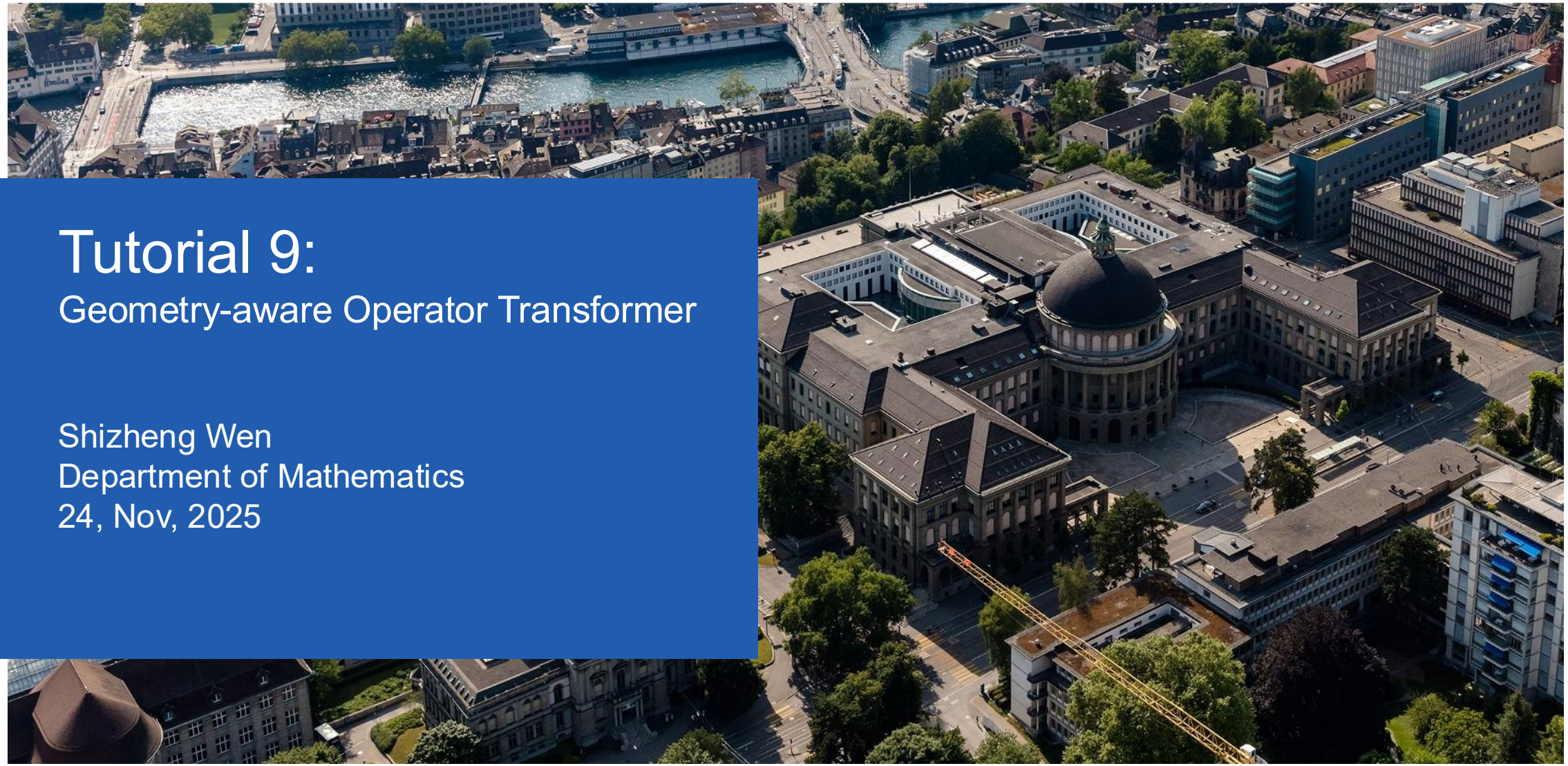
ETH zürich

CAMLab



Tutorial 9:

Geometry-aware Operator Transformer

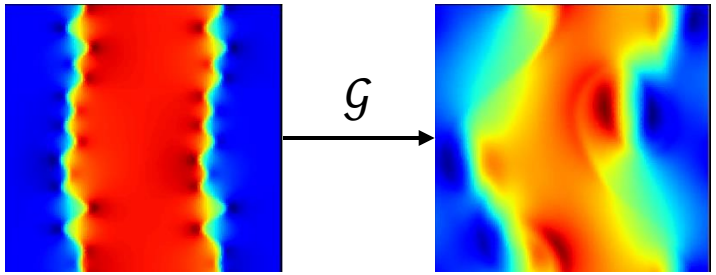
Shizheng Wen
Department of Mathematics
24, Nov, 2025

SAM Seminar for Applied Mathematics

Motivation

The neural operator discussed so far are all structured.

In real scenarios, the interesting problem is all unstructured.

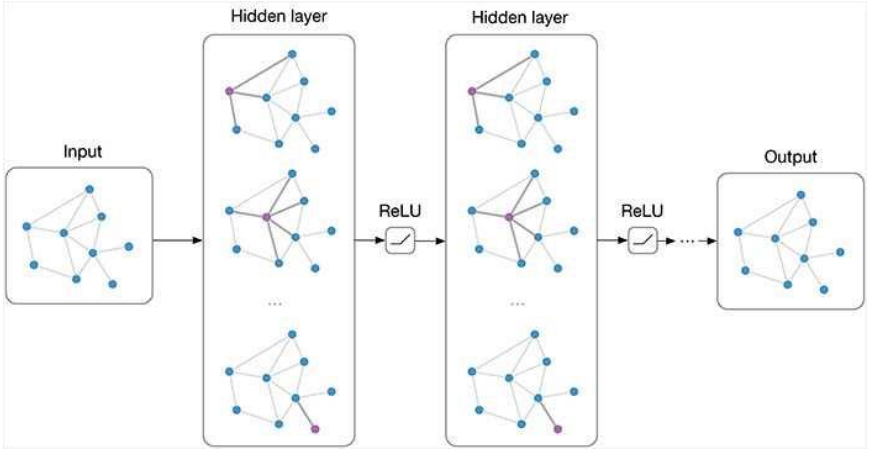


FNO, CNO, VIT, Unet, ScoT

Unstructured Mesh

Motivation

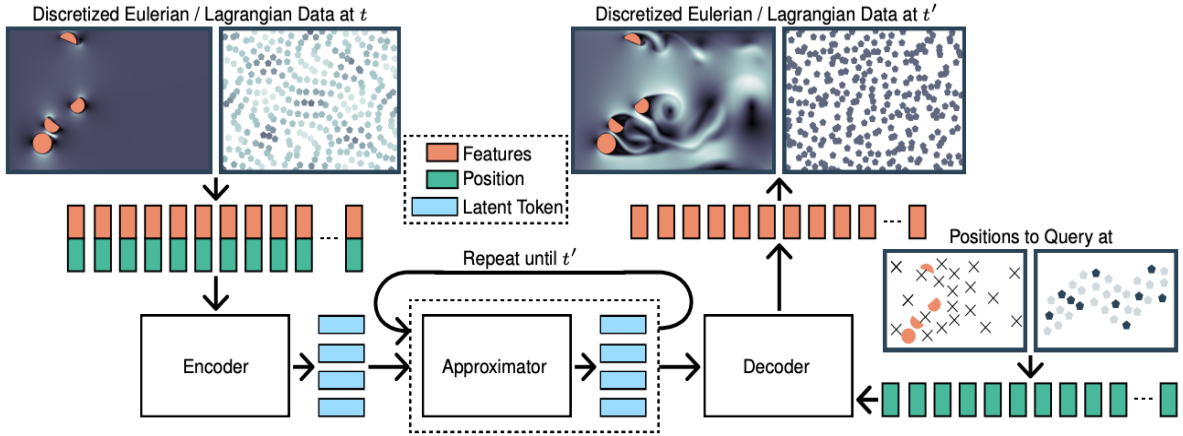
Graph-based Neural Network
Explicit Modeling



RIGNO, GNO, MPNPDE, MeshgraphNet

Low efficiency and not scalable

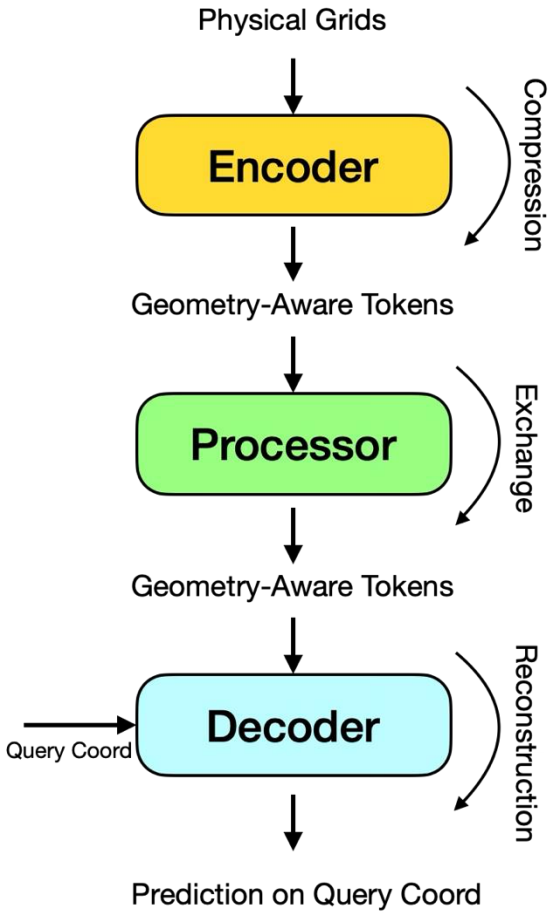
Encoder-Processor-Decoder
Implicit Modeling

