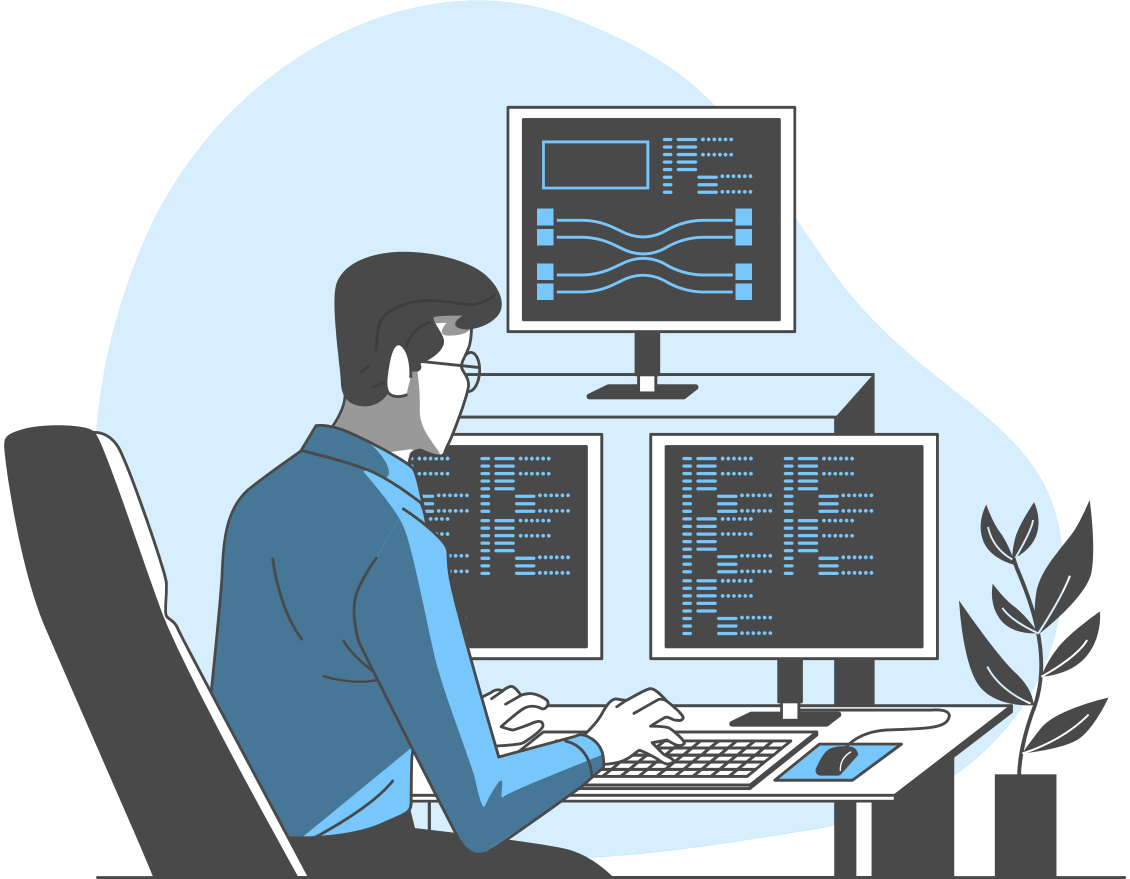


Implicit Neural Network

Jongwon Choi

Contents

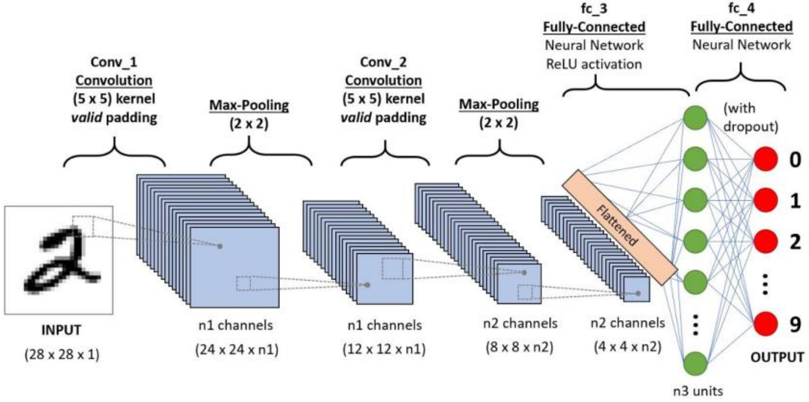
01

What is Implicit Neural Network?

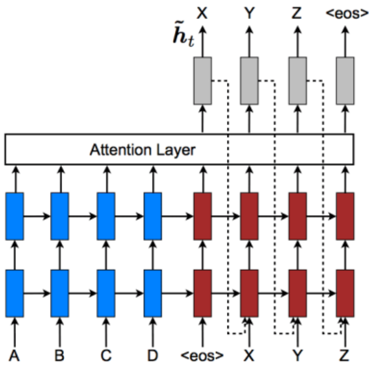
Conventional Neural Network (NN)

- A function trasforming the given input to the target output
- Ex. Convolutional Neural Network for Image, Recurrent Neural Network for Text

Convolutional Neural Network



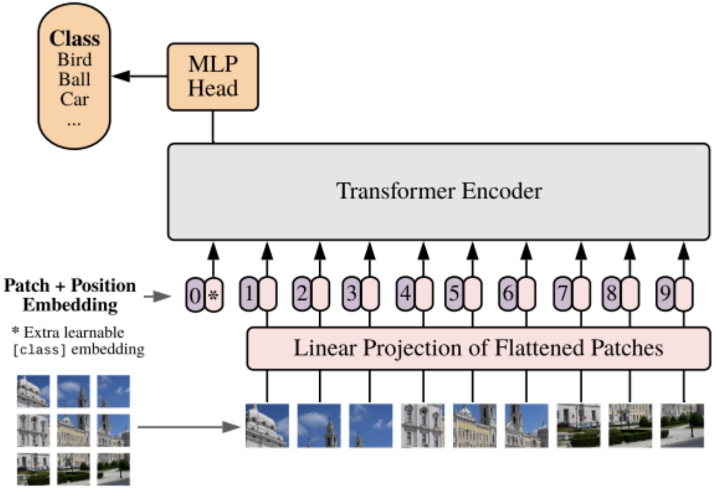
Recurrent Neural Network



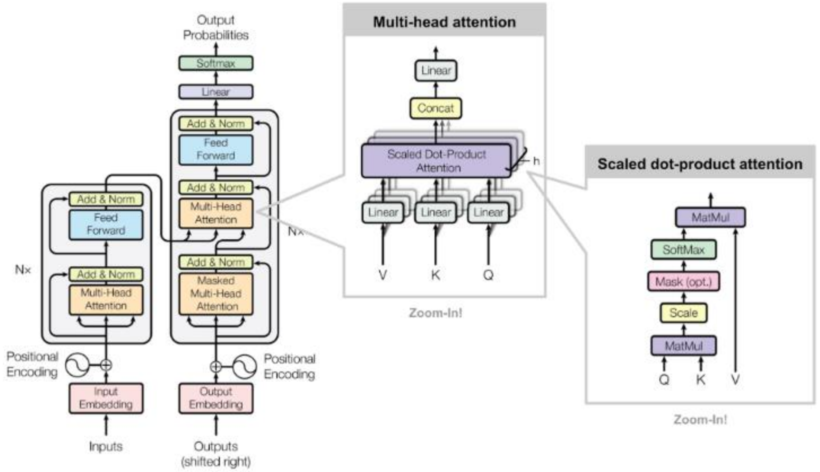
Conventional Neural Network (NN)

- A function trasforming the given input to the target output
- Recently, the transformer-based neural networks

ViT Model



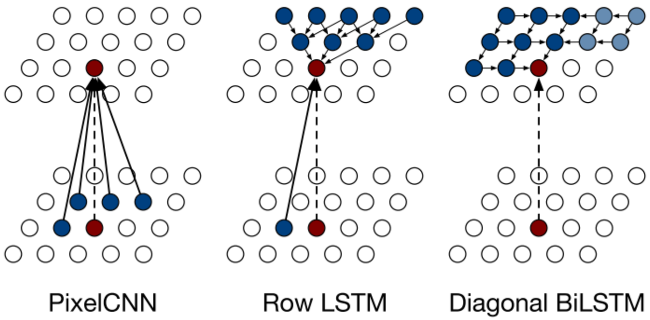
BERT

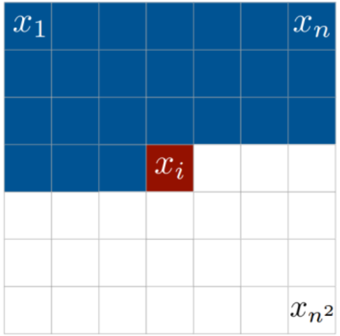


Conventional Neural Network (NN)

Definition & Limitation

- The conventional neural network trains the relation between the input data and the output labels
- Thus, it needs numerous data to obtain the distinctive features for the output labels
- Also, it is inefficient to contain the generative information of the respective input





Extension of Conventional NN

Definition & Limitation

- The conventional neural network trains the relation between the input data and the output labels
- Thus, it needs numerous data to obtain the distinctive features for the output labels
- Also, it is inefficient to contain the generative information of the respective input

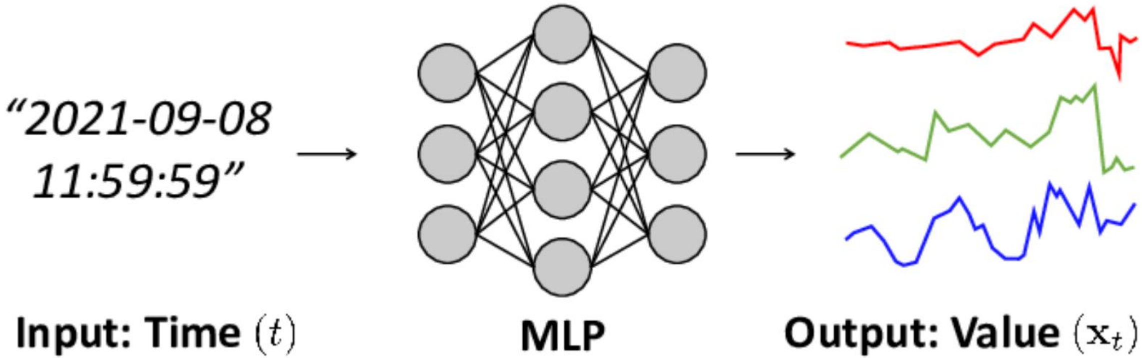
Extension

- Focus on the intra-relation of informations contained in the specific target input
- Reconstruct the original signal (Continuous domain) from the sampled signal (Discrete domain)
- Solve the partial derivative equations!

