Geometric Operator Learning with Optimal Transport

Xinyi Li
California Institute of Technology
Los Angeles, CA 91125
xinyili@caltech.edu

Zongyi Li
California Institute of Technology
Los Angeles, CA 91125
zongyili@caltech.edu

Nikola Kovachki
Nvidia
nkovachki@nvidia.com

Anima Anandkumar
California Institute of Technology
Los Angeles, CA 91125
anima@caltech.edu

Abstract

We propose integrating optimal transport (OT) into operator learning for partial differential equations (PDEs) on complex geometries, for applications in fluid dynamics. Our approach generalizes discretized meshes to mesh density functions, formulating geometry embedding as an OT problem that maps these functions to a uniform density in a reference space. Unlike prior methods that use shared deformation, our OT-based method employs instance-dependent deformation, providing enhanced flexibility and effectiveness. For 3D surface simulations, our neural operator embeds the geometry into a 2D parameterized latent space. By performing computations directly on this 2D representation, the method achieves significant computational efficiency gains over volumetric simulation. Experiments with RANS on the ShapeNet-Car and DrivAerNet-Car datasets show improved accuracy and reduced computational expenses in both time and memory. Additionally, our model shows significantly improved accuracy on the FlowBench dataset, highlighting the benefits of instance-dependent deformation for highly variable geometries.

1 Introduction

Handling complex geometric domains in partial differential equations (PDEs) is a fundamental challenge in scientific computing, particularly for fields like fluid dynamics where simulations on geometries like automobiles can take hundreds of hours on a CPU or ten hours on a GPU [Elrefaie et al., 2024, Li et al., 2023b]. While machine learning offers a computationally efficient alternative by operating at lower resolutions compared to traditional methods [Bhatnagar et al., 2019, Pfaff et al., 2021, Thuerey et al., 2020, Hennigh et al., 2021], most existing methods are limited by their resolution dependency. To address this, we focus on neural operators, a recent innovation that provides a resolution-independent approach to solving PDEs.

In this work, we generalize geometry learning from discretized meshes to mesh density functions. Our key innovation is the use of optimal transport (OT) to embed geometry by mapping an input mesh density function to a uniform density in a canonical reference space, enabling us to learn neural operators in this space. Based on this idea, we introduce the Optimal Transport Neural Operator (OTNO), which combines OT-based geometry encoding/decoding with (Spherical) Fourier Neural Operators[Bonev et al., 2023, Li et al., 2020] (Figure 1).

Our main contributions are summarized as follows:

Machine Learning and the Physical Sciences Workshop, NeurIPS 2025.

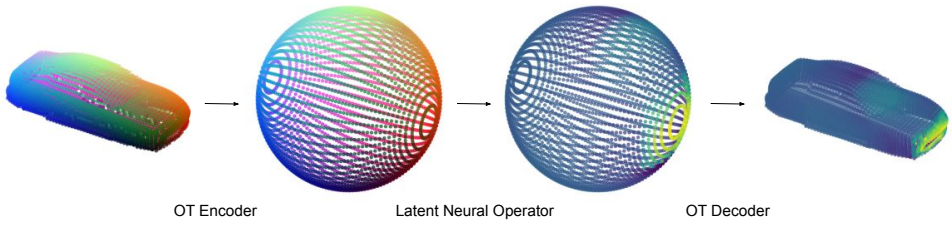


Figure 1: Illustration of the optimal transport neural operator (OTNO). (a) **OT Encoder**: The surface mesh is encoded onto a latent computational mesh, and the OT coupling is visualized by representing the coordinates of points as RGB colors. (b) **Latent Neural Operator**: Within the latent space, we apply S/FNO to calculate solutions on the latent mesh. (c) **OT Decoder**: We decode the solutions from the latent space back to the original surface mesh; here, the colors indicate solution values.