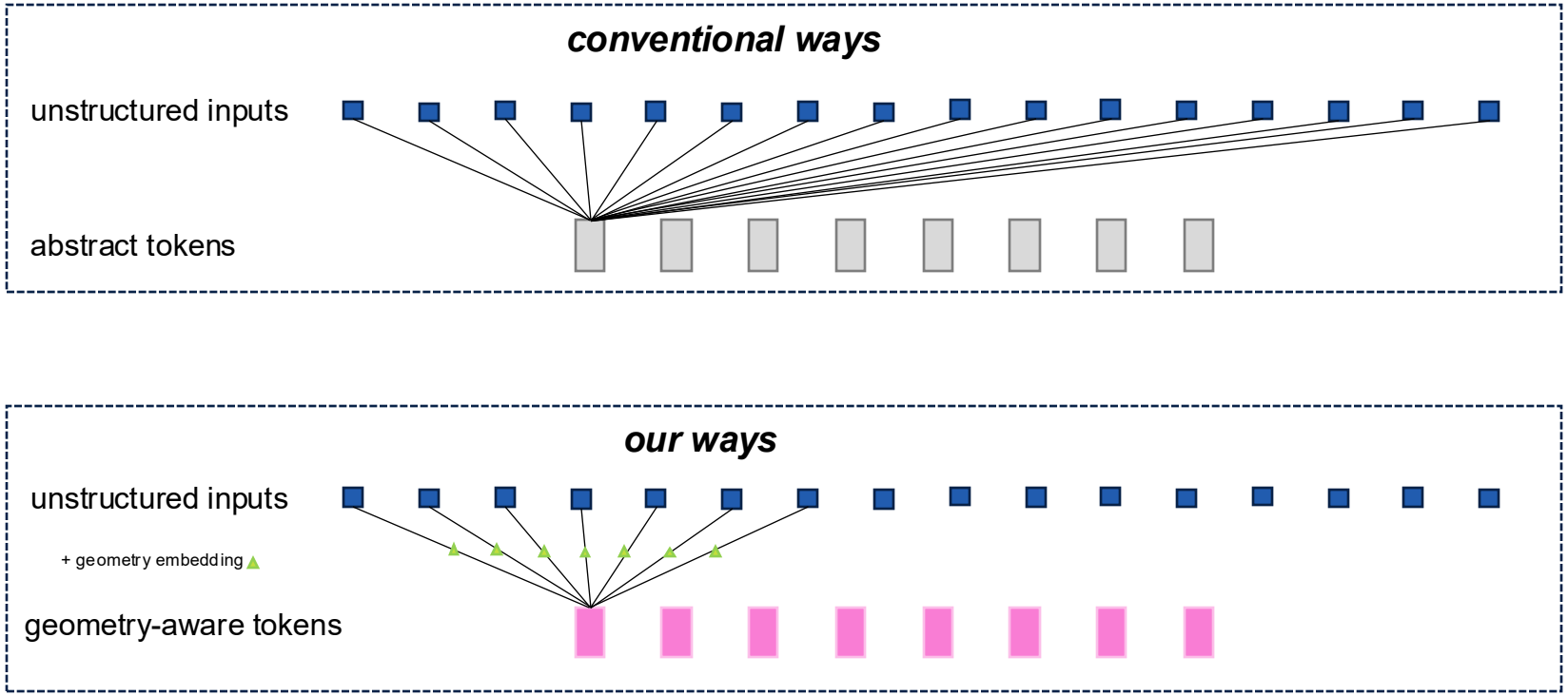


UPT, GNOT, Oformer, GT

Abstract tokens: no physics priors

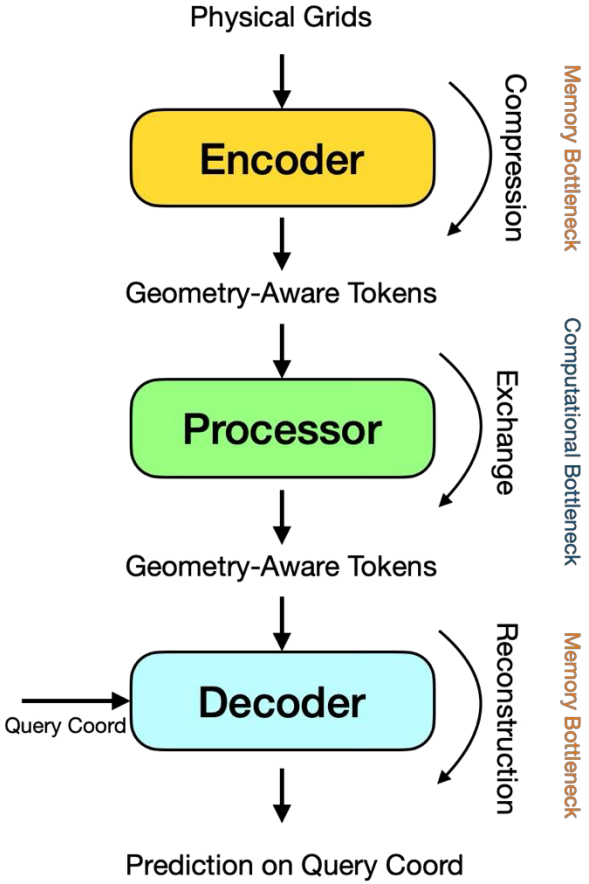
Overview of GAOT

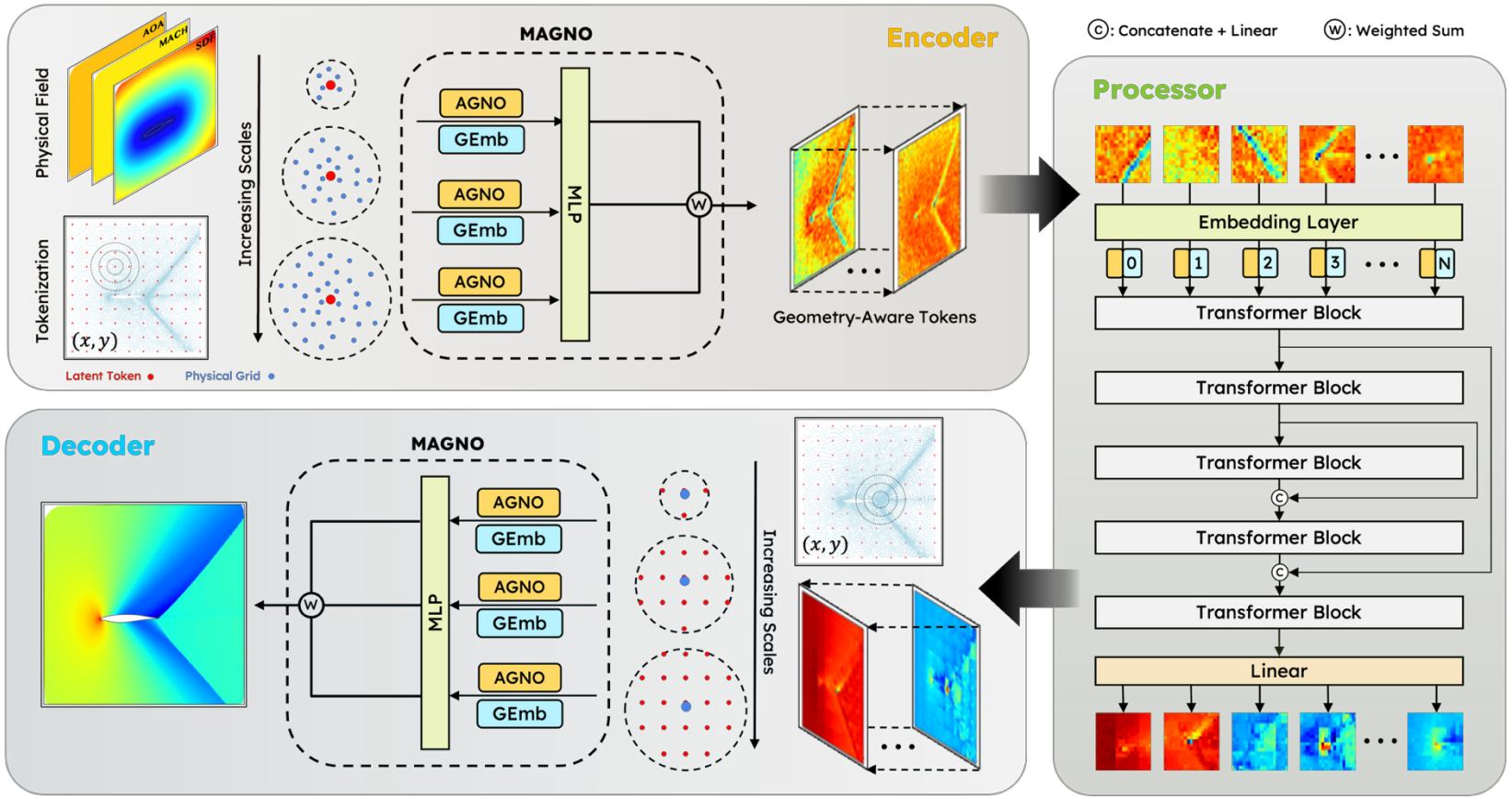




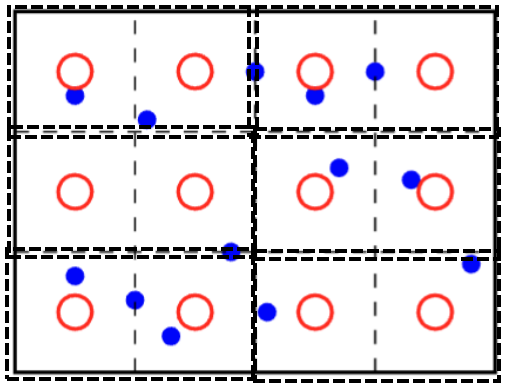
- The compression of information is based on geometric proximity.
- Geometric embedding further enhances the geometry awareness of these tokens.

