Content

Part 1 (Theory):

- Machine Learning & Deep Learning

Part 2 (Lecture):

- Self-supervised deep learning with differentiable physics engine

Content

- Supervised Learning
- Do we have the functions?
- Supervised Learning for known systems/equations
- Self-supervised Learning
- Objective functions and constrained optimization
- Implicit Neural Representation
- Published Works
- Let’s intuitively solve a problem!
- Things we did not discuss!
- QA

Supervised Learning

- We need data (x_i, y_i)
  - More data → higher resolution of sampling
  - More data → Closer to continuous representation

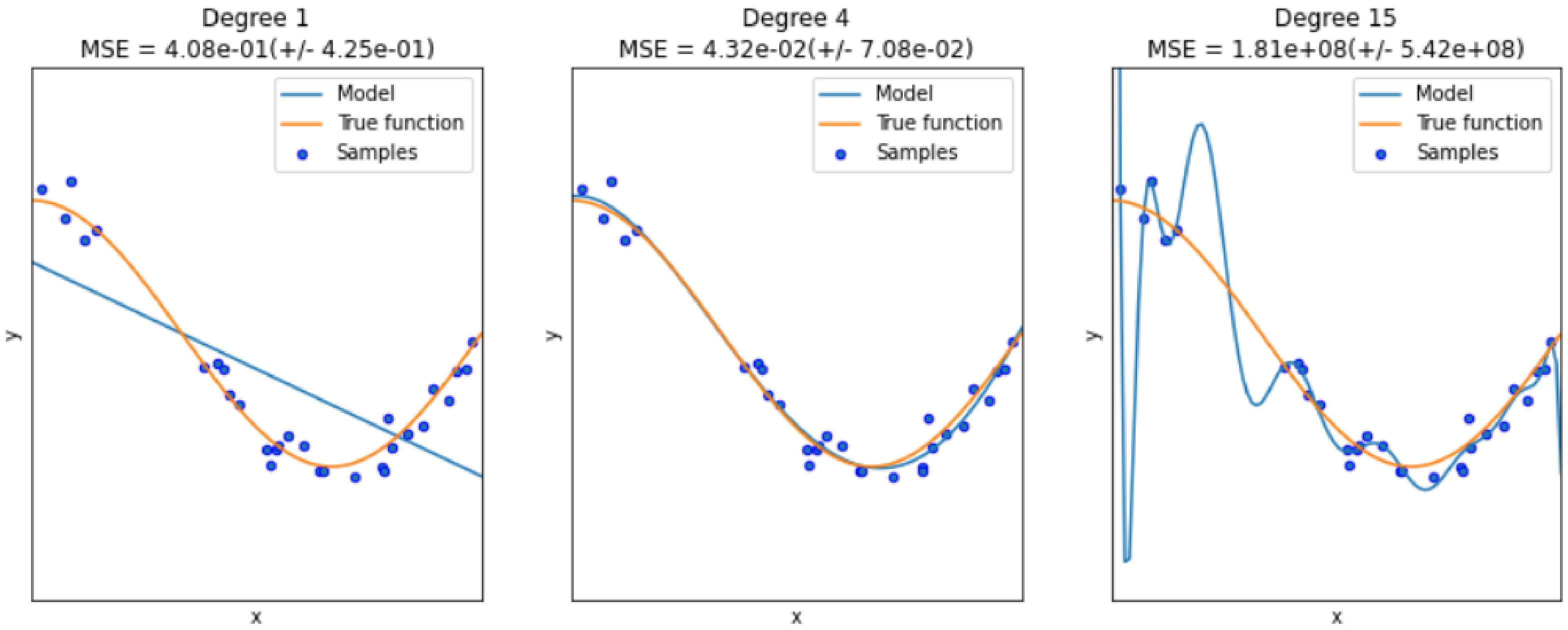


Image: datascience.foundation

Supervised Learning