1. A novel optimal transport framework for mesh embedding that maps mesh density functions to uniform latent densities. This framework integrates with a latent neural operator to create the Optimal Transport Neural Operator (OTNO) solving PDEs on complex geometries.
2. The Sub-Manifold Method, a dimension-reduction technique that restricts solution operators to surface manifolds with one lower dimension, enabling efficient PDE resolution in reduced dimensional space.
3. Comprehensive validation on industry-standard datasets—ShapeNet-Car (3.7k points) [Chang et al., 2015] and DrivAerNet-Car (400k points) [Elrefaie et al., 2024]—demonstrates unprecedented efficiency in RANS pressure field prediction, with 2-8x faster processing and 2-8x less memory usage compared to current ML methods, and approximately 7,000 times faster than traditional approaches.
4. Our model leverages instance-dependent geometry embeddings to excel across diverse geometries, as evidenced on the FlowBench dataset[Tali et al., 2024].

2 Problem Setting

Our aim is to learn the operator from the boundary geometries of PDEs to their boundary solutions. In this work, we model the geometries as supports of mesh density functions defined on an ambient Euclidean space. Let $\mathcal{F}_c$ be the set of densities whose supports are d-dimensional bounded manifolds. For any density f in the set $\mathcal{F}_c$, we define the d-dimensional manifold Ω_f by its support, with its boundary denoted by $\partial\Omega_f$ which is a $(d-1)$-dimensional sub-manifold. Let $\mathcal{L}$ and $\mathcal{B}$ be two, possibly non-linear, partial differential operators and consider the PDE:

$$\begin{aligned} \mathcal{L}(u) &= h, \quad \text{in } \Omega_f, \\ \mathcal{B}(u) &= b, \quad \text{in } \partial\Omega_f, \end{aligned} \tag{1}$$

where h and b are some fixed functions on $\mathbb{R}^d$. For any open set $A \subseteq \mathbb{R}^d$, let $\mathcal{U}(A)$ be a Banach function space whose elements have domain A. We will assume that there exists a family of extension operators $E_A : \mathcal{U}(A) \to \mathcal{U}(\mathbb{R}^d)$ and fix such a family. We now consider the following set of densities:

$$\mathcal{F}_c^{PDE} = \mathcal{F}_c^{PDE}(\mathcal{L}, \mathcal{B}, h, b) = \{f \in \mathcal{F}_c : \exists! u \in \mathcal{U}(\Omega_f) \text{ s.t. } u \text{ satisfies (1)}\}.$$

We can thus define a solution operator which maps a density f, defining the domain of the PDE, to a uniquely extended solution of the PDE. In particular,

$$\begin{aligned} \mathcal{G}^\dagger : \mathcal{F}_c^{PDE} \subset L^1(\mathbb{R}^d) &\to \mathcal{U}(\mathbb{R}^d), \\ f &\mapsto E_{\Omega_f}(u), \end{aligned} \tag{2}$$

where $u \in \mathcal{U}(\Omega_f)$ is the unique solution to (1). Our aim is to approximate $\mathcal{G}^\dagger$ or the associated sub-manifold mapping defined subsequently from data.

Sub-Manifold Solution Operator For the associated boundary solution problems, the solution operator on the sub-manifold $\partial\Omega_f$ is given by:

$$\mathcal{G}^\dagger : f^{\text{sub}} \mapsto u^{\text{sub}} \quad \text{in } \partial\Omega_f, \tag{3}$$

where $f^{\text{sub}} = \frac{f|_{\partial\Omega_f}}{\int_{\partial\Omega_f} f(x)dS}$, with dS a surface measure, denotes the normalized function of f constrained to $\partial\Omega_f$ which is a density function on $\partial\Omega_f$, and u^{sub} denotes our target solution function on $\partial\Omega_f$.

3 Optimal Transport Neural Operator (OTNO)

3.1 Methodological formulations

For convenience, we use f, u and $\Omega \in \mathbb{R}^d$ to denote the f^{sub}, u^{sub} and $\partial\Omega_f \in \mathbb{R}^{d-1}$ in Eq (3). According to the settings from Sec 2, f is a density function defined on the complex geometric domain Ω and $\Omega = \text{supp}(f)$. We define a measure μ built from the density f as follows: $d\mu = f(x)\,dx$, on Ω, and create a latent density function g on a simple geometric domain Ω^* within the same metric space $\mathbb{R}^d$. Solving the OT problem from latent measure λ to physical measure μ yields OT map $T : (\Omega^*, \lambda) \to (\Omega, \mu), \xi \mapsto x$; or OT plan P, which is a probability measure on $\Omega^* \times \Omega$ with marginals μ on Ω and λ on Ω^*. Then we build the composition $\mathcal{P} \circ \mathcal{G}^* \circ \mathcal{Q} : f \mapsto u$ to approximate the operator $\mathcal{G}^\dagger$ from Eq (3) (also works for Eq (2)), where $\mathcal{P}, \mathcal{G}^*, \mathcal{Q}$ are detailed as follows.