Overview of GAOT





Tokenization

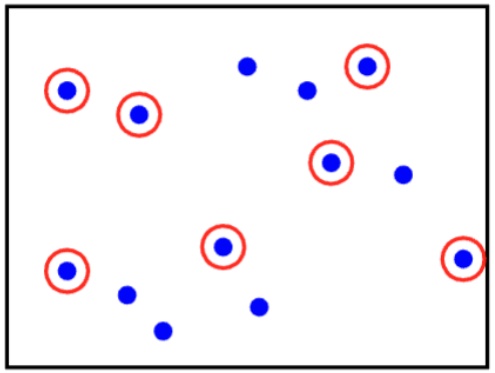


(a) Strategy I

Structured Stencil Grid

Merits:
- Adequate Coverage
- Patches for quicker computation

Demerits:
- Data in low-dimensional manifold?
- Computational heavy for 3D

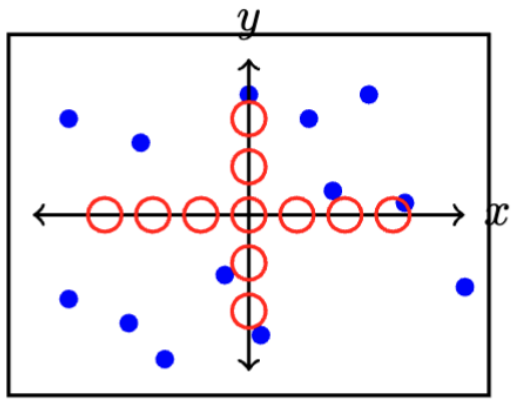


(b) Strategy II

Down-sampled Unstructured Grid

Merits:
- Efficient Token
- Suitable for data in low-dimensional manifold

Demerits:
- Unable for patches



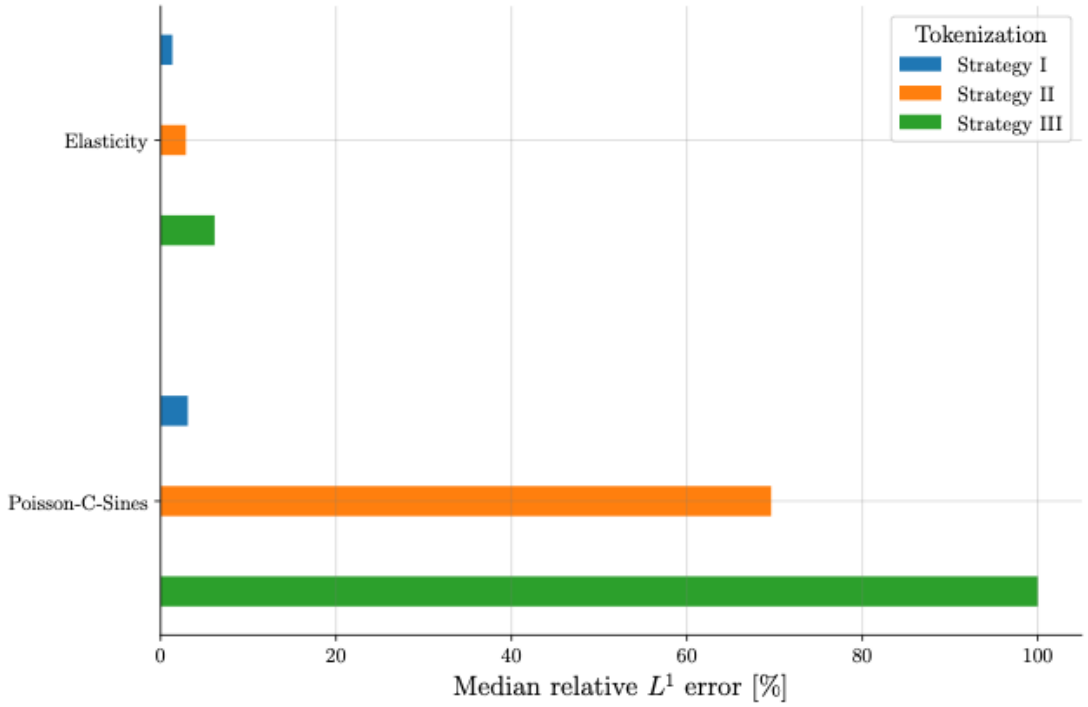
(c) Strategy III

Projected Low-Dimensional Grid

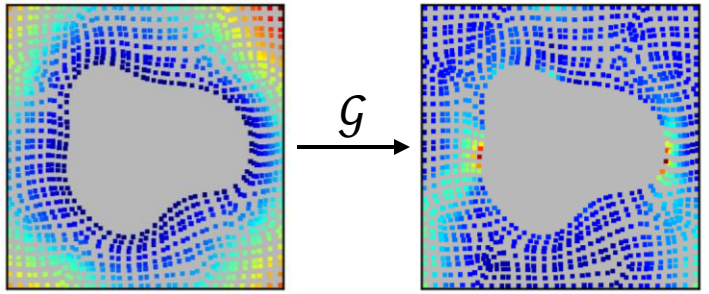
Merits:
- Efficient for 3D data point (Triplane)
- Avoid the data in low-dimensional manifold