- Neural network ***interpolates*** unseen points

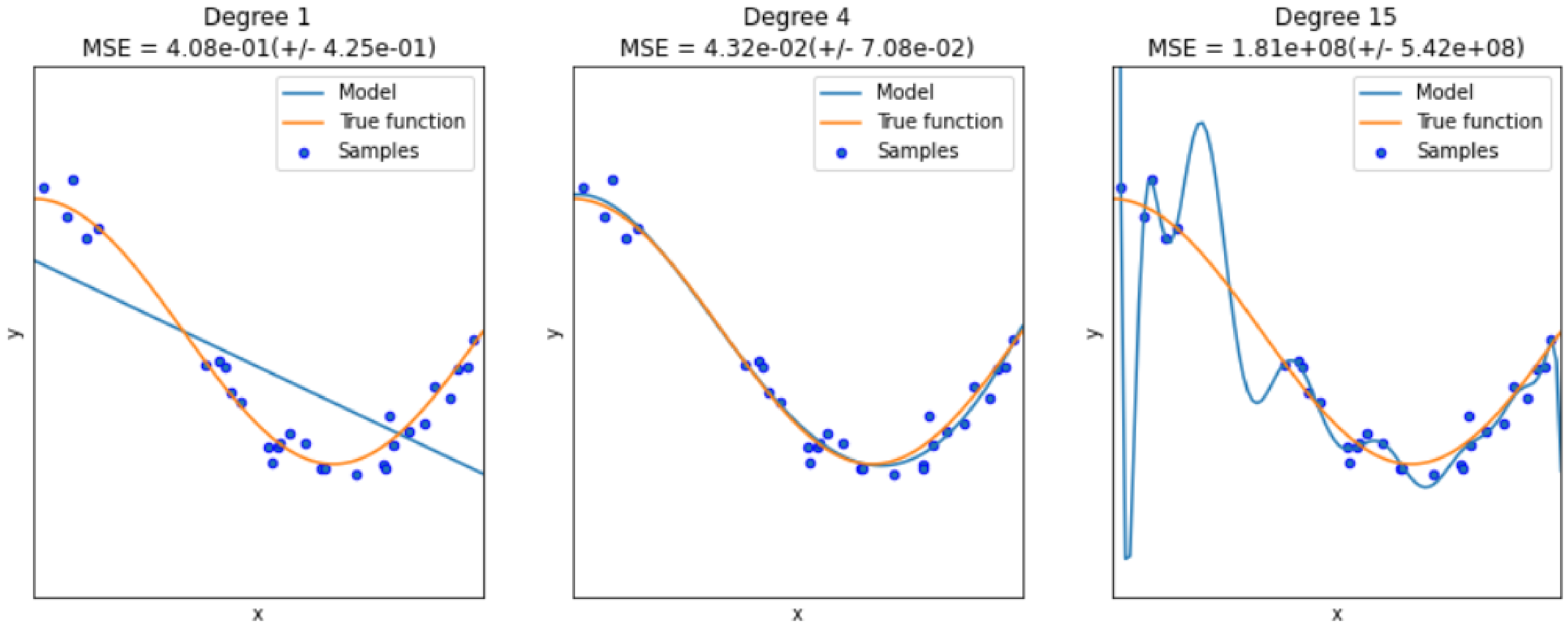


Image: datascience.foundation

Supervised Learning

- Supervised Learning is the most straightforward way of observing environment (e.g. target function)
  - In nature, we first observe samples
  - Then design a system that can capture observed samples
  - Finally, we want our model of system to give us an estimation of samples we have not seen

- We do not know the target function (we have samples of them)
  - Having more and more samples enables us to
    - Reconstruct the original function (***interpolation***)

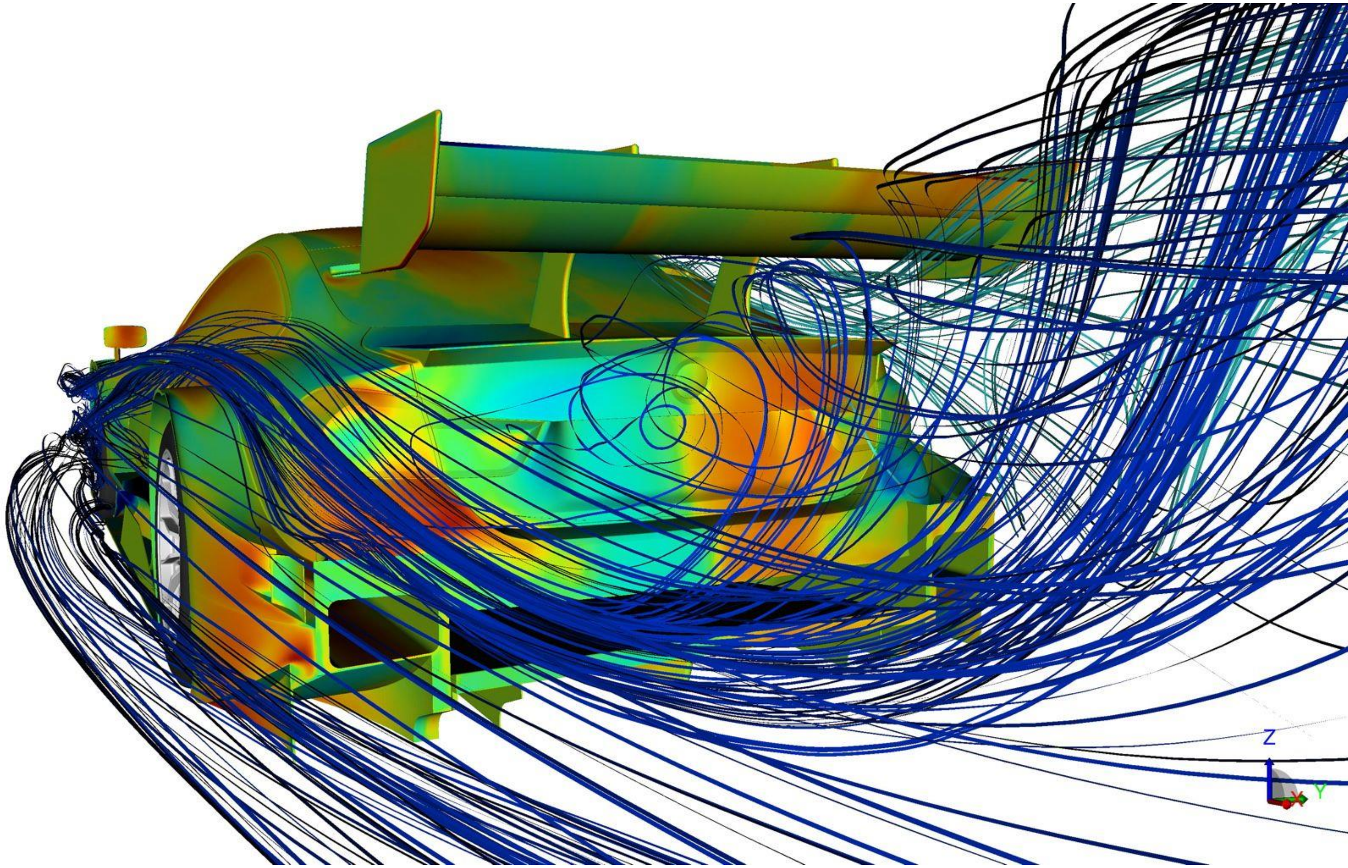
Image: friendlystock.com

Do We Have The Functions?