The Encoder the encoder is defined as the optimal transport solver that maps the density function to the associated transport maps / the marginals of the transport plans.

$$\begin{aligned} &\mathcal{Q} : L^1(\Omega; \mathbb{R}) \to L^1(\Omega^*; \mathbb{R}^d) \\ &f \mapsto \begin{cases} T, & \text{for the map case} \\ \displaystyle\int_\Omega P(\cdot, x)x\, d\mu(x), & \text{for the plan case} \end{cases} \end{aligned} \tag{4}$$

The Latent Neural Operator latent neural operator is then defined such that it maps the transport map / the marginal of the transport plan to a latent solution function v on the latent domain Ω^*:

$$\mathcal{G}^* : T/\int_\Omega P(\cdot, x)x d\mu(x) \mapsto v \quad \text{on } \Omega^*. \tag{5}$$

The Decoder We then define the decoder $\mathcal{P} : v \mapsto u$ as

$$u(x) = \begin{cases} v \circ T^{-1}(x), & \text{for the map case} \\ \int_{\Omega^*} P(\xi, x)v(\xi)d\lambda(\xi), & \text{for the plan case} \end{cases}, \forall x \in \Omega \tag{6}$$

3.2 OTNO Algorithm

Given a dataset $\{(\mathcal{X}_j, u_j)\}_{j=1}^N$ of surface sampling meshes and corresponding PDEs' solution values on surface, where $\mathcal{X}_j = [x_{j,k}]_{k=1}^{n_j^1}$ and $u_j = [u_{j,k}]_{k=1}^{n_j^1}$. For each $\mathcal{X}_j$, generate a latent surface mesh $\Xi_j = [\xi_{j,l}]_{l=1}^{n_j^2}$ using a 2D parametric mapping from a square grid, where n_j^2 is a perfect square. Compute the normals $\mathcal{N}_j$ on $\mathcal{X}_j$ and $\mathcal{H}_j$ on Ξ_j. By solving OT problem from each latent mesh to each physical mesh, we obtain a set of transported mesh $\{\mathcal{X}'_j\}_{j=1}^N$, where $\mathcal{X}'_j = \left[x'_{j,l}\right]_{l=1}^{n_j^2}$.

According to the above settings, we introduce the Algorithm 1 as the implementation of our method, applicable for both OT map and OT plan scenarios. To enhance the quantity of surface representation, we substitute points in $\mathcal{X}'_j$ with their closest counterparts in $\mathcal{X}_j$, i.e. use $\mathcal{M}_j = \mathcal{X}_j(\mathcal{E})$ to replace $\mathcal{X}'_j$, similarly we use $v_j(\mathcal{D})$ in the decoder. Considering T as a deformation map, we include deformation-related features by using the cross-product over normals $\mathcal{H}_j \times \mathcal{N}_j(\mathcal{E})$.

4 Experiments

The code for these experiments is available at `https://github.com/Xinyi-Li-4869/OTNO`.

Algorithm 1 Optimal Transport Neural Operator (OTNO)

