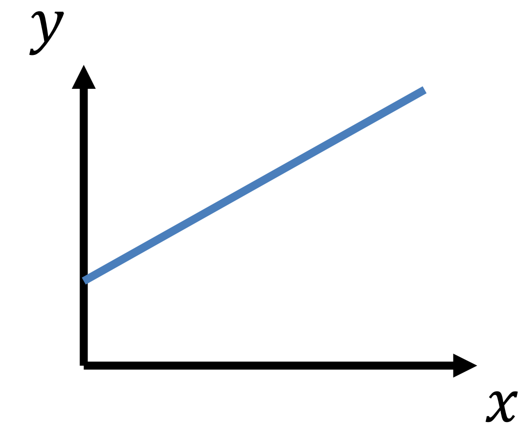



$f$ is also called **Signed Distance Function** (SDF)

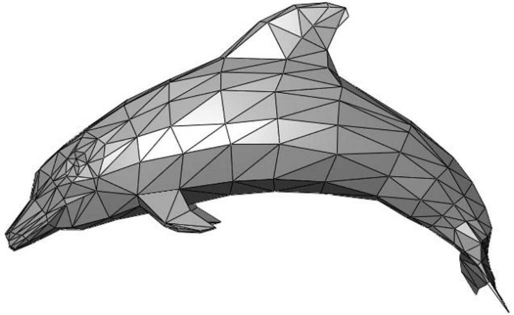
# Explicit (Surfaces)

- Explicit form:

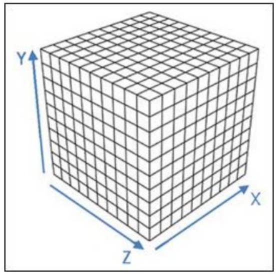
$$f(x) = y = m \cdot x + c$$

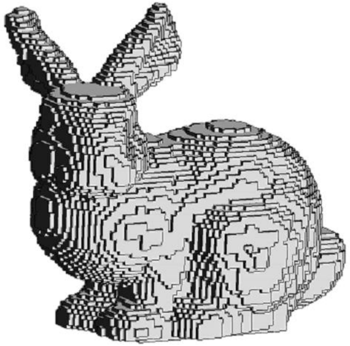



Polygonal Meshes

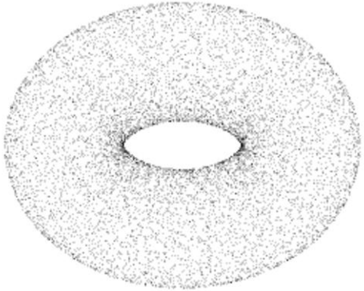


Voxels



Point Clouds



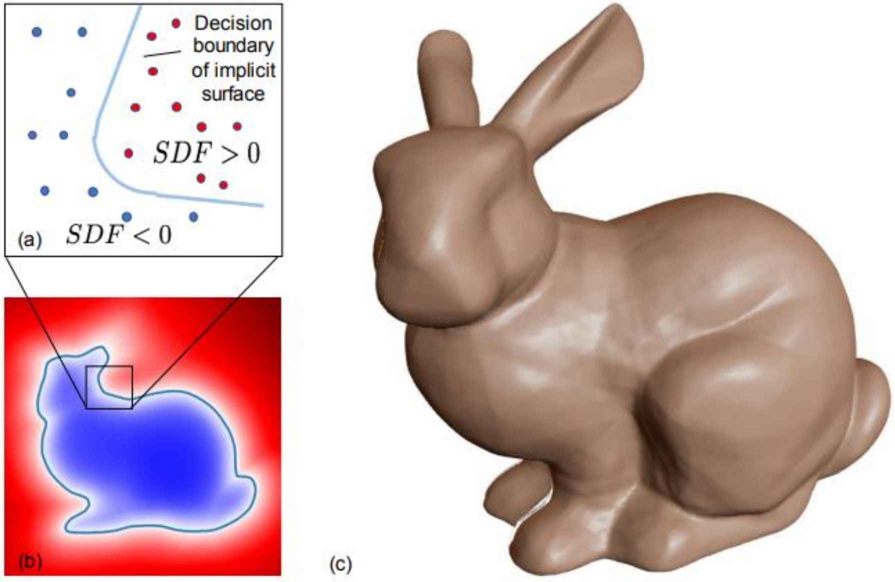
# Signed Distance Fields

**Signed Distance Fields (SDFs)**

SDF(***x***) = D

**x** denotes any point sampled in 3D space.
D is distance value from ***x*** to the surface.

D > 0 indicates ***x*** is outside of the shape
D < 0 indicates ***x*** is inside of the shape
D = 0 indicates the zero-level set (i.e., **x** is located on the surface.)



# Signed Distance Fields vs Occupancy

**Truncated Signed Distance Fields (TSDFs)**

$$TSDF(\mathbf{x}) = max(-1, min(1, \frac{SDF(\mathbf{x})}{t}))$$

$$\text{TSDF}(\boldsymbol{x}) = \begin{cases} \text{SDF(x)} / t & -t < \text{D} < t \\ 1 & \text{D} > t \\ -1 & \text{D} < -t \end{cases}$$

**Occupancy Fields**

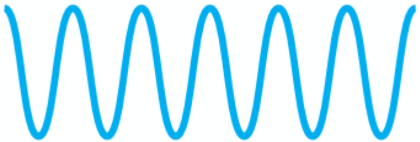
$$Occ(x) = \mathbf{1}[SDF(x) \leq 0]$$

$$\text{Occ}(\boldsymbol{x}) = \begin{cases} 1 & \text{SDF(x)} <= 0 \\ 0 & \text{SDF(x)} > 0 \end{cases}$$

# Why Neural Fields

There are many types of signals in natural world.

**Natural Signals:**

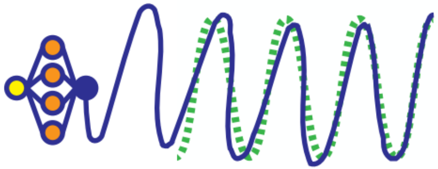


Continuous

**Sampled Signals:**

Discrete

**Neural Fields:**



Neural

Use continuous parametric functions to approximate natural signals

# Overfitting vs Generalization

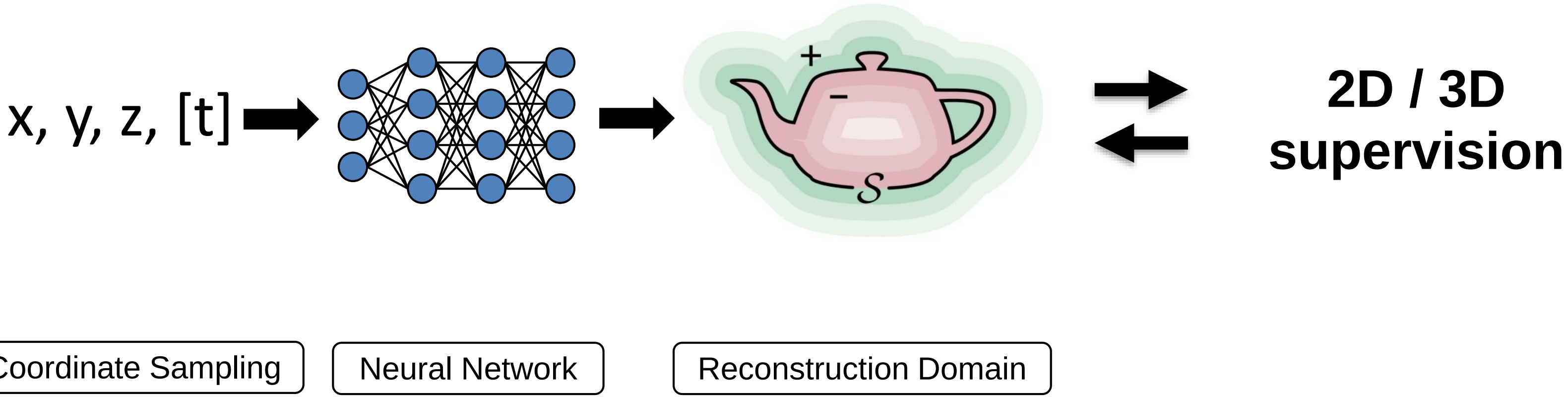
# Overfitting vs Generalization

- Overfitting as a goal
- Overfitting as a debugging tool
- Overfitting as an artifact