- There are many events in nature that have been studied very well
  - Newton physics
  - ***Partial Differential Equation***
    - Fluid dynamics
    - Motion
    - Elasticity
    - etc

- These models are general, could be solved for by them

- Why not using them?



school of ai
Rasht chapter

Image: Pinterest

Do We Have The Functions?

- The problem:
  - The equations related to these problems are hard or some times impossible to solve
  - There is almost no general method that can be used for all of problems expressed by a single PDE
  - For example, peer-reviewed papers in these topics, some times only solve a specific example (e.g. only a plane)

Supervised Learning For Known Equations

- It is counter-intuitive
  - Why sample a set of solved examples?

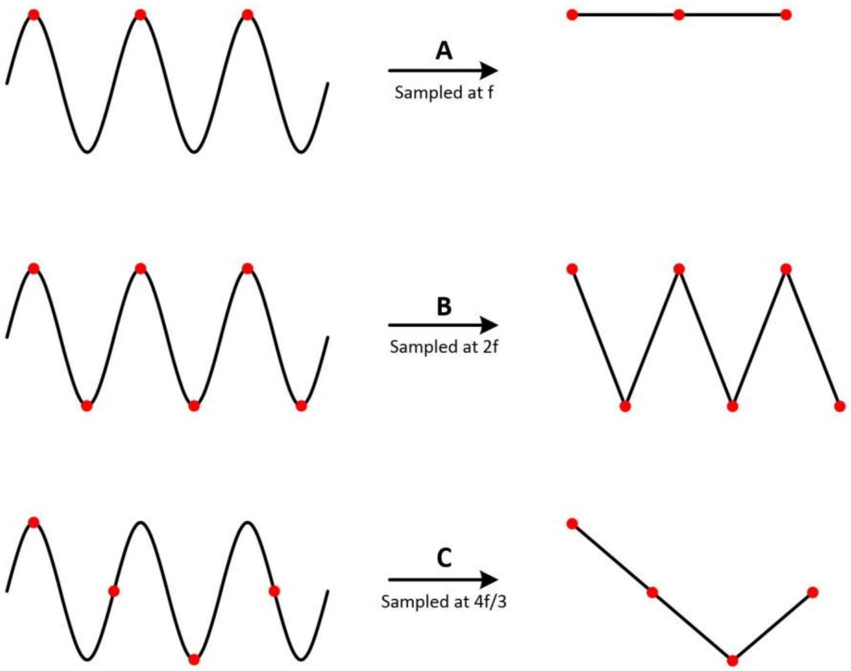


image: Psychology & Neuroscience Stack Exchange

Supervised Learning For Known Equations

- It is counter-intuitive
  - Why sample a set of solved examples?

- Physics-related simulations are computationally very expensive
  - We rather solve a system accurately rather than spending 100x more resources to inaccurately generalize to all of them
  - A small change in condition of problem usually creates a very different solution (hard to interpolate)

- Why not just solve for a single problem?
  - Only a single pair of (x_0, y_0)