Given physical mesh $\{\mathcal{X}_j\}_{j=1}^N$, latent mesh $\{\Xi_j\}_{j=1}^N$, transported mesh $\{\mathcal{X}'_j\}_{j=1}^N$ and solution values $\{u_j\}_{j=1}^N$.
Initialize a FNO $\mathcal{G}_\theta$.
for $j = 1$ to N **do**
    1. Build Index Mapping:
    Encoder indices $\mathcal{E} = \left(\arg \min_{k=1,\dots,n_j^1} \|x'_{j,l} - x_{j,k}\|_2 : l = 1, \dots, n_j^2 \right)$
    Decoder indices $\mathcal{D} = \left(\arg \min_{l=1,\dots,n_j^2} \|x'_{j,l} - x_{j,k}\|_2 : k = 1, \dots, n_j^1 \right)$
    2. OT encoder: $\mathcal{T}_j = (\Xi_j, \mathcal{M}_j, \mathcal{H}_j \times \mathcal{N}_j(\mathcal{E})) \in \mathbb{R}^{n_j^2 \times 9}$, where $\mathcal{M}_j = \mathcal{X}_j(\mathcal{E}) \in \mathbb{R}^{n_j^2 \times 3}$ selects rows from $\mathcal{X}_j$ according to the indices specified in $\mathcal{E}$.
    3. Latent FNO: $v_j = \mathcal{G}_\theta(\mathcal{T}_j)$
    4. OT decoder: $u'_j = v_j(\mathcal{D}) \in \mathbb{R}^{n_j^1 \times s}$, where u_j selects rows from v_j according to the indices specified in $\mathcal{D}$.
end for
Compute the empirical loss over all dataset instances: $\sum_{j=1}^{N} \|u'_j - u_j\|_{\mathcal{U}}$.

4.1 Main Experiments - 3D Car Datasets

We conducted experiments on the ShapeNet-Car and DrivAerNet-Car datasets, following the experimental settings of Li et al. [2023b] and Elrefaie et al. [2024], respectively. As shown in Tables 1 and 2, our method, OTNO, achieves the lowest error and consumes less memory than 3D-based methods like GINO, UNet, and RegDGCNN. And its superior performance against other surface deformation methods, such as Geo-FNO, underscores the effectiveness of our optimal transport-based approach.

Table 1: Predict pressure field on ShapeNet Car Dataset (single P100)

Model	Relative L2	Total Time (hr)	GPU Memory (MB)
Geo-FNO[Li et al., 2023a]	13.50%	1.96	12668
UNet[Wolny et al., 2020]	13.14%	6.86	7402
GINO[Li et al., 2023b]	7.21%	10.45	12734
OTNO(Plan)	**6.70%**	**1.52**	**1890**

Table 2: Predict drag coefficient (Cd) on DrivAerNet Car Dataset (single A100).

Model	**MSE (e-05)**	**R2 Score**	**Total Time (hr)**	**Memory (MB)**
RegDGCNN[Elrefaie et al., 2024]	6.63	0.887	10.78	72392
GINO	3.33	0.955	7.73	14696
OTNO(Map)	3.93	0.947	10.63	**2896**
OTNO(Plan)	**3.28**	**0.956**	**5.26**	9702

4.2 Showcase of Dataset with Diverse Geometries - 2D Flow Datasets

As shown in Table 3, OTNO demonstrates significantly superior accuracy on the FlowBench dataset [Tali et al., 2024], which spans diverse geometries. More details are in Appendix C.3.

4.3 Discussion of Experimental Results

For car surface prediction, OTNO represents the latent space as a 2D surface, whereas GINO uses the full 3D space. This enables OTNO to significantly outperform GINO in computational efficiency. When handling diverse geometries, methods like GINO and GeoFNO rely on a shared network to learn

Table 3: Prediction under **M1** Metric (global) on FlowBench Dataset (single A100) for Group G2

Model	Relative L2	Time per Epoch (sec)	GPU Memory (MB)
FNO	56.67%	43	4548
GINO	43.16%	390	58970
OTNO	**7.16%**	603	22868

deformations/decoding, requiring substantial data to approximate the wide variation effectively. In contrast, OTNO employs a fixed cost function and solves an optimal transport problem independently for each geometry. This yields instance-specific deformations governed by the same underlying principle, leading to superior performance on diverse datasets.

5 Conclusion

Our proposed model, OTNO, uses OT to embed geometry into a latent space where a neural operator is applied, significantly improving computational efficiency and speed while effectively handling diverse geometries. The model's primary limitations are the trade-off between complexity and accuracy (OTNO(Plan) has $O(n^2)$ complexity, while OTNO(Map) reduces this at the cost of accuracy) and the lack of a systematic method for determining the optimal latent (sub) manifold.

