Learning time-dependent PDE via graph neural networks and deep operator network for robust accuracy on irregular grids

Sung Woong Cho

Assistant Professor,
Department of Data Science,
Inha University

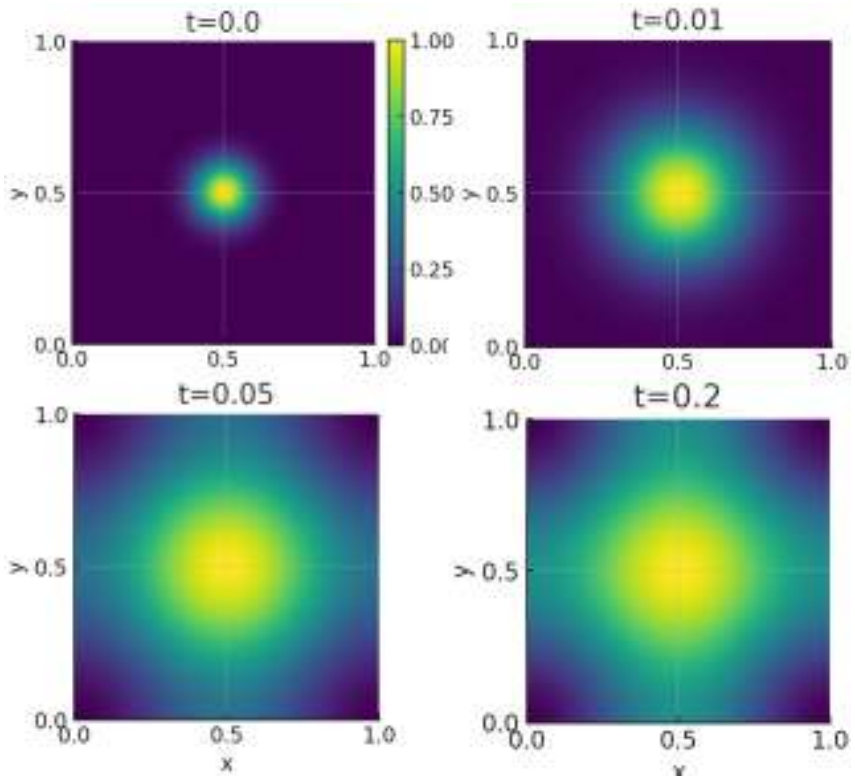
Jan 6, 2026

Outline

Outline

Motivation for PDEs

Partial Differential Equations(PDEs) are useful for modeling and predicting the dynamics.



2d Heat equation

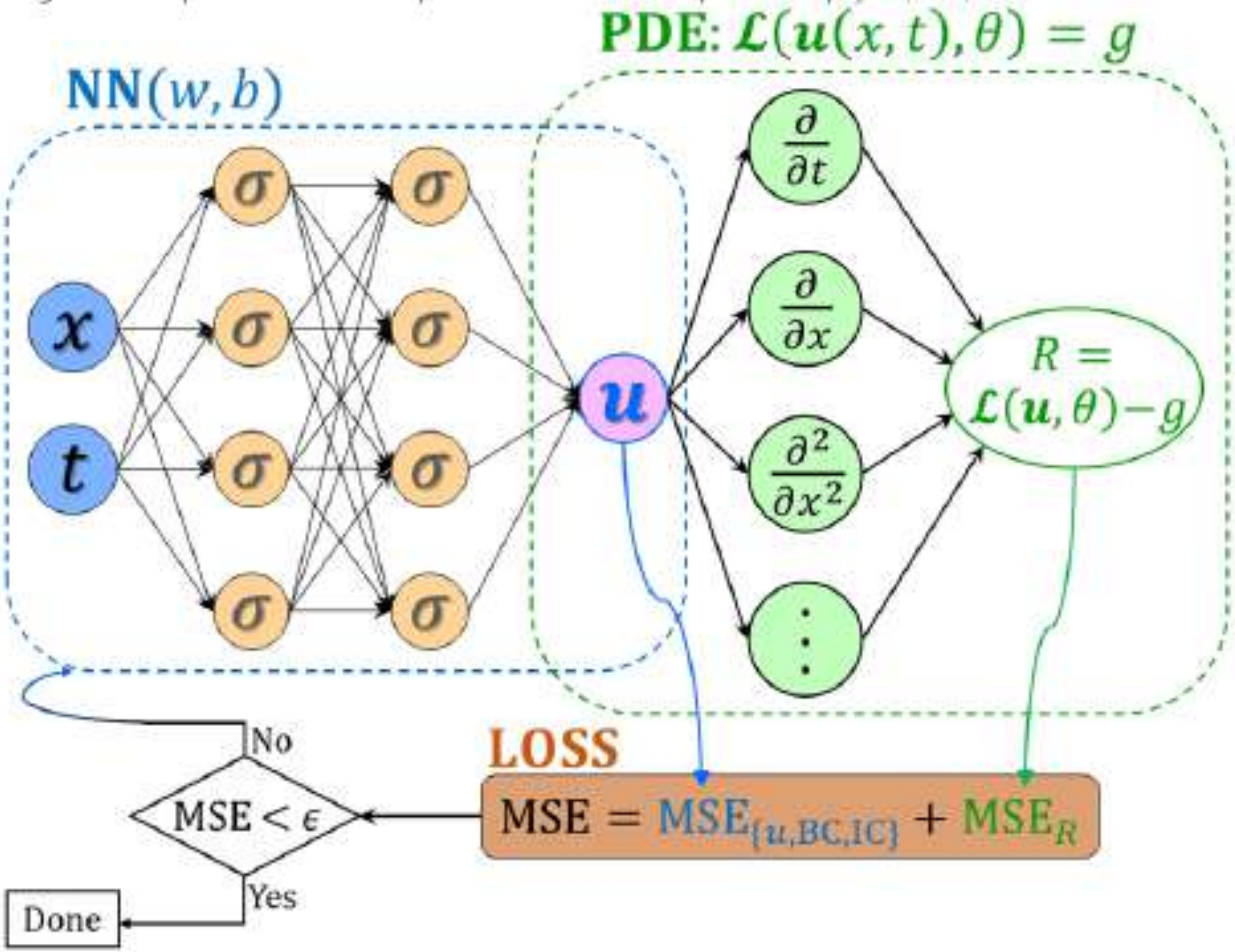
$$\frac{\partial u}{\partial t} - \alpha \nabla^2 u = 0, \quad \text{for } \mathbf{x} \in \Omega \text{ and } t > 0,$$
$$u(\mathbf{x}, t) = g(\mathbf{x}, t), \quad \text{for } \mathbf{x} \in \partial\Omega \text{ and } t > 0,$$
$$u(\mathbf{x}, 0) = f(\mathbf{x}), \quad \text{for } \mathbf{x} \in \Omega.$$

Q) How do we approximate the solutions when we cannot solve them directly?

Physics Informed Neural Networks(PINNs)

Neural networks could provide an approximation after sufficient training with proper loss functions.

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, *378*, 686-707.



PINNs for Burgers' equation

$$u(x,0) = u_0(x), x \in [-1,1]$$
$$u(-1,t) = g_1(t), u(1,t) = g_2(t) \text{ for } t \in [0,1]$$

The residual of the Burgers' equation, represented by the neural network $\hat{u}$, is denoted by $f(x,t;\theta)$:

$$f(x,t;\theta) = \frac{\partial \hat{u}}{\partial t} + \hat{u}\frac{\partial \hat{u}}{\partial x} - \nu\frac{\partial^2 \hat{u}}{\partial x^2}$$

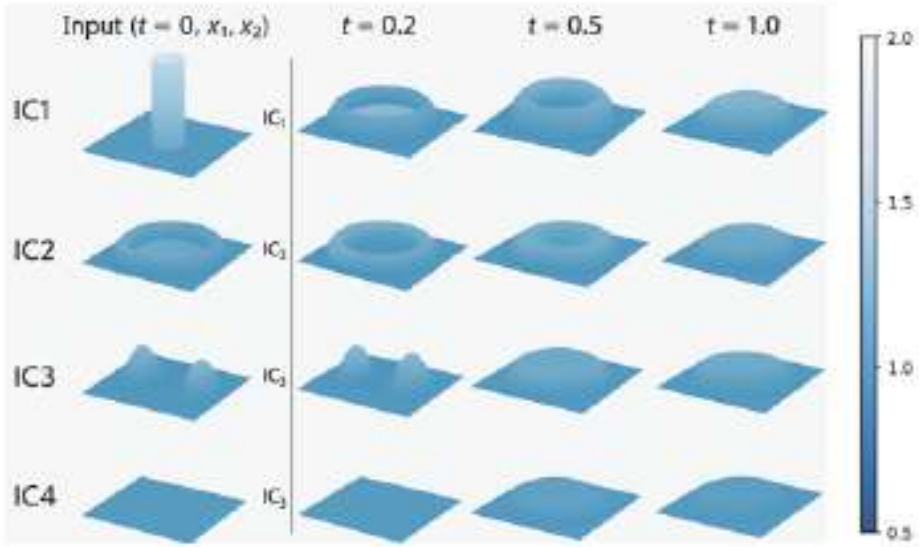
We apporximate solution u by $\hat{u}$ by minimizing the loss:

$$\mathcal{L}(\theta) = \underbrace{\frac{1}{N_B}\sum_{i=1}^{N_B}\left(\left|\hat{u}(-1,t_i^B;\theta) - g_1(t_i^B)\right|^2 + \left|\hat{u}(1,t_i^B;\theta) - g_2(t_i^B)\right|^2\right)}_{\text{Boundary condition term}}$$
$$+ \underbrace{\frac{1}{N_I}\sum_{i=1}^{N_I}\left|\hat{u}(x_i^I,0;\theta) - u_0(x_i^I)\right|^2}_{\text{Initial condition term}} + \underbrace{\frac{1}{N_P}\sum_{i=1}^{N_P}\left|f(x_i^P,t_i^P;\theta)\right|^2}_{\text{PDE residual term}}$$

Operator Learning

We aim to quickly predict solutions for arbitrary initial condition through trained neural networks without retraining. (PINNs need retraining)

- For example, based on the height of water at an initial time $t = 0$, we need a model to predict the solution behavior very fast.



- In this problem, a neural network acts as an operator that maps a **function** to corresponding **function**.