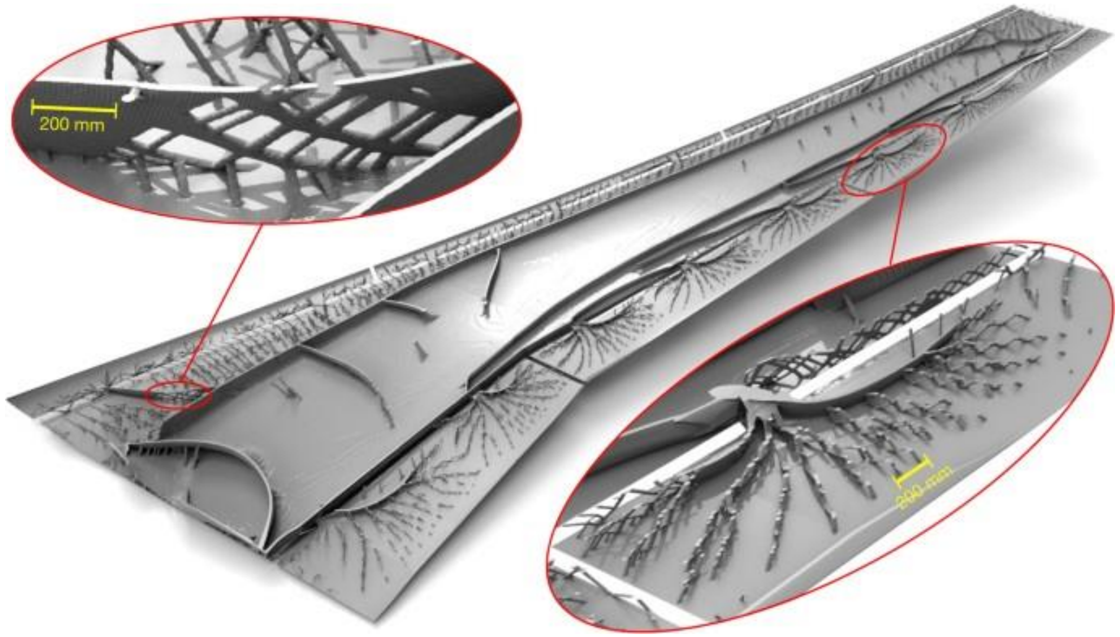


Image: Giga-voxel computational morphogenesis for structural design | Nature

Self-Supervised Learning

- No pre-defined ground truth

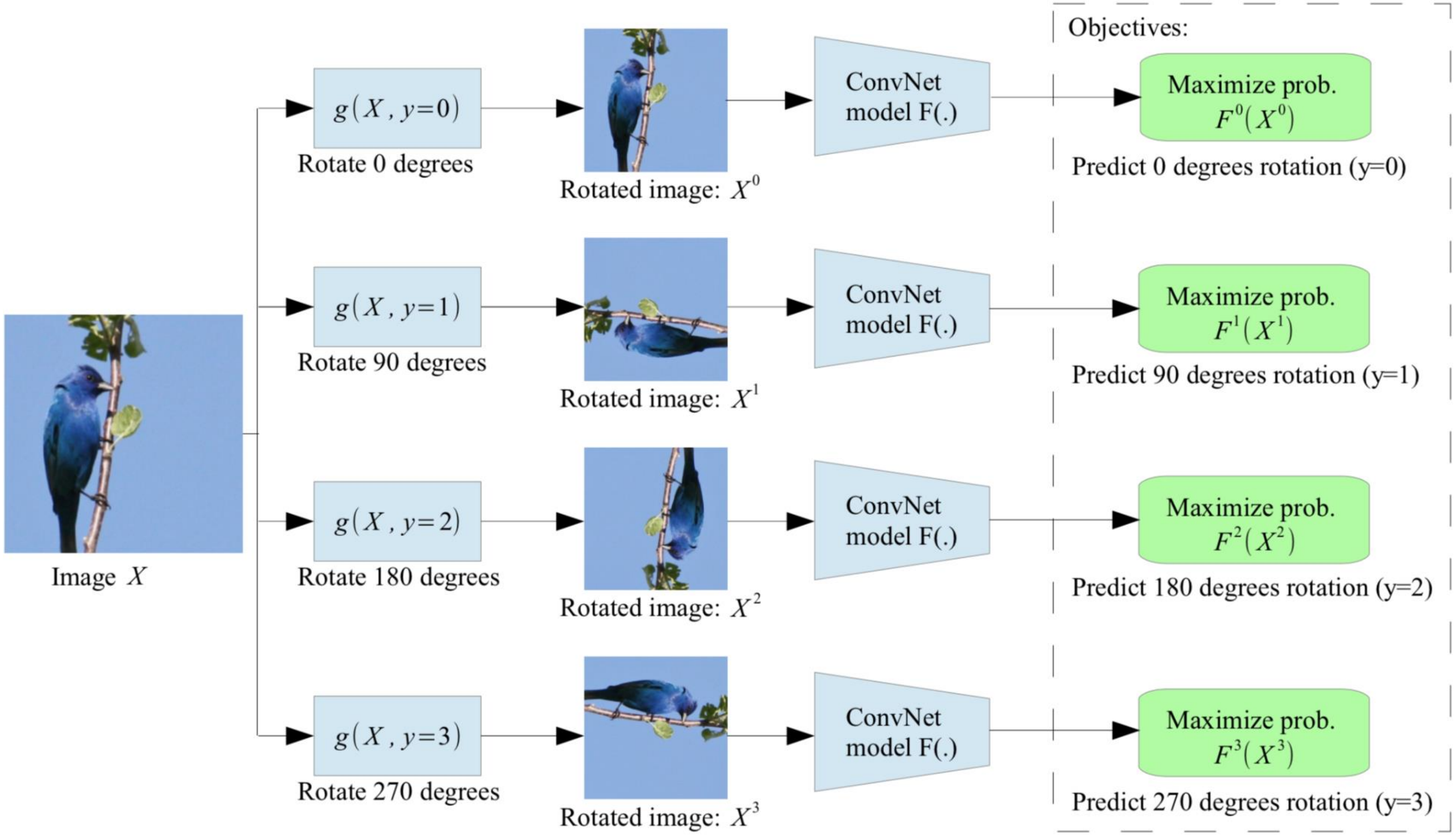
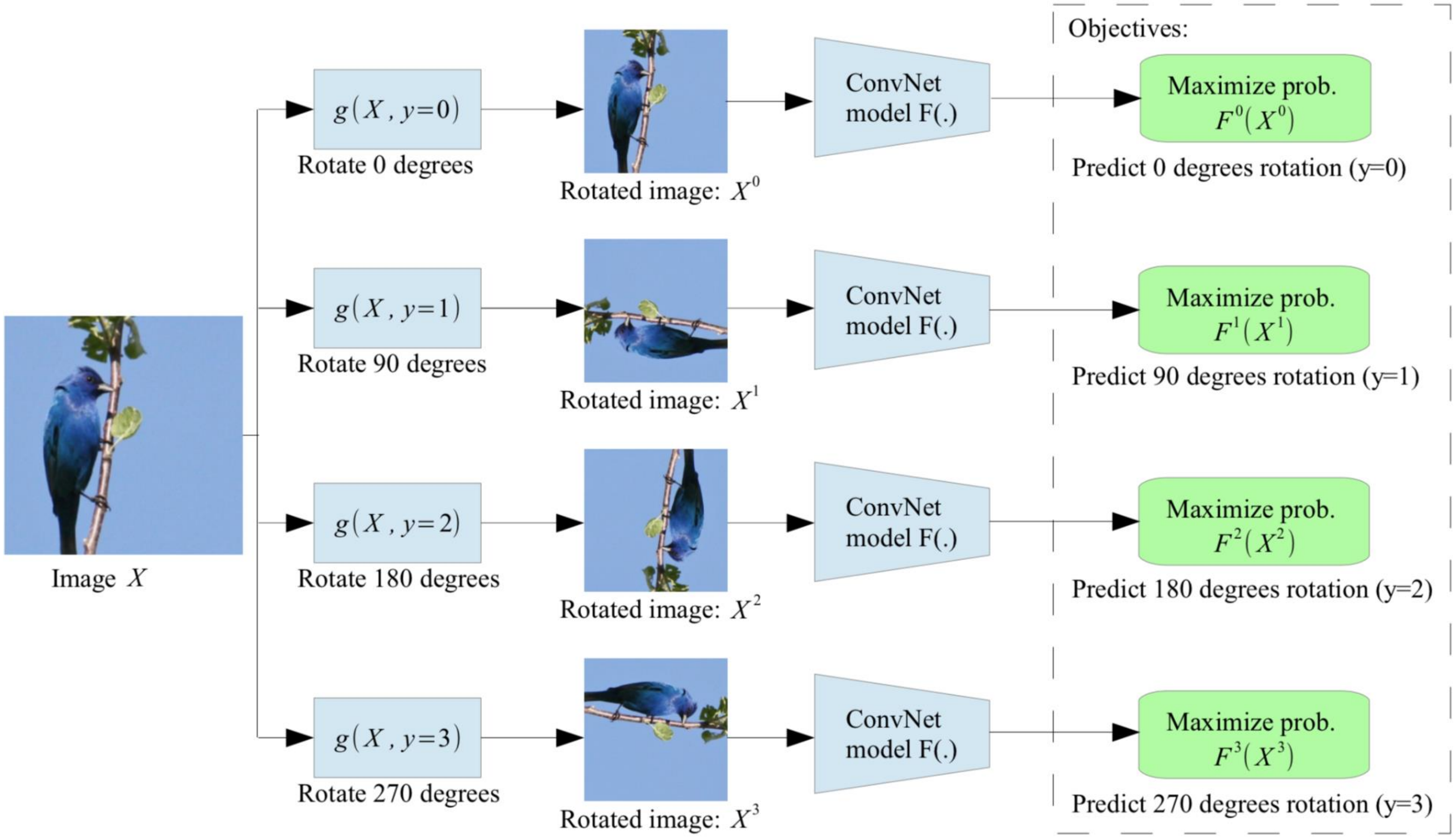