# Overfitting

## Overfitting a single scene

What we want to reconstruct:

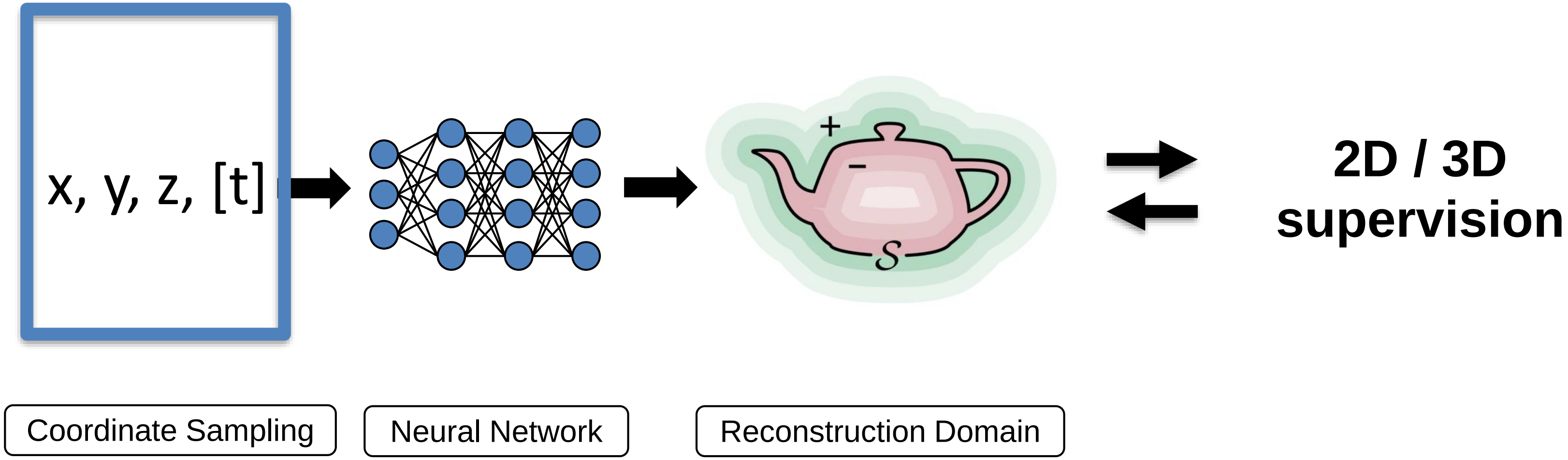


# Overfitting

## Overfitting a single scene

What we want to reconstruct:

**Without any data prior**



# Overfitting

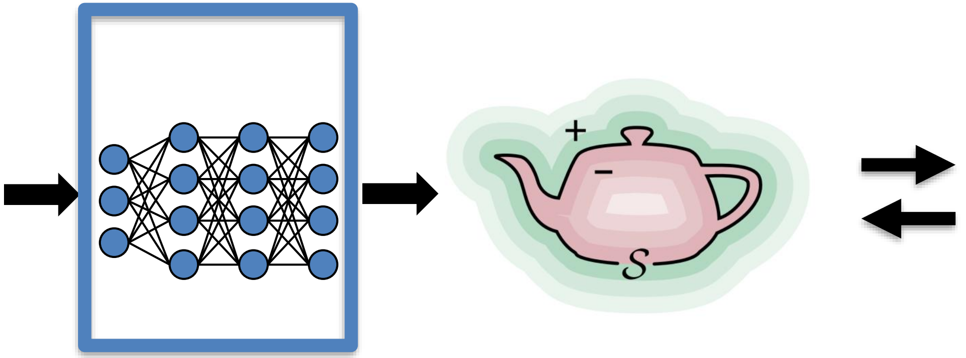
## Overfitting a single scene

**The learned network is not adaptive to other scenes.**

What we want to reconstruct:

**Overfitting!**

x, y, z, [t]




**2D / 3D supervision**

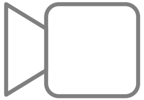
Coordinate Sampling

Neural Network

Reconstruction Domain

# Example: Scene Representation Networks

Overfitting a single scene from multi-view images

**Input:** $\left\{ \right.$  +  $\left. , \ldots \right\}$

$\mathcal{I}_i$ Image

$\xi_i$ Camera parameter

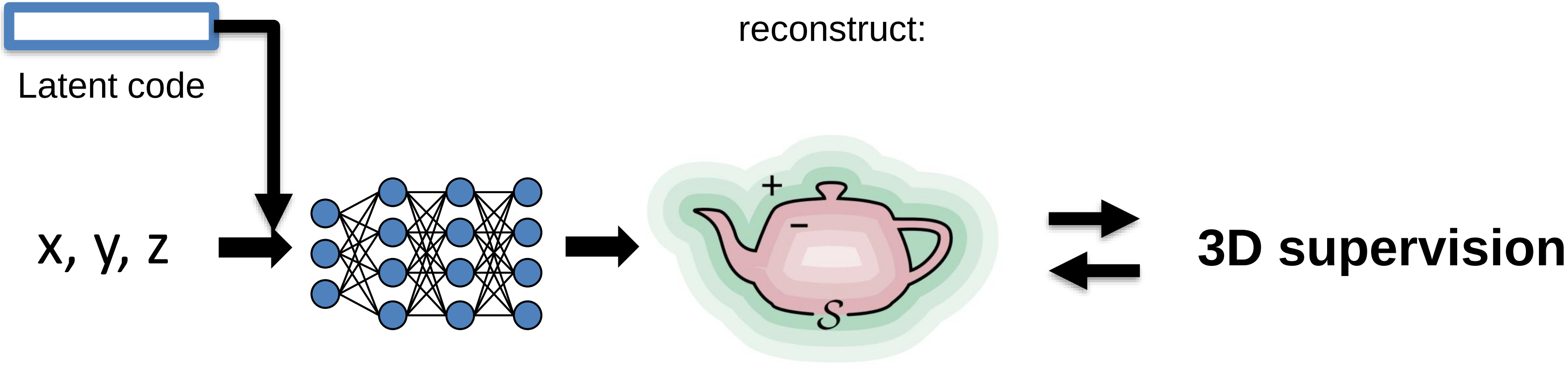
[Sitzmann et al, 2020]

# Fitting more scenes

DeepSDF: Each scene should have a unique description

Unique description

What we want to reconstruct:




