Inducing Point Operator Transformer: A Flexible and Scalable Architecture for Solving PDEs

06/20/2024

Seungjun Lee

Motivation

Motivation

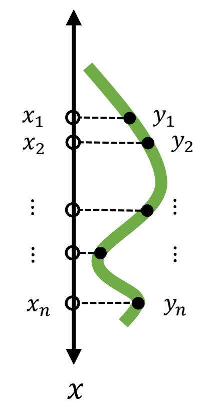
- **Function:** mapping between input variable to output variable

$$f: x \rightarrow y$$

Ex) if $f(x) = x^2$, $f(2) = 4$ and $f(3) = 9$.

  - **Regression problem:** approximate function f

Function, $\boldsymbol{y = f(x)}$



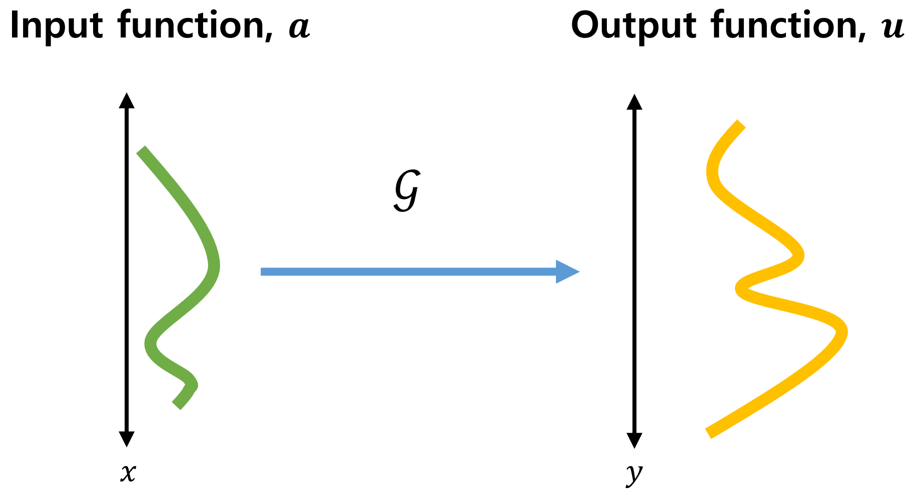
- **Operator:** mapping between input function to output function

$$\mathcal{G}: a(x) \rightarrow u(y)$$

Ex) if $\mathcal{G} = \nabla_x$,

$$a_1(x) = x^2 \quad \rightarrow \quad \nabla_x a_1 = 2x$$
$$a_2(x) = \sin x \quad \rightarrow \quad \nabla_x a_2 = \cos x$$

  - **Operator learning problem:** approximate operator $\mathcal{G}$



Input function, $\boldsymbol{a}$
Output function, $\boldsymbol{u}$
$\mathcal{G}$
x
y

Motivation

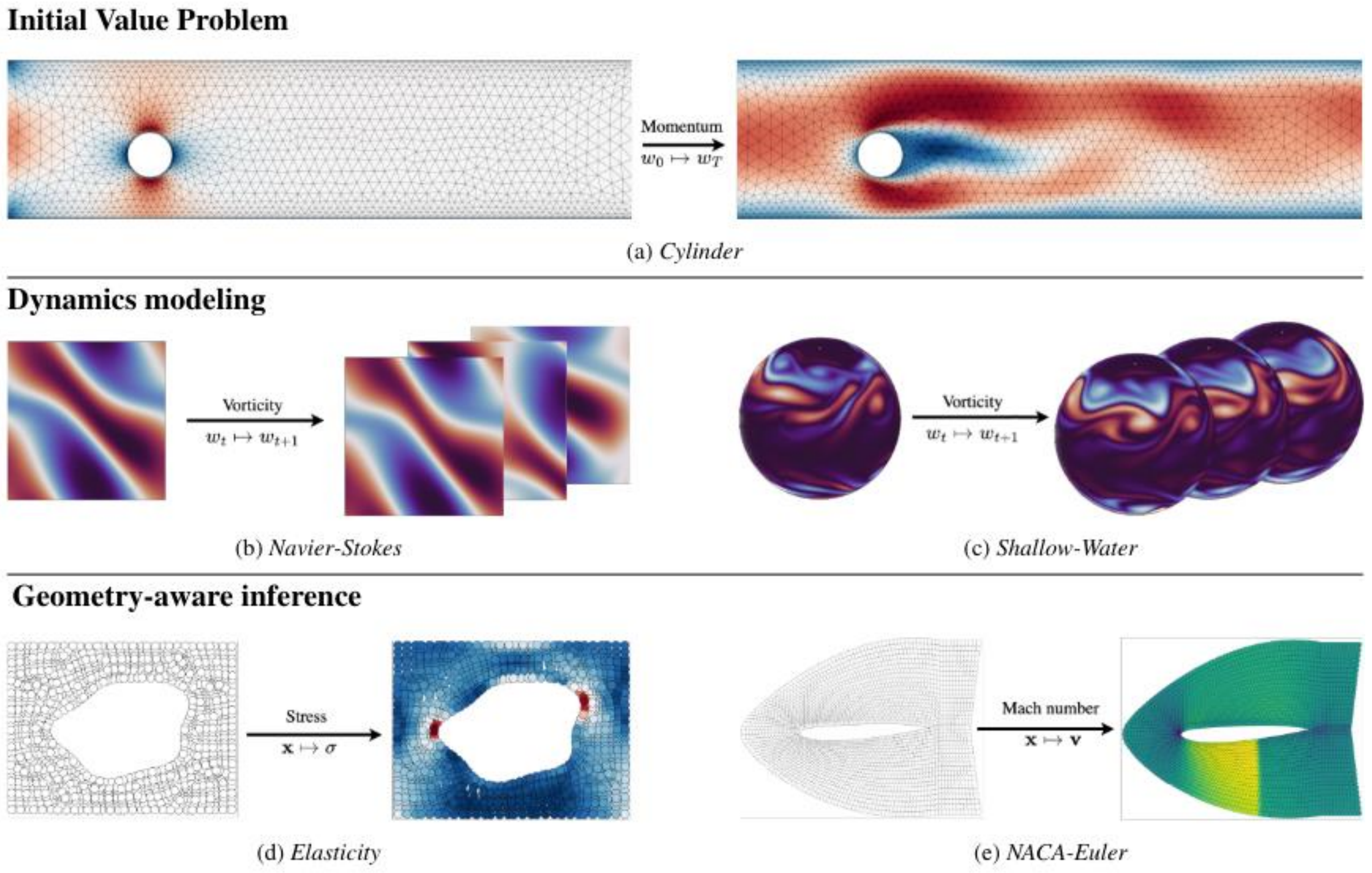


Figure 1: Illustration of the problem classes addressed in this work: Initial Value Problem (IVP) (a), dynamic forecasting (b and c) and geometry-aware inference (d and e).

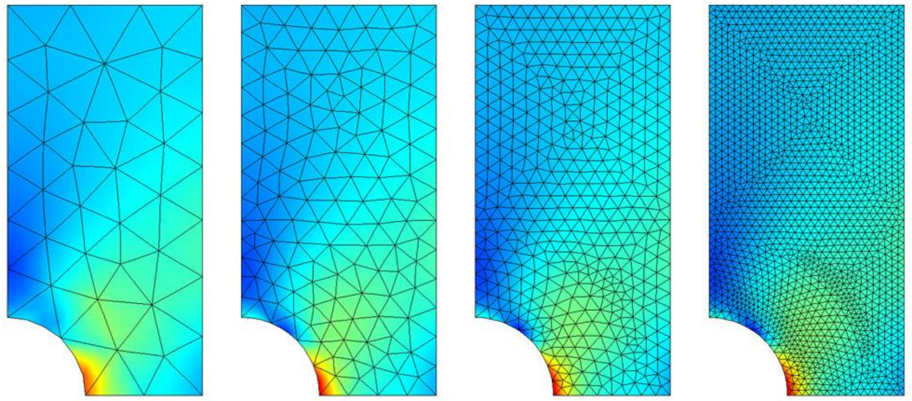
1. L. Serrano, et. al, "Operator Learning with Neural Fields: Tackling PDEs on General Geometries", Neurips, 2023.