References

Saakaar Bhatnagar, Yaser Afshar, Shaowu Pan, Karthik Duraisamy, and Shailendra Kaushik. Prediction of aerodynamic flow fields using convolutional neural networks. *Computational Mechanics*, 64:525–545, 2019.

Boris Bonev, Thorsten Kurth, Christian Hundt, Jaideep Pathak, Maximilian Baust, Karthik Kashinath, and Anima Anandkumar. Spherical fourier neural operators: Learning stable dynamics on the sphere. In *International conference on machine learning*, pages 2806–2823. PMLR, 2023.

Yann Brenier. Polar factorization and monotone rearrangement of vector-valued functions. *Communications on pure and applied mathematics*, 44(4):375–417, 1991.

Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. In *arXiv:1512.03012 [cs.GR]*, 2015.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

Mohamed Elrefaie, Angela Dai, and Faez Ahmed. Drivaernet: A parametric car dataset for data-driven aerodynamic design and graph-based drag prediction, 2024. URL `https://arxiv.org/abs/2403.08055`.

Rémi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z. Alaya, Aurélie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, Léo Gautheron, Nathalie T.H. Gayraud, Hicham Janati, Alain Rakotomamonjy, Ievgen Redko, Antoine Rolet, Antony Schutz, Vivien Seguy, Danica J. Sutherland, Romain Tavenard, Alexander Tong, and Titouan Vayer. Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021. URL `http://jmlr.org/papers/v22/20-451.html`.

Oliver Hennigh, Susheela Narasimhan, Mohammad Amin Nabian, Akshay Subramaniam, Kaustubh Tangsali, Zhiwei Fang, Max Rietmann, Wonmin Byeon, and Sanjay Choudhry. Nvidia simnet™: An ai-accelerated multi-physics simulation framework. In *International conference on computational science*, pages 447–461. Springer, 2021.

LK Kantorovich. On a problem of monge. *Journal of Mathematical Sciences*, 133(4), 2006.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *ICLR*, 2021.

Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries. *Journal of Machine Learning Research*, 24(388):1–26, 2023a.

Zongyi Li, Nikola Borislavov Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Prakash Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, and Anima Anandkumar. Geometry-informed neural operator for large-scale 3d PDEs. *NIPS*, 2023b.

Cheng Meng, Yuan Ke, Jingyi Zhang, Mengrui Zhang, Wenxuan Zhong, and Ping Ma. Large-scale optimal transport map estimation using projection pursuit. *Advances in Neural Information Processing Systems*, 32, 2019.

Sophie Mildenberger and Michael Quellmalz. A double fourier sphere method for d-dimensional manifolds. *Sampling Theory, Signal Processing, and Data Analysis*, 21(2):23, 2023.

Gaspard Monge. Mémoire sur la théorie des déblais et des remblais. *Mem. Math. Phys. Acad. Royale Sci.*, pages 666–704, 1781.

Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W Battaglia. Learning mesh-based simulation with graph networks. *ICLR*, 2021.

Filippo Santambrogio. *Optimal transport for applied mathematicians*, volume 87. Springer, 2015.

Ronak Tali, Ali Rabeh, Cheng-Hau Yang, Mehdi Shadkhah, Samundra Karki, Abhisek Upadhyaya, Suriya Dhakshinamoorthy, Marjan Saadati, Soumik Sarkar, Adarsh Krishnamurthy, et al. Flowbench: A large scale benchmark for flow simulation over complex geometries. *arXiv preprint arXiv:2409.18032*, 2024.

Nils Thuerey, Konstantin Weißenow, Lukas Prantl, and Xiangyu Hu. Deep learning methods for reynolds-averaged navier–stokes simulations of airfoil flows. *AIAA Journal*, 58(1):25–36, 2020.

Adrian Wolny, Lorenzo Cerrone, Athul Vijayan, Rachele Tofanelli, Amaya Vilches Barro, Marion Louveaux, Christian Wenzl, Sören Strauss, David Wilson-Sánchez, Rena Lymbouridou, et al. Accurate and versatile 3d segmentation of plant tissues at cellular resolution. *Elife*, 9:e57613, 2020.