Implicit Neural Network!

Definition

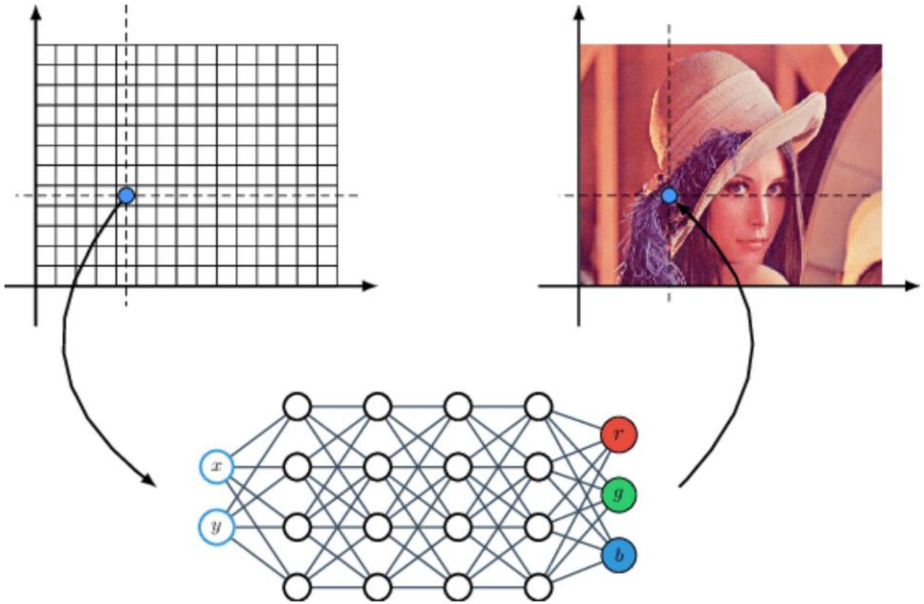
- Train the target signal itself using the sampled signal values
- Assume that the non-linearity between the sampled values can be generalized by neural network
- Or, reduce the discontinuity through the additional regularization term or the boundary conditions
- Very simple to implement a network when we can design a partial difference equation for our tasks



Implicit Neural Network!

Example

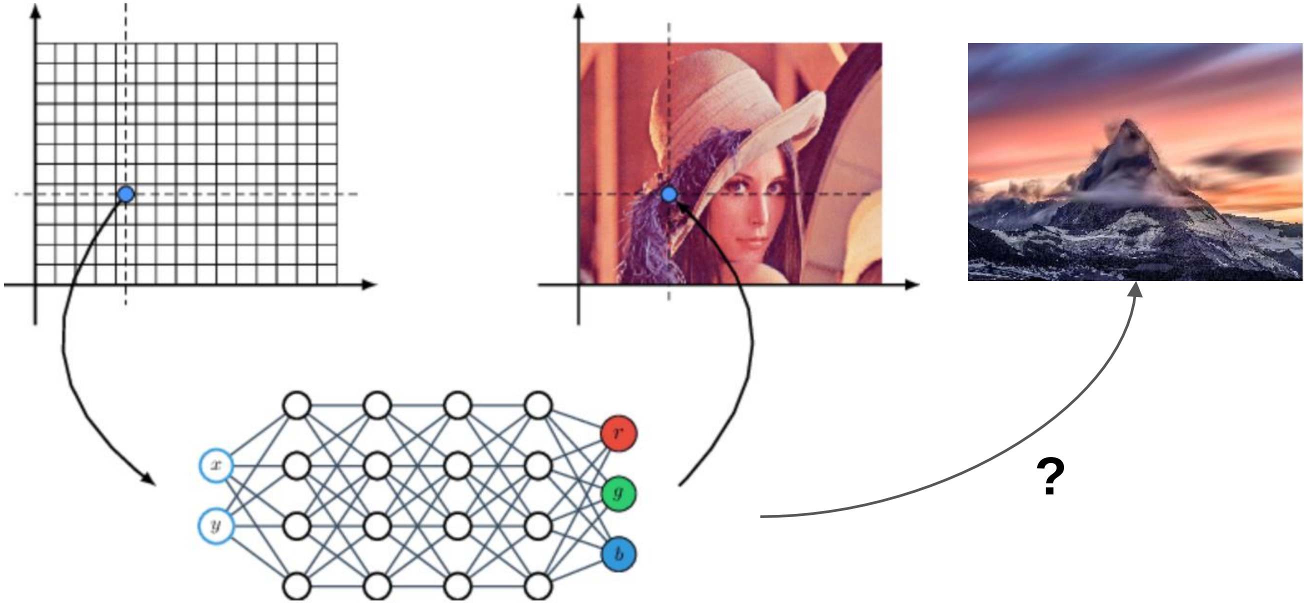
- For the image, we can train the intra-relation by learning a function given input coordinates



Implicit Neural Network!

What??

- Then, how can we handle the other image? – We cannot..



02

Partial Differential Equation (PDE)

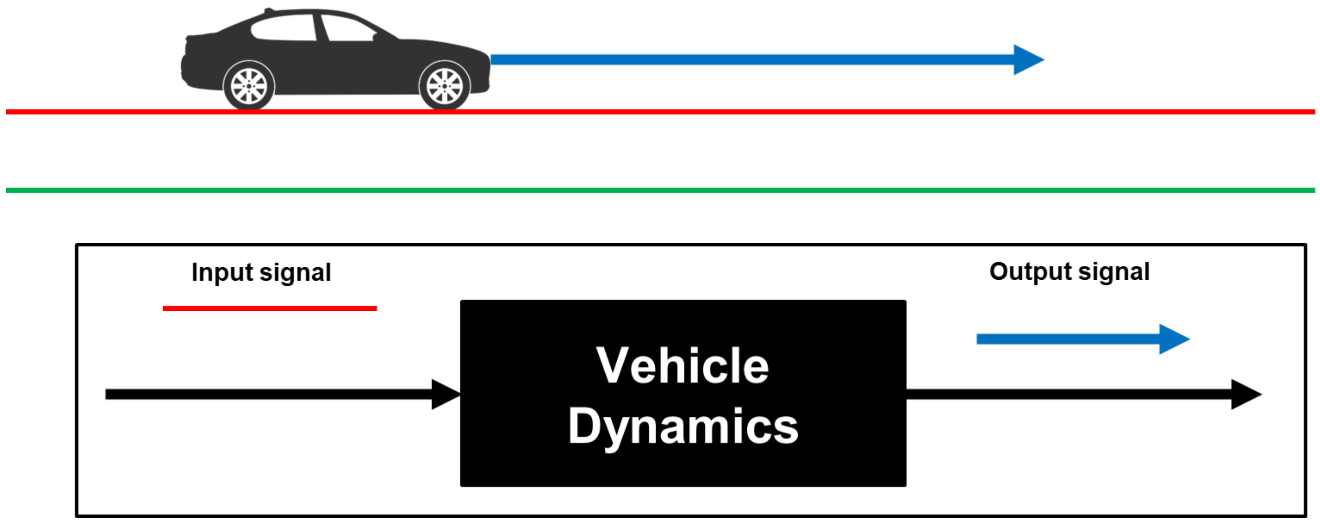
Partial Differential Equation (PDE)

Definition

- Generally, PDE is used in Physics and system design
- The equation represents the relations among the target function and its derivatives
- Solution of partial differential equation = Reconstructed target function

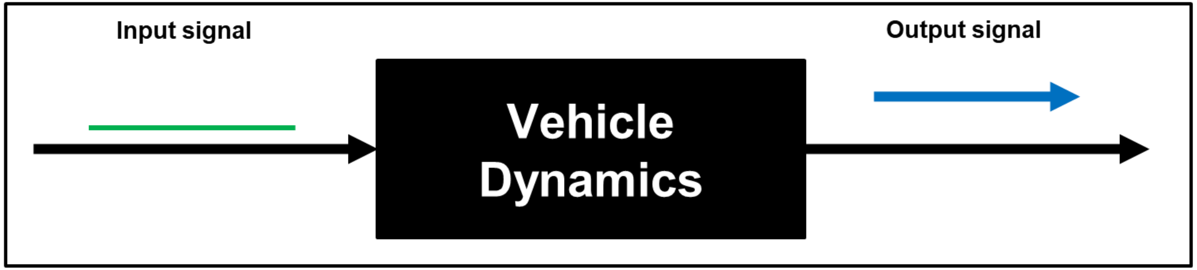
Partial Differential Equation (PDE)

Example – Difference Equation & z- transform



Partial Differential Equation (PDE)

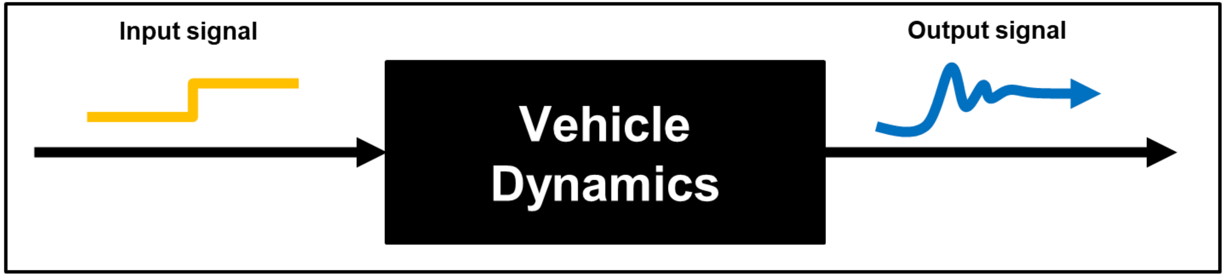
Example – Difference Equation & z- transform



The amplitude of input signal doesn't affect the result!