Motivation

- Architecture for solving partial differential equations (PDEs) faces two main challenges:
    - **Flexibility** in handling arbitrary discretization formats, and
    - **Scalability** to large discretization.

- Irregular grids
- Varying discretization
- Unordered/unstructured

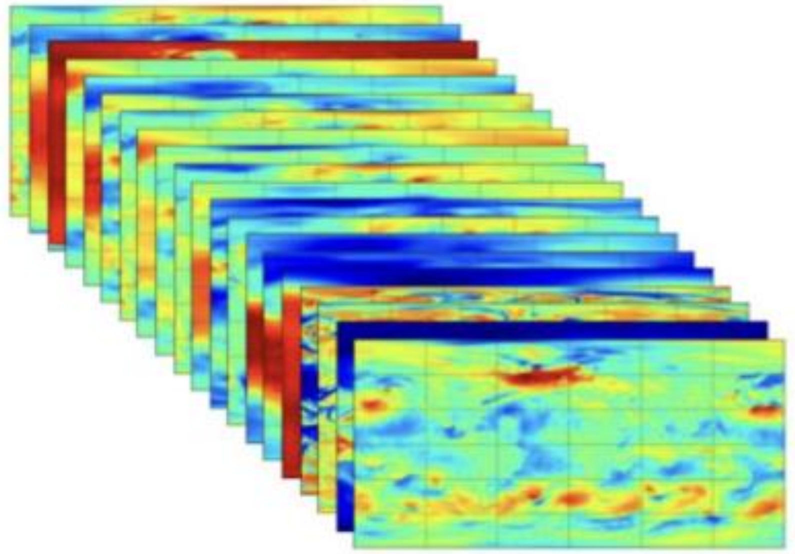
1



Flexible in any discretization

- Most existing architectures are limited by
    - Their desired discretization formats
        - Fourier neural operators,[3] DeepONets[4]
    - Infeasible to scale large inputs and outputs
        - Existing Transformer-based operators [5]

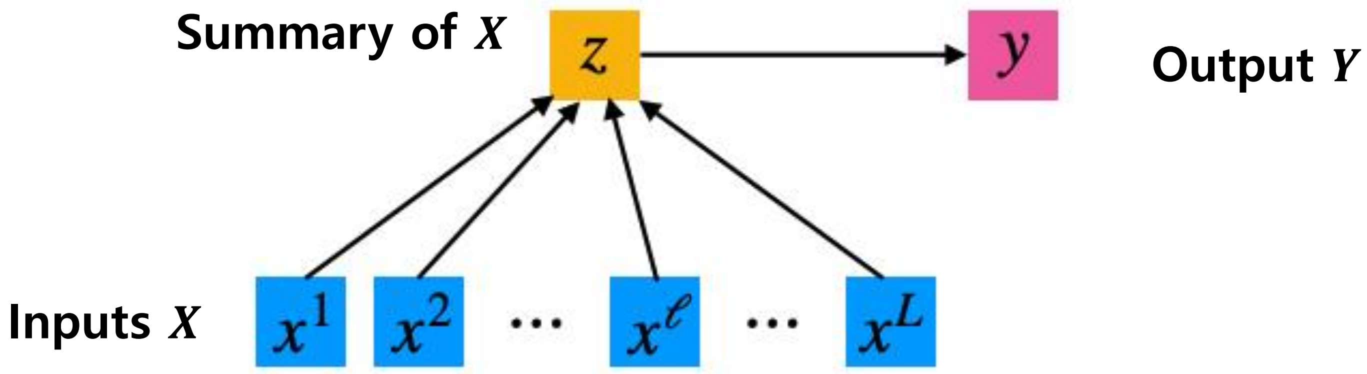
2



Scalable to large discretization or long-term forecasting

1. https://www.comsol.com/multiphysics/mesh-refinement
2. S. Chen, et. al., "Foundation Models for Weather and Climate Data Understanding: A Comprehensive Survey", arXiv:2312.03014, 2023
3. Z. Li, et. al., "Fourier Neural Operator for Parametric Partial Differential Equations", ICLR, 2021.
4. L. Lu, et. al., "Learning Nonlinear Operators via DeepONet Based on the Universal Approximation Theorem of Operators", Nature, 2021.
5. Z. Li, et. al., "Transformer for Partial Differential Equations' Operator Learning", TMLR, 2023.

Motivation



[1,2]

- **De Finetti's theorem**
  - Let $X = \{x_1, x_2, \ldots\}$ be an exchangeable sequence. Then, for $n \in N_+$, there exists a latent variable z induced by the exchangeability,

$$p(x_1, \ldots, x_n) = \int \prod_{i=1}^{n} p(x_i|z) \cdot p(z)dz\,,$$

  - where z is a desirable representation which is expected to include a sufficient statistic of the input X for predicting the target variable y.

$$p(y|X) = \int p(y|z) \cdot p(z|X)dz$$

1. M. Zaheer, et. al, "Deep Sets", Neurips, 2018.
2. Y. Zhang, et. al, "An Analysis of Attention via the Lens of Exchangeability and Latent Variable Models", arXiv:2212.14852, 2023.

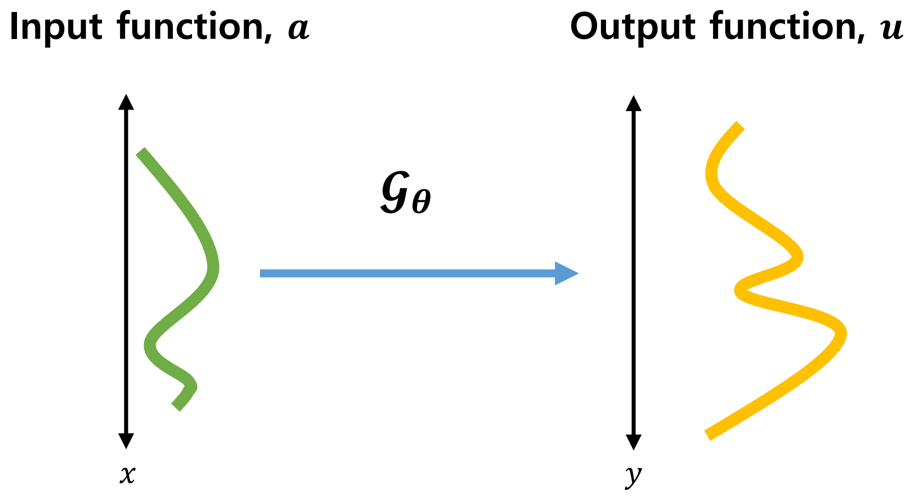
Preliminaries

Preliminaries

- **Partial Differential Equation (PDE)**

$$\begin{aligned} \mathcal{L}_a u &= f, & x &\in \Omega \\ \mathcal{B} u &= 0, & x &\in \partial\Omega \end{aligned}$$

  - $\mathcal{L}$: differential operator, $\mathcal{B}$: boundary conditions, f: source function
  - a: system parameter (input)
  - u: solution (output)



Solution function: $\boldsymbol{u} = \mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{a})$

- **Operator learning on PDE**
  - **Goal:** approximate $\mathcal{G} = \mathcal{L}_a^{-1} f$: $\mathcal{A} \to \mathcal{U}$ with $\mathcal{G}_\theta$ from input-output pairs $\{(a_i, u_i)\}_{i=1}^N$
  - **Objective:** learning the model $\mathcal{G}_\theta: \mathcal{A} \to \mathcal{U}$ with $a \sim \mu$ is i.i.d. on $\mathcal{A}$