A Preliminaries

A.1 Neural Operator

Suppose there is a set of function set $\mathcal{F}$, and for each function $f \in \mathcal{F}$, it determines a specific PDE with a specific solution function $u \in \mathcal{U}$. The target is to learn the solution operator for a family of PDEs $\mathcal{G}^\dagger : f \mapsto u$.

The neural operator $\mathcal{G}_\theta$ proposed by Kovachki et al. [2023] composes point-wise and integral layers to approximate the target operator:

$$\mathcal{G}_\theta = \mathcal{Q} \circ \mathcal{K}_L \circ \cdots \circ \mathcal{K}_1 \circ \mathcal{P}$$

where $\mathcal{Q}$ and $\mathcal{P}$ are pointwise neural network that lifting the lower dimension input to higher dimension latent space D and project them back to lower dimension output respectively. $\mathcal{K}_l$ is integral operator $\mathcal{K}_l : v_{l-1} \mapsto v_l$:

$$v_l(x) = \int_D \kappa_l(x, y) v_{l-1}(y) \, \mathrm{d}y$$

where κ_l is a learnable kernel function.

Usually we assume the dataset $\{f_j, u_j\}_{j=1}^N$ is available, where $\mathcal{G}(f_j) = u_j$. Then we can optimize the neural operator by minimizing the empirical data loss:

$$Loss(\mathcal{G}_\theta) = \frac{1}{N} \sum_{j=1}^{n} l(u_j, \mathcal{G}_\theta(f_j))$$

where $l(\cdot, \cdot)$ denotes an appropriate error metric (e.g., MSE or relative L^2 error). In this work, we propose to

A.2 Neural Operator on Geometric Problems

It is a standard method to embed the varying physical domain Ω into a latent space Ω^* with various types of encoders and decoders

$$\mathcal{G}^\dagger \approx \mathcal{Q} \circ \mathcal{G}^* \circ \mathcal{P} \tag{7}$$

where $\mathcal{G}^*$ is a latent operator defined on the latent space Ω^*. Previous works have explored using interpolation and deformation as the encoders and decoders.

Geometric Informed Neural Operator Geometric Informed Neural Operator (GINO) Li et al. [2023b] assign a uniform latent grid. It defines encoder $\mathcal{Q}$ and decoder $\mathcal{P}$ as graph neural operator Li et al. [2021] to learn non-linear interpolation from the physical mesh to the latent grid.

$$v_l(x) = \sum_{y \in \mathcal{N}} \kappa_l(x, y) v_{l-1}(y) \, \mu(y)$$

where $\mathcal{N}$ is the neighbor of the bipartite graph between the physical mesh and the latent grid.

Geometric Aware Fourier Neural Operator Geometric Aware Fourier Neural Operator (GeoFNO) assigns a canonical manifold as the latent space, and constructs an invertible deformation map (diffeomorphism) as the encoder

$$T : \Omega^* \to \Omega \tag{8}$$

The deformation map induces a pullback of input function a on the latent space, $T^\# a := a \circ T$. Then GeoFNO applies the latent operator to the pullback $T^\# a$.

In this work, we will investigate using optimal transport as the encoder and decoder to embed the geometries into a uniform mesh.

A.3 Optimal Transport

Monge originally formulated the OT problem as finding the most economical map to transfer one measure to another [Monge, 1781]. Later, Kantorovich introduced a relaxation of these strict transportation maps to more flexible transportation plans, solved using linear programming techniques [Kantorovich, 2006].

A.3.1 Monge Problem

Let Ω and Ω^* be two separable metric spaces such that any probability measure on Ω (or Ω^*) is a Radon measure (i.e. they are Radon spaces). Let $c : \Omega^* \times \Omega \to \mathbb{R}^+$ be a Borel-measurable function. Given probability measures μ on Ω and λ on Ω^* with corresponding density functions f and g,

Monge's formulation of the optimal transportation problem is to find a transport map $T : \Omega^* \to \Omega$ that minimizes the total transportation cost:

$$\text{(MP)} \quad \inf\{\int_{\Omega^*} c(\xi, T(\xi)) d\lambda(\xi) \mid T_{\#}\lambda = \mu\}, \tag{9}$$

where $T_{\#}\lambda = \mu$ denotes that map T is measure preserving (i.e. $\int_{T^{-1}(B)} d\lambda(\xi) = \int_B d\mu(x)$, for any Borel set $B \subseteq \Omega$).