Partial Differential Equation (PDE)

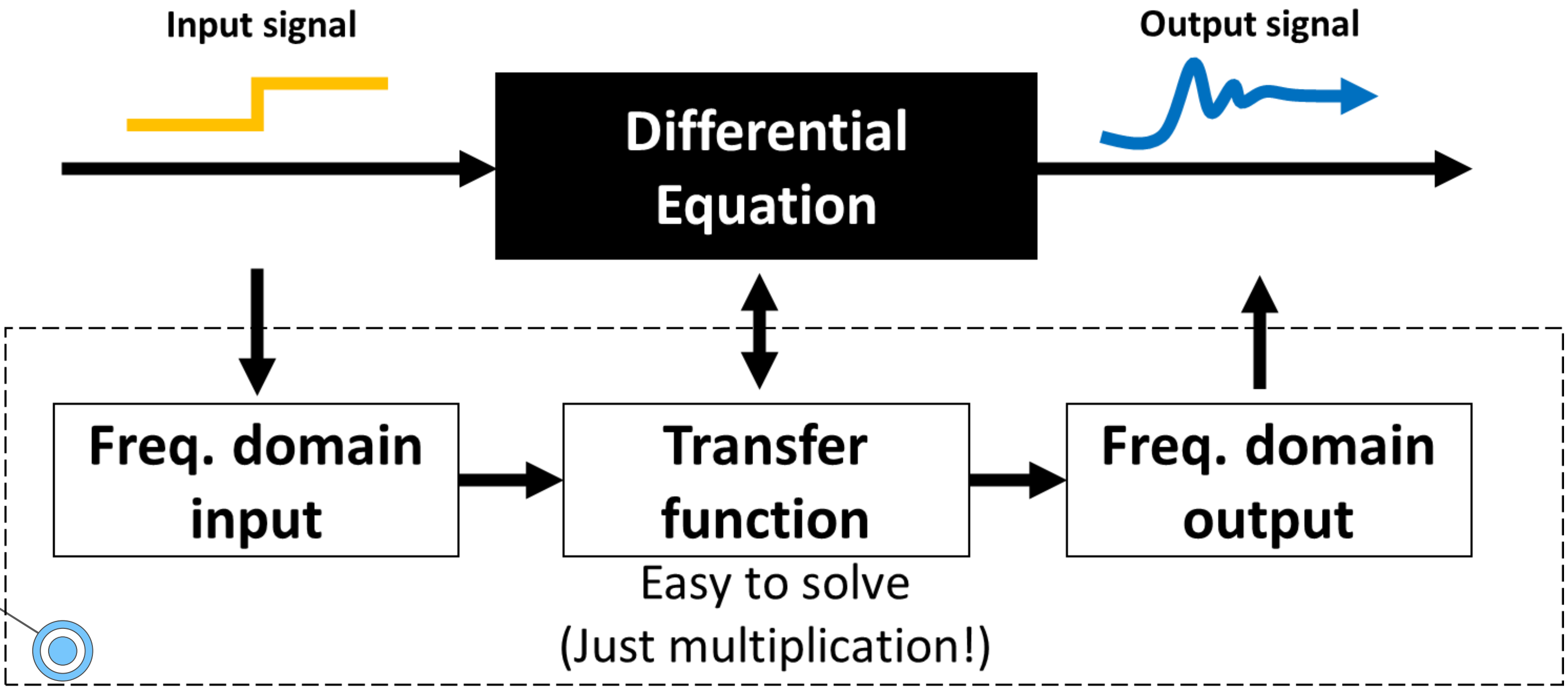
Example – Difference Equation & z- transform



Vehicle Dynamics = Differential Equation!

Partial Differential Equation (PDE)

Example – Difference Equation & z- transform



Partial Differential Equation (PDE)

Example – Difference Equation & z- transform

$\sum_{k=0}^{N} b_k y[n-k] = \sum_{k=0}^{M} a_k x[n-k]$,where $M \le N$

⇩

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{M} a_k z^{-k}}{\sum_{k=0}^{N} b_k z^{-k}}$$

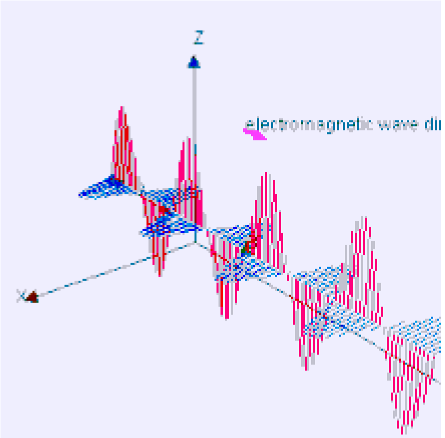
Property	Time Domain	Z Domain	ROC
Linearity	$a_1x_1(n)+a_2x_2(n)$	$a_1X_1(z)+a_2X_2(z)$	At least $R_1 \cap R_2$
Translation	$x(n-n_0)$	$z^{-n_0}X(z)$	R except possible addition/deletion of 0
Modulation	$a^n x(n)$	$X(a^{-1}z)$	$\|a\|R$
Time Reversal	$x(-n)$	$X(1/z)$	R^{-1}
Upsampling	$(\uparrow M)x(n)$	$X(z^M)$	$R^{1/M}$
Downsampling	$(\downarrow M)x(n)$	$\frac{1}{M}\sum_{k=0}^{M-1} X\left(e^{-j2\pi k/M}z^{1/M}\right)$	R^M
Conjugation	$x^*(n)$	$X^*(z^*)$	R
Convolution	$x_1 * x_2(n)$	$X_1(z)X_2(z)$	At least $R_1 \cap R_2$
Z-Domain Diff.	$nx(n)$	$-z\frac{d}{dz}X(z)$	R
Differencing	$x(n)-x(n-1)$	$(1-z^{-1})X(z)$	At least $R \cap \|z\| > 0$
Accumulation	$\sum_{k=-\infty}^{n} x(k)$	$\frac{z}{z-1}X(z)$	At least $R \cap \|z\| > 1$

Property	
Initial Value Theorem	$x(0) = \lim_{z\to\infty} X(z)$
Final Value Theorem	$\lim_{n\to\infty} x(n) = \lim_{z\to 1}[(z-1)X(z)]$

Partial Differential Equation (PDE)

Example – Maxwell Equation

Name	Integral equations	Differential equations
Gauss's law	$\oiint_{\partial\Omega} \mathbf{E} \cdot \mathrm{d}\mathbf{S} = \frac{1}{\varepsilon_0} \iiint_{\Omega} \rho \, \mathrm{d}V$	$\nabla \cdot \mathbf{E} = \frac{\rho}{\varepsilon_0}$
Gauss's law for magnetism	$\oiint_{\partial\Omega} \mathbf{B} \cdot \mathrm{d}\mathbf{S} = 0$	$\nabla \cdot \mathbf{B} = 0$
Maxwell–Faraday equation (Faraday's law of induction)	$\oint_{\partial\Sigma} \mathbf{E} \cdot \mathrm{d}\boldsymbol{\ell} = -\frac{\mathrm{d}}{\mathrm{d}t} \iint_{\Sigma} \mathbf{B} \cdot \mathrm{d}\mathbf{S}$	$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}$
Ampère's circuital law (with Maxwell's addition)	$\oint_{\partial\Sigma} \mathbf{B} \cdot \mathrm{d}\boldsymbol{\ell} = \mu_0 \left(\iint_{\Sigma} \mathbf{J} \cdot \mathrm{d}\mathbf{S} + \varepsilon_0 \frac{\mathrm{d}}{\mathrm{d}t} \iint_{\Sigma} \mathbf{E} \cdot \mathrm{d}\mathbf{S} \right)$	$\nabla \times \mathbf{B} = \mu_0 \left(\mathbf{J} + \varepsilon_0 \frac{\partial \mathbf{E}}{\partial t} \right)$



*** “Maxwell’s equations”, Berkeley Lab.

Partial Differential Equation (PDE)

Example – Material Derivative

- Conservation of mass

$$\frac{\partial}{\partial t}\iiint_V \rho\, dV = -\oiint_S \rho\mathbf{u}\cdot d\mathbf{S} \quad \Rightarrow \quad \frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\mathbf{u}) = 0$$

- Conservation of energy

$$\rho\frac{Dh}{Dt} = \frac{Dp}{Dt} + \nabla\cdot(k\nabla T) + \Phi$$

*** "*The Dawn of Fluid Dynamics: A Discipline Between Science and Technology*." Wiley

Solution of PDE

Finite Difference Method (FDM)

- Discretize the target space & Approximate the differential values

$$f(x+\delta) = f(x) + \delta f'(x) + \frac{\delta^2}{2} f''(x) + \frac{\delta^3}{6} f'''(x) + \cdots$$

$$[f(x+h) - f(x)]/h = f'(x) + O(h)$$

$$\frac{f(x+h) - f(x-h)}{2h} = f'(x) + O(h^2).$$

$$\frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$$

1. We need to set the grid size (h)
2. We need the initial value

*** “Computational Electromagnetics,” Bondeson et al., 2000

Boundary Condition

Dirichlet boundary condition

- Constraint for the function value upon a surface/volume boundary

$$u(x,y) = 0, \quad (x,y) \in \partial\Omega.$$

Neumann boundary condition

- Constraint for the function derivative value upon a surface/volume boundary

$$-\Delta u(x,y) = 1, \quad (x,y) \in \Omega,$$

03

PDE and Implicit Neural Network

PDE and Implicit Neural Network

01 PINN

Solve PDE for Physics

02 NeRF

Learn PDE for 3D Rendering

03 LIIF

Learn PDE for image super-resolution

04 VideoINR

Learn PDE for video manipulation

PDE and Implicit Neural Network

01 PINN

Solve PDE for Physics

02 NeRF

Learn PDE for 3D Rendering

03 LIIF

Learn PDE for image super-resolution

04 VideoINR

Learn PDE for video manipulation

*** "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations", JCP, 2019

PINN

Target Problem

Diffusion equation

$$\frac{\partial u}{\partial t} = \lambda \frac{\partial^2 u}{\partial x^2}$$

Subject to

$$\begin{aligned} -\Delta u(x,y) &= 1, \quad (x,y) \in \Omega, \\ u(x,y) &= 0, \quad (x,y) \in \partial\Omega. \end{aligned}$$