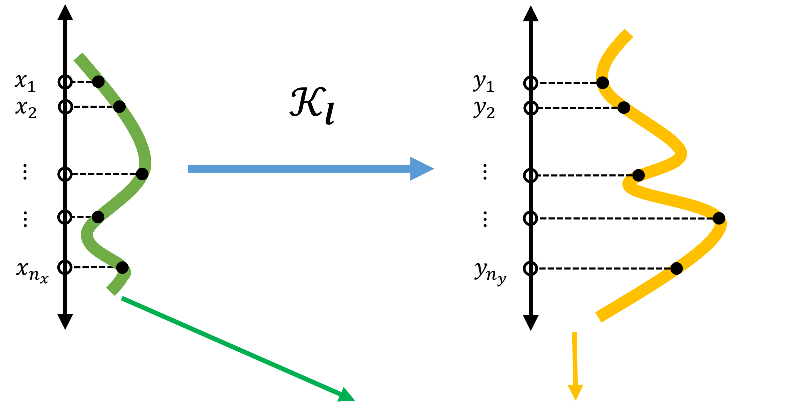
$$E_{a\sim\mu}[\mathcal{L}(\mathcal{G}_\theta(a), u)] \approx \frac{1}{N} \sum_{i=1}^{N} \|u_i - \mathcal{G}_\theta(a_i)\|^2$$

Preliminaries

- **Kernel integral operation**
  - Operator learning models consists of series of **kernel integral operators $\boldsymbol{\mathcal{K}}: \boldsymbol{v_l(x)} \rightarrow \boldsymbol{v_{l+1}(y)}$,**

$$v_{l+1}(y) = [\mathcal{K}_l(v_l)](y) = \int_{\Omega_x} \kappa_\phi(y,x) v_l(x) dx\,, \qquad (x,y) \in \Omega_x \times \Omega_y$$

Input function, $v_l(x)$ **Output function, $v_{l+1}(y)$**



$$\underbrace{\begin{bmatrix} \kappa_\phi(y_1,x_1) & \cdots & \kappa_\phi(y_1,x_{n_x}) \\ \vdots & \ddots & \vdots \\ \kappa_\phi\left(y_{n_y},x_1\right) & \cdots & \kappa_\phi\left(y_{n_y},x_{n_x}\right) \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} v_l(x_1) \\ \vdots \\ v_l(x_{n_x}) \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} v_{l+1}(y_1) \\ \vdots \\ v_{l+1}\left(y_{n_y}\right) \end{bmatrix}}_{b}$$

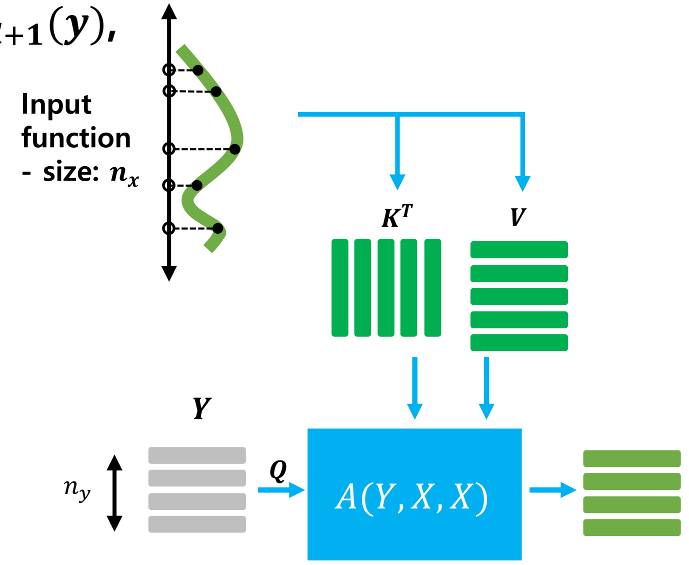
  - $\mathcal{K}$: kernel operator, $\kappa_\phi(y,x)$: kernel function
  - $\Omega_x = \{x_1, \dots, x_{n_x}\}$: (discretized) input domain, $\Omega_y = \left\{y_1, \dots, y_{n_y}\right\}$: (discretized) output domain

- $\kappa_\phi(y,x)$ can be interpreted as **interactions (correlations)** between the **infinitesimal** segments **at location $\boldsymbol{x}$ and $\boldsymbol{y}$.**

Preliminaries

- **Kernel integral operation**
  - Operator learning models consists of series of **kernel integral operators $\boldsymbol{\mathcal{K}}: \boldsymbol{v_l(x) \rightarrow v_{l+1}(y)}$,**

$$v_{l+1}(y) = [\mathcal{K}_l(v_l)](y) = \int_{\Omega_x} \underline{\kappa_\phi(y,x)} v_l(x) dx \,, \qquad (x,y) \in \Omega_x \times \Omega_y$$



- **Attention mechanism**
  - Let input vectors $X \in R^{n_x \times d}$, query vectors $Y \in R^{n_y \times d}$, then the attention can be

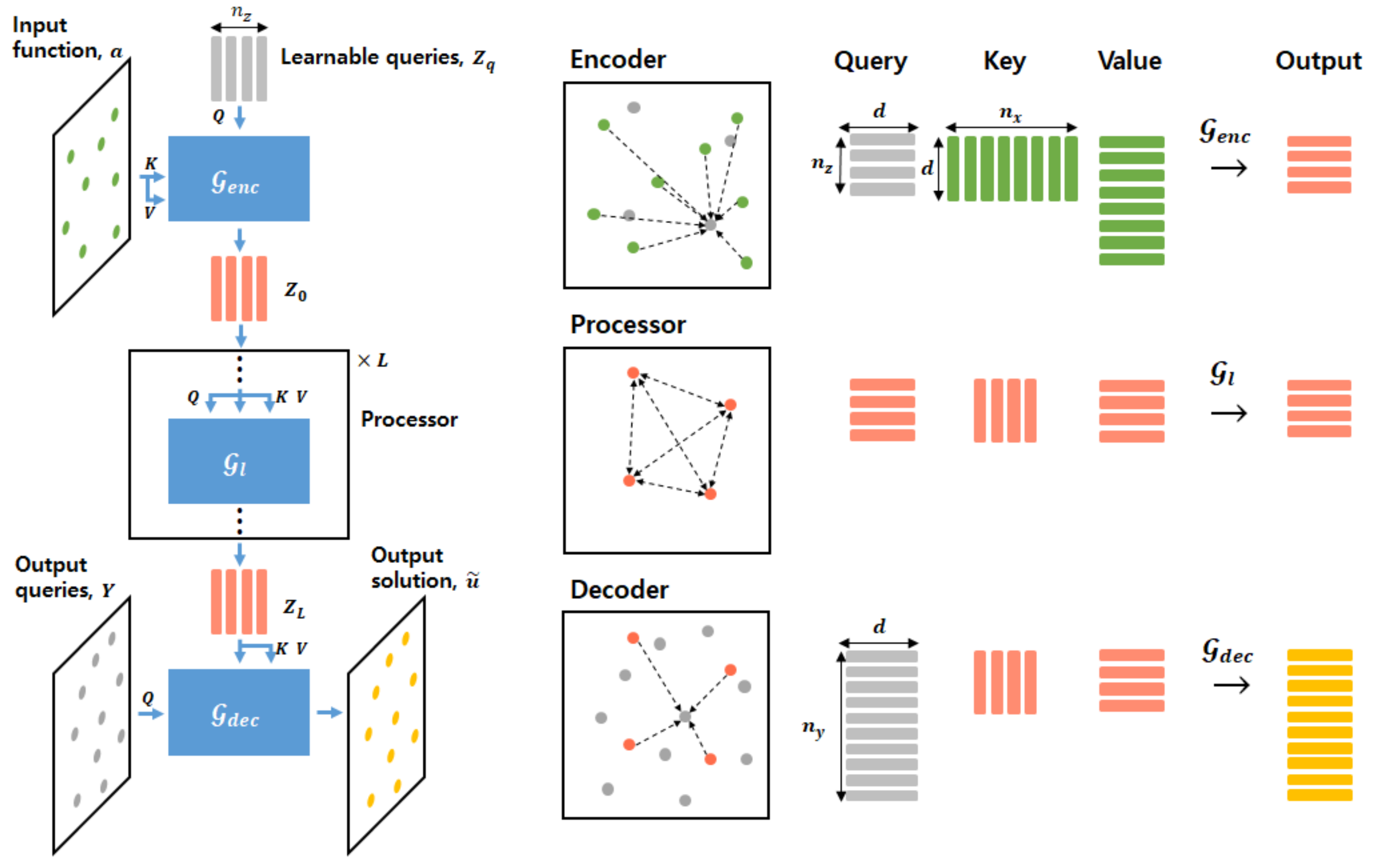
$$Attention(Y,X,X) = \sigma\underline{(QK^T)}V \approx \int_{\Omega_x} \underline{\big(q(Y) \cdot k(x)\big)} v(x) dx \,,$$