Image: Self-Supervised Representation Learning (lilianweng.github.io)

Self-Supervised Learning

- No pre-defined ground truth
- It is still “supervised”
  - ***Objective function*** as the ***supervision signal***



[Image: Self-Supervised Representation Learning (lilianweng.github.io)](lilianweng.github.io)

Self-Supervised Learning

- No pre-defined ground truth

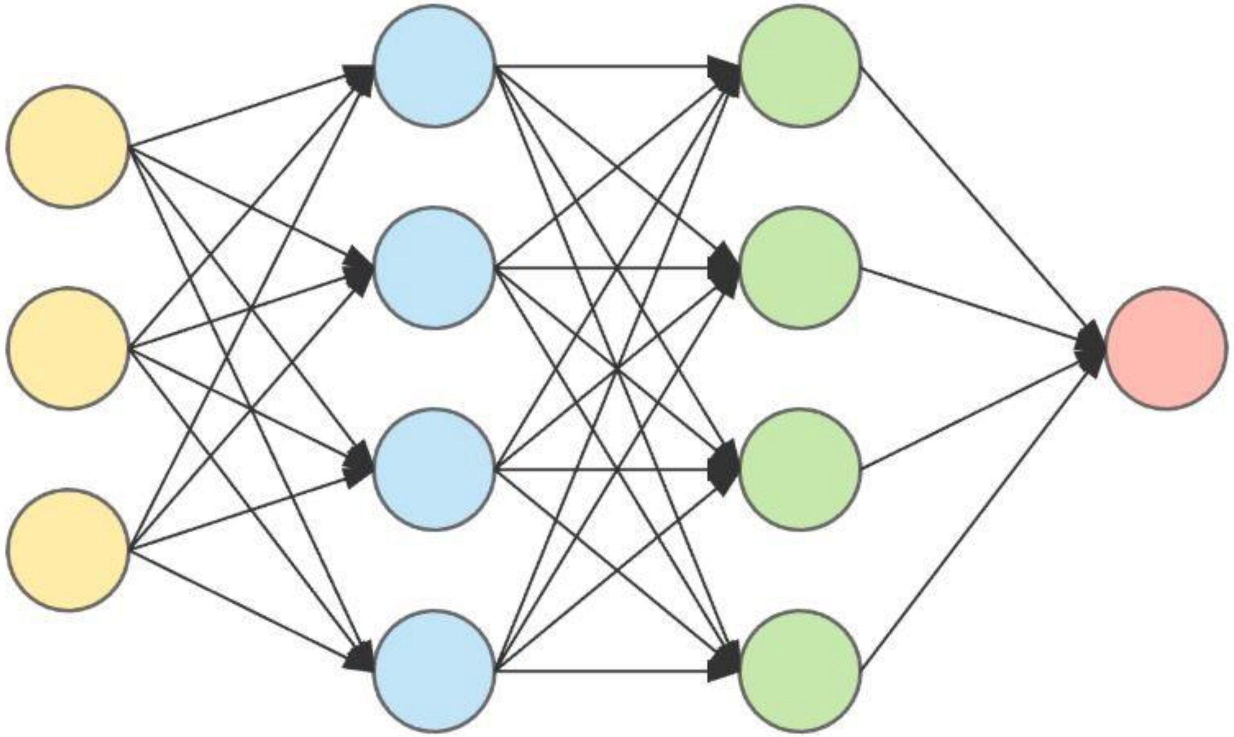
- It is still “supervised”
  - ***Objective function*** as the ***supervision signal***