Coordinate Sampling

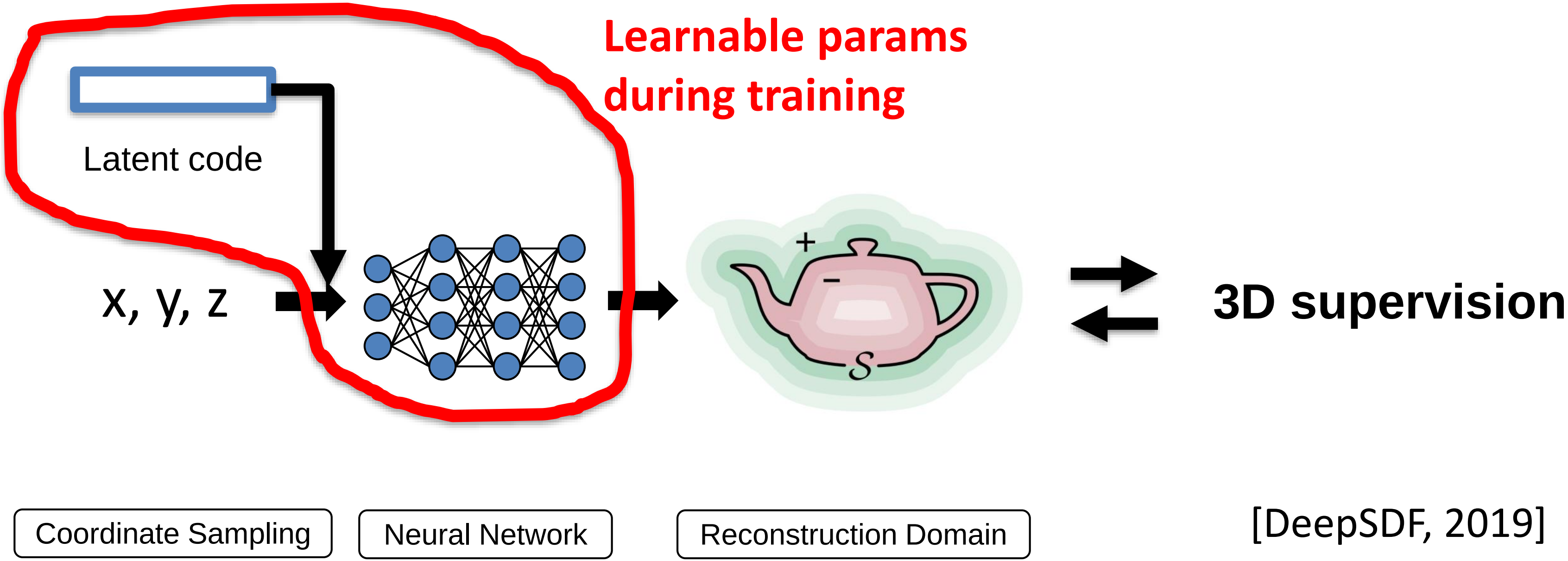
Neural Network

Reconstruction Domain

[DeepSDF, 2019]

# Fitting more scenes

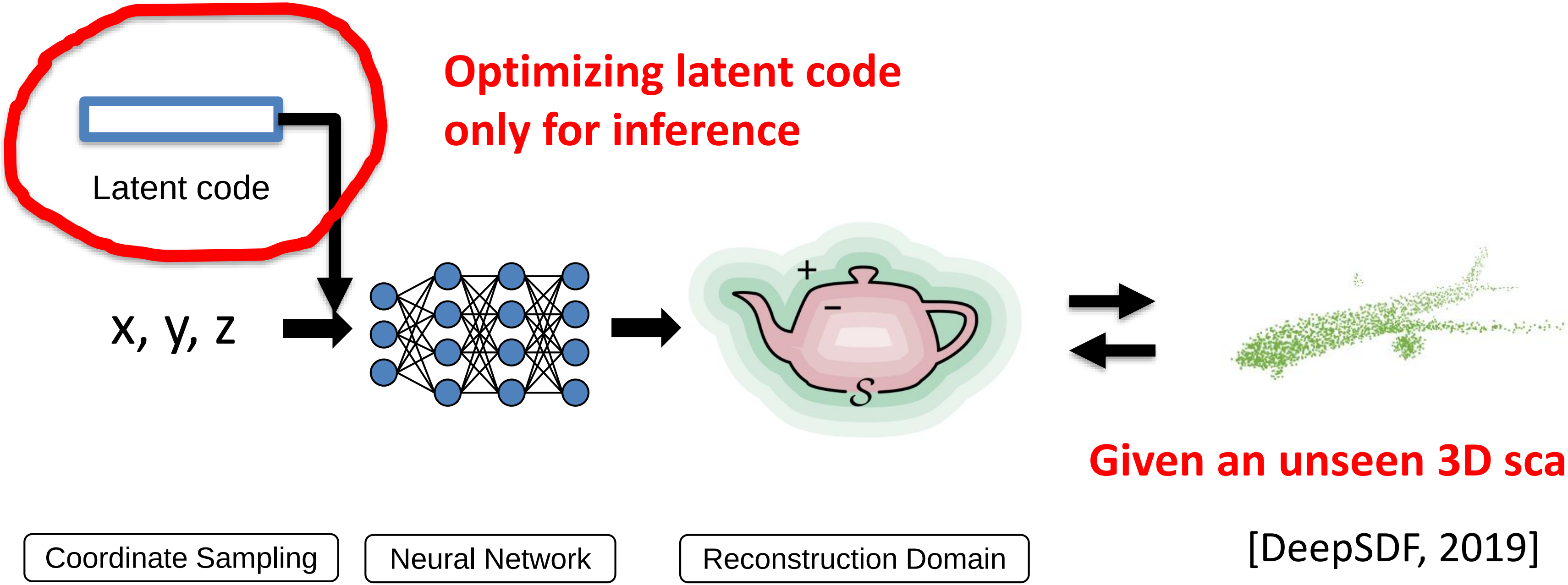
DeepSDF: Each scene should have a unique description




[DeepSDF, 2019]

# Fitting more scenes

## DeepSDF: Each scene should have a unique description



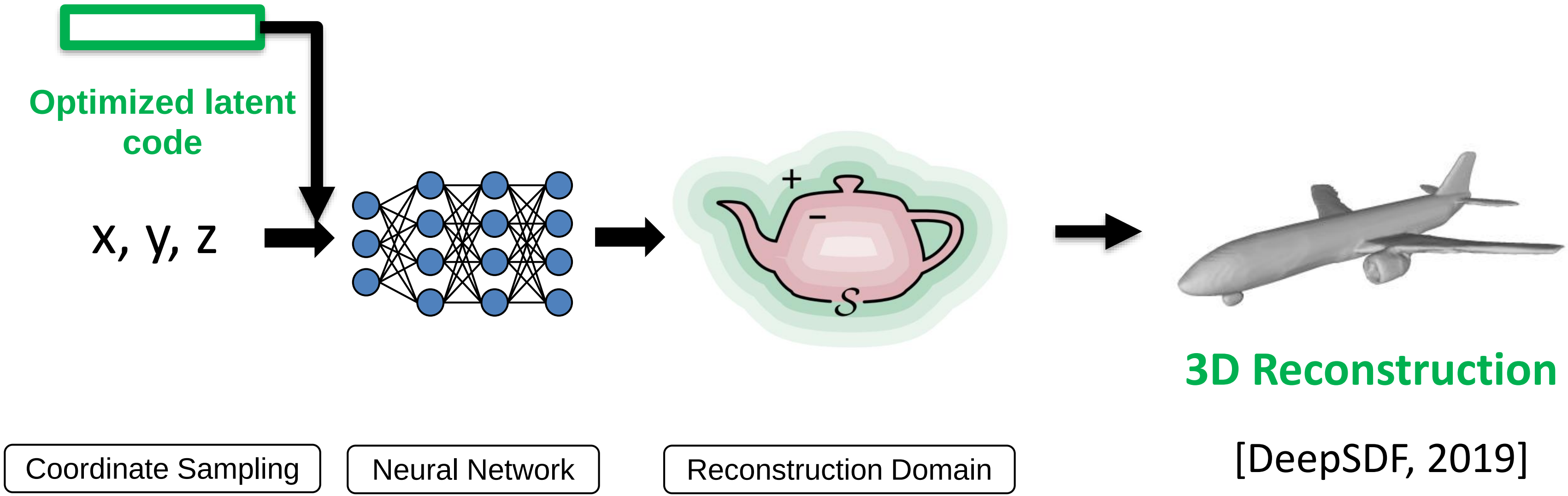

**Optimizing latent code only for inference**

**Given an unseen 3D scan**

[DeepSDF, 2019]