  - $Q = YW^q \in R^{n_y \times d}$, $K = XW^k \in R^{n_x \times d}$, and $V = XW^v \in R^{n_x \times d}$ are query, key, and value matrices respectively.
  - **Kernel integral operation** can be approximated by the **attention mechanism.**
  - Input vector X is projected to query embedding space Y, which can **change the number of discretization** from $\boldsymbol{n_x \rightarrow n_y}$

- **Complexity:** when the size of input and output discretization are $n_x \approx n_y \approx n$.
  - **Self-attention:** $Attention(X,X,X) \sim \mathcal{O}(n^2 d)$, **Cross-attention:** $Attention(Y,X,X) \sim \mathcal{O}(n n_z d)$

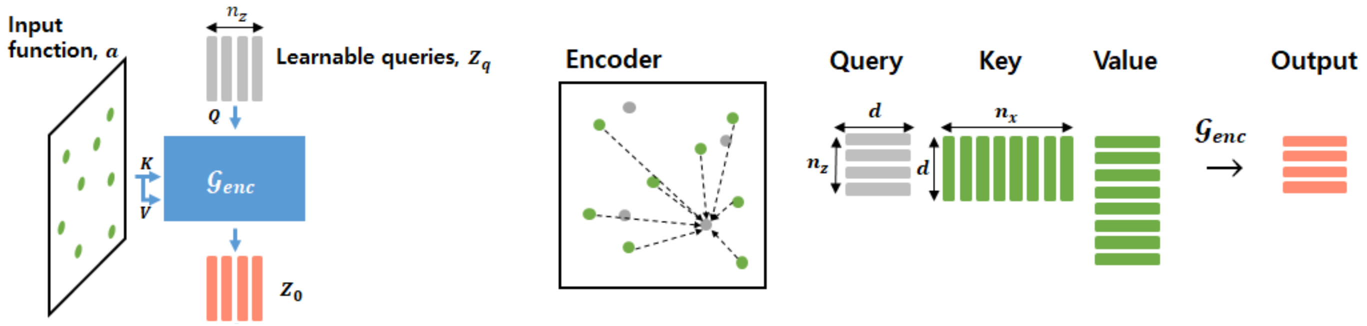
Approach

Approach



- **Inducing point operator transformer (IPOT)**
  - IPOT can capture **global interactions** and provide **flexibility** for handling **arbitrary input and output formats** with **feasible computational complexity.**
  - IPOT employs a **smaller number of n_z learnable query vectors** into the encoder, allowing most of the attention mechanisms to be computed in the **latent space** instead of the larger observational space.

Approach



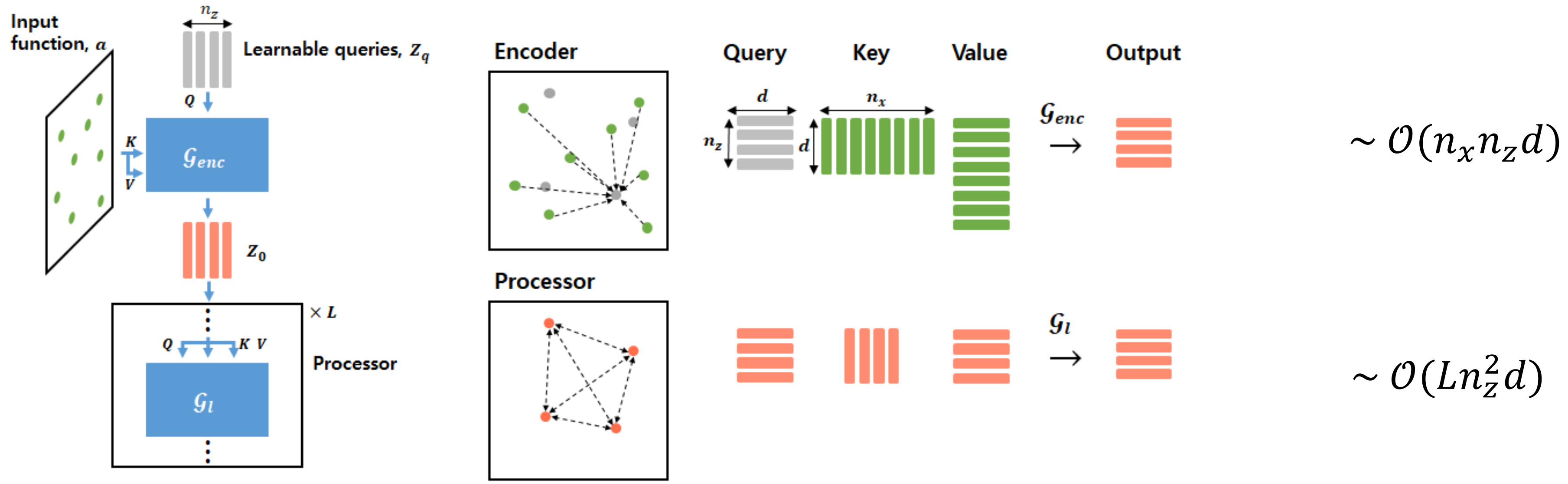
$\sim \mathcal{O}(n_x n_z d)$

- **Encoder**
  - We use cross-attention block to encode inputs $a \in R^{n_x \times d}$ to a smaller fixed number n_z of **learnable queries $\boldsymbol{Z_q} \in \boldsymbol{R^{n_z \times d}}$** $(n_z \ll n_x)$,

$$Z_0 = \mathcal{G}_{enc}(a) = Attention\left(Z_q, a, a\right) \in R^{n_z \times d},$$

  - **Projecting** input function to a **smaller size of $\boldsymbol{n_z}$** elements called **"inducing points"** which behave like the summary of X.
  - Computational cost $\sim \mathcal{O}(n_x n_z d)$: **linear complexity** with the size of input discretization n_x.