Demerits:
- Disjoint projections introduce additional errors

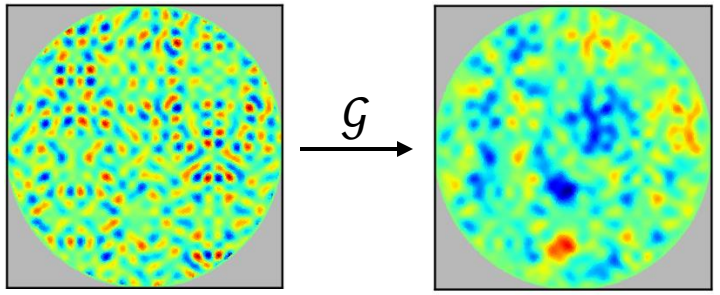
Tokenization



elasticity

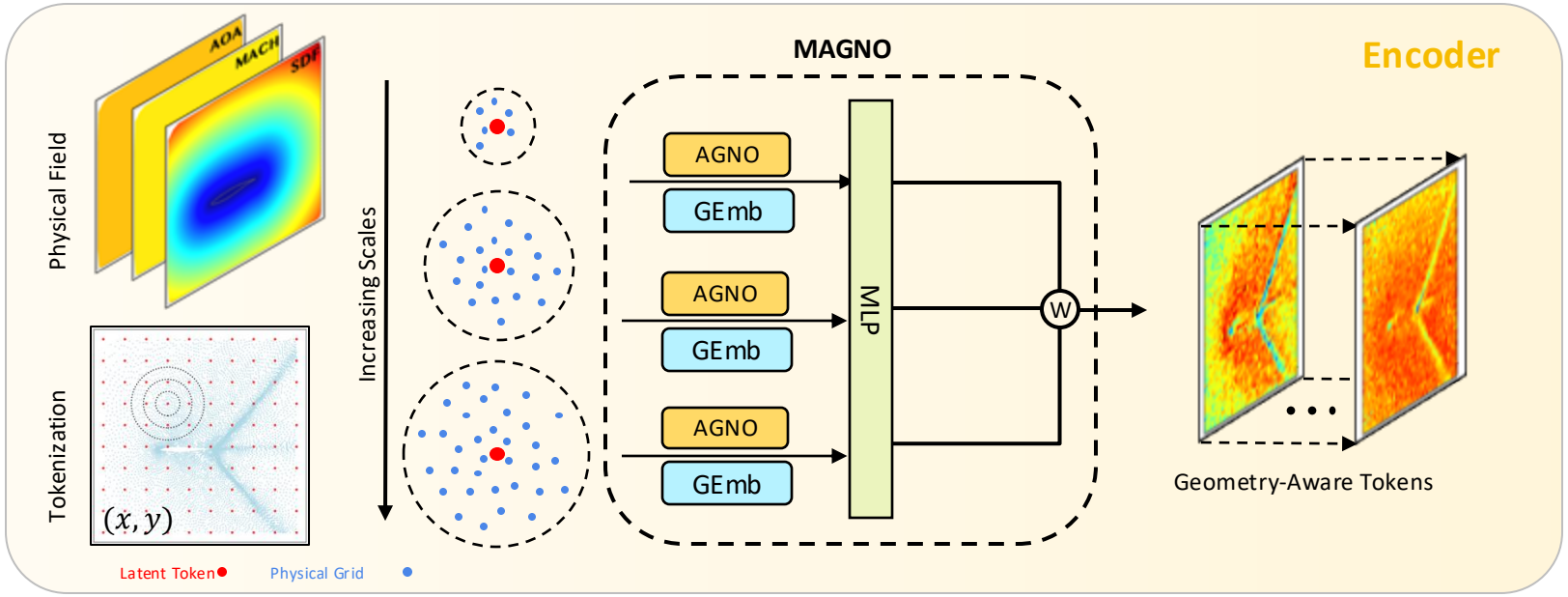


Poisson-C-Sines



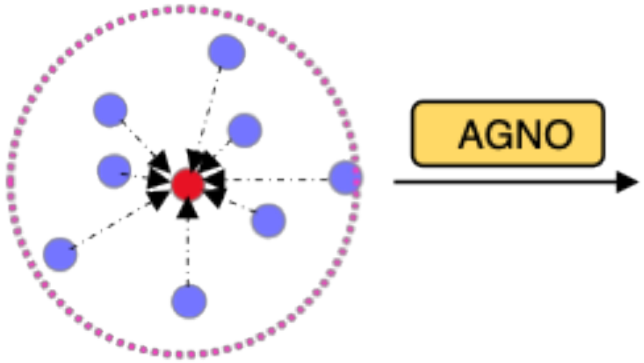
- Are these void tokens really useless?

Multiscale Attentional Graph Neural Operator



- Attentional Graph Neural Operator
- Geometric Embedding
- Multiscale Features

Multiscale Attentional Graph Neural Operator (MAGNO)



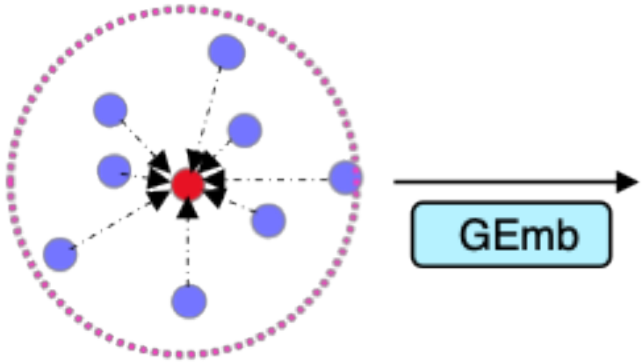
Trick 1: Attentional Weighting in Local Integration

$$(\mathcal{L}_r f)(x) = \int_{A_r(x)} K_\ell(x, y, f(y))\, \varphi(f(y))\, \mathrm{d}y \approx \sum_{i=1}^{n_y} \alpha_i K_\ell\big(x, y_i, f(y_i)\big)\, \varphi\big(f(y_i)\big).$$

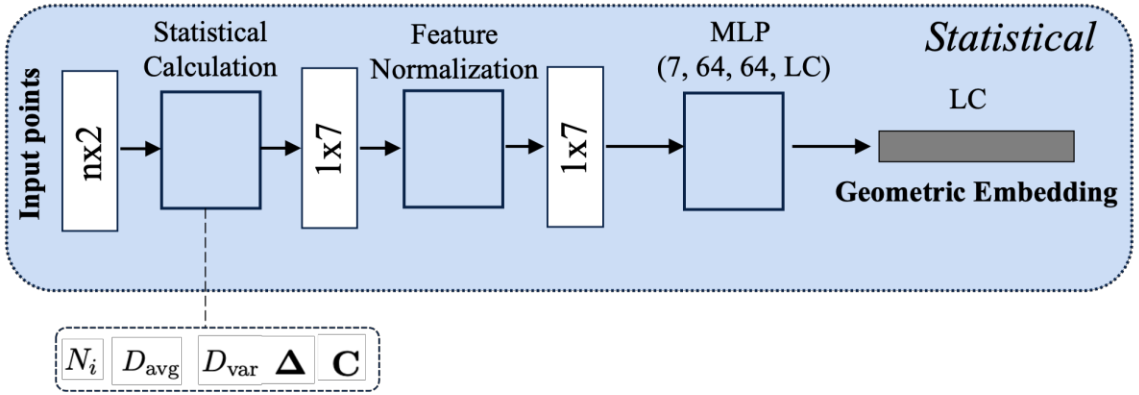
$$\mathbf{q} = W_q\, \mathbf{x} \qquad \mathbf{k}_i = W_k\, \mathbf{y}_i \qquad e_i = \frac{\langle \mathbf{q}, \mathbf{k}_i \rangle}{\sqrt{d}} \qquad \alpha_i = \frac{\exp(e_i)}{\sum_{j=1}^{n_y} \exp(e_j)}.$$

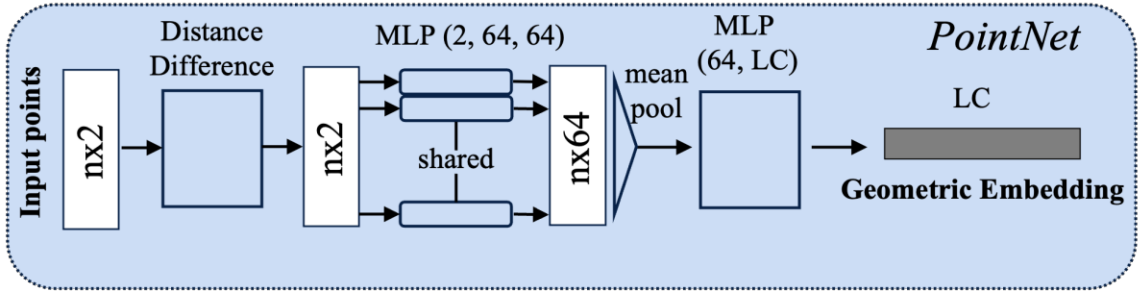
- Introduce learning components in determining the importance of neighboring nodes.

Multiscale Attentional Graph Neural Operator (MAGNO)



Trick 2: Geometric Embedding

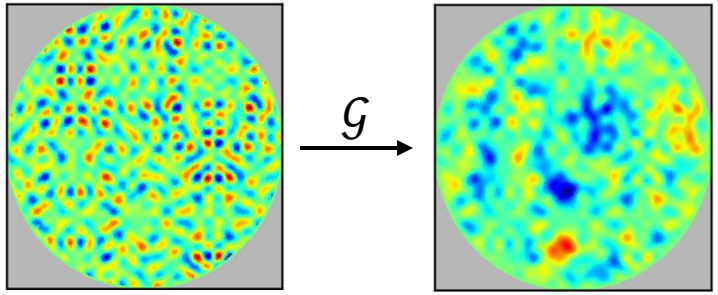




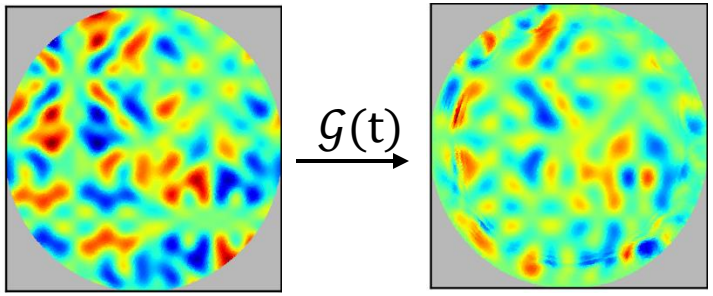
Geometric Embedding

Dataset	Median relative L^1 error [%]		
	original	statistical	pointnet
Wave-C-Sines	6.50	5.69	6.07
Poisson-C-Sines	6.60	4.66	23.7

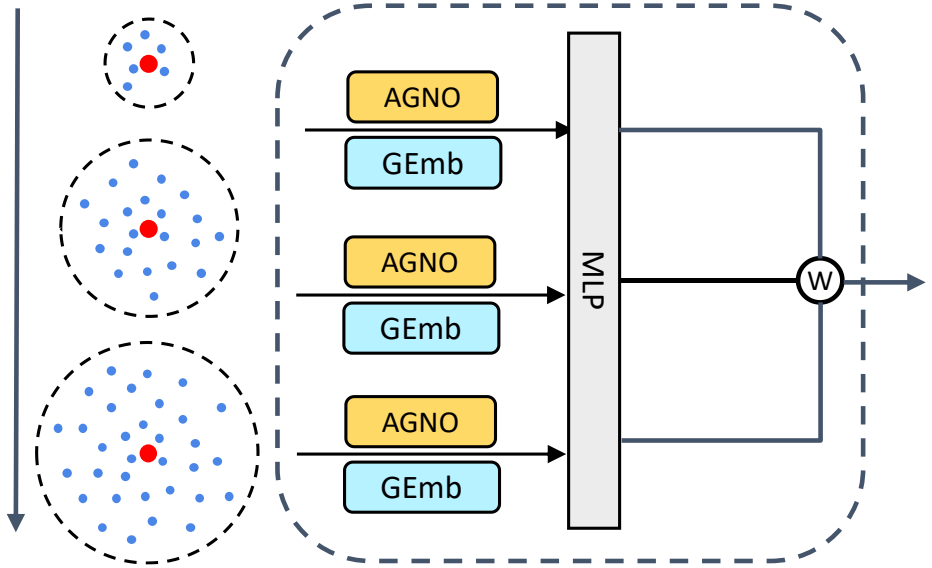
Poisson-C-Sines



Wave-C-Sines



Multiscale Attentional Graph Neural Operator (MAGNO)