Find a map $\mathcal{G} : g(x) \mapsto u(t, x)$,

$$\mathcal{L}_{PDE} = f(u, u_t, u_x, u_{xx}, \ldots) = 0$$

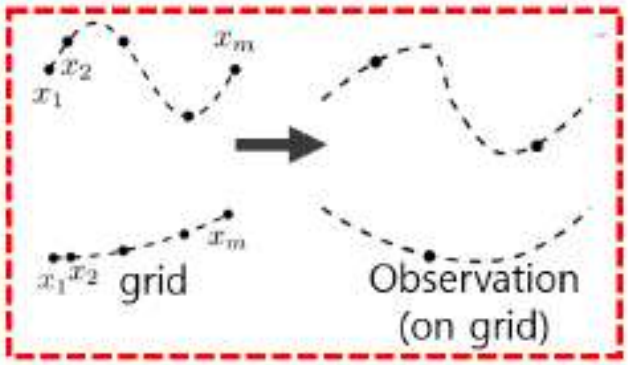
$$\mathcal{L}_{IC} = u(0, x) - g(x) = 0$$

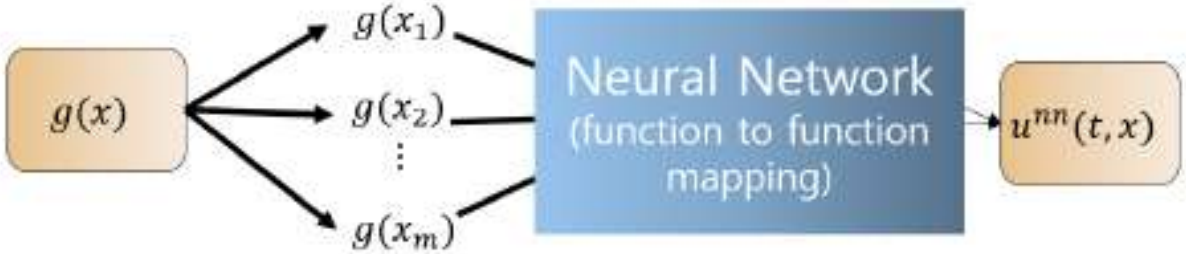
$$\mathcal{L}_{BC} = u|_{\partial\Omega} - h(t, x)|_{\partial\Omega}$$

Handling irregular grid points

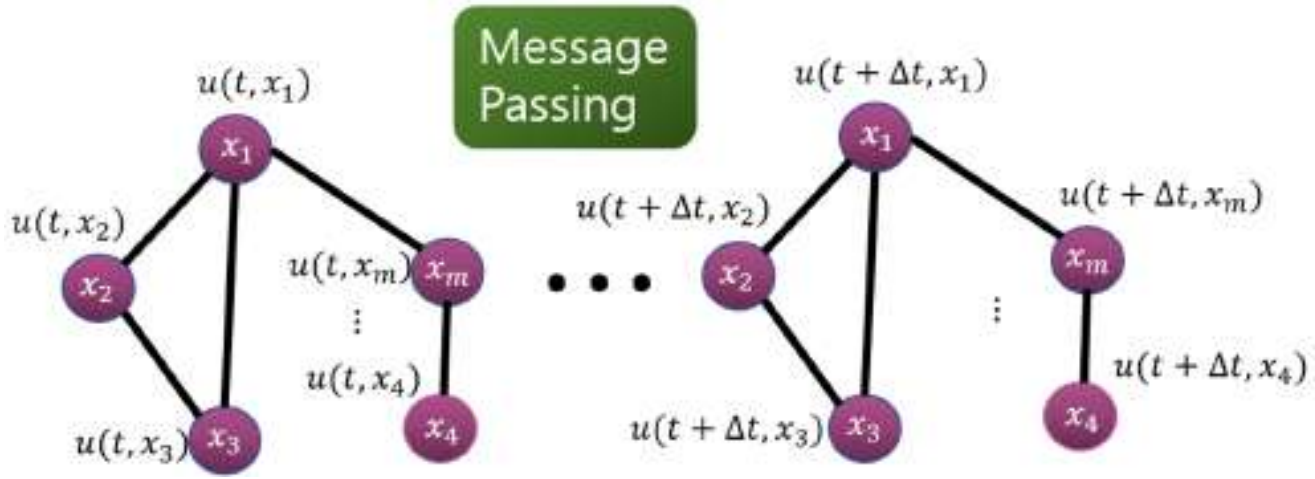
Operator learning usually depend on the observations at some grid.

- Previous methods relied on a regular grid.





On the one hand, Graph Neural Networks can consider arbitrary grids.



Outline

Deep Operator Network (DeepONet)

Deep Operator Network (DeepONet) uses two neural networks: Branch net and Trunk net. (Mapping $u(x) \mapsto \mathcal{G}(u)(y)$)

- Branch net takes **finite** observations of the input function $u(x)$.
- Trunk net takes the point y in the output function's domain to form **basis** functions.

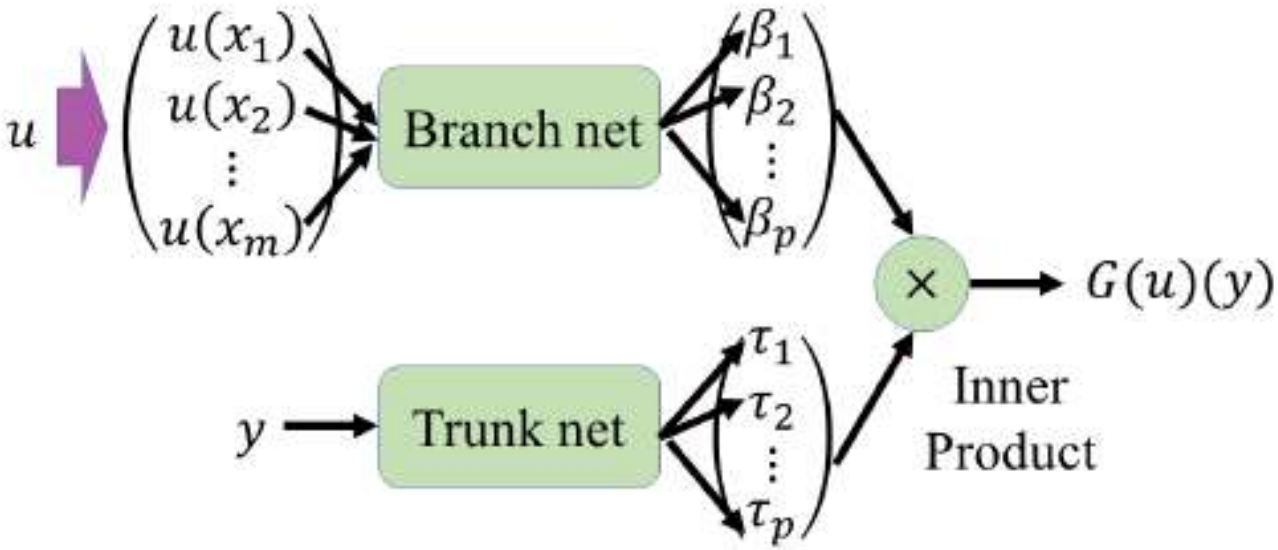


Figure: The structure of DeepONet

Loss function for operator learning (Mapping $u(x) \mapsto \mathcal{G}(u)(y)$

- Suppose we have (**real** or **numerical**) observations $\{u_i, \mathcal{G}(u_i)\}_{i=1}^N$ where $u_i \in \mathcal{U}$ and $\mathcal{G}(u_i) \in \mathcal{S}$.
- We aim to find an approximation $\mathcal{G}_\theta : \mathcal{U} \to \mathcal{S}$ with parameter θ using the N observations so that $\mathcal{G}_\theta \approx \mathcal{G}$.

$$\min \sum \|\mathcal{G}_\theta(u_i) - \mathcal{G}(u_i)\|_{L^2(\mathcal{Y})}$$

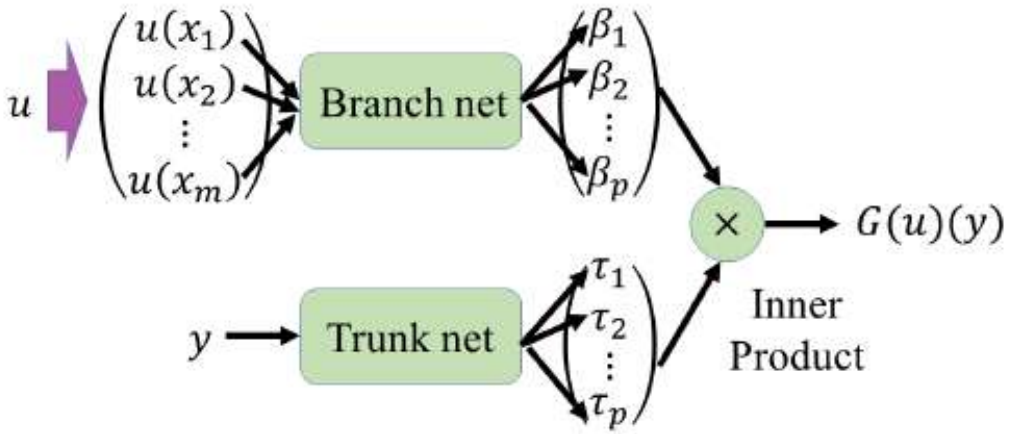


Figure: The structure of DeepONet

Q) Is an MLP good architecture for Branch net when we deal with irregular grid points?

Outline

Operator Learning based on graph neural networks

Recently, graph neural networks have been applied to handle irregular grids.

- Each grid point is represented as a node (point) in the graph.
- Note that the edges are **not** automatically defined.