# Fitting more scenes

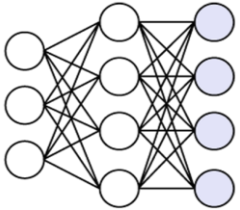
DeepSDF: Each scene should have a unique description




[DeepSDF, 2019]

# How to generalize?

What is generalization?



Neural Network
(Φ)

Unseen input observation
(e.g., image, 3D scan)

Reconstruction

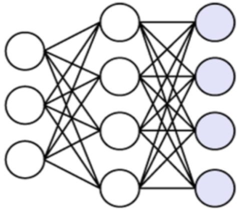
[Lars Mescheder et al, 2019]

# How to generalize?

Able to infer from unseen input

Unseen input observation
(e.g., image, 3D scan)

[Lars Mescheder et al, 2019]



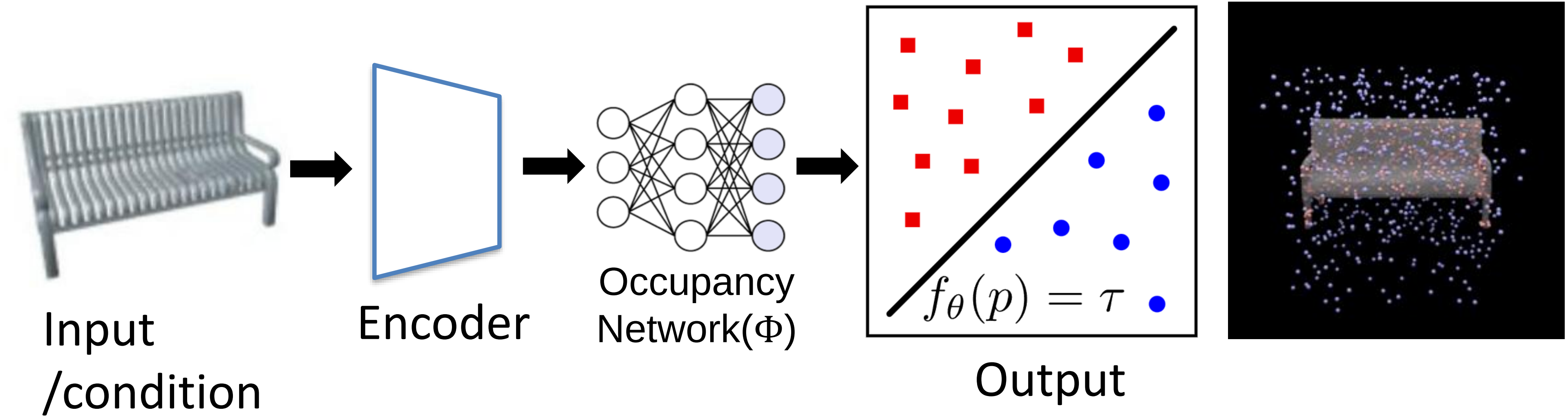
Neural Network
($\Phi$)

Reconstruction

Capable to learn prior from dataset !

# Occupancy Network

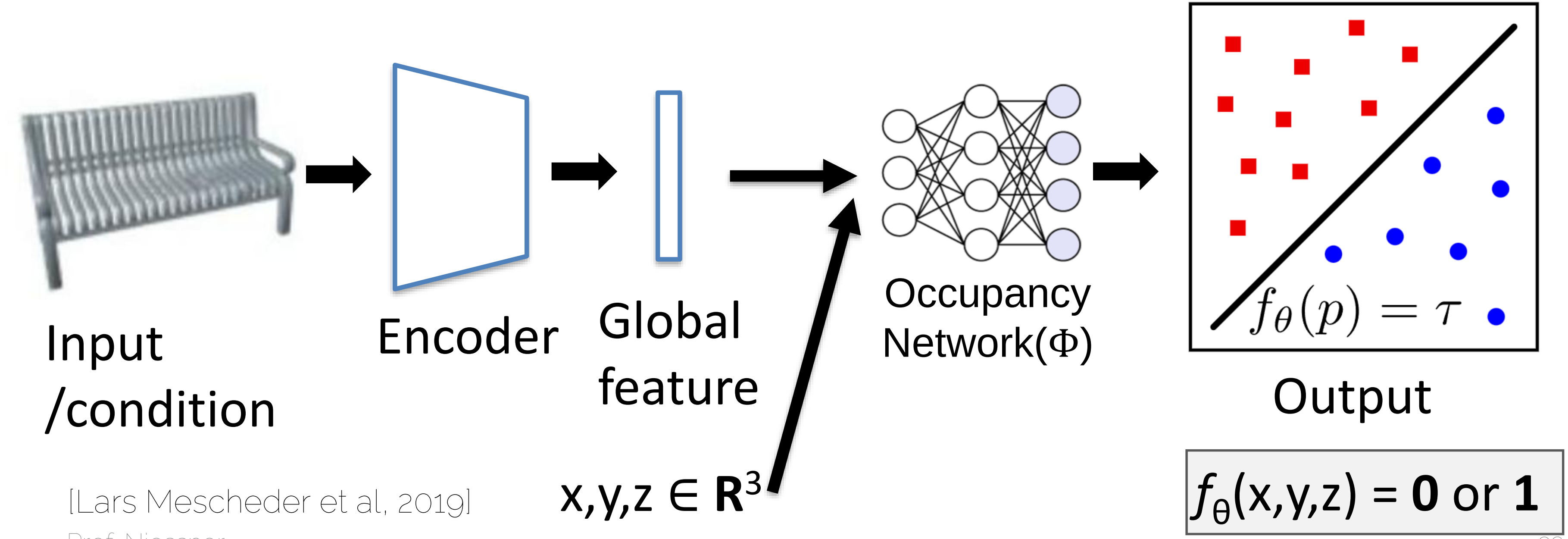
Target: Learning the occupancy field of a shape conditioned on different observations (e.g., images, point clouds)




[Lars Mescheder et al, 2019]