Trick 2: Multiscale Neighborhood Construction + self-adaptive weighting sum

$$A_{r_1}(x), \quad A_{r_2}(x), \quad \ldots, \quad A_{r_M}(x) \qquad \left(\mathcal{L}_{\text{multi}} f\right)(x) = \sum_{m=1}^{M} \beta_m(x) \left(\mathcal{L}_{r_m} f\right)(x).$$

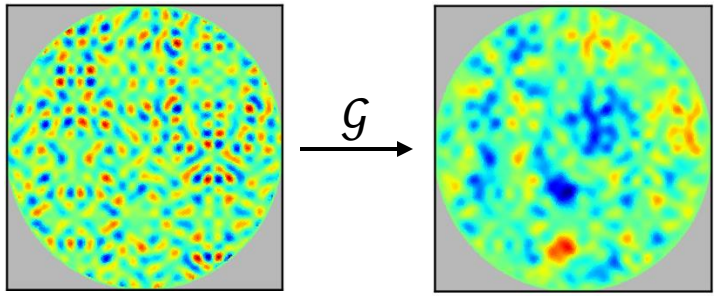
$$\boldsymbol{g}(x) = \text{MLP}_\alpha(\mathbf{x}) \qquad \beta_m(x) = \frac{\exp\left(g_m(x)\right)}{\sum_{m'=1}^{M} \exp\left(g_{m'}(x)\right)}, \quad m = 1, \ldots, M.$$

- Introduce learning components in determining the importance of different scales features.
- Analogy of the different size of kernels in convolutional neural network.

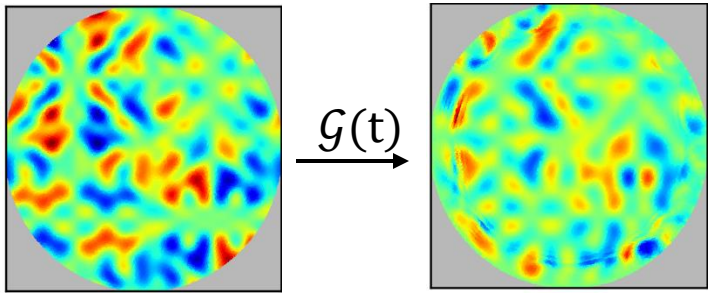
Multiscale Neighborhood Construction

Dataset	Median relative L^1 error [%] Single-scale	multiscale
Wave-C-Sines	5.69	4.6
Poisson-C-Sines	4.66	3.04

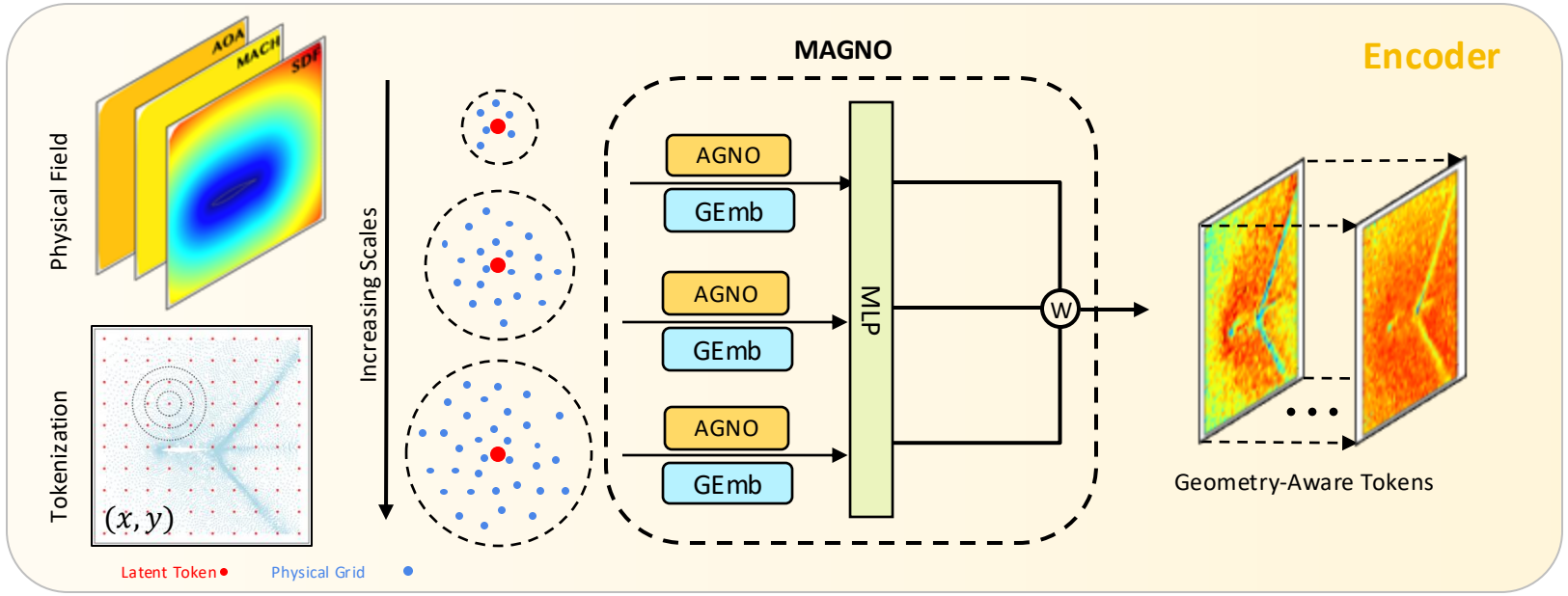
Poisson-C-Sines



Wave-C-Sines

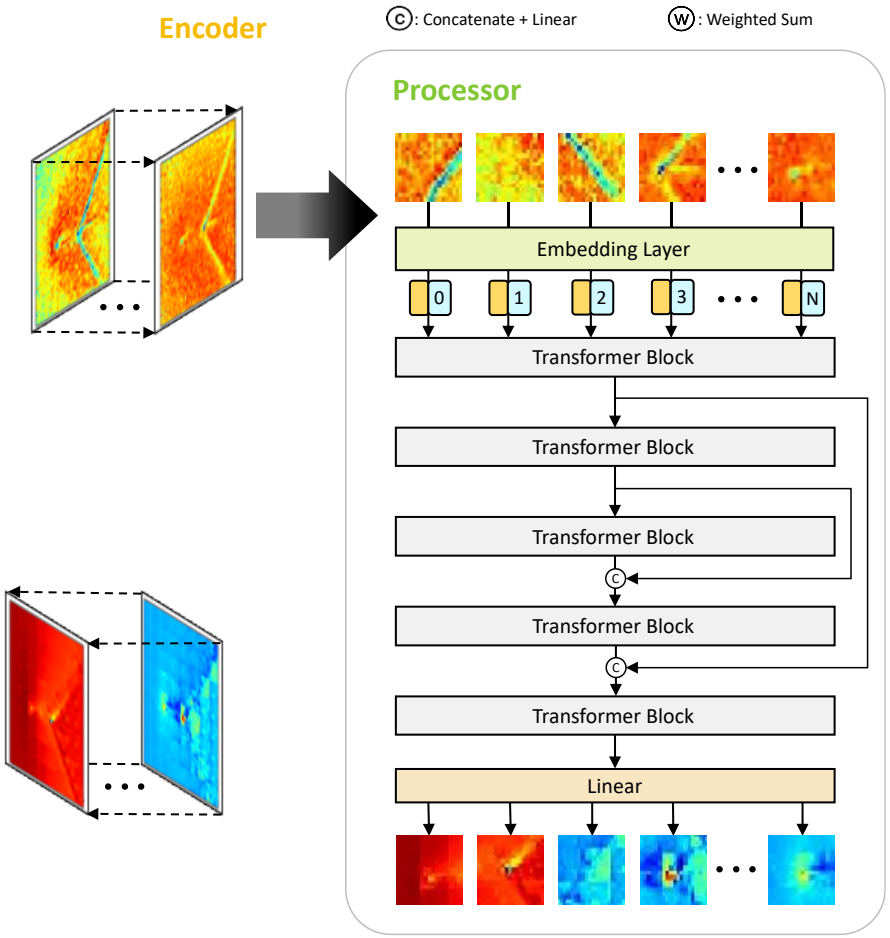


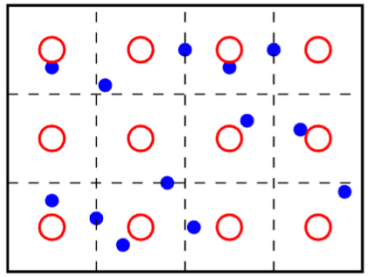
Encoder: Multiscale features compression



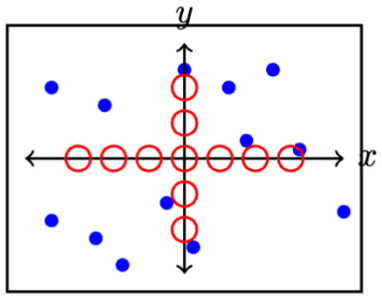
- Similarly, you can have other modal-specific modules for adding multi-modal information into tokens.