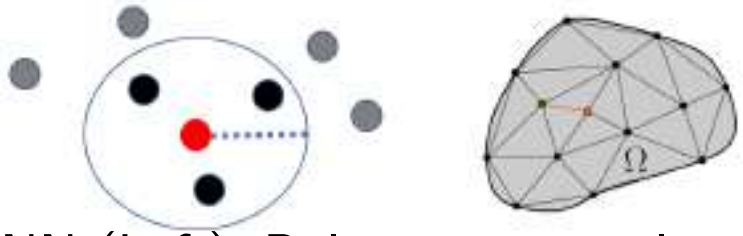
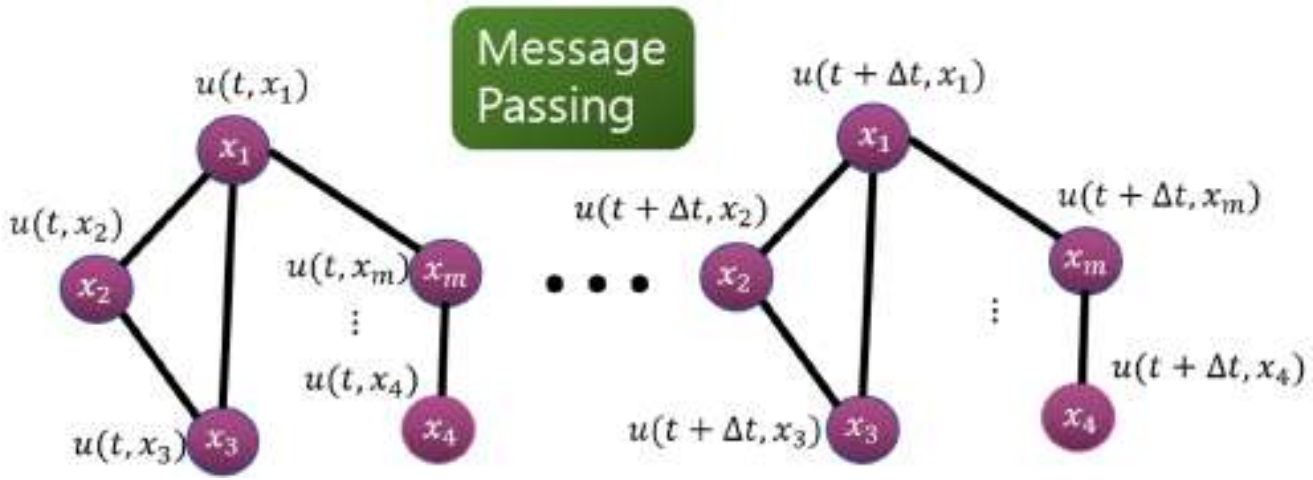


Figure: KNN (Left), Delaunay triangulation (Right)

Graph Neural Network predict the solution over time by updating the node value.



Challenging example for graph neural networks (MP-PDE & MAgNet

Previous graph neural networks may **not work** for irregular grid points.

(Ex) A transport equation on $\mathbb{T}^2 = [0,1]^2$ with the constant velocity vector $\boldsymbol{v} = (1,0)$.

$$\frac{\partial u}{\partial t} + \nabla \cdot (\boldsymbol{v} u) = 0.$$

We aim to approximate two operators $u(0, \boldsymbol{x}) \mapsto u(1/2, \boldsymbol{x}), u(1, \boldsymbol{x})$.

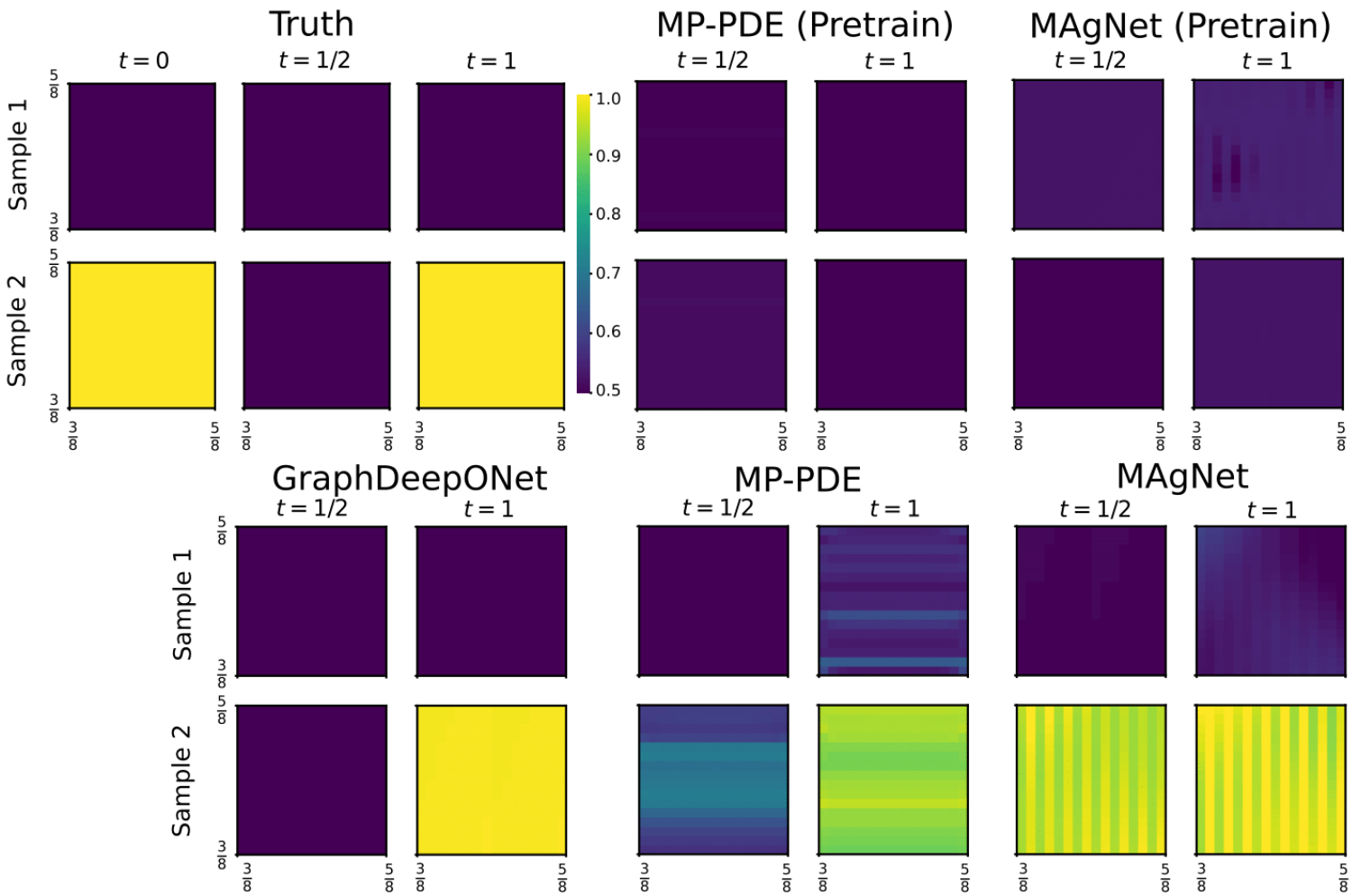
f_1: a zero function, $f_2(\boldsymbol{x}) = 1$ for $\boldsymbol{x} \in A$ and $f_2(\boldsymbol{x}) = 1/2$ for $\boldsymbol{x} \in B$, where

- $A = [3/8, 5/8]^2$,
- $B = \mathbb{T}^2 \cap A^C$.

We **assume** that the grid point lies in A.

Challenging example for graph neural networks (MP-PDE & MAgNet

In this case, MP-PDE and MAgNet have a difficulty in multiple predictions.



- At $t = 0.5$, the two states cannot be **distinguished**.

Challenging example for graph nerual networks

Graph neural networks were developed for **irregular** grids.

- However, it may not be effective when used in the previous model.

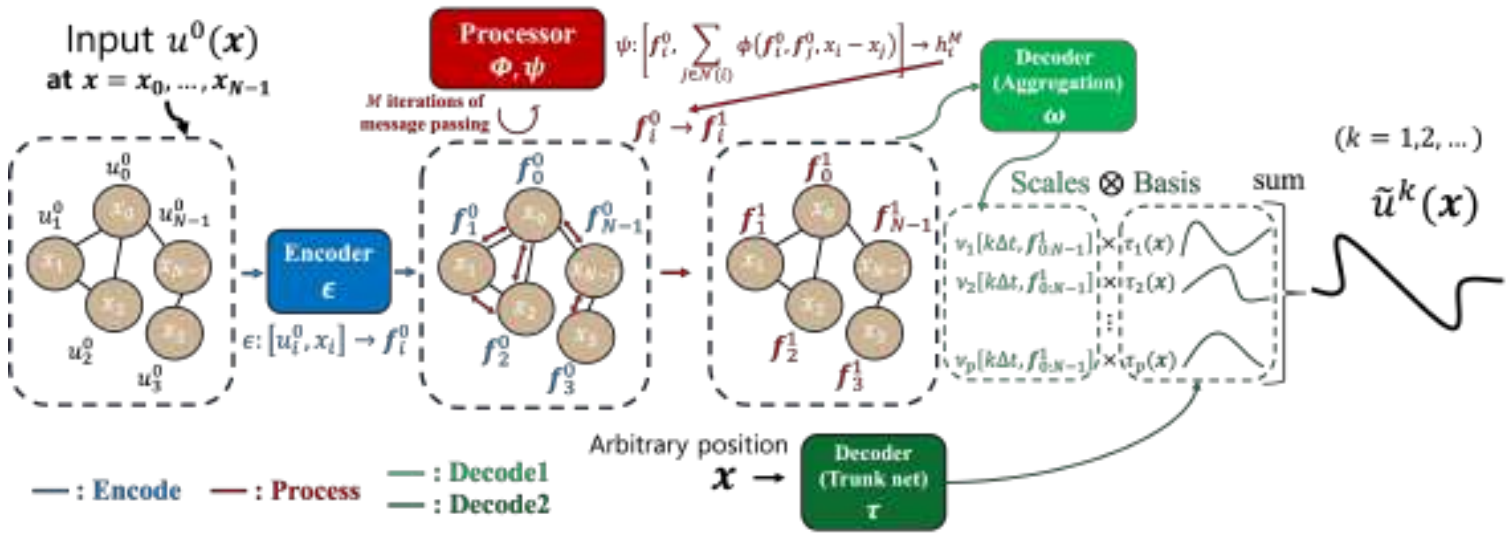
Theorem

(Failure to learn operator using other graph-based models) *Assume that the sensor points* $\{\boldsymbol{x}_i\}_{i=0}^{N-1}$ *are independently selected following a uniform distribution on* $\mathbb{T}^d$*. Then, there exist Lipshitz continuous operators* $\mathcal{G}^{(1)}, \mathcal{G}^{(2)} : H^s(\mathbb{T}^d) \to H^s(\mathbb{T}^d)$ *such that the following inequality holds with a non-zero probability* $\delta > 0$*:*

$$\sum_{k=1}^{2} \left\| \mathcal{G}^{(k)}(\bar{u}^0) - \mathcal{G}_{graph}(\bar{u}^0, k\Delta t) \right\|_{L^2(\mu)} \geq 1/2.$$

Our Proposed Structure

Now, we propose a structure which combines graph neural networks with deep operator network.



- Encoder (takes finite observations)
- Processor (updates the value for each node)
- First Decoder (aggregates all node values to produce coefficients)
- Second Decoder (assigns coefficient to its respective basis function.)