Approach

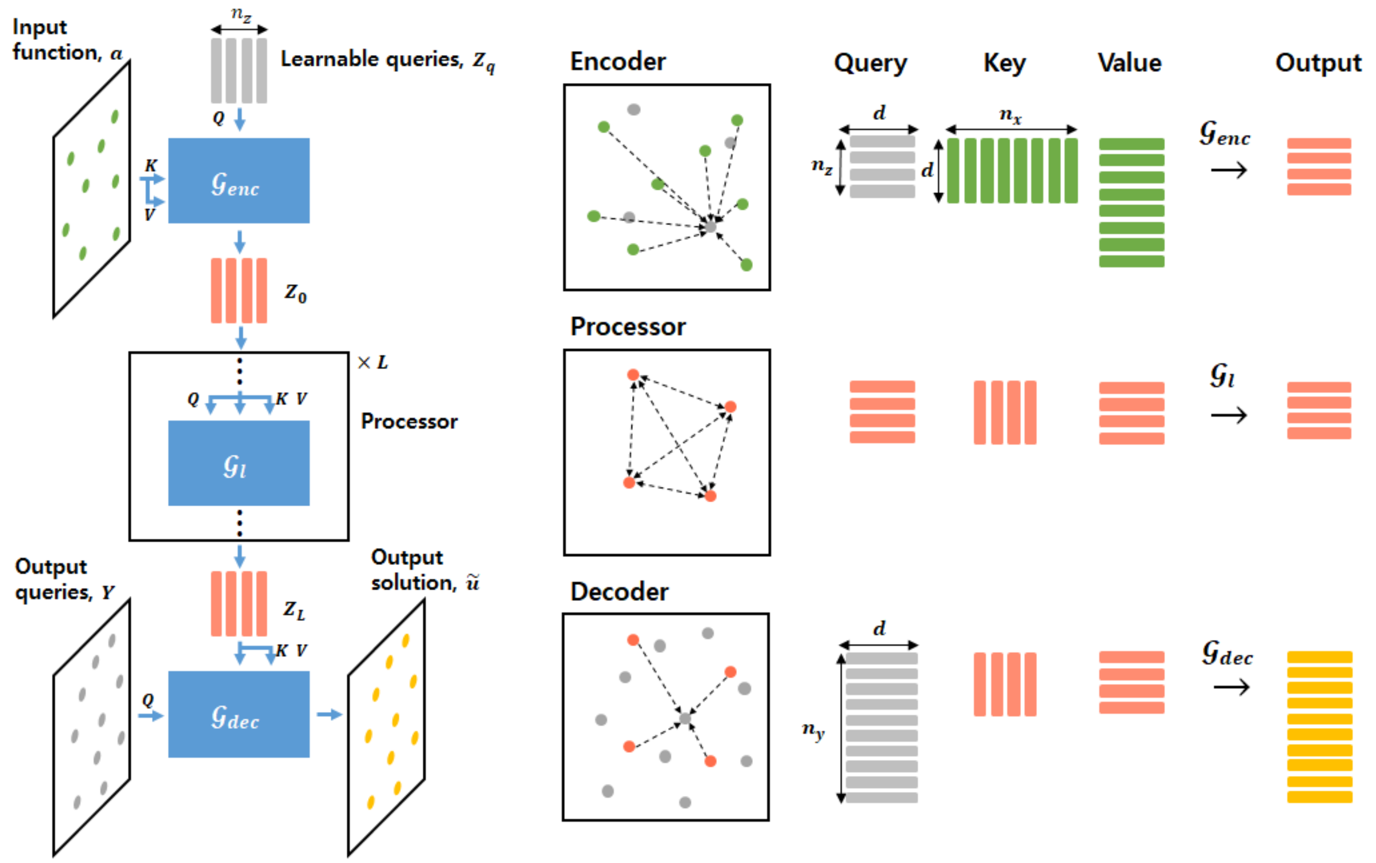


- **Processor**
  - We use self-attention blocks to process latent vectors $Z_l \in R^{n_z \times d}$ as the input of the query, key, and value components,

$$Z_{l+1} = \mathcal{G}_l(Z_l) = Attention(Z_l, Z_l, Z_l) \in R^{n_z \times d},$$

  - which are **decoupled from the discretization** of the input and output functions: **applicable to any discretization formats.**
  - Computational cost $\sim \mathcal{O}(L n_z^2 d)$: the size of $\boldsymbol{n_z}$ makes it feasible for high-fidelity modeling or long-term forecasting.

Approach



$$\sim \mathcal{O}(n_x n_z d)$$

$$\sim \mathcal{O}(L n_z^2 d)$$

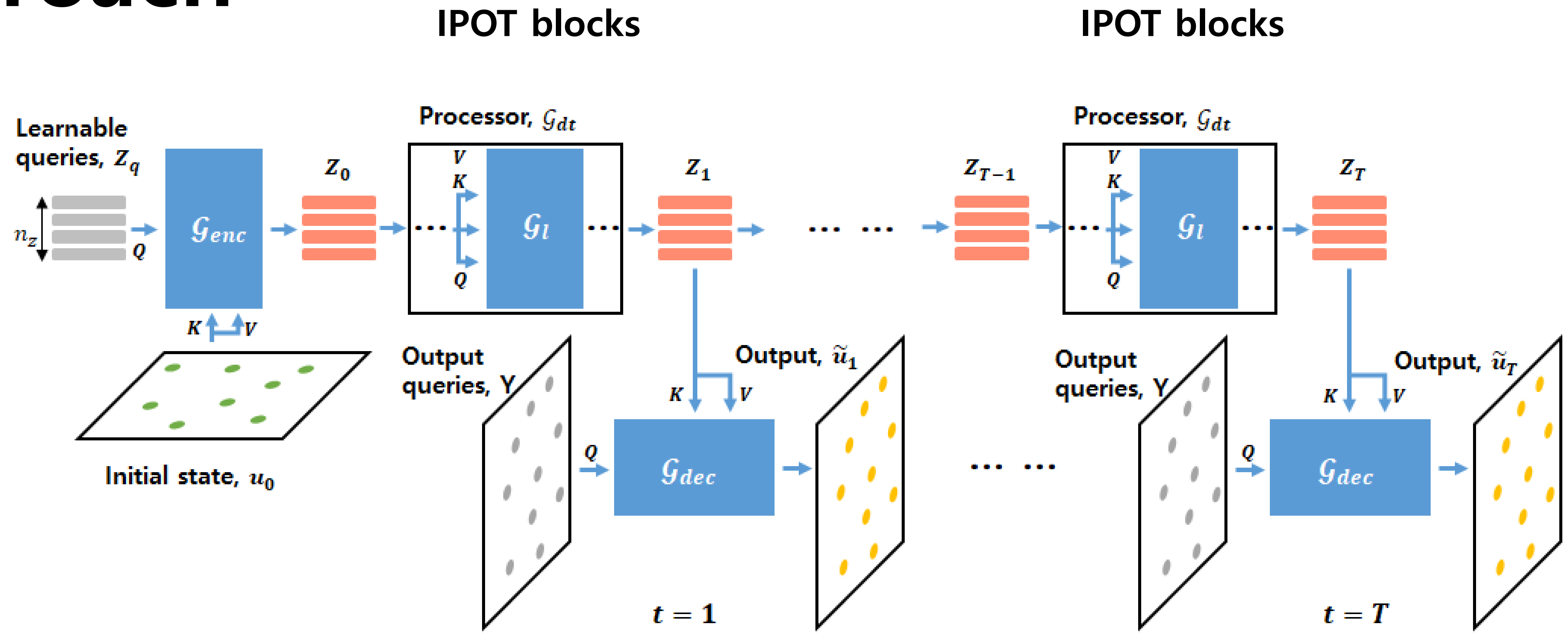
$$\sim \mathcal{O}(n_z n_y d)$$

- **Decoder**
  - We use cross-attention block to decode the **latent vectors** from the processor $Z_L \in R^{n_z \times d}$ at **output queries $\boldsymbol{Y} \in \boldsymbol{R}^{n_y \times d}$,**

$$\tilde{u} = \mathcal{G}_{dec}(Y, Z_L) = Attention(Y, Z_L, Z_L) \in R^{n_y \times d},$$

  - The entire model is applicable to **arbitrary input discretization** and can be **queried out at arbitrary output queries.**
  - Computational cost $\sim \mathcal{O}(n_z n_y d)$: **linear complexity** with the size of output discretization n_y.

Approach



- **Time-stepping through latent space**
  - We model the time-dependent PDEs as an **autoregressive process.**
  - The initial states are **compressed into the latent space**, where most computations are **time-stepping through the latent space**.
  - Finally, the latent states are decoded at each time step with **arbitrary query points.**

Experiments

Experiments

- **Datasets.** conducted on several scientific data benchmarks
  - **Regular grids:** Burgers, Darcy flow, Navier-Stokes[1]
  - **Irregular grids:** Airfoil, Elasticity, Plasticity,[2] spherical shallow water[3]
  - **Real-world data:** ERA5 reanalysis[4]

- **Training.** Given the input-output pairs (full grids of input and output in case of regular grids)

$$E_{a\sim\mu}[\mathcal{L}(\mathcal{G}_\theta(a),u)] \approx \frac{1}{N}\sum_{i=1}^{N}\|u_i - \mathcal{G}_\theta(a_i)\|^2$$

- **Evaluation.** Empirical test errors where discretization of a_i^{test} (input) and y (output) can be changing.

$$\frac{1}{N'}\sum_{i=1}^{N'}\sum_{y\in Y}\left\|u_i(y) - \mathcal{G}_\theta\left(a_i^{test}\right)(y)\right\|^2$$