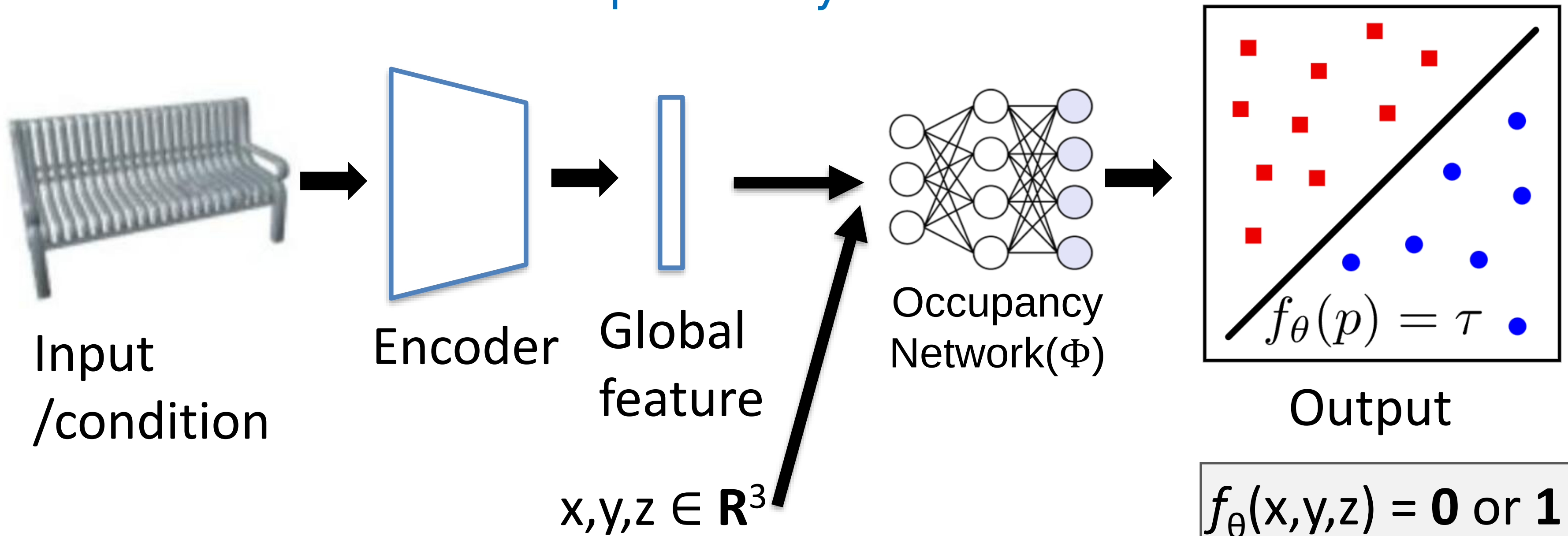
# Occupancy Network

Target: Learning the occupancy field of a shape conditioned on different observations (e.g., images, point clouds)




[Lars Mescheder et al, 2019]

# Occupancy Network



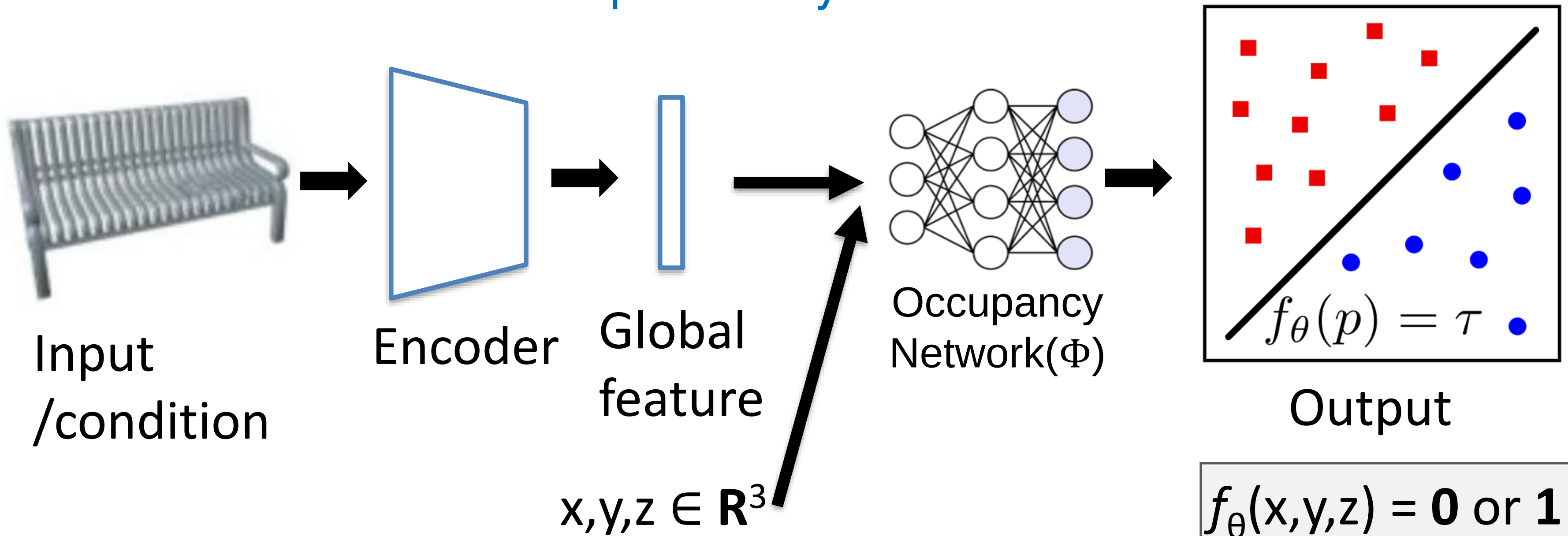

$f_\theta(x,y,z)$ = **0** or **1**

When *input = image, Encoder = ResNet*

When *input = point cloud, Encoder = PointNet*

[Lars Mescheder et al, 2019]

# Occupancy Network




$f_\theta(x,y,z)$ = **0** or **1**

Occupancy Network: a fully-connected neural network with **5 ResNet blocks** and condition it on the input using **conditional batch normalization**.

[Lars Mescheder et al, 2019]

# Encodings and Activations

# Input Encoding

## Why input encoding (or positional encoding)?

On the Spectral Bias of Neural Networks

Nasim Rahaman [*1 2] Aristide Baratin [*1] Devansh Arpit [1] Felix Draxler [2] Min Lin [1] Fred A. Hamprecht [2] Yoshua Bengio [1] Aaron Courville [1]

Abstract

Neural networks are known to be a class of highly expressive functions able to fit even random input-output mappings with 100% accuracy. In this work we present properties of neural networks that complement this aspect of expressivity. By using tools from Fourier analysis, we highlight a learning bias of deep networks towards low frequency functions – i.e. functions that vary globally without local fluctuations – which manifests itself as a frequency-dependent learning speed. Intuitively, this property is in line with the observation that over-parameterized networks prioritize learning simple patterns that generalize across data samples. We also investigate the role of the shape of the data manifold by presenting empirical and theoretical evidence that, somewhat counter-intuitively, learning higher frequencies gets *easier* with increasing manifold complexity.

1. Introduction

The remarkable success of deep neural networks at generalizing to natural data is at odds with the traditional notions of model complexity and their empirically demonstrated ability to fit arbitrary random data to perfect accuracy (Zhang et al., 2017a; Arpit et al., 2017). This has prompted recent investigations of possible implicit regularization mechanisms inherent in the learning process which induce a bias towards low complexity solutions (Neyshabur et al., 2014; Soudry et al., 2017; Poggio et al., 2018; Neyshabur et al., 2017).

In this work, we take a slightly shifted view on implicit regularization by suggesting that low-complexity functions are *learned faster* during training by gradient descent. We expose this bias by taking a closer look at neural networks through the lens of Fourier analysis. While they can approximate arbitrary functions, we find that these networks prioritize learning the low frequency modes, a phenomenon we call the *spectral bias*. This bias manifests itself not just in the process of learning, but also in the parameterization of the model itself: in fact, we show that the lower frequency components of trained networks are more robust to random parameter perturbations. Finally, we also expose and analyze the rather intricate interplay between the spectral bias and the geometry of the data manifold by showing that high frequencies get easier to learn when the data lies on a lower-dimensional manifold of complex shape embedded in the input space of the model. We focus the discussion on networks with rectified linear unit (ReLU) activations, whose continuous piece-wise linear structure enables an analytic treatment.