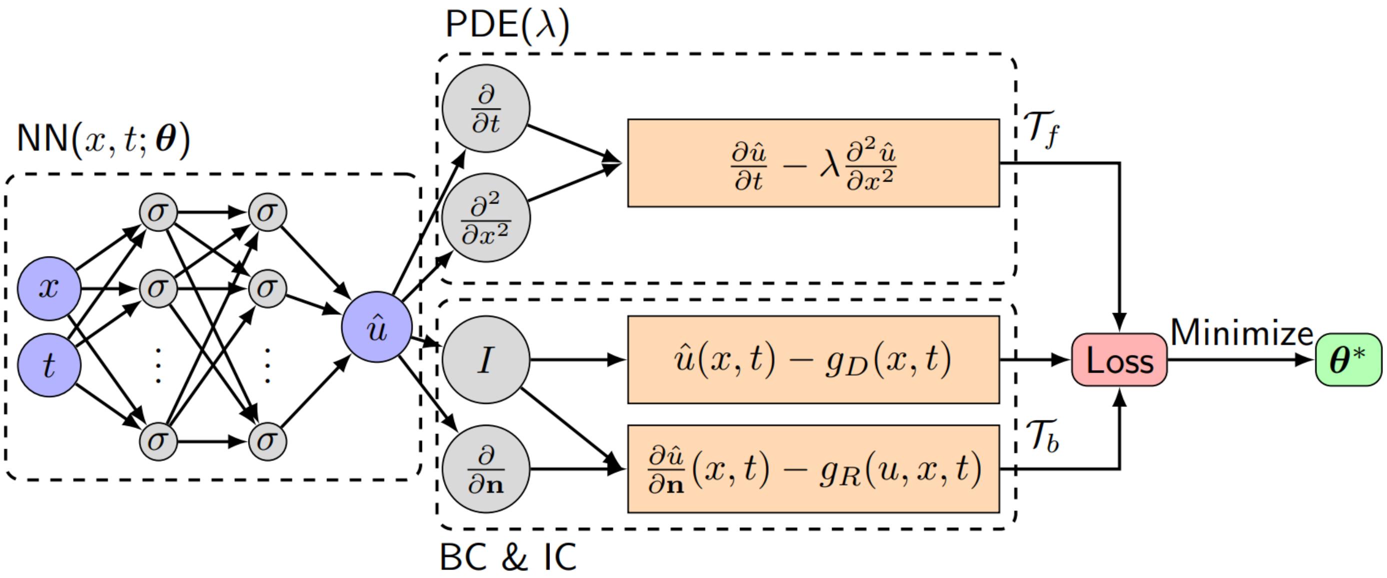
$$\Omega = [-1,1]^2 \setminus [0,1]^2$$

Name	Integral equations	Differential equations
Gauss's law	$\oint\!\!\!\oint_{\partial\Omega} \mathbf{E} \cdot \mathrm{d}\mathbf{S} = \frac{1}{\varepsilon_0} \iiint_{\Omega} \rho \, \mathrm{d}V$	$\nabla \cdot \mathbf{E} = \frac{\rho}{\varepsilon_0}$
Gauss's law for magnetism	$\oint\!\!\!\oint_{\partial\Omega} \mathbf{B} \cdot \mathrm{d}\mathbf{S} = 0$	$\nabla \cdot \mathbf{B} = 0$
Maxwell–Faraday equation (Faraday's law of induction)	$\oint_{\partial\Sigma} \mathbf{E} \cdot \mathrm{d}\boldsymbol{\ell} = -\frac{\mathrm{d}}{\mathrm{d}t} \iint_{\Sigma} \mathbf{B} \cdot \mathrm{d}\mathbf{S}$	$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}$
Ampère's circuital law (with Maxwell's addition)	$\oint_{\partial\Sigma} \mathbf{B} \cdot \mathrm{d}\boldsymbol{\ell} = \mu_0 \left(\iint_{\Sigma} \mathbf{J} \cdot \mathrm{d}\mathbf{S} + \varepsilon_0 \frac{\mathrm{d}}{\mathrm{d}t} \iint_{\Sigma} \mathbf{E} \cdot \mathrm{d}\mathbf{S} \right)$	$\nabla \times \mathbf{B} = \mu_0 \left(\mathbf{J} + \varepsilon_0 \frac{\partial \mathbf{E}}{\partial t} \right)$

PINN

Framework

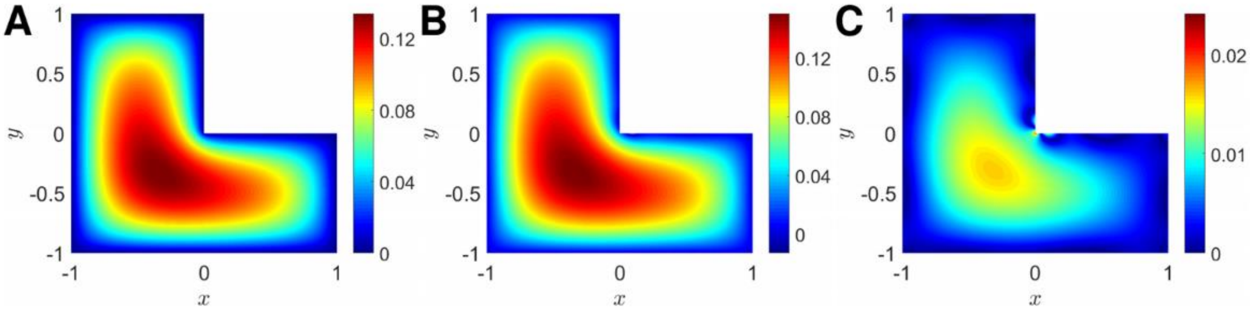
- Build the loss terms for the PDE and the constraints (Like penalty loss term)
- Remind the back-propagation algorithm for neural network!



PINN

Experimental Result

- Similar results to the conventional solution of Maxwell equation



PINN

PINN vs. FEM

- Constraint for the function value upon a surface/volume boundary

	PINN	FEM
Basis function	Neural network (nonlinear)	Piecewise polynomial (linear)
Parameters	Weights and biases	Point values
Training points	Scattered points (mesh-free)	Mesh points
PDE embedding	Loss function	Algebraic system
Parameter solver	Gradient-based optimizer	Linear solver
Errors	$\mathcal{E}_{\text{app}}$, $\mathcal{E}_{\text{gen}}$ and $\mathcal{E}_{\text{opt}}$ (subsection 2.4)	Approximation/quadrature errors
Error bounds	Not available yet	Partially available [14, 26]