Theoretical Results - Proposed Method

Our proposed model (GDON) is indeed proper for handling **any irregular** grid points in a periodic domain with **appropriate** assumptions.

- $\mathcal{G} : H^s(\mathbb{T}^d) \to H^s(\mathbb{T}^d)$ is an operator.
- μ: probability measure on $H^s(\mathbb{T}^d)$.

Theorem

Suppose that grid points $\boldsymbol{x}_i$ $(1 \leq i \leq m)$ are independently selected based on a uniform distribution over $\mathbb{T}^d$. If m is sufficiently large, then there exists a GraphDeepONet $\mathcal{G}_{GDON} : (\mathbb{R} \cup \mathbb{T}^d)^m \to H^s(\mathbb{T}^d)$ can approximate $\mathcal{G}$ well with probability 1.

$$\sum_{k=1}^{K_{frame}} \left\| \mathcal{G}^{(k)}(u^0) - \mathcal{G}_{GDON}(\bar{u}^0, k\Delta t) \right\|_{L^2(\mu)} \leq C \left(\sum_{j > N/C\log(N)} \lambda_j \right)^{1/2} + Cp^{-s/d},$$

for every $p \in \mathbb{N}$, where the constant $C > 0$ is a function of N, K_{frame}, $\mathcal{G}$, and μ.

Outline

Experimental Results (Equations)

We consider the following three equations with multiple initial conditions.

Burgers type PDEs:

$$\partial_t u + \partial_x(\alpha u^2 - \beta \partial_x u + \gamma \partial_{xx} u) = \delta(t,x), \quad t \in T = [0,4],\ x \in \Omega = [0,16),$$
$$u(0,x) = \delta(0,x). \quad x \in \Omega,$$

The shallow water equations:

$$\begin{cases} \frac{\partial h}{\partial t} + \frac{\partial}{\partial x}(hu) + \frac{\partial}{\partial y}(hv) = 0, \\ \frac{\partial (hu)}{\partial t} + \frac{\partial}{\partial x}(u^2 h + \frac{1}{2} g h^2) + \frac{\partial}{\partial y}(huv) = 0, \\ \frac{\partial (hv)}{\partial t} + \frac{\partial}{\partial y}(v^2 h + \frac{1}{2} g h^2) + \frac{\partial}{\partial x}(huv) = 0, \\ h(0, x_1, x_2) = h_0(x_1, x_2). \end{cases}$$

Incompressible N-S equation:

$$\begin{cases} \frac{\partial w}{\partial t} + u \cdot \nabla w - \nu \Delta w = f, & (t, \boldsymbol{x}) \in [0, T] \times (0,1)^2, \\ \nabla \cdot u = 0, & (t, \boldsymbol{x}) \in [0, T] \times (0,1)^2, \\ w(0, \boldsymbol{x}) = w_0(\boldsymbol{x}), & \boldsymbol{x} \in (0,1)^2. \end{cases}$$

Experimental Results (The shallow water equations)

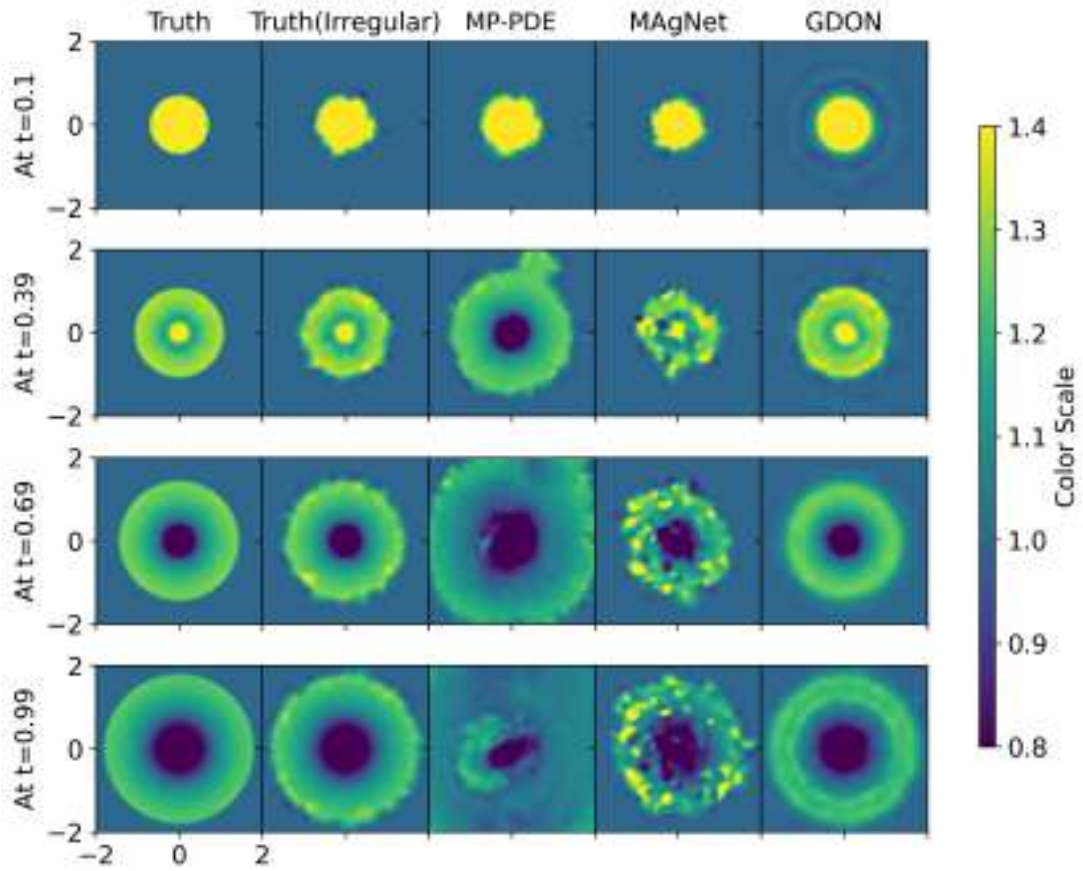
Our method showcases robust accuracy regardless of the grid points.

Table: Mean relative L^2 test errors.

2D Shallow Water Equation						
Type of Sensor Points	FNO-2D	F-FNO	GINO	MP-PDE	MAgNet	GraphDeepONet (Ours)
Regular	0.0051 ±0.0024	0.0033 ±0.0013	0.0127 ±0.0044	**0.0015** ±0.0006	0.0073 ±0.0014	0.0094 ±0.0027
Irregular I	-	0.0483 ±0.0014	0.0188 ±0.0019	0.1693 ±0.0338	0.0949 ±0.0635	**0.0137** ±0.0078
Irregular II	-	0.0494 ±0.0012	0.0185 ±0.0031	0.1698 ±0.0395	0.0899 ±0.0599	**0.0148** ±0.0121
Irregular III	-	0.0478 ±0.0018	0.0174 ±0.0017	0.0155 ±0.0023	0.0709 ±0.0184	**0.0140** ±0.0086
Average across all data types	-	0.0372 ±0.0205	0.0169 ±0.0036	0.0890 ±0.0871	0.0657 ±0.0527	**0.0130** ±0.0076
2D N-S Equation						
Type of Sensor Points	FNO-2D	F-FNO	GINO	MP-PDE	MAgNet	GraphDeepONet (Ours)
Regular	**0.0351** ±0.0132	0.0323 ±0.0016	0.1729 ±0.0103	0.4940 ±0.0185	0.3761 ±0.0010	0.1205 ±0.0128
Irregular I	-	0.1979 ±0.0275	0.2391 ±0.0052	0.6174 ±0.0052	0.4139 ±0.0584	**0.1223** ±0.0020
Irregular II	-	0.3121 ±0.1293	0.2463 ±0.0010	**0.1163** ±0.0053	0.4142 ±0.0462	0.1271 ±0.0022
Irregular III	-	0.2335 ±0.0719	0.2432 ±0.0038	**0.1240** ±0.0037	0.3982 ±0.0283	0.1279 ±0.0056
Average across all data types	-	0.1939 ±0.1244	0.2254 ±0.0322	0.3379 ±0.2321	0.4006 ±0.0379	**0.1244** ±0.0069

Experimental Results (Interpolation by DeepONet)

- The Truth (with irregular grid points), MP-PDE, and MAgNet plot the solutions through interpolation using values from the irregular grid points.
- Our model(GraphDeepONet) predicts solutions for all grids directly.



Experimental Results (Extrapolation by Graph)

Our model(GraphDeepONet, GDON) predicts the solution more accurately beyond the training domain for 1D Burgers' type PDE.

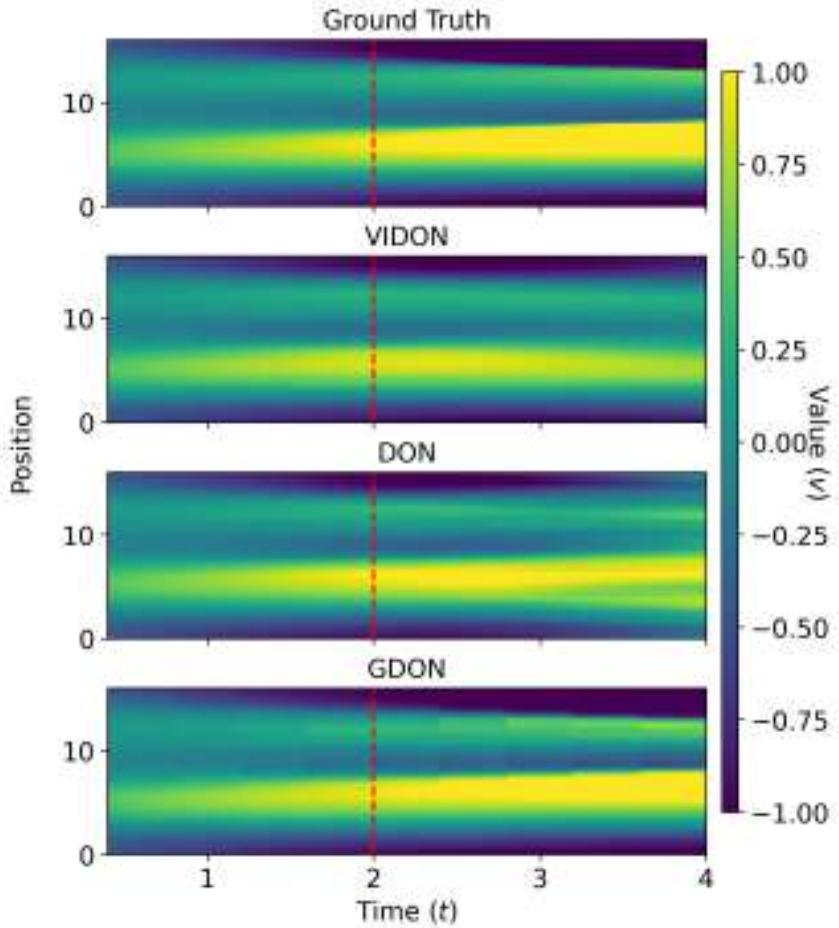


Figure: Solution profile in Burgers' equation for time extrapolation simulation using DeepONet, VIDON, and GraphDeepONet.