Moreover, the following theorem and continuity property hold for the Monge formulation.

Theorem 1 (Existence and uniqueness of transport map [Brenier, 1991])**.** *Suppose the measures μ and λ are with compact supports $\Omega, \Omega^* \subseteq \mathbb{R}^d$ respectively, and they have equal total mass $\mu(\Omega) = \lambda(\Omega^*)$. Assume the corresponding density functions satisfy $f, g \in L^1(\mathbb{R}^d)$, and the cost function is $c(\xi, x) = \frac{1}{2}|\xi - x|^2$, then the OT map from λ to μ exists and is unique. It can be expressed as $T(\xi) = \xi + \nabla\phi(\xi)$, where $\phi : \Omega^* \to \mathbb{R}$ is a convex function, and ϕ is unique up to adding a constant.*

Lemma 2 (Continuity of transport map)**.** *Given that cost function is the squared Euclidean distance and λ is a measure with uniform density function with a compact support, if μ is absolutely continuous and strictly positive also with a compact support, then the OT map T is continuous almost everywhere. (This lemma can be easily derived from Theorem 1.)*

In practical applications, especially in computational settings, the continuous problem (9) is often discretized. Suppose the measures λ and μ are supported on finite point sets $\Xi = \{\xi_1, \ldots, \xi_{n_1}\} \subset \Omega^*$ and $\mathcal{X} = \{x_1, \ldots, x_{n_2}\} \subset \Omega$, and are represented by discrete probability vectors $a = (\lambda_1, \ldots, \lambda_{n_1})$ and $b = (\mu_1, \ldots, \mu_{n_2})$, respectively.

Then Monge problem then seeks a transport map T that minimizes the following total cost:

$$\min_{T \in \Psi} \sum_{i=1}^{n_1} \lambda_i \cdot c(\xi_i, T(\xi_i)), \tag{10}$$

where $\Psi = \{T \,|\, T_{\#}\lambda = \mu\}$ denotes the set of all feasible transport maps.

A.3.2 Kantorovich Problem

Relaxing the map constraint, the Kantorovich formulation seeks probability measures P on $\Omega^* \times \Omega$ that attains the infimum,

$$\text{(KP)} \quad \inf\{\int_{\Omega^* \times \Omega} c(\xi, x) dP(\xi, x) \mid P \in \Gamma(\lambda, \mu)\}. \tag{11}$$

where $\Gamma(\lambda, \mu)$ denotes the collection of all probability measures on $\Omega^* \times \Omega$ with marginals λ on Ω^* and μ on Ω, and $c : \Omega^* \times \Omega \to \mathbb{R}^+$ is transportation cost function. The existence and uniqueness are guaranteed with similar assumptions (c.f. Theorem 1.17 Santambrogio [2015]).

In the discrete setting (using the same notation $\Xi, \mathcal{X}, a, b$ as in the Monge formulation), the cost function is evaluated as a matrix $M \in \mathbb{R}^{n_1 \times n_2}$, with entries

$$M_{ij} = c(\xi_i, x_j), \quad 1 \le i \le n_1,\, 1 \le j \le n_2. \tag{12}$$

The Kantorovich problem then reduces to the following linear program:

$$\text{(KP)} \quad \min_{P \in \Gamma(a,b)} \langle P, M \rangle = \min_{P \in \Gamma(a,b)} \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} M_{ij} \cdot P_{ij}$$

where $\Gamma(a, b)$ represents the set of all feasible coupling matrices, defined as the discrete probability measures $\Gamma(a, b) = \{P \geq 0 \mid P\mathbf{1}_{n_2} = a, P^T\mathbf{1}_{n_1} = b\}$. Moreover, the following sparity property holds for the discrete implementation of the Kantorovich formulation [Peyré et al., 2019]:

Lemma 3 (Sparsity of Transport Plan)**.** *The solution to the linear programming is sparse; the number of non-zero entries in the transport plan P is at most $n_1 + n_2 - 1$.*

In practice, optimal transport plan is often computed with entropy regularization using the Sinkhorn Algorithm [Cuturi, 2013]. In this case, the smoothened transport plan is not sparse anymore.