- General approximation theorem
  - An neural network with one hidden layer can approximate any function

- So, in previous example
  - If we assume the ***same network for all problems***
  - Then ***objective function is the representation of problem***

Self-Supervised Learning

- MLP + Objective function



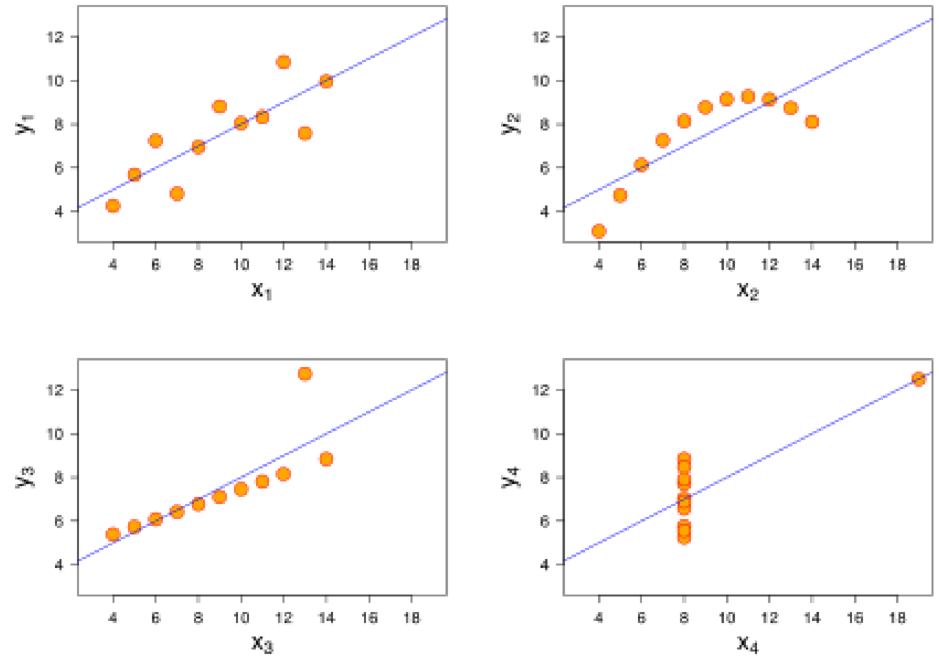


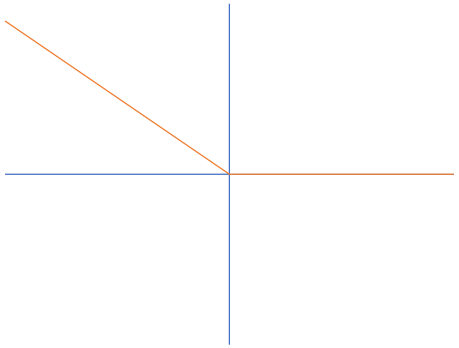
Image: Kdnuggets

Image: Wikipedia

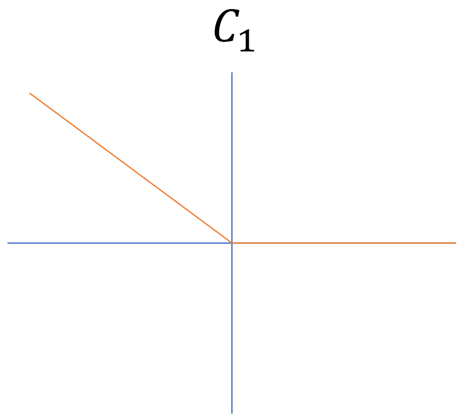
Objective Function (Loss, Error, etc)

- We can encode any function and its constraints into a single ***weighted sum of function and its constraints***
  - $L_{total} = \lambda_1 L_{obj} + \lambda_2 L_{C_1} + \cdots + \lambda_i L_{C_i}$
  - Constrained problem → unconstrained

- For example
  - $L_{obj} = \left|\left|ax + b - y\right|\right|^2$
  - Constraint: we only want *positive part of the function*
    - $C_1 = ax + b \geq 0 \;\rightarrow C_1 = f(x) = \begin{cases} -(ax+b), & o.w. \\ 0, & ax + b \geq 0 \end{cases}$



Objective Function (Loss, Error, etc)