Contributions[1]

1. We exploit the continuous piecewise-linear structure of ReLU networks to evaluate its Fourier spectrum (Section 2).
2. We find empirical evidence of a *spectral bias*: i.e. lower frequencies are learned first. We also show that lower frequencies are more robust to random perturbations of the network parameters (Section 3).
3. We study the role of the shape of the data manifold: we show how complex manifold shapes can facilitate the learning of higher frequencies and develop a theoretical understanding of this behavior (Section 4).

2. Fourier analysis of ReLU networks

2.1. Preliminaries

Throughout the paper we call 'ReLU network' a scalar function $f : \mathbb{R}^d \mapsto \mathbb{R}$ defined by a neural network with $L$ hidden layers of widths $d_1, \cdots d_L$ and a single output neuron:

$$f(\mathbf{x}) = \left(T^{(L+1)} \circ \sigma \circ T^{(L)} \circ \cdots \circ \sigma \circ T^{(1)}\right)(\mathbf{x}) \quad (1)$$

*Equal contribution [1]Mila, Quebec, Canada [2]Image Analysis and Learning Lab, Ruprecht-Karls-Universität Heidelberg, Germany. Correspondence to: Nasim Rahaman <nasim.rahaman@live.com>, Aristide Baratin <aristide.baratin@umontreal.ca>, Devansh Arpit <devansharpit@gmail.com>.

*Proceedings of the 36th International Conference on Machine Learning*, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

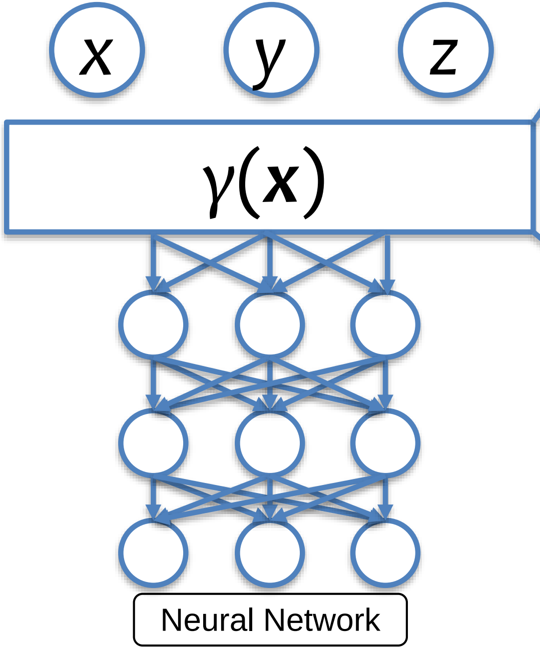
[1]Code: https://github.com/nasimrahaman/SpectralBias

***Neural networks are biased to fit lower frequency signals for generalization (mising high dimensional details)***

-> Involve input encoding to alleviate this issue by lifting coordinates into higher dimensional features.

# Input Encoding

## Positional Encodings




$$\gamma(\mathbf{x}) = [\gamma_1(\mathbf{x}), \gamma_2(\mathbf{x}), \ldots, \gamma_m(\mathbf{x})]$$

$$\gamma_{(2i)}(x) = sin(2^{i-1}\pi x),$$

$$\gamma_{(2i+1)}(x) = cos(2^{i-1}\pi x)$$

Spatial coordinates are embedded to higher dimension with **sinusoidal functions**.

[Vaswani et al. 2017]

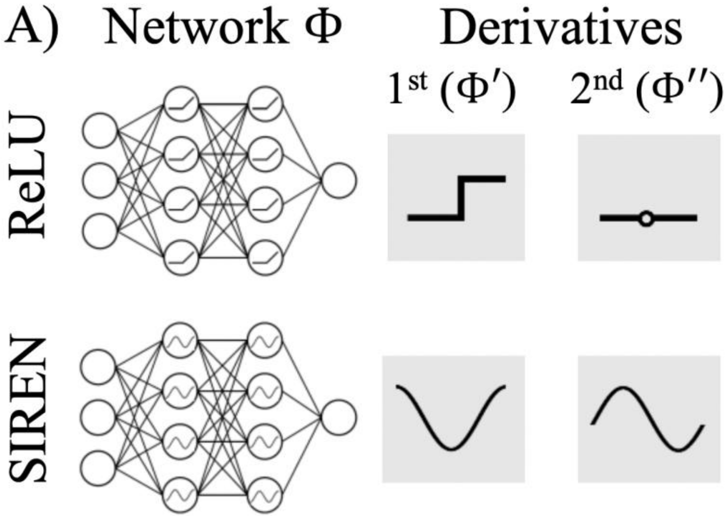
# Input Encoding

## More Positional Encodings:

1. Random Fourier Encodings [Tancik et al. 2020]
2. One-blob Encodings [Müller et al. 2020]
3. Super Gaussian Encodings [Ramasinghe et al. 2021]

# Activation Functions

## SIREN vs ReLU




SIREN uses sinusoidal activation functions to fit high-frequency signals.

[Sitzmann et al. 2021]

# Activation Functions

## More activation functions

[Ramasinghe et al. 2021]