$$\frac{f(x+h)-f(x-h)}{2h} = f'(x) + O(h^2).$$

$$\frac{f(x+h)-2f(x)+f(x-h)}{h^2} = f''(x) + O(h^2).$$

PDE and Implicit Neural Network

01 PINN

Solve PDE for Physics

02 NeRF

Learn PDE for 3D Rendering

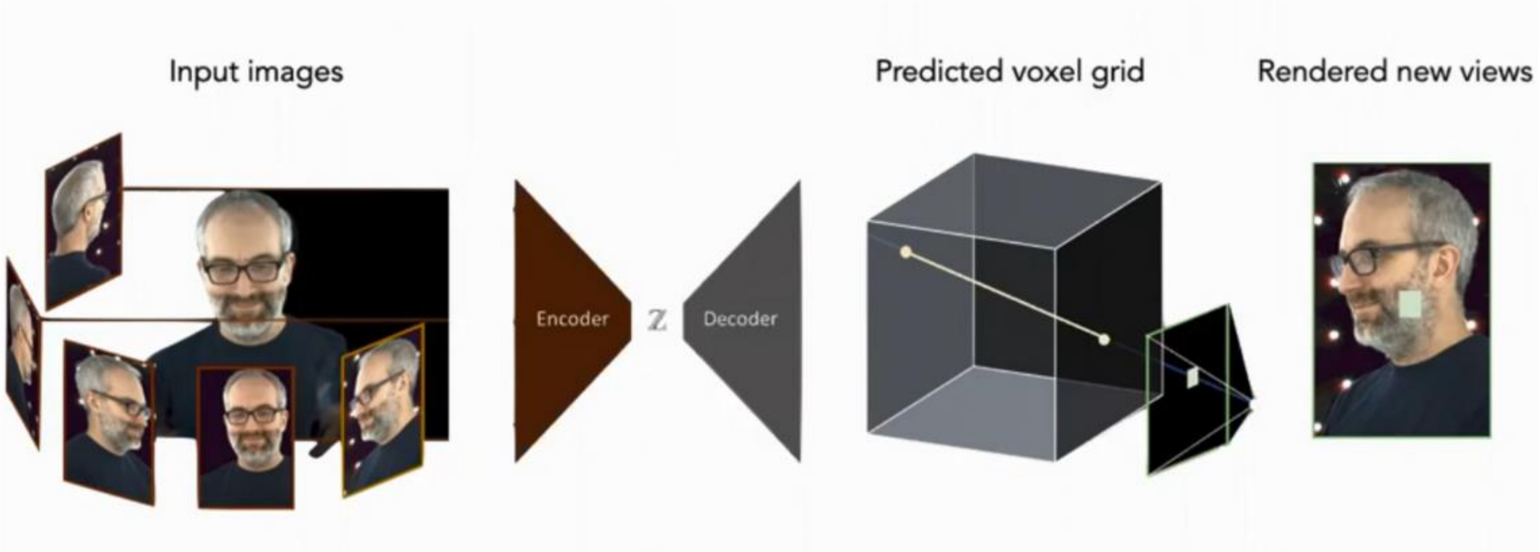
03 LIIF

Learn PDE for image super-resolution

04 VideoINR

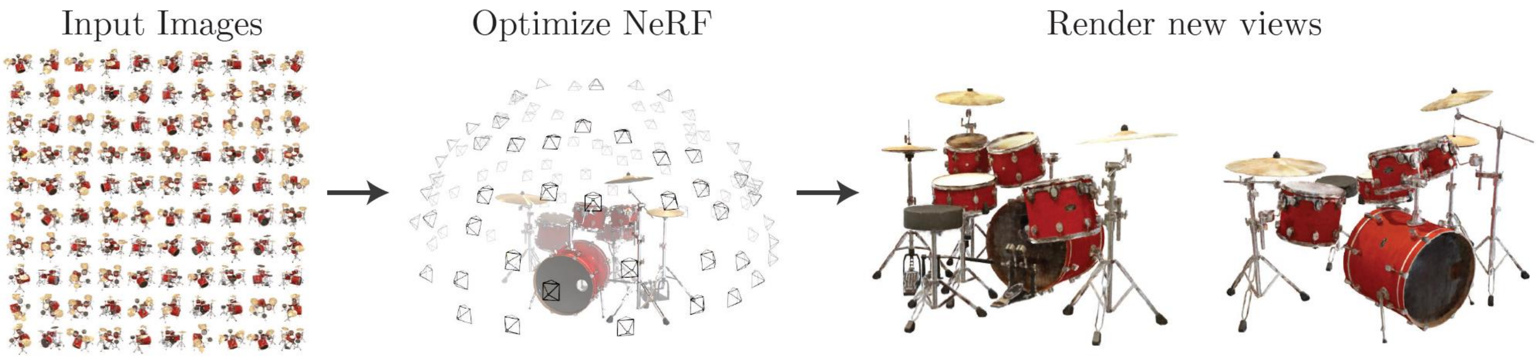
Learn PDE for video manipulation

Previous Studies – Synthetic new view



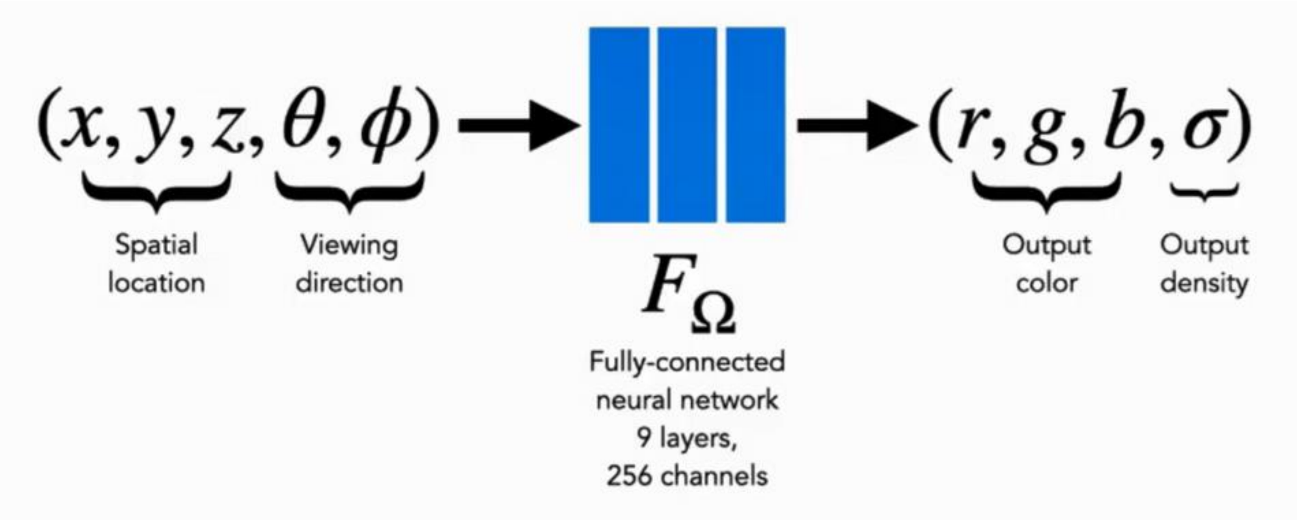
NeRF

Teaser



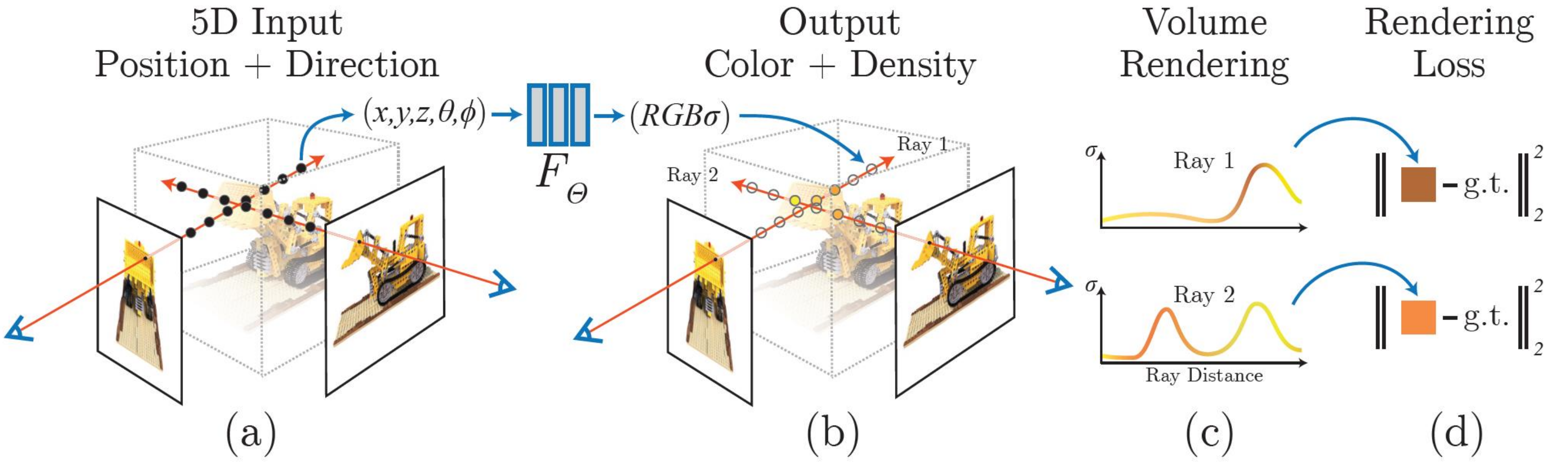
NeRF

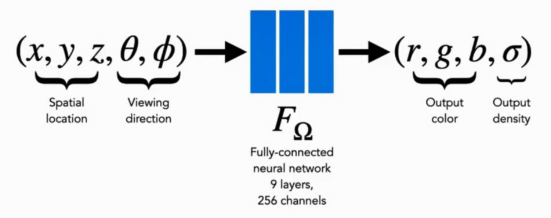
Target Problem



NeRF

Framework

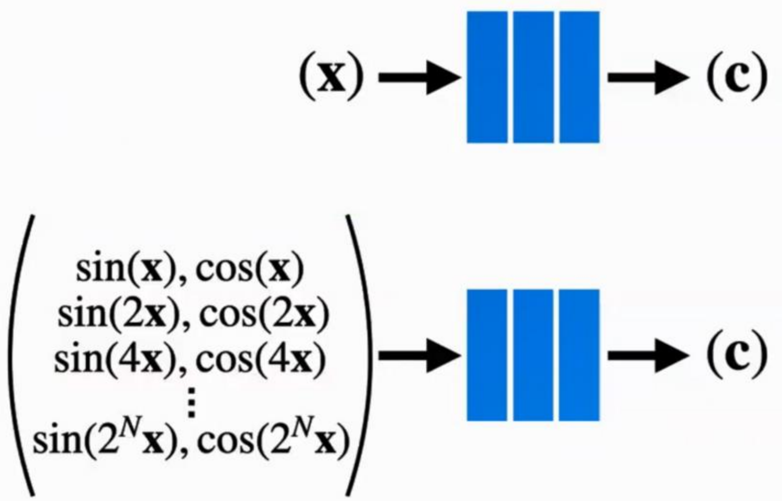


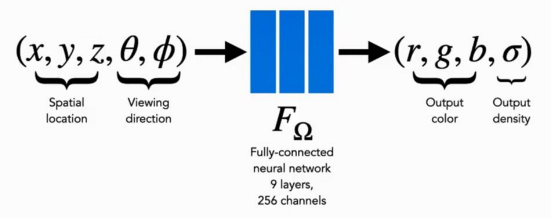


NeRF

Positional Encoding

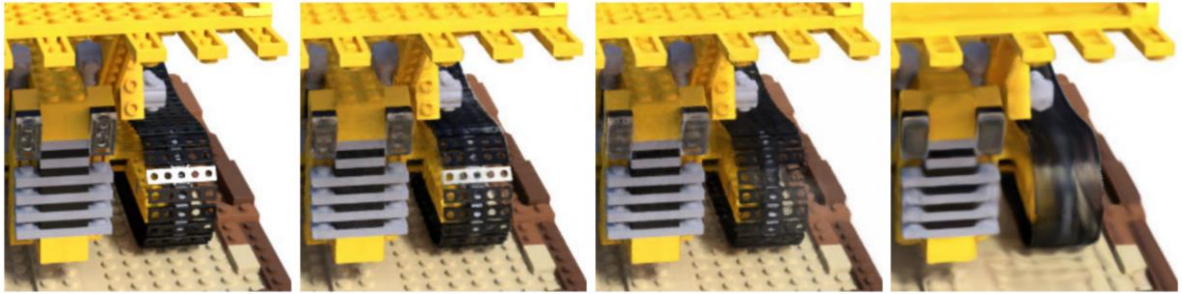
- Enlarge the information for the position values
- Effectively represent the high-frequency information (Like Fourier transform)





NeRF

Positional Encoding



Ground Truth | Complete Model | No View Dependence | No Positional Encoding

NeRF

Experimental Result

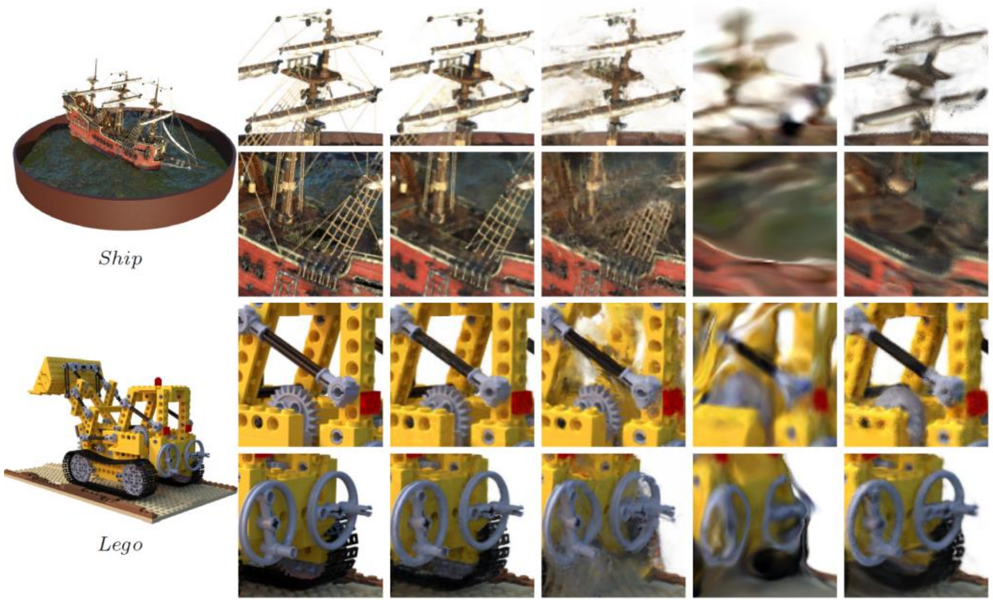
- Rendering quality – quantative comparison

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

NeRF

Experimental Result

- Qualitative comparison



NeRF

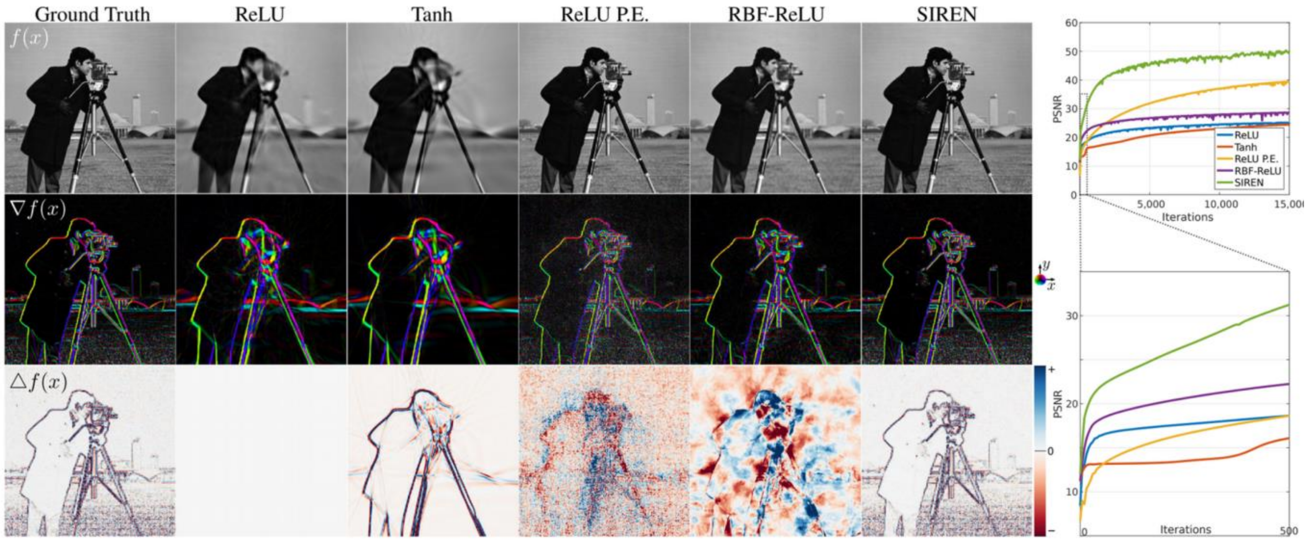
Experimental Result

- https://www.matthewtancik.com/nerf

NeRF – Improved

Implicit Neural Representations with Periodic Activation Functions

- SIREN – sine-function-based activation function instead of tanh and ReLU
- Much better performance than P.E.