Processor





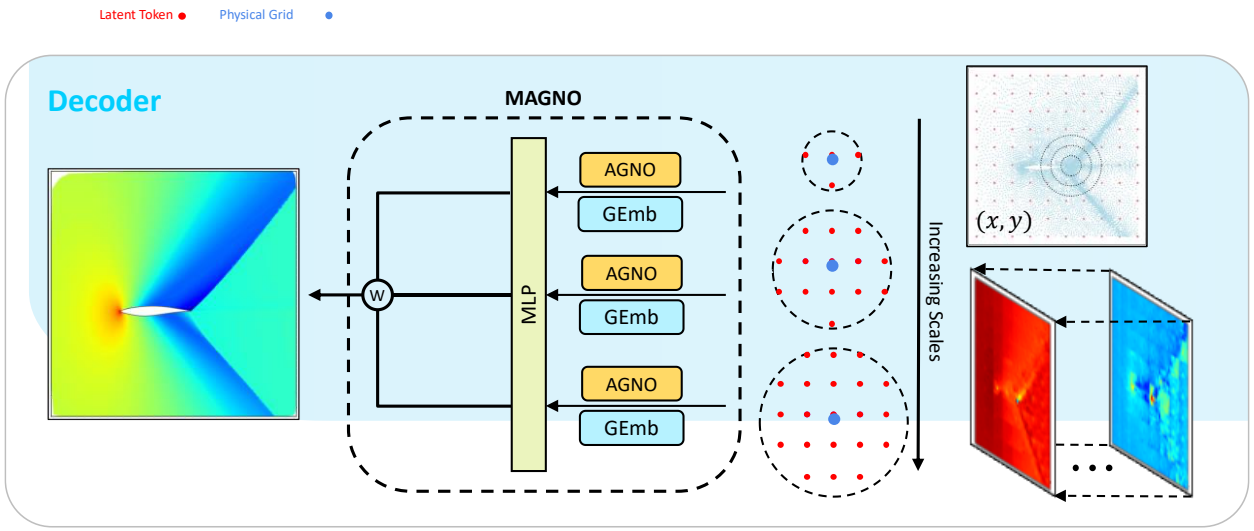
(a) *Strategy I*



(c) *Strategy III*

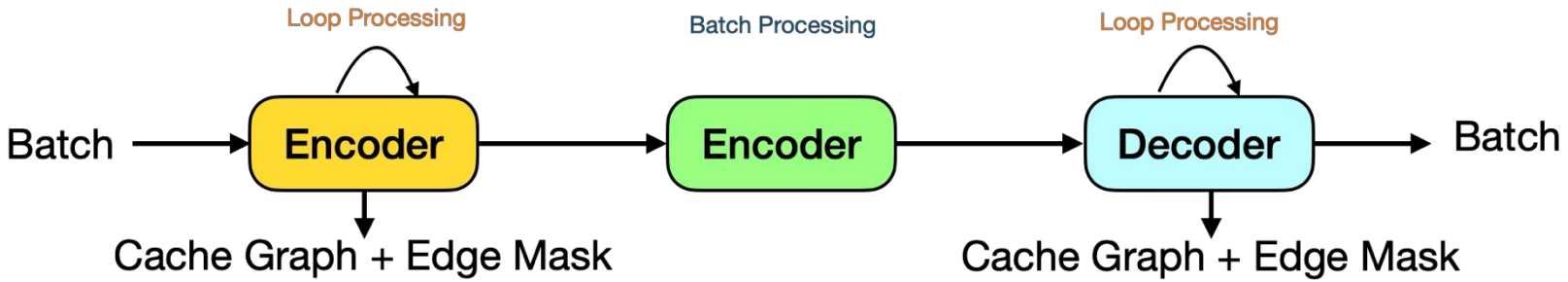
- Transformer backbone for global interaction
- Patches for *Strategy I* and *III*
- Residual connection to avoid gradient vanishing

Decoder



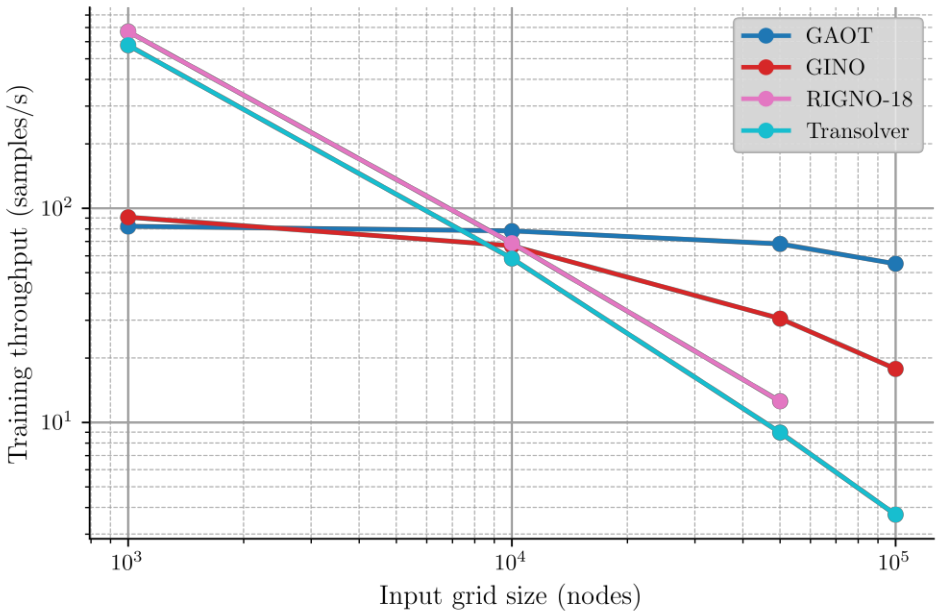
- Similar as Encoder
- Decode the solution Field based on locality
- Neural Field Properties

Efficient Implementation

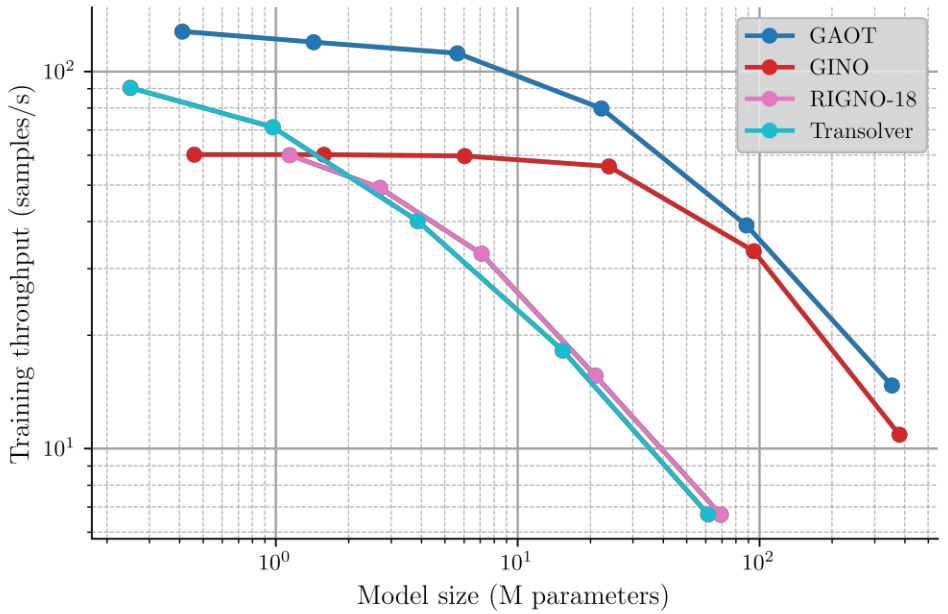


- Lightweight graph-based encoder/decoder (sparse operations) + structured-grid transformer processor (computational-intensive).
- Loop processing for encoder and decoder (alleviate memory bottlenecks) + batch processing for processor (improving training throughput).
- Edge mask to alleviate memory bottlenecks and enhance generalization.
- Graph Cache to avoid redundant graph construction.