Burgers type PDE - periodic boundary condition by DeepONet

Our model(GraphDeepONet) predicts the solution near the boundary by enforcing periodic boundary condition.

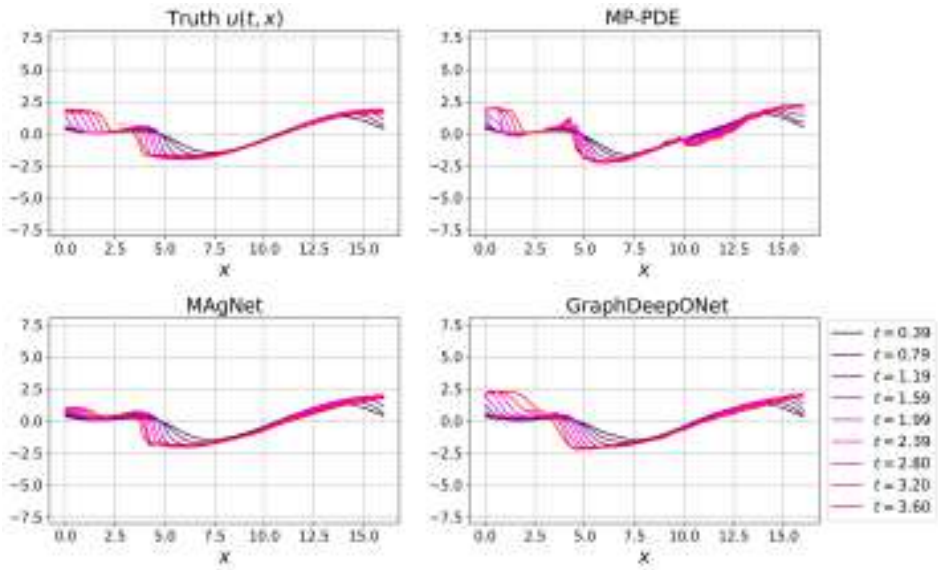


Figure: Ground truth solution and a prediction profile for Burgers' equation on a uniform grid, showcasing the small-scale shock in the solution.

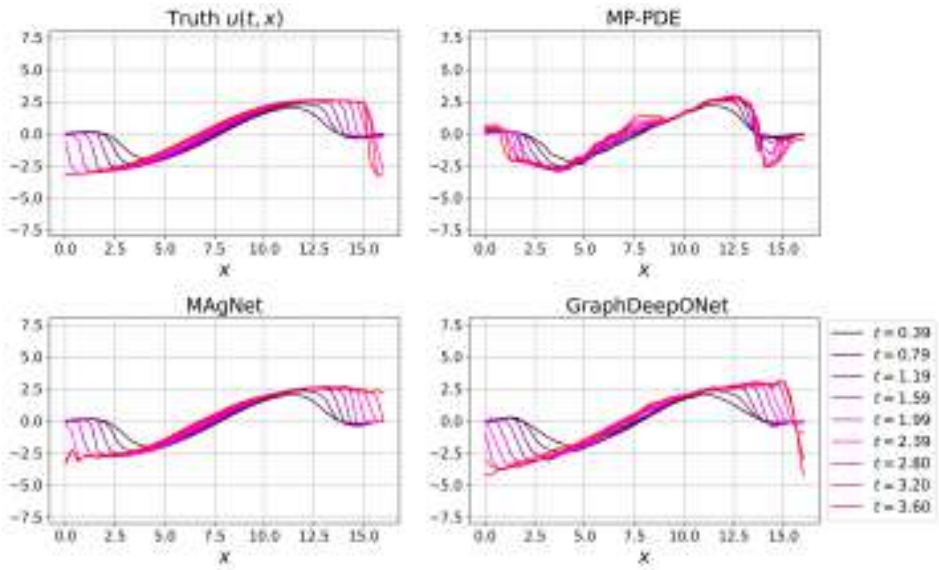


Figure: Ground truth solution and a prediction profile for Burgers' equation on a uniform grid, showcasing the large-scale shock in the solution.

2D N-S equation - periodic boundary condition by DeepONet

Our model(GraphDeepONet) satisfy periodic boundary condition.

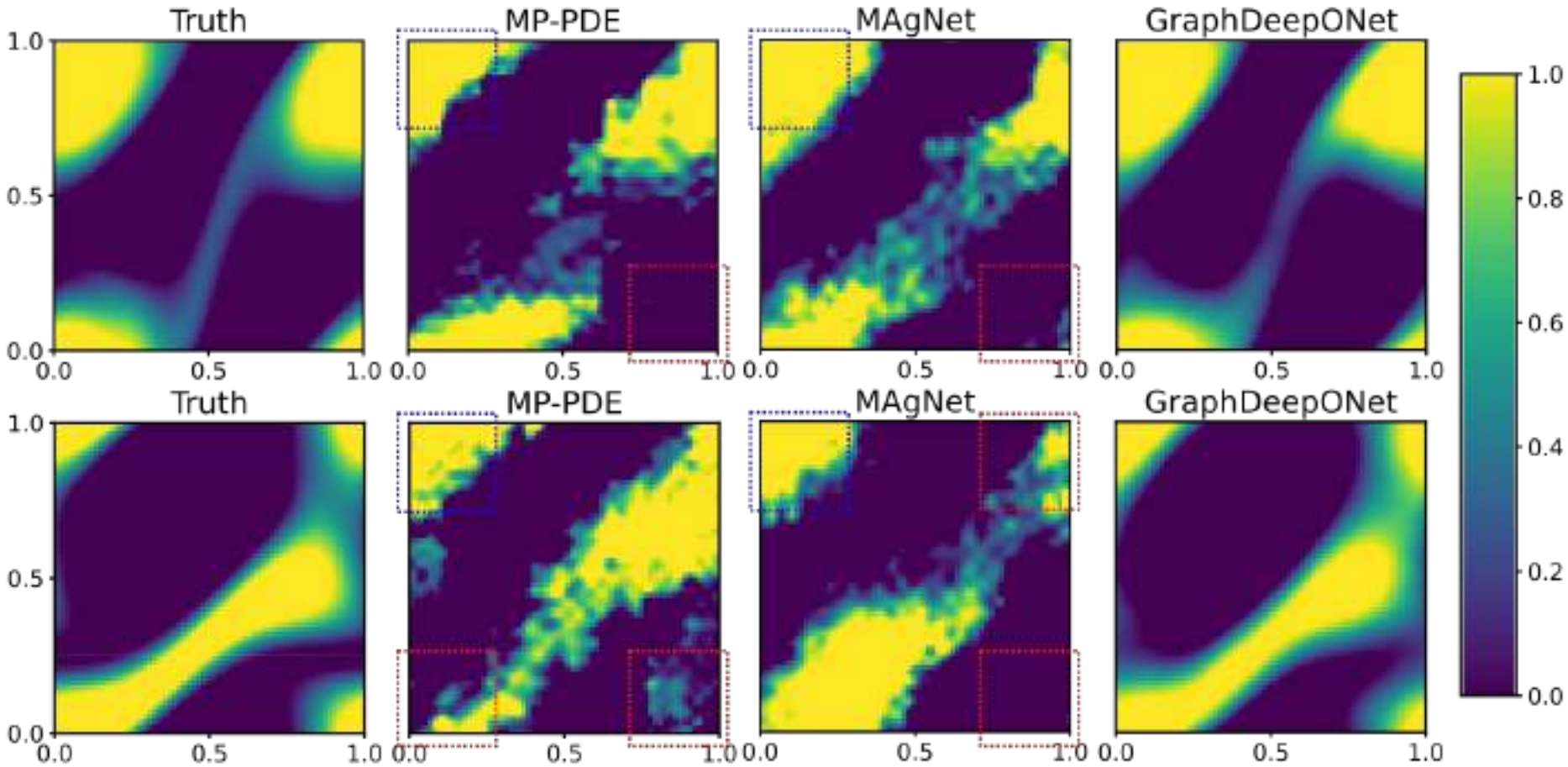


Figure: A snapshot for Navier-Stokes equation data.

Training dataset with varying grid points by Graph

Our model(GraphDeepONet) can be trained with dataset where each data has different grid points (Relative error: near 0.13).