- For example
  - $L_{obj} = \left||ax + b - y|\right|^2$
  - Constraint: we only want *positive part of the function*
    - $C_1 = ax + b \geq 0 \ \rightarrow C_1 = f(x) = \begin{cases} -(ax+b), & o.w. \\ 0, & ax + b \geq 0 \end{cases}$

- Why not adding this constraint to the network (e.g. ReLU)?
  - Yes you can! We call them hard constraints against soft (C_i)
  - Intuitively, any constraint could be encoded into loss function

Implicit Neural Representation

- Problem
  - Fitting any signal in form of $F\left(\mathbf{x}, \Phi, \nabla_{\mathbf{x}} \Phi, \nabla_{\mathbf{x}}^2 \Phi, \ldots\right)=0, \quad \Phi: \mathbf{x} \mapsto \Phi(\mathbf{x})$

- Goal
  - Learn a network that parameterizes Φ to map x to desired function while satisfying constraints in F

- Benefits
  - **Continuous** and **differentiable**
  - Resolution independent (details)
  - Calculation of **higher-order derivatives analytically**

Φ : function

x : inputs (coords)

∇_x^i : i-th gradient wrt x

school of ai
Rasht chapter

[Implicit Neural Representations with Periodic Activation Functions (vsitzmann.github.io)](vsitzmann.github.io)

Implicit Neural Representation

- SIREN
  - We cast constraints into loss function (**soft** constrains)

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

  - Use $Sine$ activation function for all layers
    - Any derivative of SIREN is SIREN
    - Could be used to calculate nth order derivatives (PDEs)

$\phi_i : \mathbb{R}^{M_i} \to \mathbb{R}^{N_i}$ $\quad i^{th}$ layer of NN

$W_i \in \mathbb{R}^{N_i \times M_i}$: weights

$b_i \in \mathbb{R}^{N_i}$: biases

$x_i \in \mathbb{R}^{M_i}$: inputs

school of ai Rasht chapter

Implicit Neural Representations with Periodic Activation Functions (vsitzmann.github.io)

Implicit Neural Representation

- Example of image fitting on original data, gradient of image or Laplacian of it.

$$\tilde{\mathcal{L}} = \sum_i \|\Phi(\mathbf{x}_i) - f(\mathbf{x}_i)\|^2$$

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

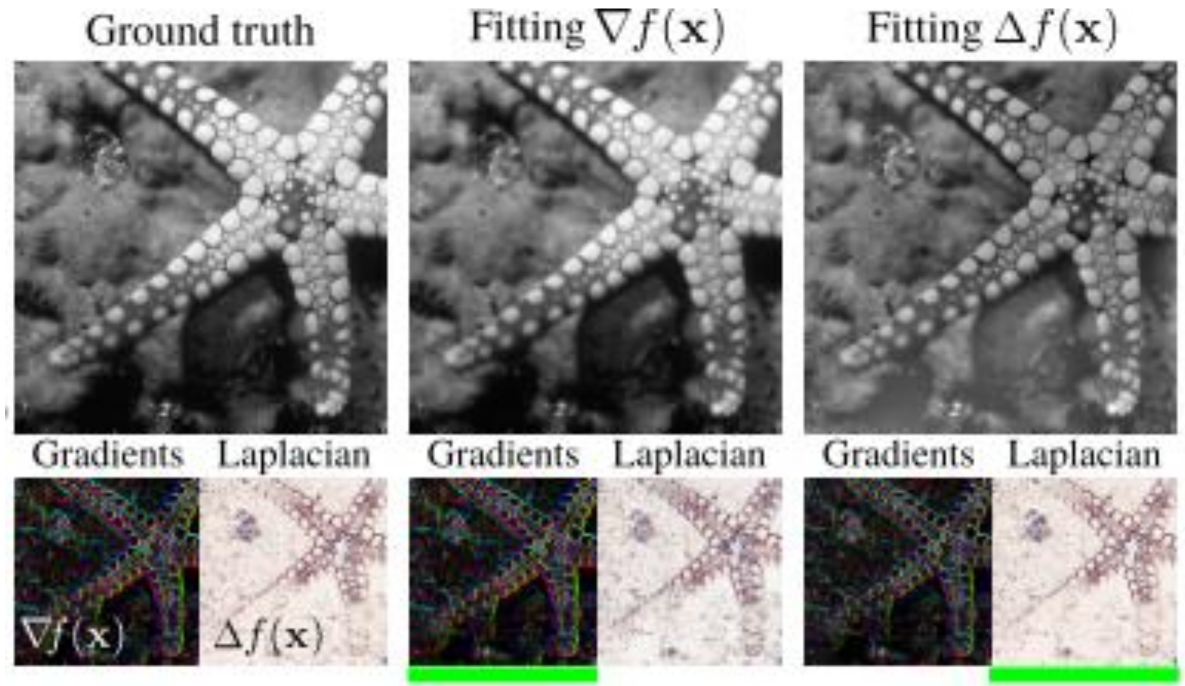
$$\mathcal{L}_{\text{lapl.}} = \int_\Omega \|\Delta\Phi(\mathbf{x}) - \Delta f(\mathbf{x})\| \, d\mathbf{x}$$

$\Phi(x_i)$: prediction of siren

$f(x_i)$: ground truth

$\nabla_x(x)$: Gradient w.r.t. coords

$\Delta_x(x)$: Laplace w.r.t. coords



[Implicit Neural Representations with Periodic Activation Functions (vsitzmann.github.io)](https://vsitzmann.github.io)