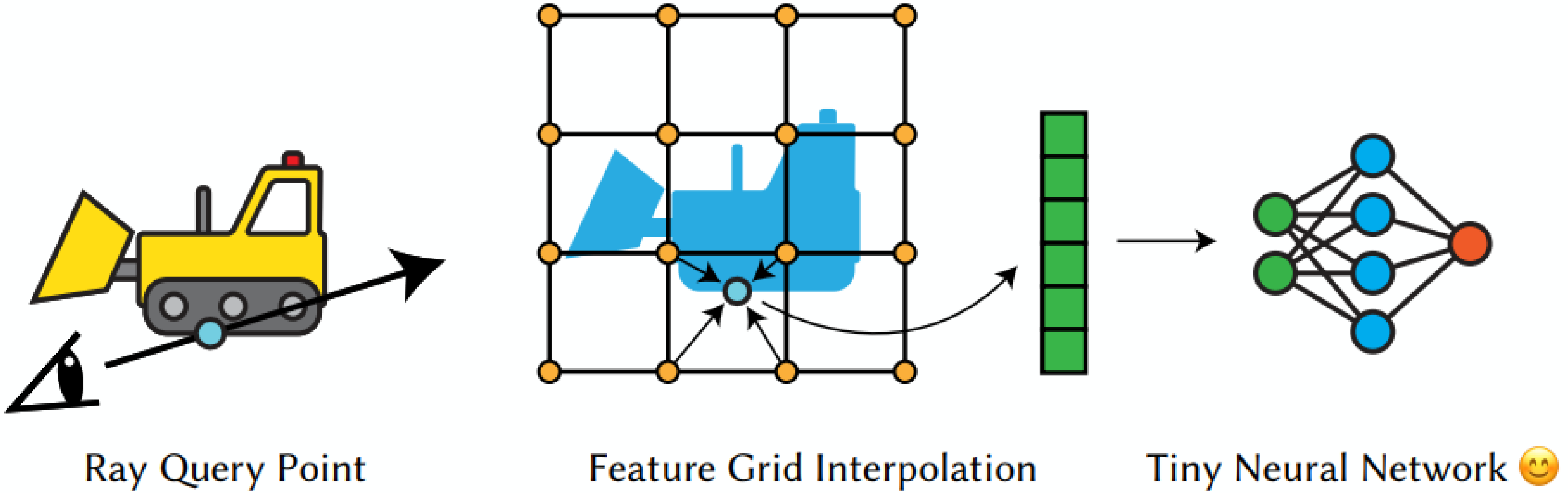
| Activation ($\psi$) | Equation | parameterized | $\psi'$ | $\psi''$ | R1 | R2 |
|---|---|---|---|---|---|---|
| ReLU | $\max(0,x)$ | ✗ | $\begin{cases}1, & \text{if } x>0\\ 0, & \text{otherwise}\end{cases}$ | $0$ | ✗ | ✗ |
| PReLU | $\begin{cases}x, & \text{if } x>0\\ ax, & \text{otherwise}\end{cases}$ | ✓ | $\begin{cases}1, & \text{if } x>0\\ a, & \text{otherwise}\end{cases}$ | $0$ | ✓ | ✗ |
| Sin | $\sin(ax)$ | ✓ | $a\cos(ax)$ | $-a^2\sin(ax)$ | ✓ | ✓ |
| Tanh | $\frac{e^x-e^{-x}}{e^x+e^{-x}}$ | ✗ | $\frac{4e^{2x}}{(e^{2x}+1)^2}$ | $-\frac{8(e^{2x}-1)e^{2x}}{(e^{2x}+1)^3}$ | ✗ | ✓ |
| Sigmoid | $\frac{1}{1+e^{-x}}$ | ✗ | $\frac{e^x}{(e^x+1)^2}$ | $-\frac{(e^x-1)e^x}{(e^x+1)^3}$ | ✗ | ✓ |
| SiLU | $\frac{x}{1+e^{-x}}$ | ✗ | $\frac{e^x(e^x+x+1)}{(e^x+1)^2}$ | $-\frac{e^x((x-2)e^x-x-2)}{(e^x+1)^3}$ | ✗ | ✓ |
| SoftPlus | $\frac{1}{a}\log(1+e^{ax})$ | ✓ | $\frac{e^{cx}}{1+e^{cx}}$ | $\frac{ce^{cx}}{(e^{cx}+1)^2}$ | ✓ | ✗ |
| Gaussian | $e^{\frac{-0.5x^2}{a^2}}$ | ✓ | $-\frac{xe^{-\frac{x^2}{2a^2}}}{a^2}$ | $\frac{(x^2-a^2)e^{-\frac{x^2}{2a^2}}}{a^4}$ | ✓ | ✓ |
| Quadratic | $\frac{1}{1+(ax)^2}$ | ✓ | $-\frac{2a^2x}{(a^2x^2+1)^2}$ | $\frac{2a^2(3a^2x^2-1)}{(a^2x^2+1)^3}$ | ✓ | ✓ |
| Multi Quadratic | $\frac{1}{\sqrt{1+(ax)^2}}$ | ✓ | $-\frac{a^2x}{(a^2x^2+1)^{\frac{3}{2}}}$ | $\frac{2a^4x^2-a^2}{(a^2x^2+1)^{\frac{5}{2}}}$ | ✓ | ✓ |
| Laplacian | $e^{\left(\frac{-\lvert x\rvert}{a}\right)}$ | ✓ | $\frac{xe^{\frac{\lvert x\rvert}{a}}}{a\lvert x\rvert}$ | $\frac{e^{\frac{\lvert x\rvert}{a}}}{a^2}$ | ✓ | ✓ |
| Super-Gaussian | $\left[e^{\frac{-0.5x^2}{a^2}}\right]^b$ | ✓ | $-\frac{bxe^{-\frac{bx^2}{2a^2}}}{a^2}$ | $\frac{b(bx^2-a^2)e^{-\frac{bx^2}{2a^2}}}{a^4}$ | ✓ | ✓ |
| ExpSin | $e^{-\sin(ax)}$ | ✓ | $ae^{\sin(ax)}\cos(ax)$ | $-a^2e^{\sin(ax)}(\sin(ax)-\cos^2(ax))$ | ✓ | ✓ |

Table 1: **Comparison of existing activation functions (top block) against the proposed activation functions (bottom block).** The proposed activations and the sine activations fulfill **R1** and **R2**, implying better suitability to encode high-frequency signals.

# Hybrid Representations

# Hybrid Representations

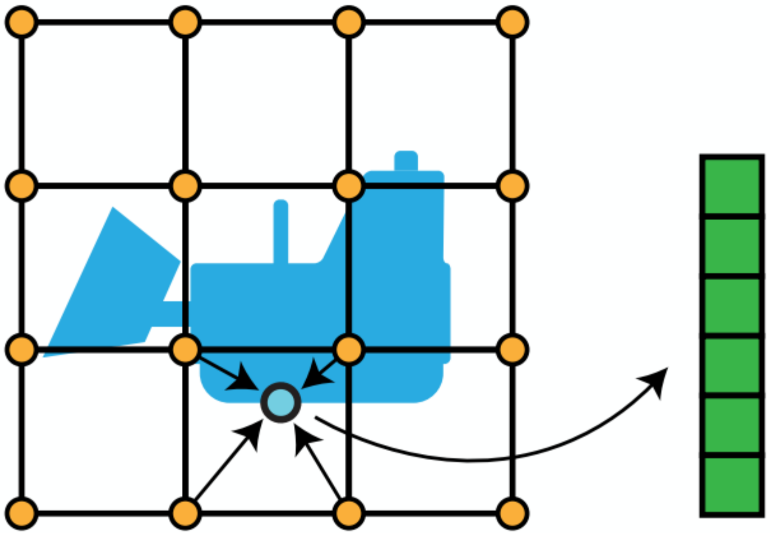
- Uniform Grids




**[PIFu (Saito et al.), Neural Volumes (Lombardi et al.), etc]**

# Hybrid Representations

- Uniform Grids



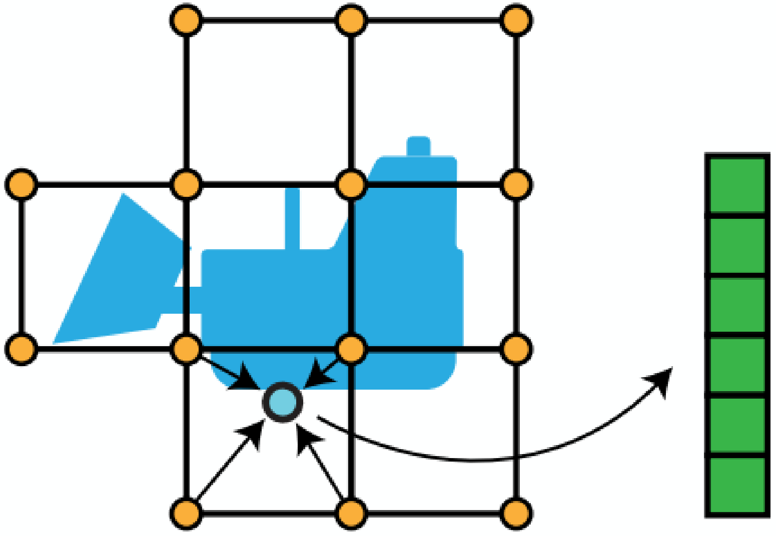
**Pros:**
- Easy to implement
- Algorithmically fast access [O(1)]
- Established operations like convolutions
- Simple topology

**Cons:**
- Expensive in memory and bandwidth
- Limited by Nyquist

**[PIFu (Saito et al.), Neural Volumes (Lombardi et al.), etc]**

# Hybrid Representations

- Sparse Grids



**Pros:**

- Memory Efficient
- Algorithmically efficient access [O(log(n))]
- GPU-compatible data structures
- Established operations like sparse 3D convs