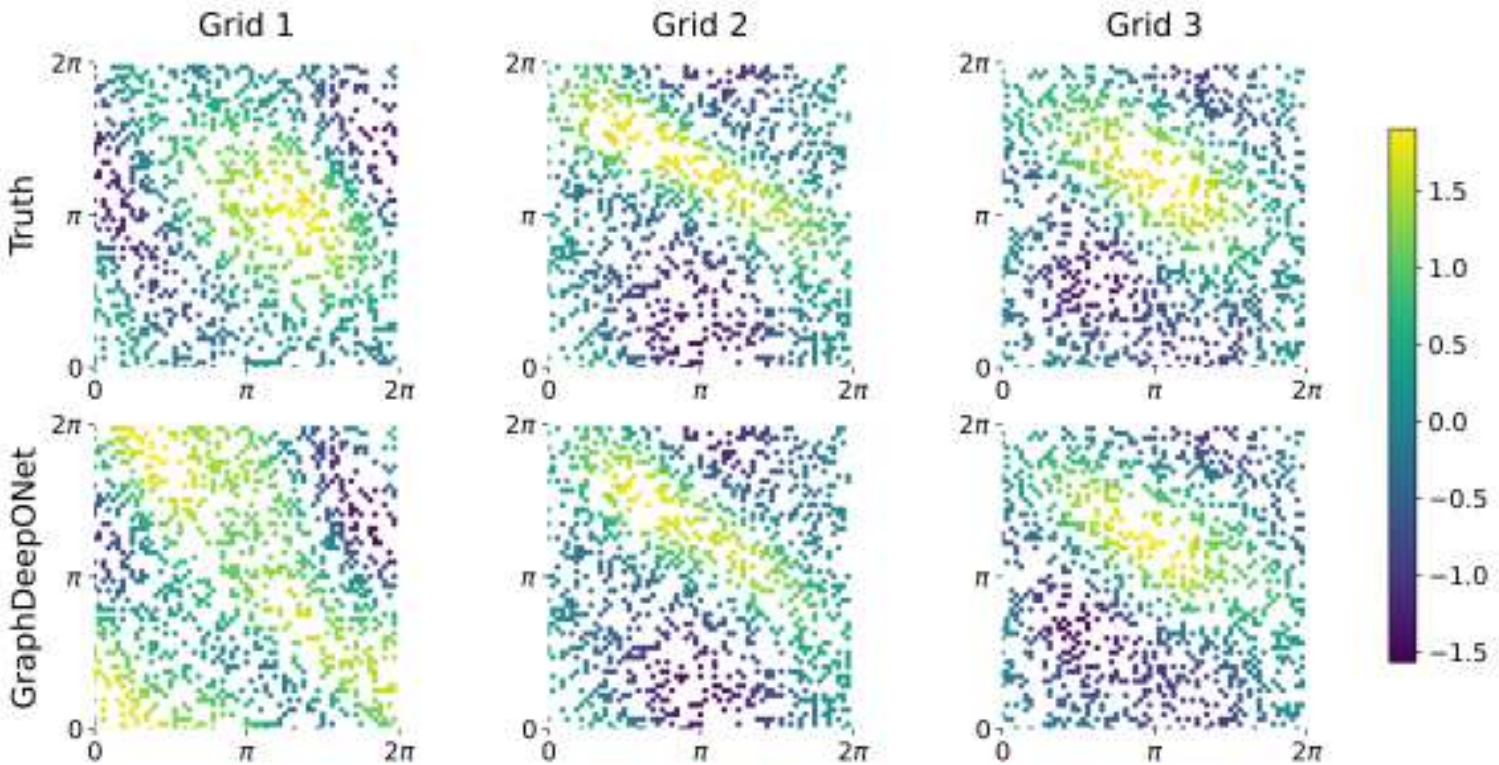


Figure: Representative snapshot of the two-dimensional Navier-Stokes equation predicted by GraphDeepONet trained at a single resolution 32^2. Each training sample employs a different irregular discretization of that resolution.

Training dataset with varying grid resolutions by Graph

Our model(GraphDeepONet) can be trained with dataset where each data has different grid resolutions (Relative error: near 0.13).

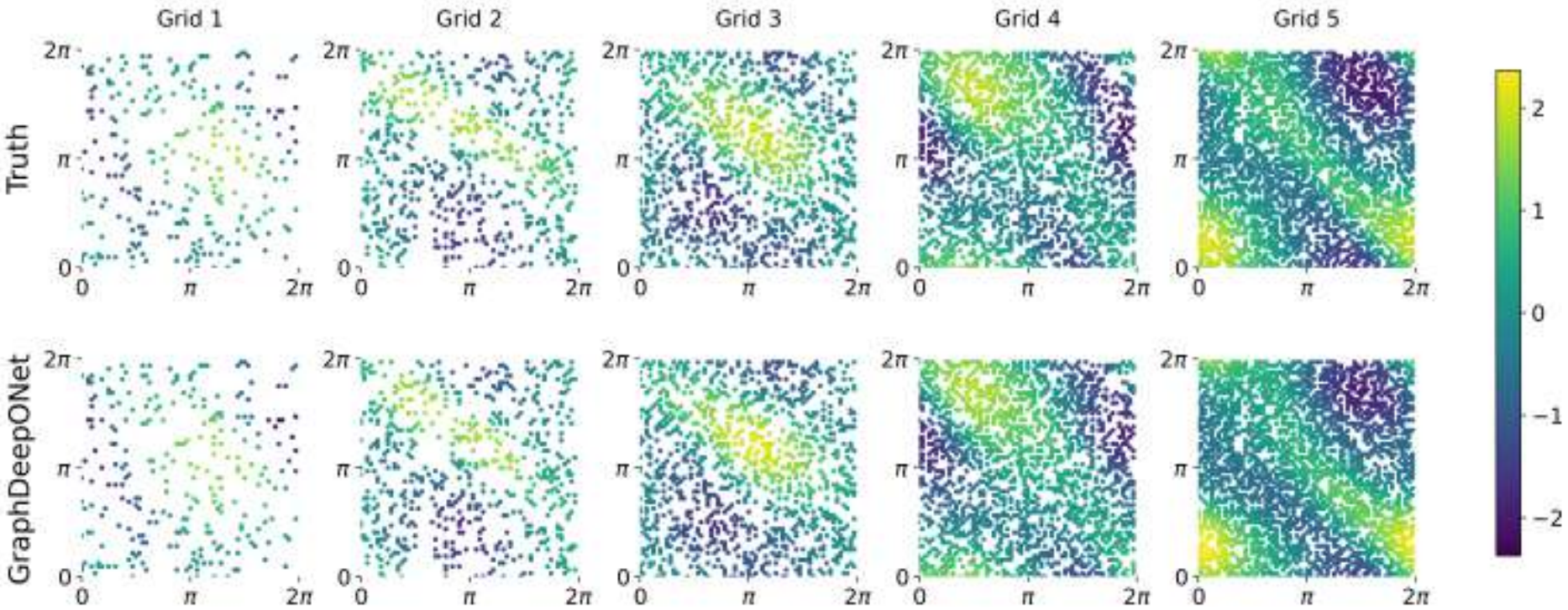
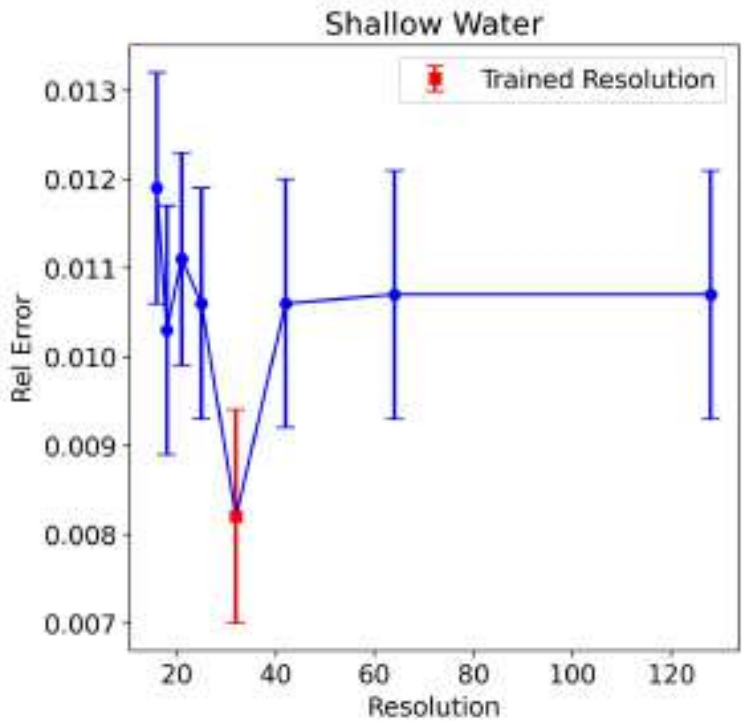


Figure: Representative snapshot for the two-dimensional Navier-Stokes equations predicted by GraphDeepONet jointly trained on five grid resolutions 16^2, 24^2, 32^2, 40^2, and 48^2, using a fixed irregular grid for each resolution.

Summary

- PINNs may require a large amount of time to predict the solution for a given initial condition.
- Learning operators can be conducted on the irregular, unstructured grid.
- Previous models with graph neural networks can suffer from lower accuracy for some specific grids.

Our model can be applied to the dataset with different types of grid points. It can achieve comparable error even with test data with unseen resolution.



Thanks for your attention!

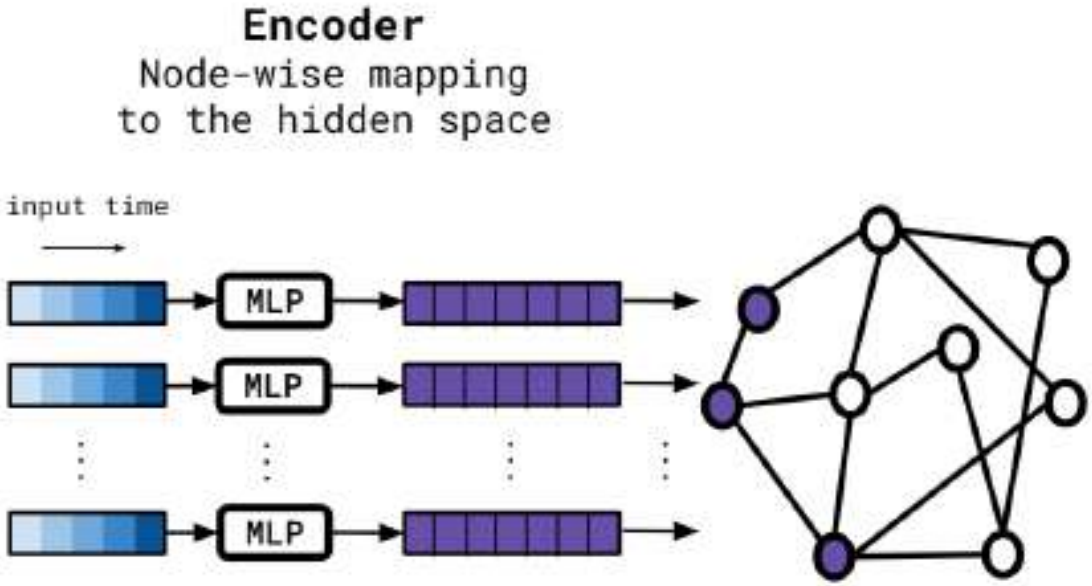
Operator Learning based on graph neural networks - I

We first encode function values $\tilde{u}^k(x_i)$ at the grid points $\{x_i\}_{i=1}^m$.

(Encoder) It encodes the value at each grid point.

$\mathbf{f}_i^0 = \epsilon^v\left(\left[\tilde{u}^k(x_i), x_i, k\Delta t\right]\right)$ where ϵ^v is a multi-layer perceptron (MLP).

Brandstetter, J, Worrall, D., & Welling, M. (2022). Message passing neural PDE solvers. *arXiv preprint arXiv:2202.03376*.