Implicit Neural Representation

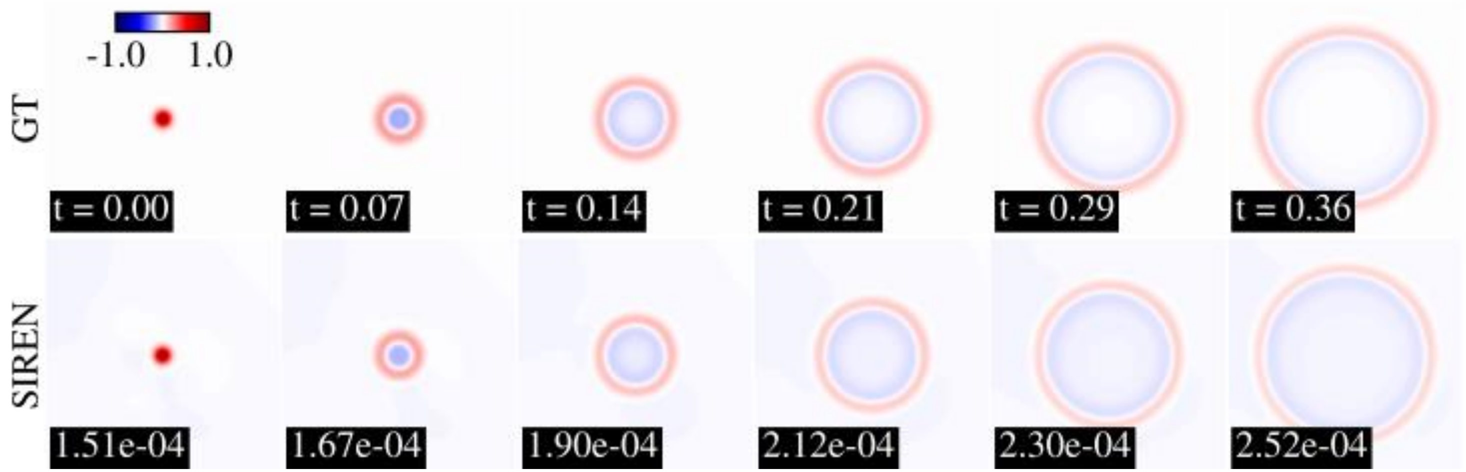
- Example of wave equation
- Wave

$$\frac{\partial \Phi}{\partial t} - c^2 \frac{\partial \Phi}{\partial \mathbf{x}} = 0.$$

- Boundary conditions

$$\frac{\partial \Phi(0, \mathbf{x})}{\partial t} = 0$$

$$\Phi(0, \mathbf{x}) = f(\mathbf{x})$$



x : Coords
t : time
$\lambda_1(x), \lambda_2(x)$: hyper parameters
$f(x)$: desired function from BCs

[Implicit Neural Representations with Periodic Activation Functions (vsitzmann.github.io)](https://vsitzmann.github.io)

Implicit Neural Representation

- Example of wave equation
- Wave

$$\frac{\partial \Phi}{\partial t} - c^2 \frac{\partial \Phi}{\partial \mathbf{x}} = 0.$$

- Boundary conditions

$$\frac{\partial \Phi(0, \mathbf{x})}{\partial t} = 0$$

$$\Phi(0, \mathbf{x}) = f(\mathbf{x})$$

$$\mathcal{L}_{\text{wave}} = \int_\Omega \left\| \frac{\partial \Phi}{\partial t} - c^2 \frac{\partial \Phi}{\partial \mathbf{x}} \right\|_1 + \lambda_1(\mathbf{x}) \left\| \frac{\partial \Phi}{\partial t} \right\|_1 + \lambda_2(\mathbf{x}) \left\| \Phi - f(\mathbf{x}) \right\|_1 \, d\mathbf{x}$$

x : Coords
t : time
$\lambda_1(x), \lambda_2(x)$: hyper parameters
$f(x)$: desired function from BCs

[Implicit Neural Representations with Periodic Activation Functions (vsitzmann.github.io)](https://vsitzmann.github.io)

Published Works

- Liang et al. differentiable **cloth simulation**
- Hu et al. a real-time differentiable simulator for **soft robotics** for applications in reinforcement learning
- Geilinger et al. handles **frictional contacts** for both rigid and deformable bodies
- Um et al. leverage a differentiable **fluid simulator** inside the training loop to reduce numerical errors in a traditional solver

- DeepXDE contains different methods of solving PDEs directly.

school of ai
Rasht chapter

Let’s Intuitively Solve An Example!

- Question: How to find integral of a function $F(x) = ?$

$$\int f(x) = F(x)$$

- We have $f(x)$, a MLP ϕ_θ and any objective function defined w.r.t. $f(x)$

$$L_{obj} = \boxed{\frac{\partial \phi_\theta}{\partial x}} - \boxed{f(x)}$$

Hint: $L_{obj} = ||ax + b - y||^2$

[2012.01714.pdf (arxiv.org)](https://arxiv.org/pdf/2012.01714.pdf)

Things We Did Not Discuss!

- A basic MLP cannot learn high frequency functions!
  - See **FourFeat, SIREN, Random Fourier Features,** etc.
  - Basic models are inefficient
    - We need smart sampling
  - Hardware is really important!
    - Nowadays, it is almost mandatory to join top-tier companies, institutes, universities to have access to powerful hardware.

QA