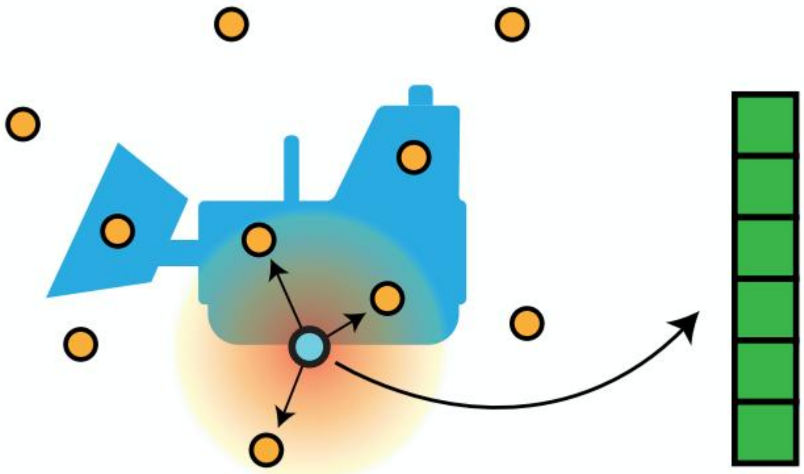
**Cons:**

- Need to manage a complex data structure
- Topology hard to generate (sparse grids)
- Still limited by Nyquist
- Sparse support region (have undefined points in space)

[DeepLS (Chabra et al.), NSVF (Liu et al.), NGLOD (Takikawa et al.), etc]

# Hybrid Representations

- Point Clouds



Aggregate nearby points feature with a kernel function

**Pros:**

- Not limited by Nyquist
- Can be densely supported in space
- Expressive

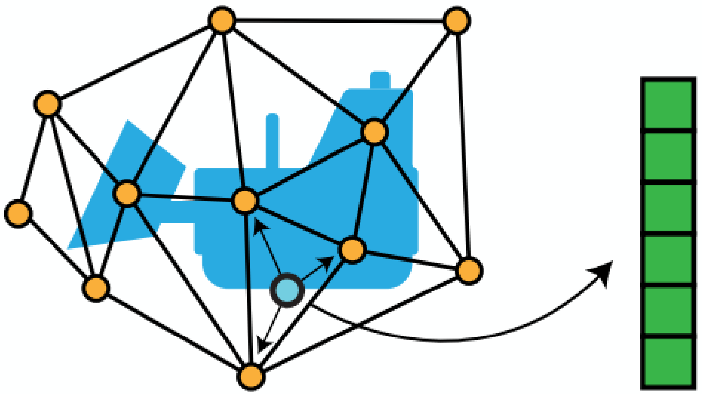
**Cons:**

- Often needs complex data structures for fast access and interpolation
- Heavily affected by choice of kernel

[Liu et al. 2019, LDIF (Genova et al.), 3DILG (Zhang et al.) etc]

# Hybrid Representations

- Mesh



**Interpolate nearby points on a face.**

**Pros:**

- Not limited by Nyquist
- Can use the rich sets of tools in mesh processing

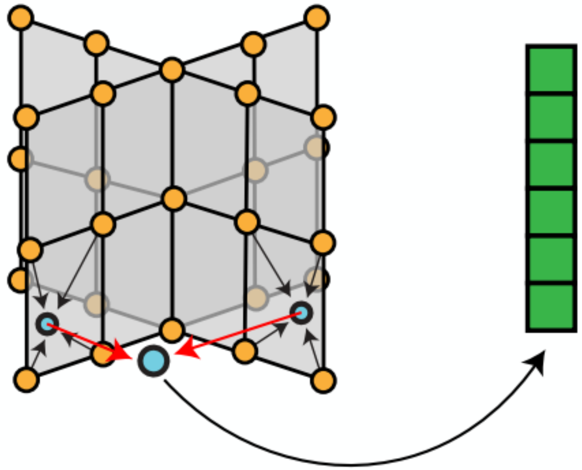
**Cons:**

- Is a mesh (difficult to process with NNs)
- Non-trivial data access especially in 3D

[DefTet (Gao et al.), NeuralBody (Peng et al.), etc]

# Hybrid Representations

- Multiplanar Images



**Aggregate from projections on multiple 2D feature planes.**

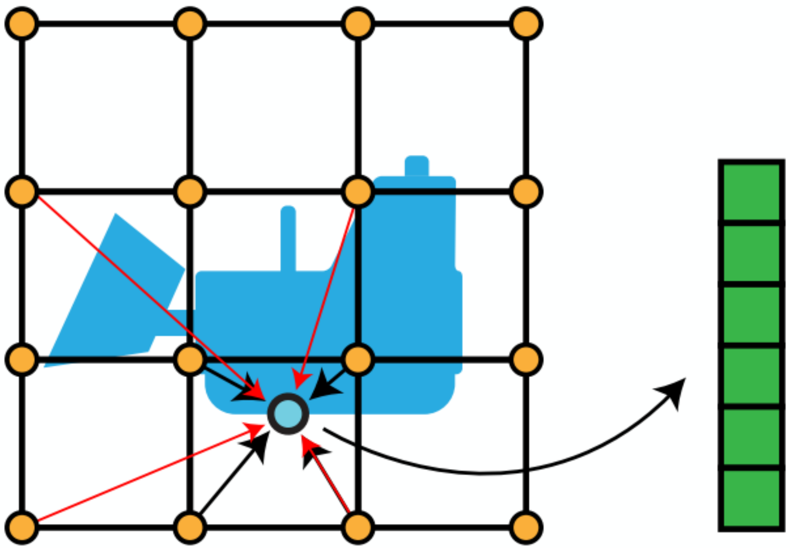
**Pros:**
- More compact than 3D dense grids
- Compatibility with 2D pipelines (2D CNNs)

**Cons:**
- Resolution bias on plane axis

[Convolutional OccNet (Peng et al), EG3D (Chan et al.), etc]

# Hybrid Representations

- Multiresolution Images



**Pros:**

- Multiple streaming levels of detail (LOD)
- Stable training
- Wider support region

**Cons:**

- More memory
- More complexity

[NGLOD (Takikawa et al.), ACORN (Lindell et al.), Instant-NGP (Muller et al.), etc]

# Hybrid Representations

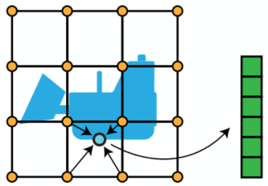
More hybrid representations:

- Hash Grids [Instant-NGP (Muller et al.)]
- Codebook Grids [Variable Bitrate Neural Fields (Takikawa et al.)]
- Bounding Volume Hierarchies [Neural Scene Graphs (Ost et al), Object-Centric Neural Scenes (Guo et al.), etc]

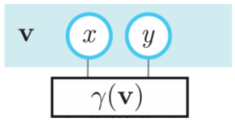
...

# Key Components in Architectures

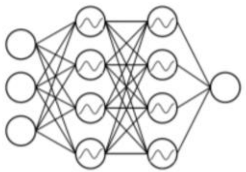
Representations



Input Encoding




Activation Functions



# Thanks for watching!

# Some Slides adapted from…

- CVPR 2022 Tutorial on Neural Fields in Computer Vision
- Tutorial on Neural Fields in Computer Vision

  from Towaki Takikawa, NVIDIA / University of Toronto
- Prior-based Reconstruction of Neural Fields

  from Prof. Vincent Sitzmann, MIT