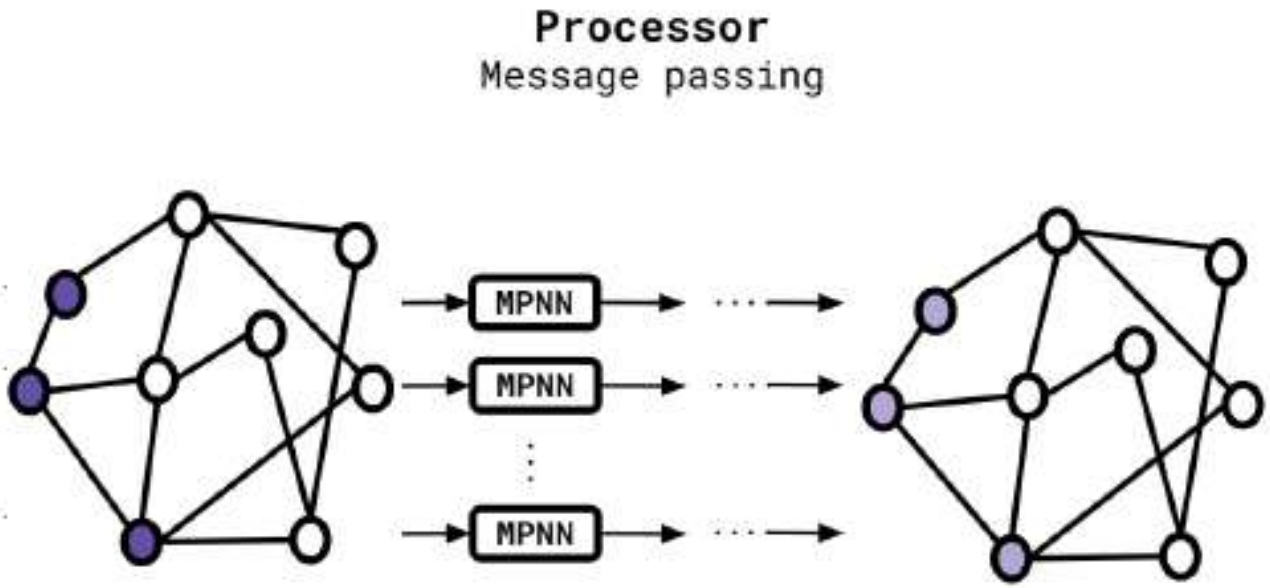
Operator Learning based on graph neural networks - II

(Processor) It updates the value for each grid point based on local interactions.

edge $j \to i$ message: $$\mathbf{m}_{ij}^{\ell} = \phi\left(\mathbf{f}_i^{\ell}, \mathbf{f}_j^{\ell}, \widetilde{u}^k(x_i) - \widetilde{u}^k(x_j), x_i - x_j\right),$$

node i update: $$\mathbf{f}_i^{\ell+1} = \psi\left(\mathbf{f}_i^{\ell}, \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij}^{\ell}\right), \ell = 0, 1, \ldots, L-1.$$

Brandstetter, J., Worrall, D., & Welling, M. (2022). Message passing neural PDE solvers. *arXiv preprint arXiv:2202.03376*.

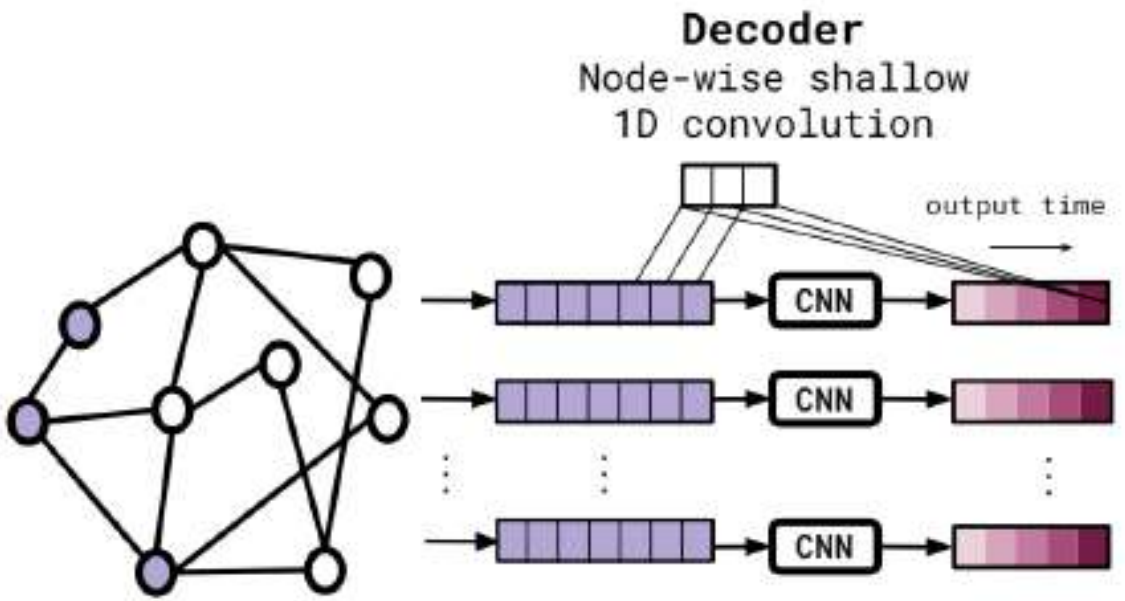


Operator Learning based on graph neural networks - III

(Decoder) It uses the previous $\mathbf{f}_i^L$ to predict the value of solution at the next time step.

$$\tilde{u}^{k+1}(x_i) = \tilde{u}^k(x_i) + \Delta t\, \mathbf{f}_i^L, \quad 1 \le i \le m.$$

Brandstetter, J, Worrall, D., & Welling, M. (2022). Message passing neural PDE solvers. *arXiv preprint arXiv:2202.03376*.



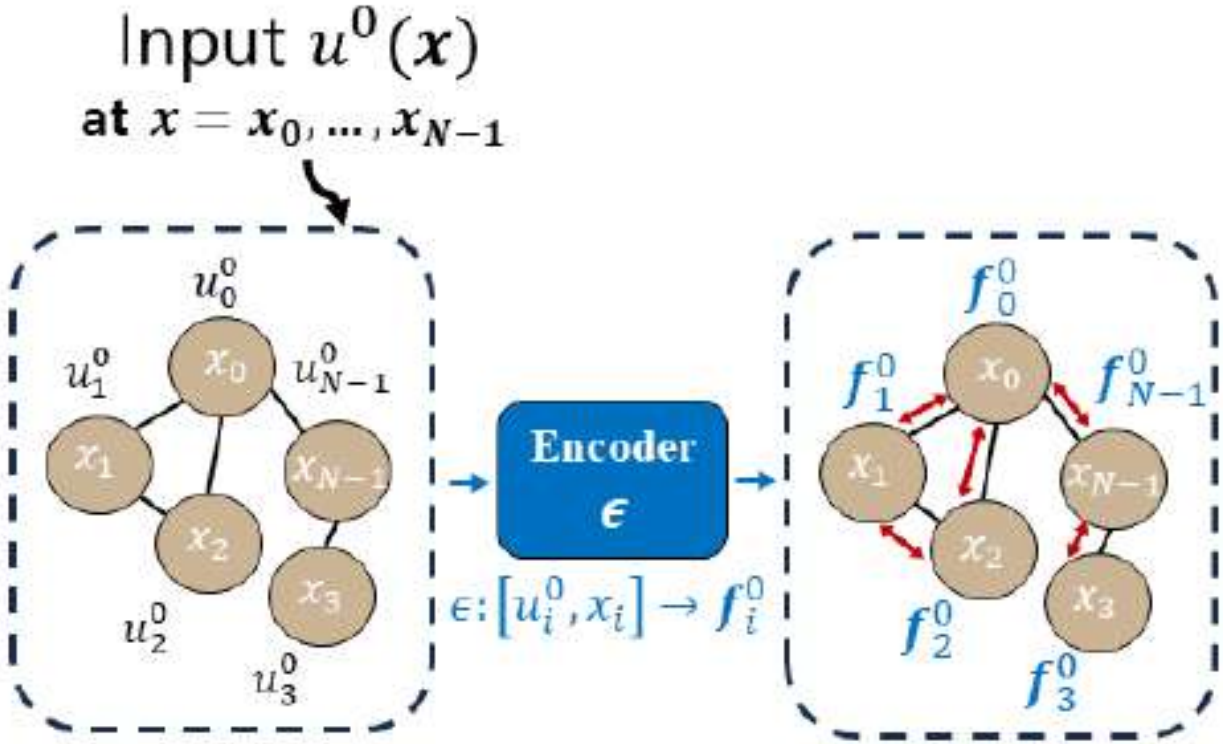
Graph neural networks were included to handle irregular grid points. (However, there is a **challenging** exmaple.)

Proposed Method - Encoder

Encoder ϵ. The encoding function $\epsilon : \mathbb{R}^{1+d} \to \mathbb{R}^{d_{\text{lat}}}$ produces the node embedding vector $\boldsymbol{f}_i^0$ as follows:

$$\boldsymbol{f}_i^0 := \epsilon\left(u_i^0, x_i\right) \in \mathbb{R}^{d_{\text{lat}}},$$

where ϵ is a multilayer perceptron (MLP).



Proposed Method - Processor

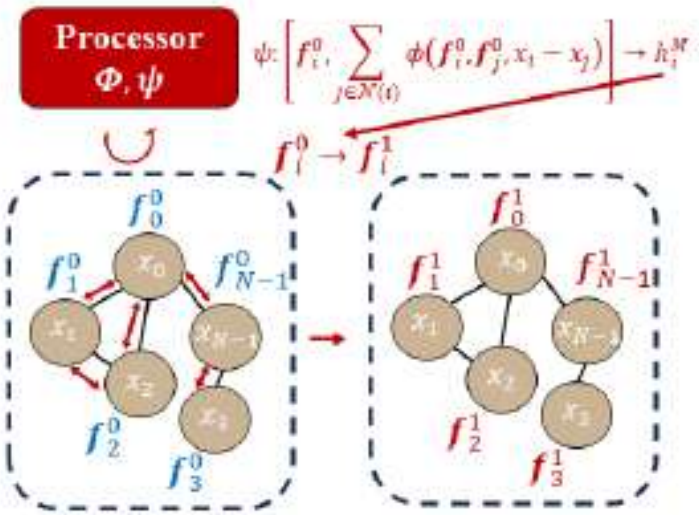
Processor ϕ, ψ. For $\ell = 0, 1, ..., L-1$ with $\boldsymbol{h}_i^0 = \boldsymbol{f}_i^0$,

$$\boldsymbol{m}_{ij}^{\ell} = \phi(\boldsymbol{h}_i^{\ell}, \boldsymbol{h}_j^{\ell}, x_i - x_j),$$

$$\boldsymbol{h}_i^{\ell+1} = \psi\left(\boldsymbol{h}_i^{\ell}, \sum_{j \in \mathcal{N}(i)} \boldsymbol{m}_{ij}^{\ell}\right),$$

ϕ, ψ; MLPs. $\mathcal{N}(i)$: the neighboring nodes of node i.