Scalable Experiments



Grid vs. Throughput



Model vs. Throughput

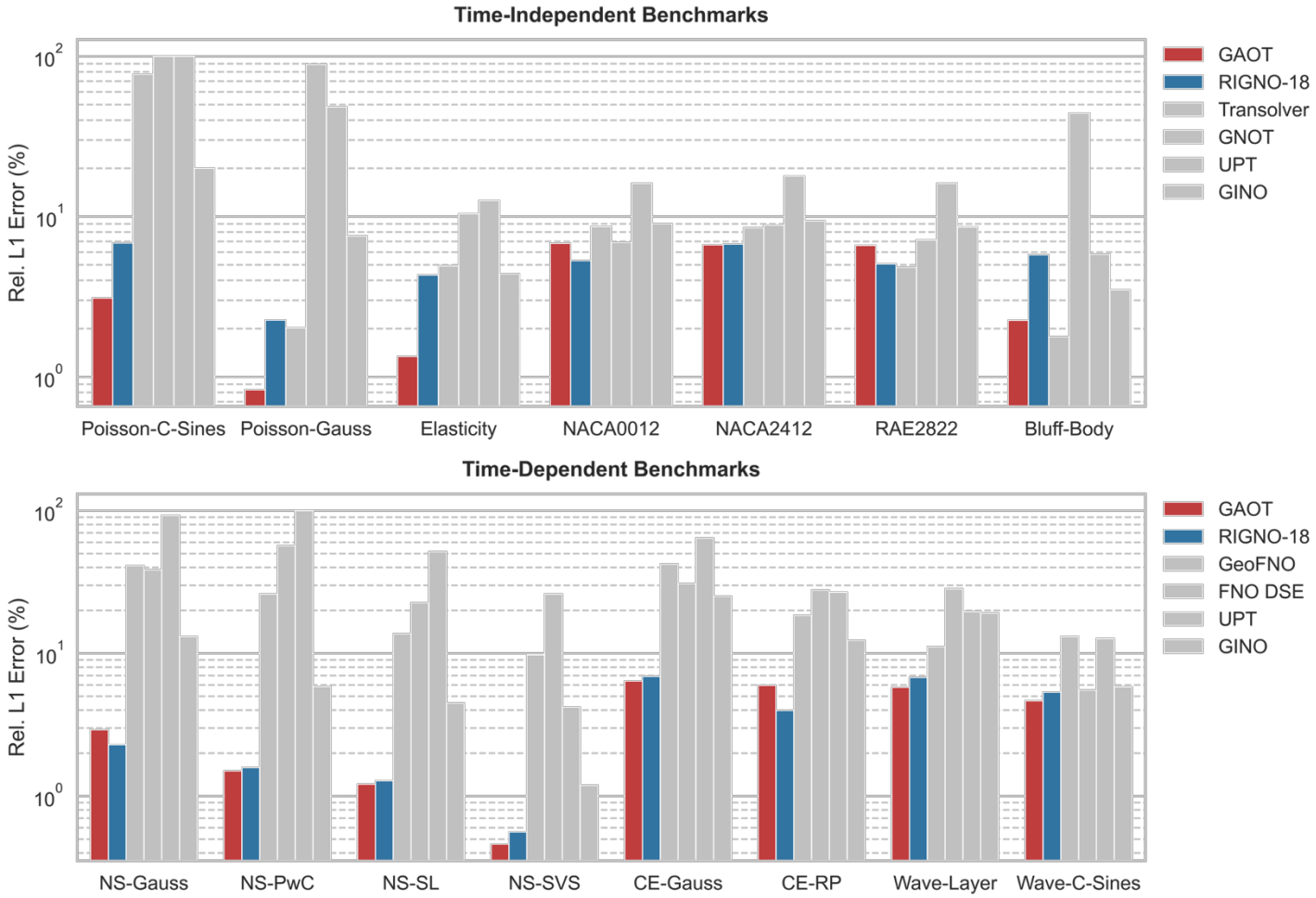
Training throughput (samples/s) with increasing input grid size and model size

- GAOT scales more favorably than baselines (FNO-based, Graph-based and Transformer-based) with respect to both input and model size.

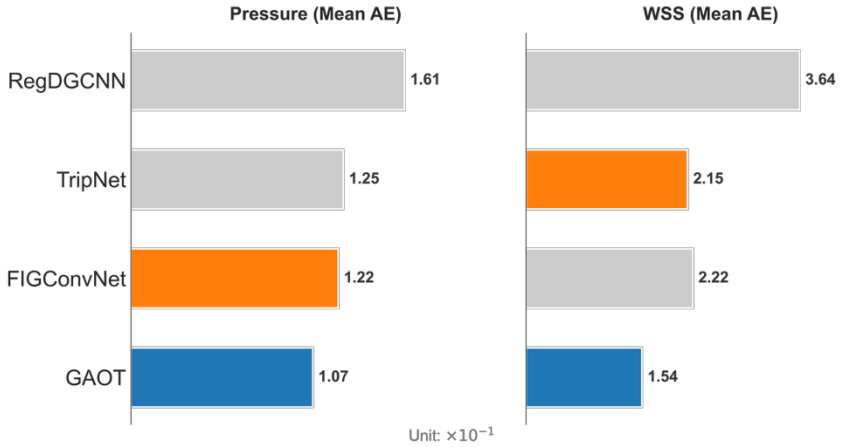
Main Results

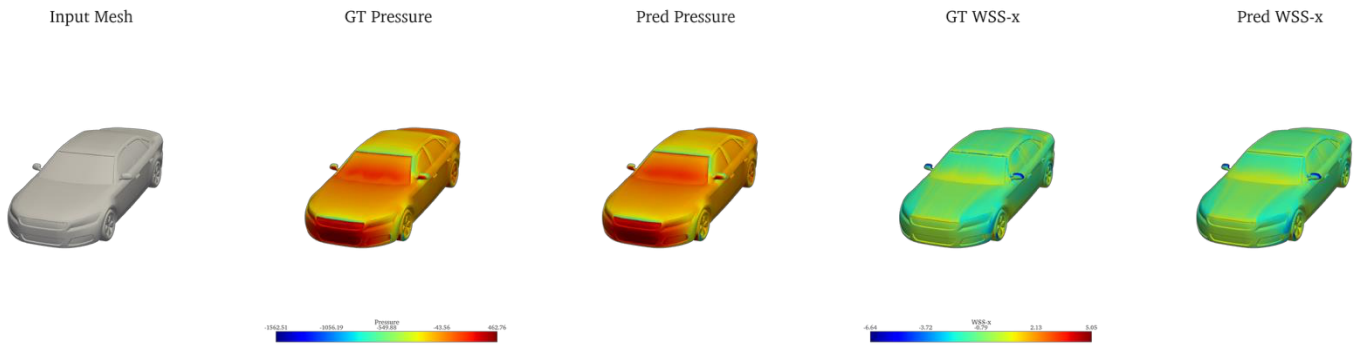
Shows GAOT is very accurate, being either the best (10) or second-best (4) model on 14 benchmarks.

- On average, over the time-independent datasets, GAOT is almost 50% more accurate than the second-best performing model (RIGNO-18) while on time-dependent datasets, it is slightly more accurate than the second-best performing model (RIGNO-18).
- GAOT has a robust performance for all the datasets.



Industrial 3D dataset – DrivAerNet++

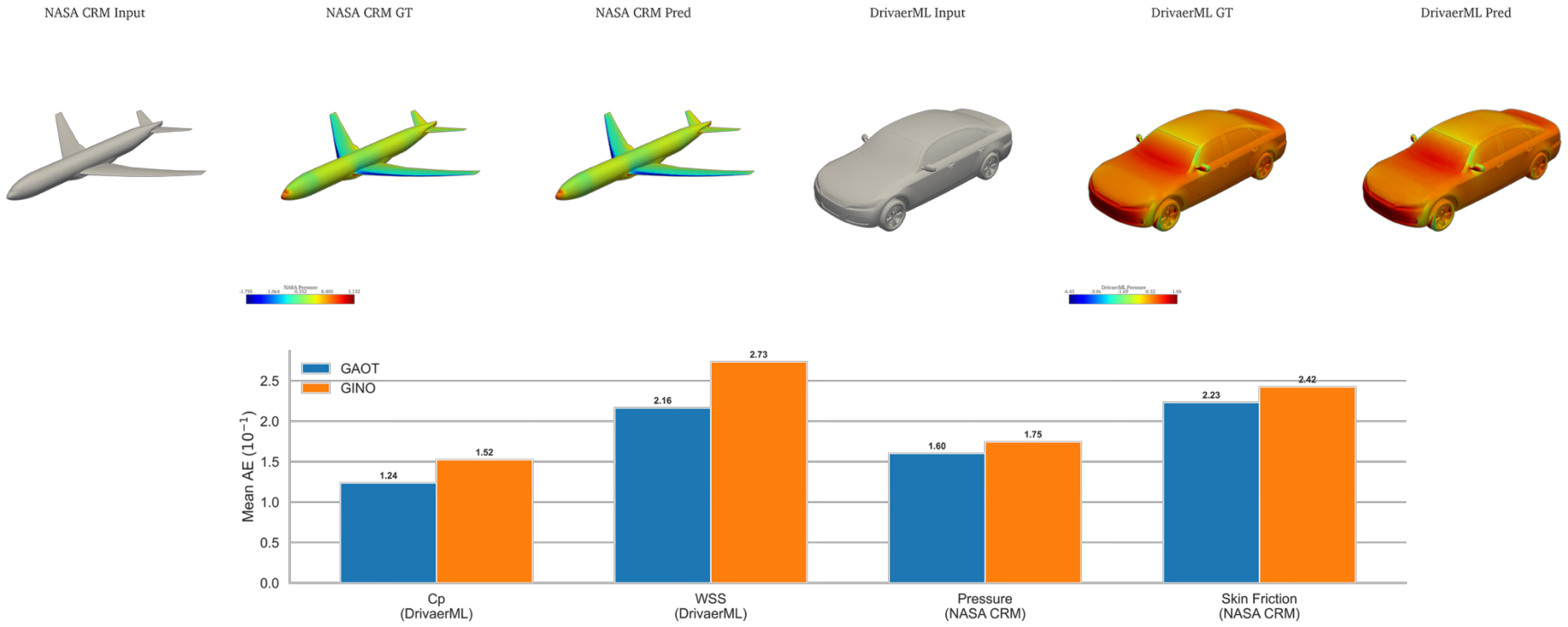




Dataset of CFD simulations (RANS) with 8000 different car shapes. Each shape has 500k mesh cells/points.

- We follow the setup from leaderboard of the DrivAerNet++ challenge and rank first in predictions of surface pressure and wall shear stress.