  - The test error can be **bounded** by the sum of the **approximation error** and **discretization error**

$$\sum_{y\in Y}\|u_i(y) - \mathcal{G}_\theta\left(a_i^{test}\right)(y)\| \le \underbrace{\sum_{y\in Y}\|u_i(y) - \mathcal{G}_\theta(a_i)(y)\|}_{\textbf{approximation error}} + \underbrace{\|\mathcal{G}_\theta(a_i)(y) - \mathcal{G}_\theta\left(a_i^{test}\right)(y)\|}_{\textbf{discretization error}}$$

1. Z. Li, et. al, "Fourier Neural Operator for Parametric Partial Differential Equations", ICLR, 2021.
2. Z. Li, et. al, "Fourier Neural Operator with Learned Deformations for PDEs on General Geometries", arXiv:2207.05209, 2022.
3. Y. Yin, et. al, "Continuous PDE Dynamics Forecasting with Implicit Neural Representations", ICLR, 2023.
4. 5 https://www.ecmwf.int/en/forecasts/datasets/browsereanalysis-datasets

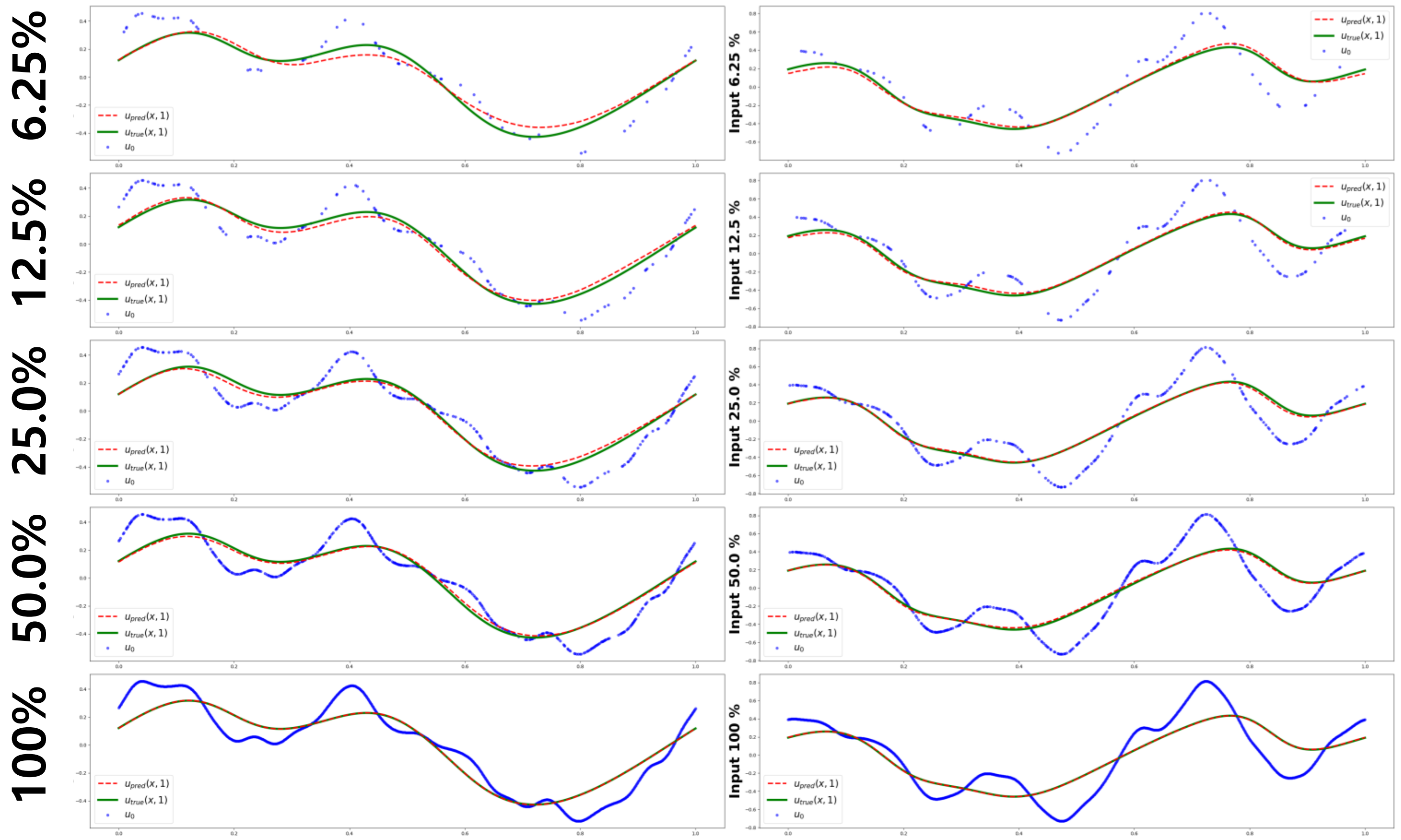
Experiments

- **Flexible to arbitrary discretization formats**

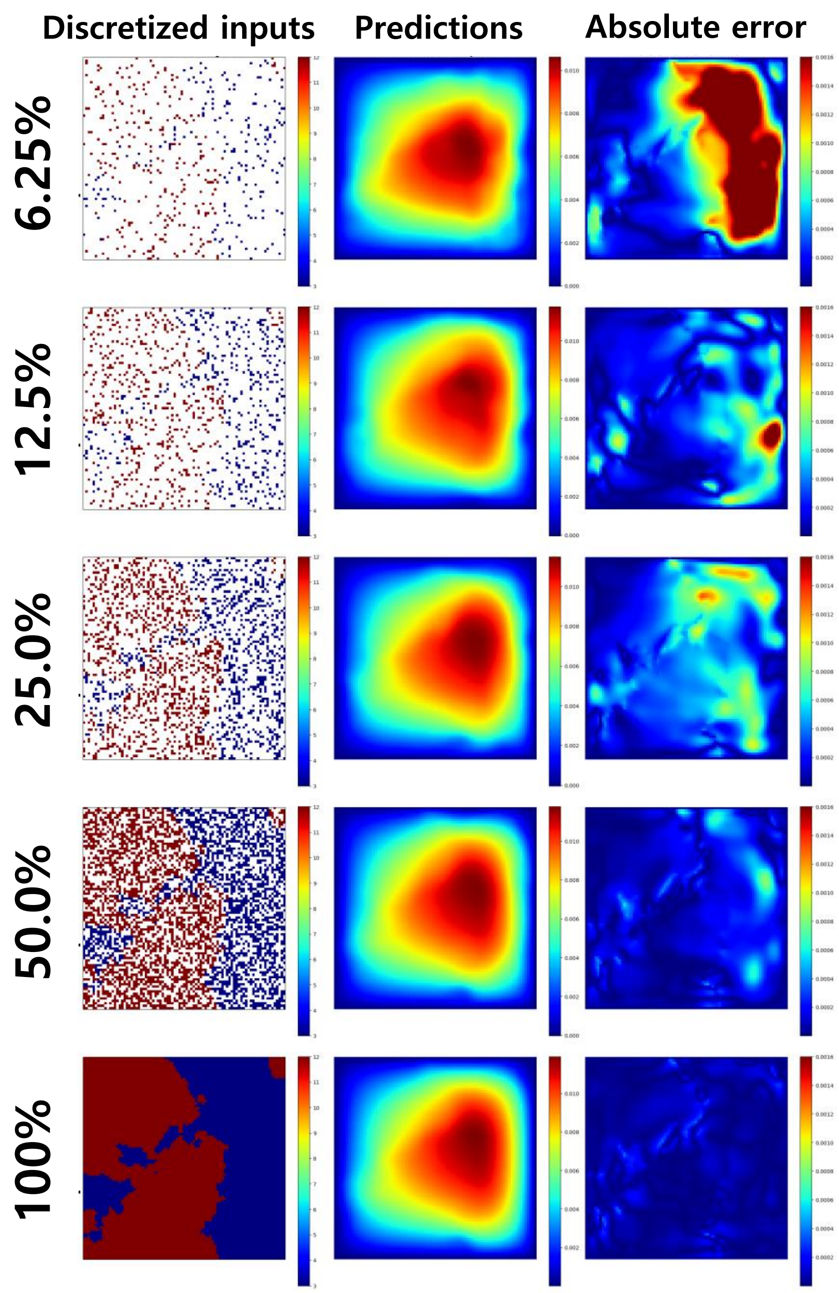
● Discretized inputs

🟥 Predictions

🟩 Truth



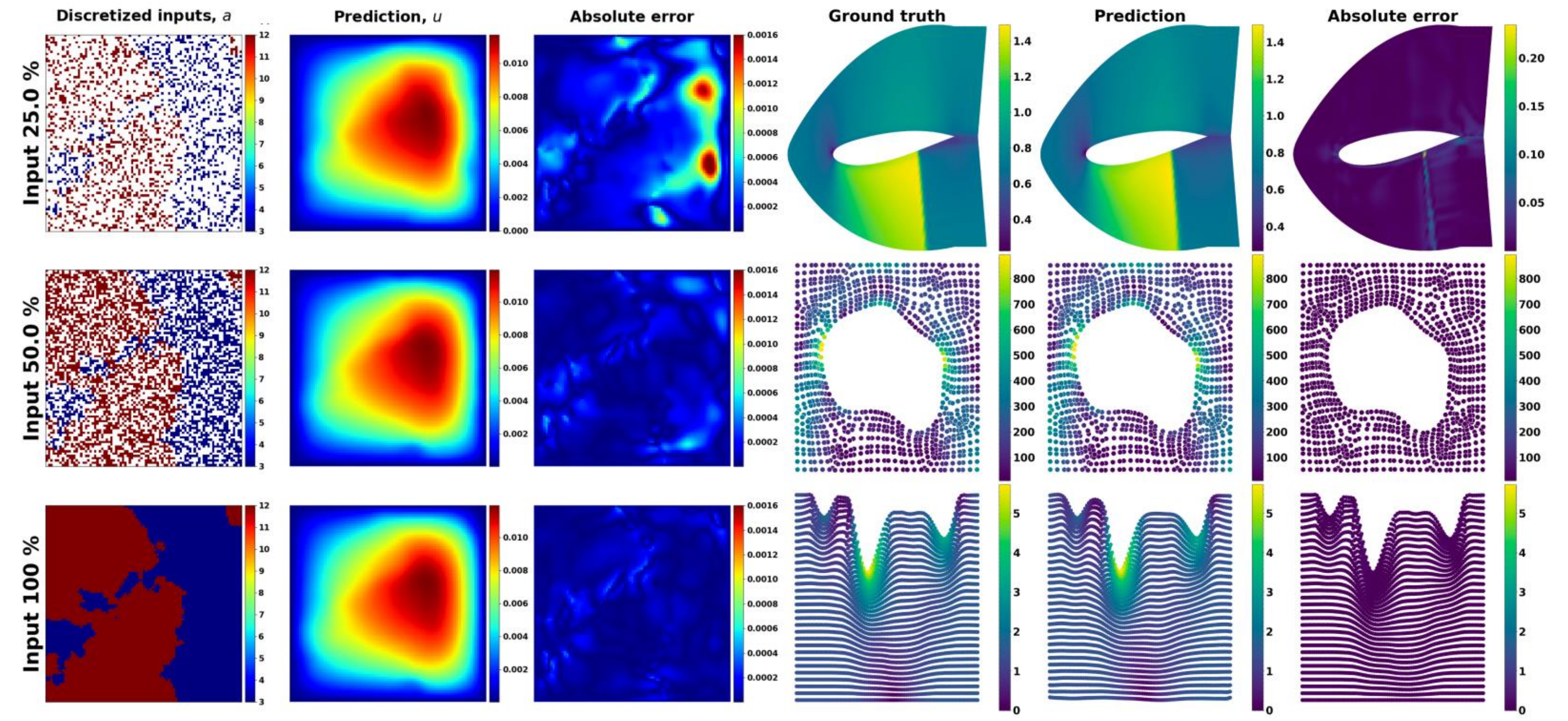
Burgers' equations



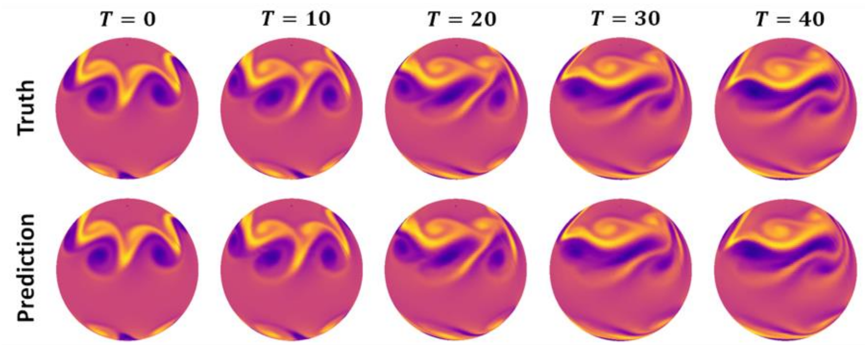
Darcy flows

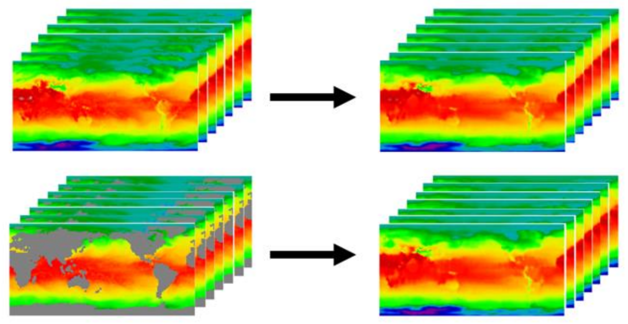
Experiments

- **Flexible to arbitrary discretization formats**



- Predictions of IPOT on the problems of Darcy flow (left), airfoil (top right), elasticity (middle right), and plasticity (bottom right).

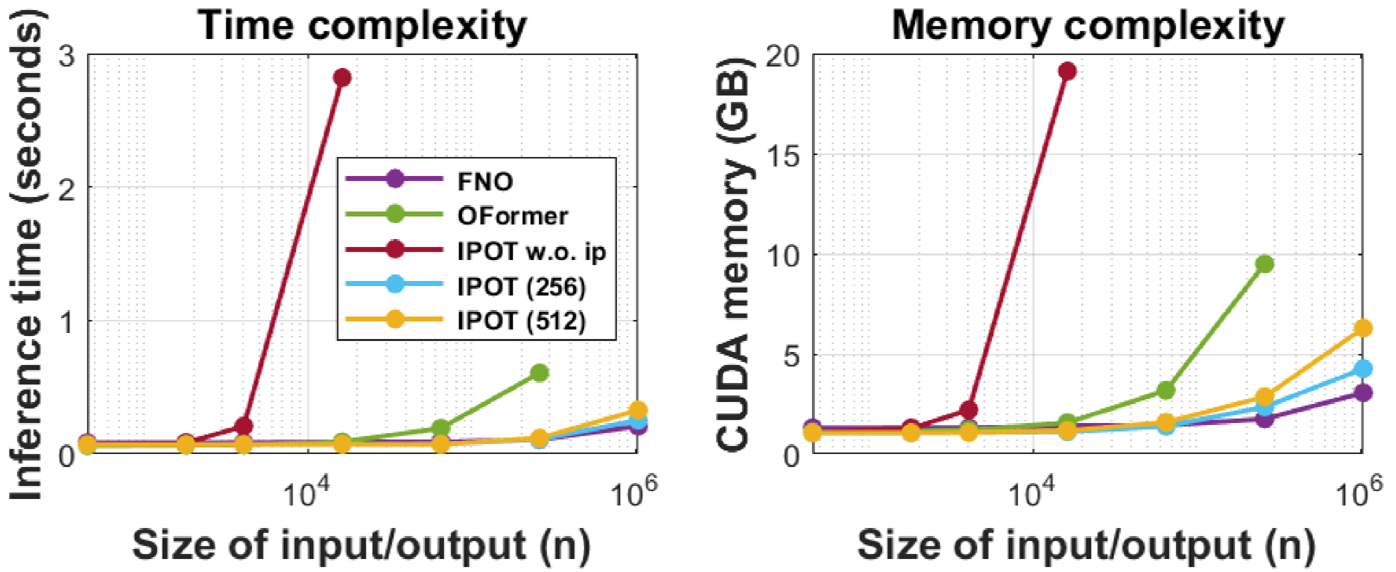




- Long-term predictions of IPOT on spherical manifolds for the shallow-water equation (left), and on real-world weather forecasting when the inputs are spatially fully given (top right), and partially given (bottom right).