PDE and Implicit Neural Network

01 **PINN**

Solve PDE for Physics

02 **NeRF**

Learn PDE for 3D Rendering

03 **LIIF**

Learn PDE for image super-resolution

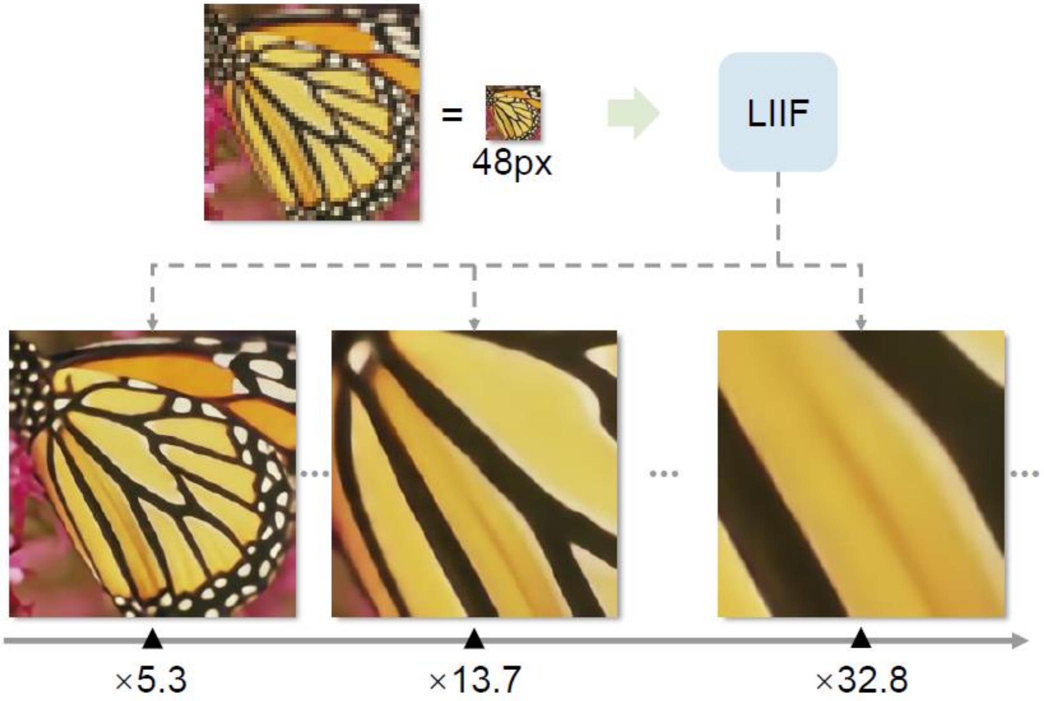
04 **VideoINR**

Learn PDE for video manipulation

*** "Learning Continuous Image Representation with Local Implicit Image Function", CVPR, 2021

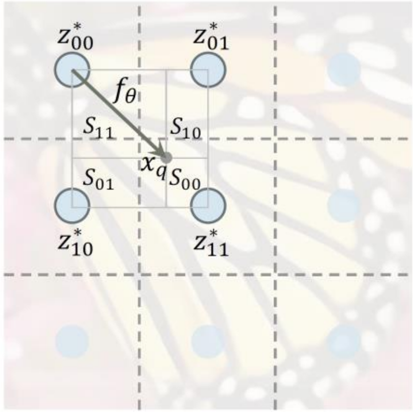
LIIF

Teaser



LIIF

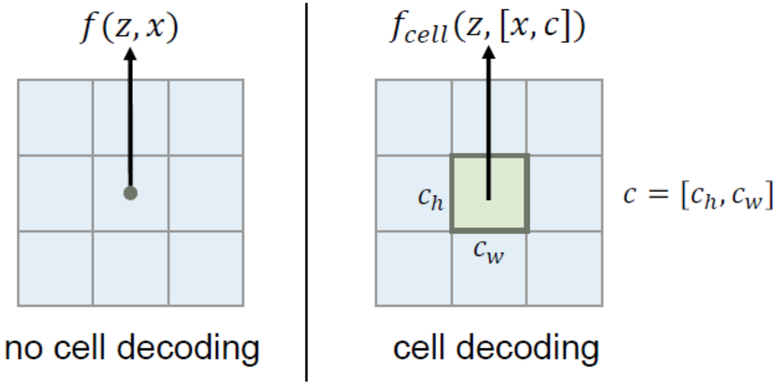
Target Problem

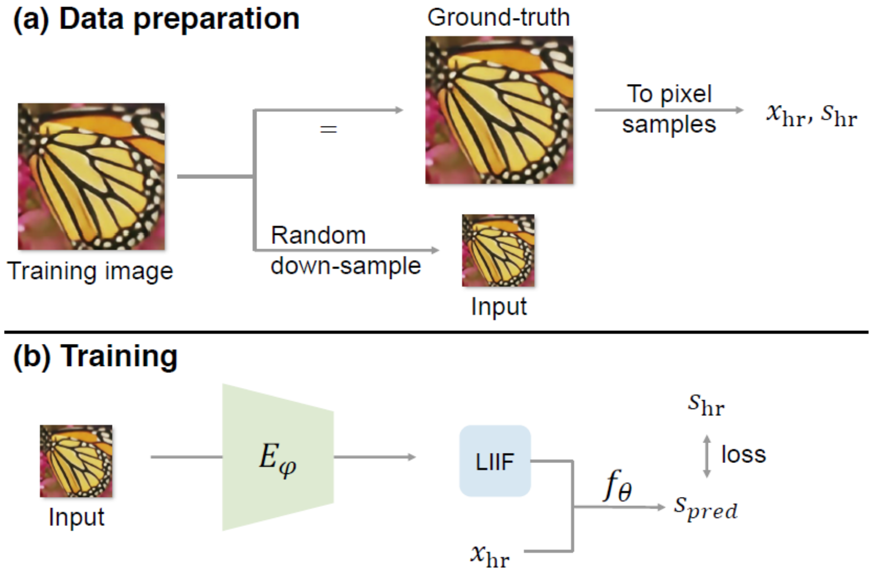


$$I^{(i)}(x_q) = f_\theta(z^*, x_q - v^*)$$

LIIF

Framework

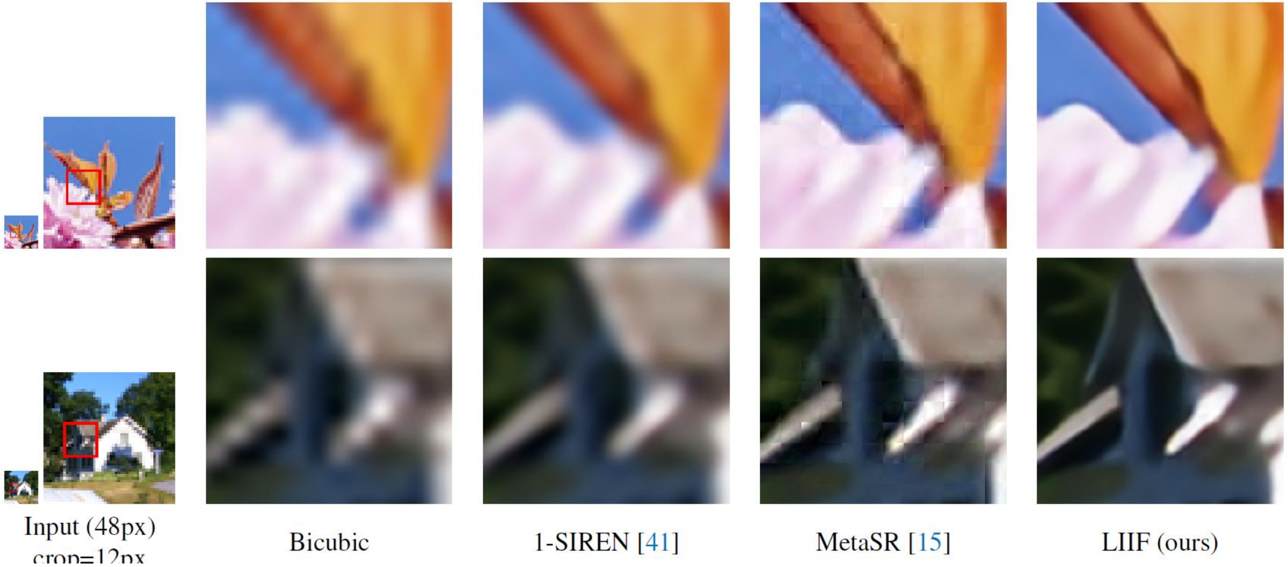




LIIF

Experimental Result

- Constraint for the function value upon a surface/volume boundary



LIIF

Experimental Result

- Constraint for the function value upon a surface/volume boundary

Method	In-distribution			Out-of-distribution				
	$\times 2$	$\times 3$	$\times 4$	$\times 6$	$\times 12$	$\times 18$	$\times 24$	$\times 30$
Bicubic [24]	31.01	28.22	26.66	24.82	22.27	21.00	20.19	19.59
EDSR-baseline [24]	34.55	30.90	28.94	-	-	-	-	-
EDSR-baseline-MetaSR$^\sharp$ [15]	34.64	30.93	28.92	26.61	23.55	22.03	21.06	20.37
EDSR-baseline-LIIF (ours)	34.67	30.96	**29.00**	**26.75**	**23.71**	**22.17**	**21.18**	**20.48**
RDN-MetaSR$^\sharp$ [15]	35.00	31.27	29.25	26.88	23.73	22.18	21.17	20.47
RDN-LIIF (ours)	34.99	31.26	29.27	**26.99**	**23.89**	**22.34**	**21.31**	**20.59**

Table 1: **Quantitative comparison on DIV2K validation set (PSNR (dB)).** $\sharp$ indicates ours implementation. The results that surpass others by 0.05 are bolded. EDSR-baseline trains different models for different scales. MetaSR and LIIF use one model for all scales, and are trained with continuous random scales uniformly sampled in $\times 1$–$\times 4$.