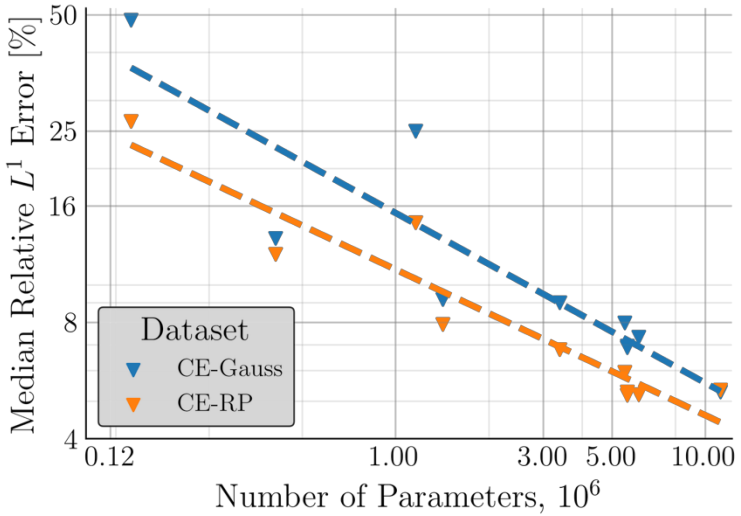
Industrial 3D dataset – DrivAerML and NASA CRM



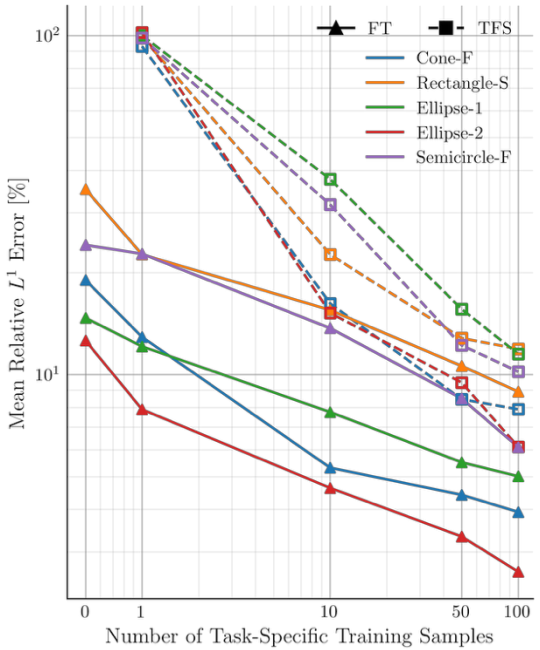
Industrial 3D dataset for automotive (DrivAerML) and aerospace (NASA-CRM) applications:

- NASA-CRM: RANS simulations using a Spalart-Allmaras turbulence model
- DrivAerML: highly accurate LES simulations. Each Car shape with ~9M surface points. GAOT can train on full surface resolution.

Scaling Properties

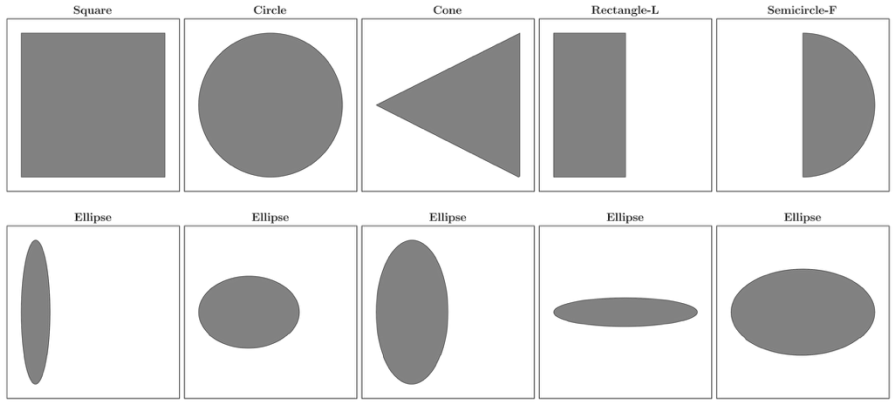


Error vs. Model Size

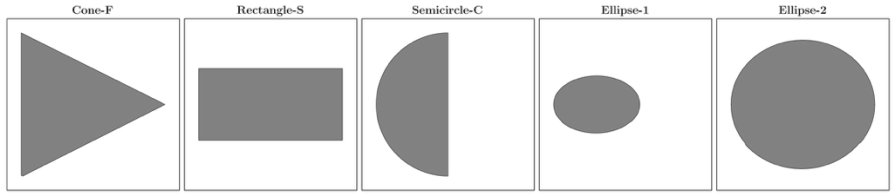


Error vs. Data Size

Transfer Learning (TL)

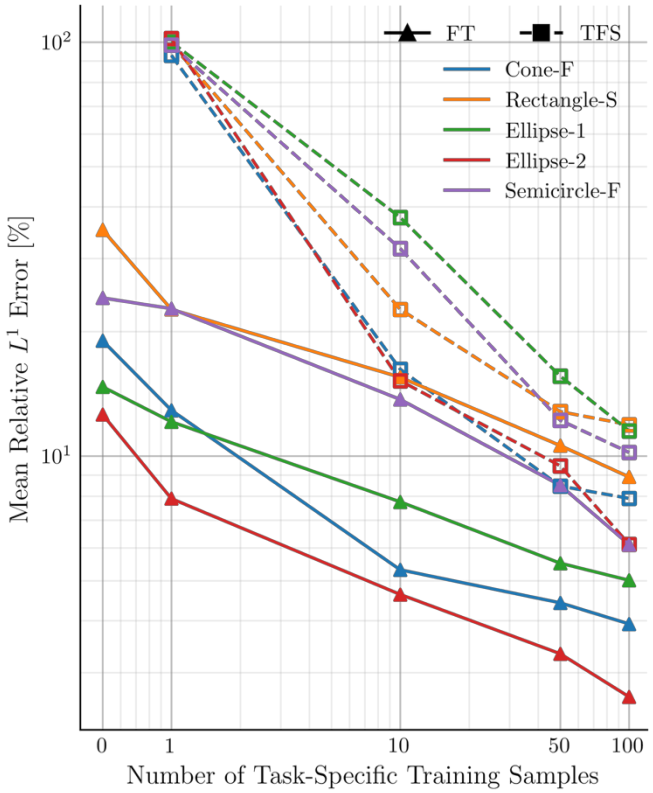


Pre-training shapes (Bluff-Body)

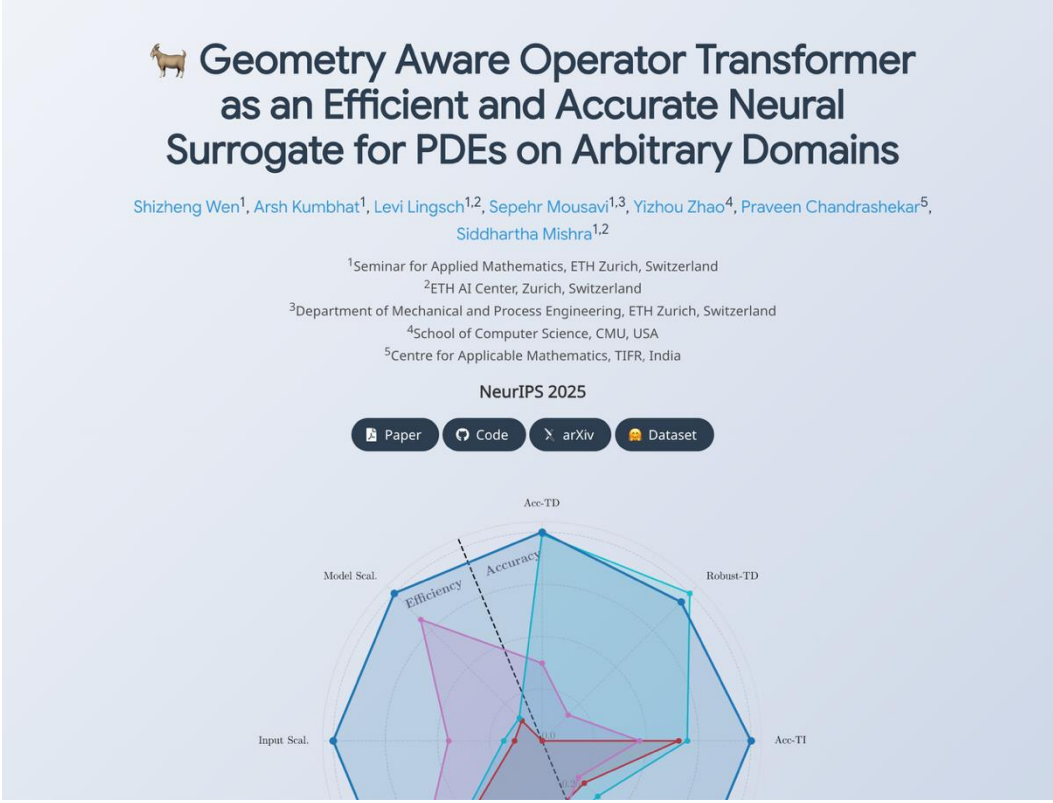


Fine-tuning shapes (Bluff-Body)

TL vs TFS (train from scratch)



Project Page



https://camlab-ethz.github.io/GAOT/