Experiments

- **Scalable to large discretization**



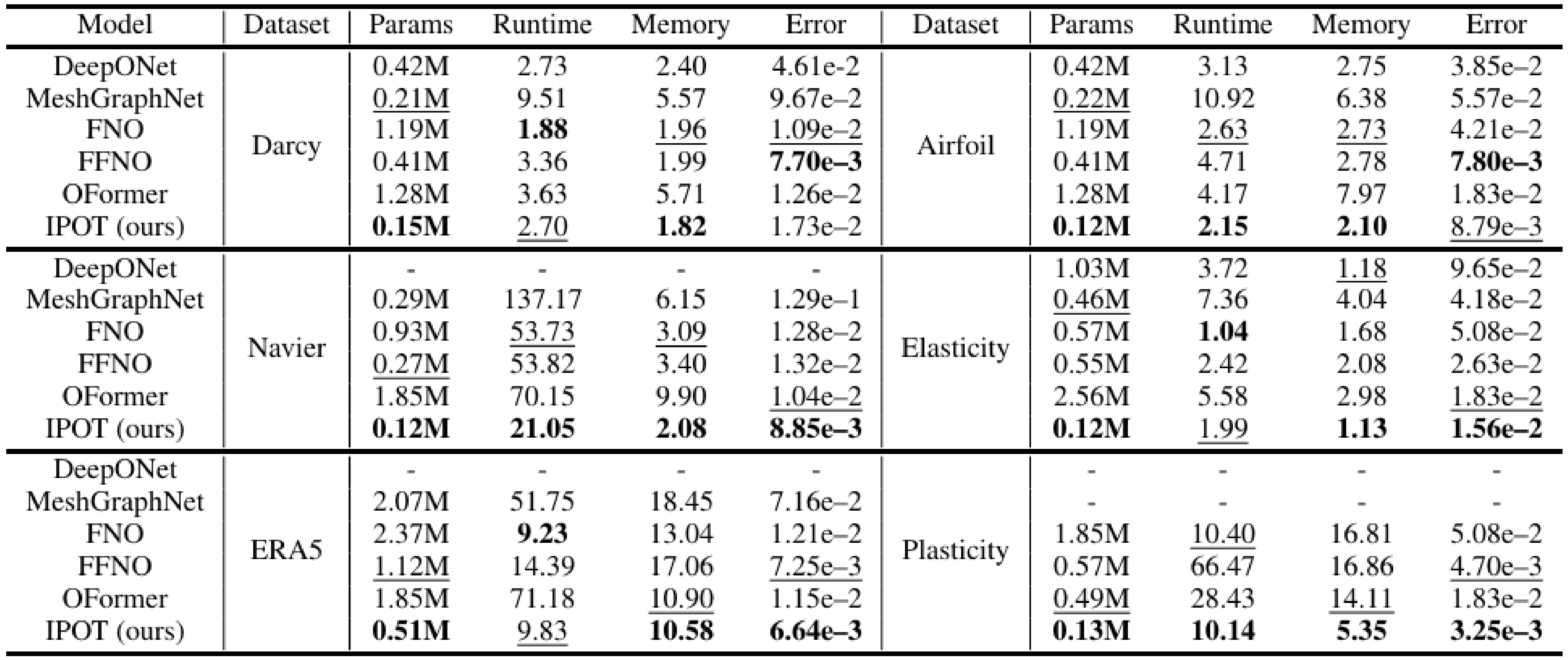
- Complexity comparisons on different re solutions. We compare the different m odels in terms of inference time (left) a nd CUDA memory usage (right) with di fferent sizes of input/output.

Model	n	n_z	L	Runtime	Error	Complexity
OFormer	16.2K	16.2K	19	71.18	1.15e–2	$\mathcal{O}(Lnd^2)$
IPOT w.o ip	16.2K	16.2K	28	$\gg 100$	–	$\mathcal{O}(Ln^2d)$
IPOT (64)	16.2K	64	28	7.44	1.45e–2	$\mathcal{O}(Ln_z^2d)$
IPOT (128)	16.2K	128	28	7.61	1.30e–2	$\mathcal{O}(Ln_z^2d)$
IPOT (256)	16.2K	256	28	7.91	6.87e–3	$\mathcal{O}(Ln_z^2d)$
IPOT (512)	16.2K	512	28	9.83	6.44e–3	$\mathcal{O}(Ln_z^2d)$

- Performance of IPOT with varying the n umber of inducing points. IPOT w.o ip denotes that IPOT without inducing poi nts which emulates the standard Transf ormer.

Experiments

- **IPOT demonstrates great flexibility and scalability compared to state-of-the-art methods.**



Model	Dataset	Params	Runtime	Memory	Error	Dataset	Params	Runtime	Memory	Error
DeepONet	Darcy	0.42M	2.73	2.40	4.61e-2	Airfoil	0.42M	3.13	2.75	3.85e–2
MeshGraphNet		0.21M	9.51	5.57	9.67e–2		0.22M	10.92	6.38	5.57e–2
FNO		1.19M	**1.88**	1.96	1.09e–2		1.19M	2.63	2.73	4.21e–2
FFNO		0.41M	3.36	1.99	**7.70e–3**		0.41M	4.71	2.78	**7.80e–3**
OFormer		1.28M	3.63	5.71	1.26e–2		1.28M	4.17	7.97	1.83e–2
IPOT (ours)		**0.15M**	2.70	**1.82**	1.73e–2		**0.12M**	**2.15**	**2.10**	8.79e–3
DeepONet	Navier	-	-	-	-	Elasticity	1.03M	3.72	1.18	9.65e–2
MeshGraphNet		0.29M	137.17	6.15	1.29e–1		0.46M	7.36	4.04	4.18e–2
FNO		0.93M	53.73	3.09	1.28e–2		0.57M	**1.04**	1.68	5.08e–2
FFNO		0.27M	53.82	3.40	1.32e–2		0.55M	2.42	2.08	2.63e–2
OFormer		1.85M	70.15	9.90	1.04e–2		2.56M	5.58	2.98	1.83e–2
IPOT (ours)		**0.12M**	**21.05**	**2.08**	**8.85e–3**		**0.12M**	1.99	**1.13**	**1.56e–2**
DeepONet	ERA5	-	-	-	-	Plasticity	-	-	-	-
MeshGraphNet		2.07M	51.75	18.45	7.16e–2		-	-	-	-
FNO		2.37M	**9.23**	13.04	1.21e–2		1.85M	10.40	16.81	5.08e–2
FFNO		1.12M	14.39	17.06	7.25e–3		0.57M	66.47	16.86	4.70e–3
OFormer		1.85M	71.18	10.90	1.15e–2		0.49M	28.43	14.11	1.83e–2
IPOT (ours)		**0.51M**	9.83	**10.58**	**6.64e–3**		**0.13M**	**10.14**	**5.35**	**3.25e–3**

Table 1: Performance and efficiency comparisons with baselines across various datasets. The efficiencies are compared in terms of the number of parameters, time spent per epoch (seconds), and CUDA memory consumption (GB). The missing entries occur when the methods are not able to handle the datasets or when encountering convergence issues.

Summary

Summary

- Since the physical systems are usually continuous, it is required to build discretization-invariant mode ls that impose few assumptions on the data structure.
- Our proposed architecture is flexible in handling arbitrary discretization formats and scalable to large discretization.
- It is important to balance accuracy and complexity effectively according to the problems.
- [From experience on operator learning] When moving beyond the benchmarks to the foundation mo del, it is crucial to building datasets that can cover the potential test set carefully and to investigate basic knowledge (bandwidth, scales, eigenbasis, geometries, …) in advance.

Thank you