PDE and Implicit Neural Network

01 PINN

Solve PDE for Physics

02 NeRF

Learn PDE for 3D Rendering

03 LIIF

Learn PDE for image super-resolution

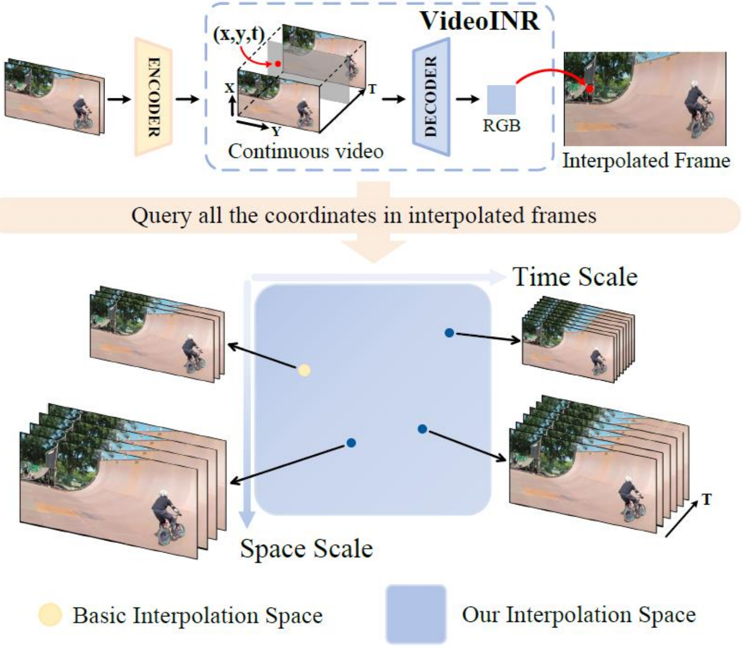
04 VideoINR

Learn PDE for video manipulation

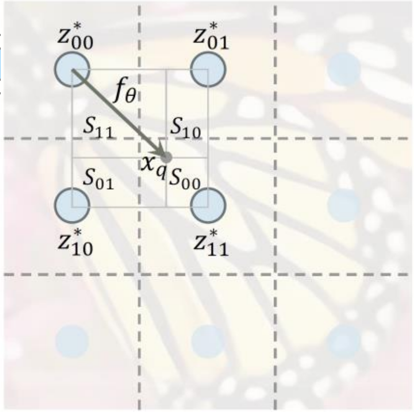
*** "VideoINR: Learning Video Implicit Neural Representation for Continuous Space-Time Super-Resolution", CVPR, 2022

VideoINR

Teaser



VideoINR



Target Problem

$$\mathcal{M}(x_s, x_t) = f_t(x_s, x_t, I_0, I_1)$$

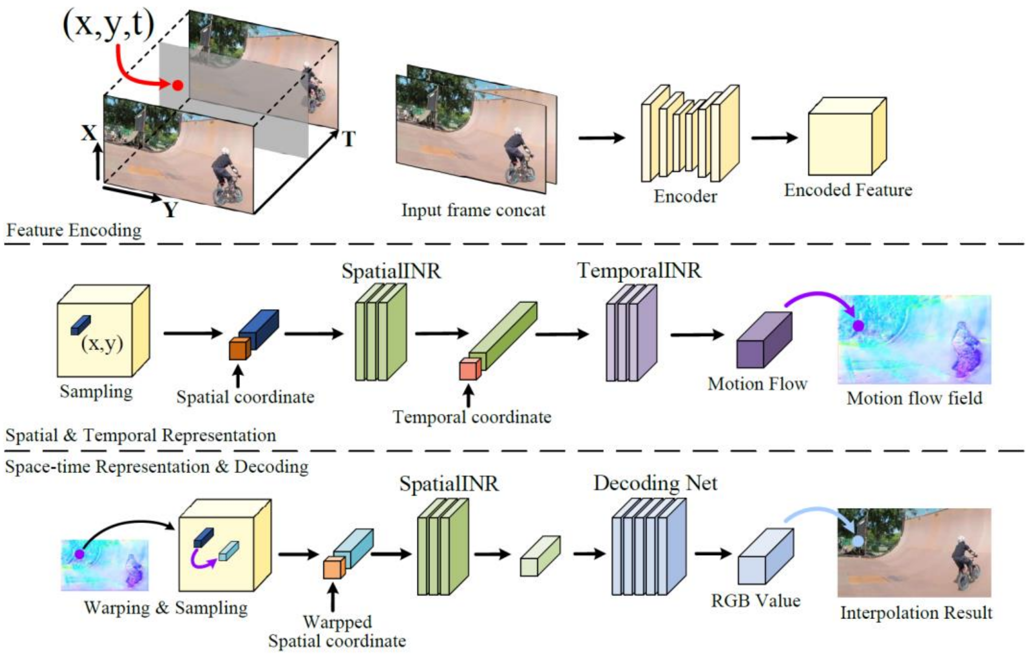
$$\mathcal{F}_{st}(x_s, x_t) = \mathcal{F}_s(x'_s) = \mathcal{F}_s(x_s + \mathcal{M}(x_s, x_t))$$

$$\mathcal{F}_s(x_s) = f_s(z^*, x_s - v^*)$$

$$\mathcal{M}(x_s, x_t) = f_t(x_t, \mathcal{F}_s(x_s))$$

VideoINR

Framework



VideoINR

Experimental Result

Table 1. **Quantitative comparison on benchmark datasets** including Vid4 [23], GoPro [29] and Adobe240 [41]. The best three results are highlighted in red, blue, and **bold**. We omit the results of Zooming SlowMo and VideoINR-*Fixed* on GoPro-*Average* and Adobe240-*Average* as the two models are trained for synthesizing frames only at fixed times.

VFI Method	SR Method	Vid4 PSNR	Vid4 SSIM	GoPro-*Center* PSNR	GoPro-*Center* SSIM	GoPro-*Average* PSNR	GoPro-*Average* SSIM	Adobe-*Center* PSNR	Adobe-*Center* SSIM	Adobe-*Average* PSNR	Adobe-*Average* SSIM	Parameters (Million)
SuperSloMo [18]	Bicubic	22.42	0.5645	27.04	0.7937	26.06	0.7720	26.09	0.7435	25.29	0.7279	19.8
SuperSloMo [18]	EDVR [45]	23.01	0.6136	28.24	0.8322	26.30	0.7960	27.25	0.7972	25.95	0.7682	19.8+20.7
SuperSloMo [18]	BasicVSR [6]	23.17	0.6159	28.23	0.8308	26.36	**0.7977**	27.28	0.7961	25.94	0.7679	19.8+6.3
QVI [18]	Bicubic	22.11	0.5498	26.50	0.7791	25.41	0.7554	25.57	0.7324	24.72	0.7114	29.2
QVI [18]	EDVR [45]	23.60	0.6471	27.43	0.8081	25.55	0.7739	26.40	0.7692	25.09	0.7406	29.2+20.7
QVI [18]	BasicVSR [6]	23.15	0.6428	27.44	0.8070	26.27	0.7955	26.43	0.7682	25.20	0.7421	29.2+6.3
DAIN [2]	Bicubic	22.57	0.5732	26.92	0.7911	26.11	0.7740	26.01	0.7461	25.40	0.7321	24.0
DAIN [2]	EDVR [45]	23.48	0.6547	28.01	0.8239	26.37	0.7964	27.06	0.7895	26.01	0.7703	24.0+20.7
DAIN [2]	BasicVSR [6]	23.43	0.6514	28.00	0.8227	**26.46**	0.7966	27.07	0.7890	**26.23**	**0.7725**	24.0+6.3
Zooming SlowMo [47]		**25.72**	**0.7717**	30.69	0.8847	-	-	30.26	0.8821	-	-	11.10
TMNet [48]		25.96	0.7803	30.14	0.8692	28.83	0.8514	29.41	0.8524	28.30	0.8354	**12.26**
VideoINR-*fixed*		25.78	0.7730	30.73	0.8850	-	-	30.21	0.8805	-	-	11.31
VideoINR		25.61	0.7709	**30.26**	**0.8792**	29.41	0.8669	**29.92**	**0.8746**	29.27	0.8651	11.31

VideoINR

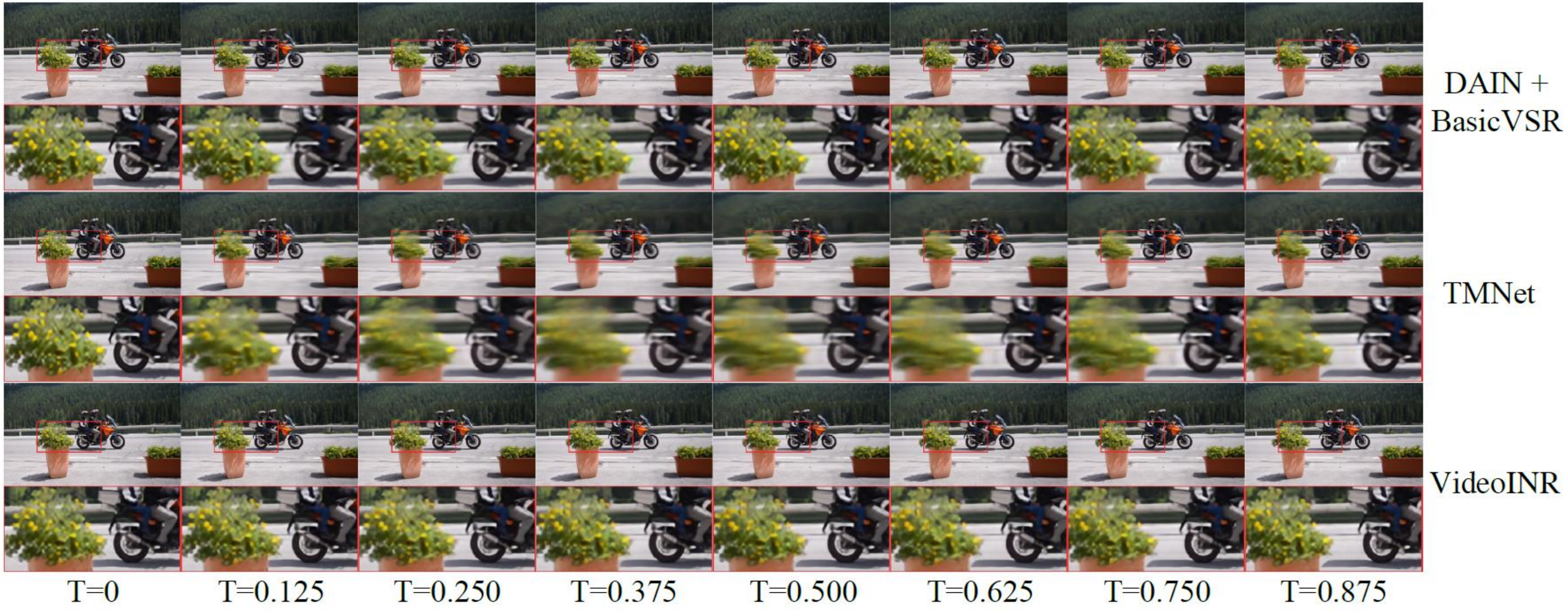
Experimental Result

Table 2. **Quantitative comparison for out-of-distribution scales** on GoPro dataset. Model performances are evaluated by PSNR and SSIM. Some results of TMNet are bolded as it does not support generalizing to out-of-training-distribution space scales.