Update the latent vector as: $\boldsymbol{f}_i^1 = \boldsymbol{f}_i^0 + \boldsymbol{h}_i^L, \quad 1 \le i \le m.$



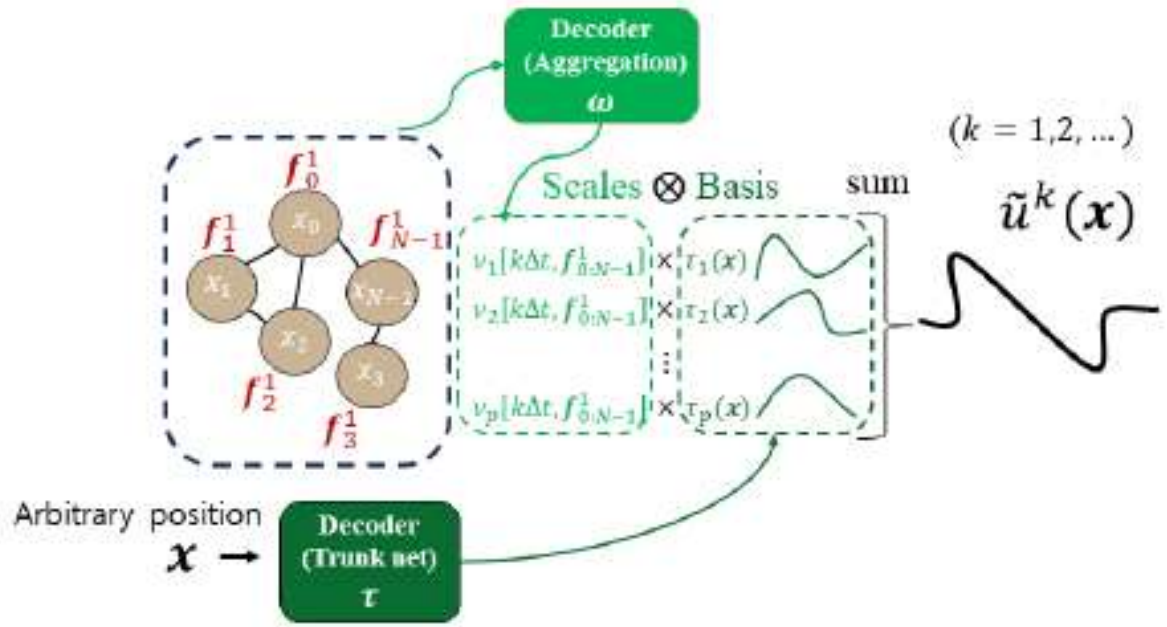
Proposed Method (Decoder 1)

The soft attention aggregation $\boldsymbol{\nu} : \mathbb{R}^{d_{\text{lat}} \times N} \to \mathbb{R}^p$ outputs coefficients:

$$\boldsymbol{\nu}[\boldsymbol{f}^1_{0:N-1}, \Delta t] := \sum_{i=0}^{N-1} \overbrace{\frac{\exp\left(\omega_{\text{gate}}(\boldsymbol{x}_i, \boldsymbol{f}^1_i)/\sqrt{d_{\text{lat}}}\right)}{\sum_{j=0}^{N-1} \exp\left(\omega_{\text{gate}}(\boldsymbol{x}_j, \boldsymbol{f}^1_j)/\sqrt{d_{\text{lat}}}\right)}}^{\text{attention score}} \odot \omega_{\text{feature}}(\Delta t, \boldsymbol{f}^1_i).$$

$\odot$: element-wise product

$\omega_{\text{gate}}(\mathbb{R}^{d_{\text{lat}}+d} \to \mathbb{R}^p)$, $\omega_{\text{feature}}(\mathbb{R}^{d_{\text{lat}}+1} \to \mathbb{R}^p)$: MLPs.

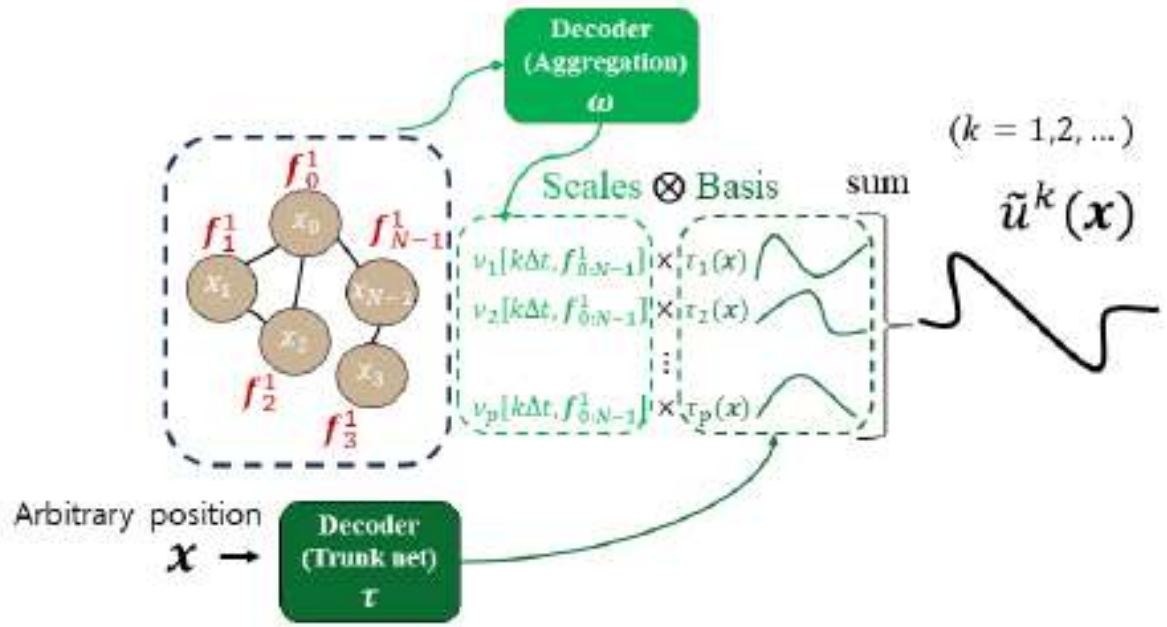


Proposed Method (Decoder 2)

The next timestep is predicted as

$$\widetilde{u}^1(\boldsymbol{x}) = \sum_{j=1}^{p} \nu_j[\boldsymbol{f}^1_{0:N-1}, \Delta t]\tau_j(\boldsymbol{x}),$$

where $\boldsymbol{\nu}[\boldsymbol{f}^1_{0:N-1}, \Delta t] := [\nu_1, \nu_2, ..., \nu_p] \in \mathbb{R}^p$.



we can predict the solution at arbitrary position x.

Number of dataset vs relative error

Overall, our model can achieve the most accurate prediction except when the number of training samples is smaller than 512.

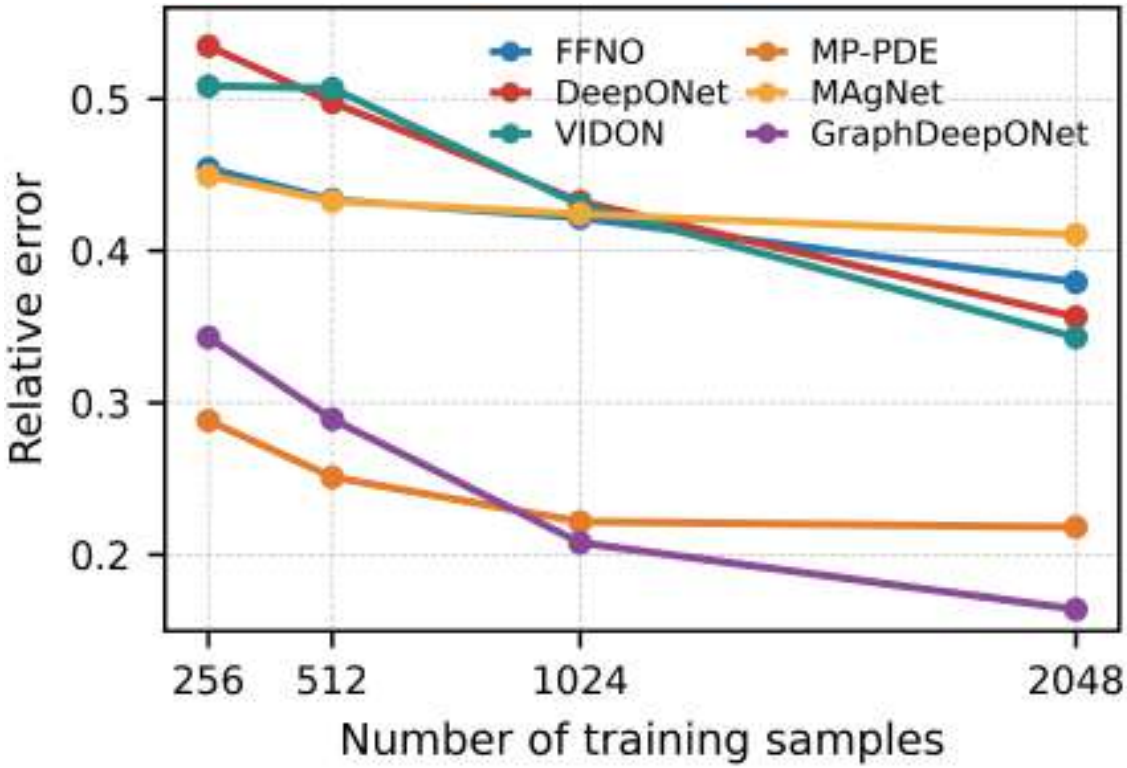


Figure: Relative prediction error of F-FNO, DeepONet, VIDON, MP-PDE, MAgNet, and GraphDeepONet for the number of training data. Errors are computed on irregular spatial grids for the Burgers-type PDE E1 dataset.