Time Scale	Space Scale	SuperSloMo [18] + LIIF [7]	DAIN [2] + LIIF [7]	TMNet [48]	VideoINR
$\times 6$	$\times 4$	26.70 / 0.7988	26.71 / 0.7998	30.49 / 0.8861	**30.78 / 0.8954**
$\times 6$	$\times 6$	23.47 / 0.6931	23.36 / 0.6902	-	**25.56 / 0.7671**
$\times 6$	$\times 12$	21.92 / 0.6495	22.01 / 0.6499	-	**24.02 / 0.6900**
$\times 12$	$\times 4$	25.07 / 0.7491	25.14 / 0.7497	26.38 / 0.7931	**27.32 / 0.8141**
$\times 12$	$\times 6$	22.91 / 0.6783	22.92 / 0.6785	-	**24.68 / 0.7358**
$\times 12$	$\times 12$	21.61 / 0.6457	21.78 / 0.6473	-	**23.70 / 0.6830**
$\times 16$	$\times 4$	24.42 / 0.7296	24.20 / 0.7244	24.72 / 0.7526	**25.81 / 0.7739**
$\times 16$	$\times 6$	23.28 / 0.6883	22.80 / 0.6722	-	**23.86 / 0.7123**
$\times 16$	$\times 12$	21.80 / 0.6481	22.22 / 0.6420	-	**22.88 / 0.6659**

VideoINR

Experimental Result

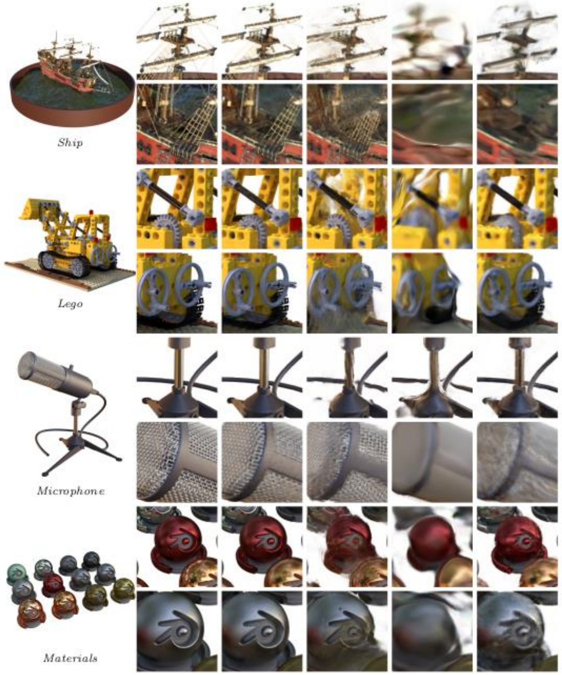


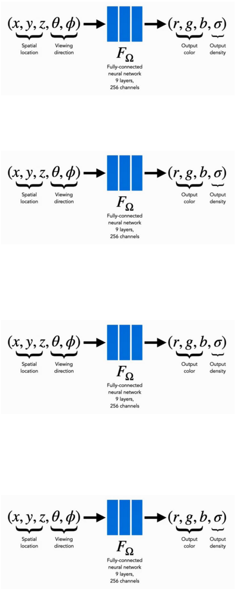
04

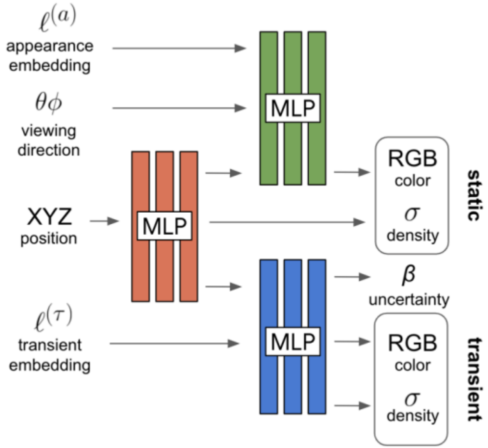
Limitation

Limitation

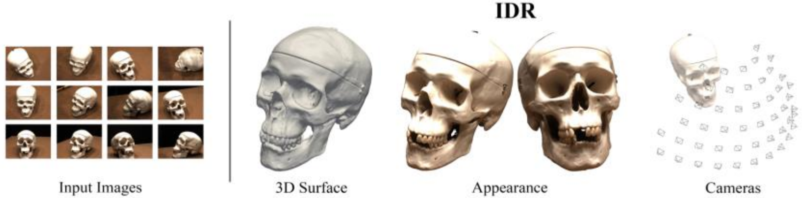
Repetitive training for each sample







“NeRF in the Wild”, CVPR2021



“Disentangling Geometry and Appearance”, NIPS2020

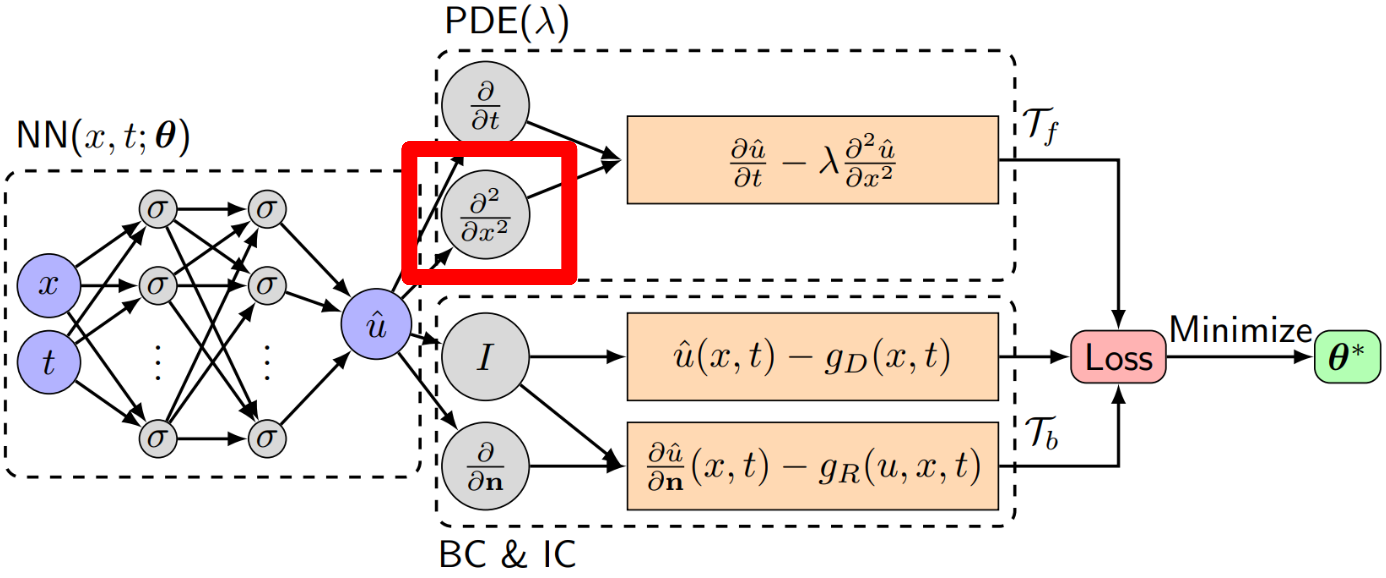
Limitation

No bound for detected error

	PINN	FEM
Basis function	Neural network (nonlinear)	Piecewise polynomial (linear)
Parameters	Weights and biases	Point values
Training points	Scattered points (mesh-free)	Mesh points
PDE embedding	Loss function	Algebraic system
Parameter solver	Gradient-based optimizer	Linear solver
Errors	$\mathcal{E}_{\text{app}}$, $\mathcal{E}_{\text{gen}}$ and $\mathcal{E}_{\text{opt}}$ (subsection 2.4)	Approximation/quadrature errors
Error bounds	Not available yet	Partially available [14, 26]

Limitation

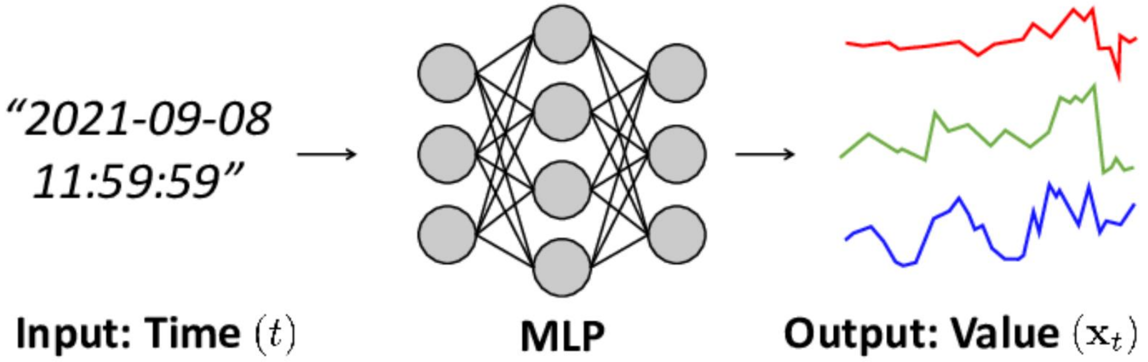
Sometimes, large computation



Implicit Neural Network!

Definition

- Train the target signal itself using the sampled signal values
- Assume that the non-linearity between the sampled values can be generalized by neural network
- Or, reduce the discontinuity through the additional regularization term or the boundary conditions
- Very simple to implement a network when we can design a partial difference equation for our tasks



Thanks!

Do you have any questions?

choijw@cau.ac.kr
Jongwon Choi