B Algorithm Implementation

Our model employs the Projection pursuit Monge map[Meng et al., 2019] to obtain an approximate solution of the Monge OT map T, and employs Sinkhorn method [Cuturi, 2013] to obtain an approximate solution of the Kantorovich OT plan T. The resulting map/plan is then utilized to construct the OT encoder/decoder for neural operator. The algorithm is detailed in Algorithm 1, applicable for both OT map and OT plan scenarios. The primary distinctions between utilizing an OT map and an OT plan are in the generation of the latent mesh Ξ_j and the construction of the transported mesh $\mathcal{X}_j'$. These distinctions are elaborated in Sections B.1 and B.2.

B.1 OTNO - Kantorovich Plan

B.1.1 Transported Mesh

Using the Sinkhorn method [Cuturi, 2013] detailed in the following Sec B.1.3, we obtain dense coupling matrices P_j that represent the OT plans from the latent computational mesh Ξ_j to the boundary sampling mesh $\mathcal{X}_j$. How to effectively utilize these dense matrices within a neural operator framework become a problem worth discussing.

As described in Eq. (5), the transported mesh is obtained by applying the function $\int_\Omega P(\cdot, x)\, x\, d\mu(x)$ to the latent mesh. Therefore, in the discrete case, the transported mesh is given by $\mathcal{X}_j' = P_j \mathcal{X}_j$. However, directly applying the dense matrix by multiplying it with the mesh will lead to a lower accuracy of the final predictions because it only approximates the solution plan for the Kantorovich problem (11). And saving large-scale dense matrices is costly. Therefore we discuss how to use these optimal coupling matrices more effectively. As the matrix P_j represents a discrete probability measure on $\mathcal{X}_j \times \Xi_j$, a practical approach is to focus on the maximum probability elements, referred to as the "Max" strategy. Additionally, a more effective "Mean" strategy involves replacing each point $x' \in \mathcal{X}_j'$ with the nearest point in $\mathcal{X}_j$, significantly reducing approximation ambiguity caused by Sinkhorn method. Further, we can encode/decode by the top-k nearest neighbors in $\mathcal{X}_j/\mathcal{X}_j'$ instead of only find the single nearest one. Details of these different strategies are further discussed in Section D.1.1.

B.1.2 Latent Mesh

In practice, we design the computational grid to have more vertices than the boundary sampling mesh to ensure maximal information retention during the encoding and decoding processes. Let α be an expansion factor, and the number of points in latent space is then set to $n_j^2 = \lceil \sqrt{\alpha \times n_j^1} \rceil^2$. This approach implies that the encoder functions as an interpolator, while the decoder acts as a selective querier. A simple illustration of the encoder and decoder processes is shown in Fig.2. And detailed ablation studies for latent mesh shape and latent mesh size are in Sec D.1.3 and Sec D.1.4.

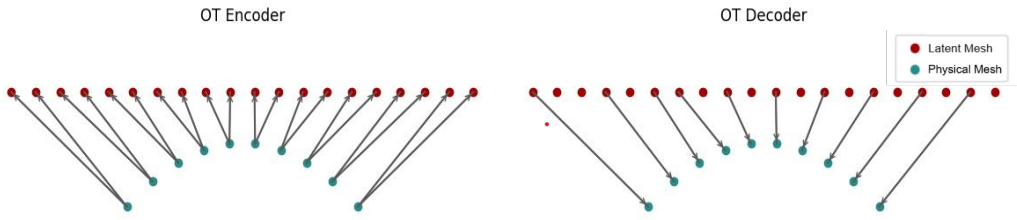


Figure 2: A simple illustration for OT encoder and OT decoder. The curve represents the boundary sampling points and the line denotes the latent computational grid

B.1.3 Sinkhorn Algorithm

Sinkhorn [Cuturi, 2013] method added an entropy regularizer to the Kantorovich potential and greatly improved efficiency. They first propose the Sinkhorn distance that added entropy constraint:

$$d_{M,\alpha}(a,b) = \min_{P \in \Gamma_\alpha(a,b)} \langle P, M \rangle, \tag{13}$$

where

$$\Gamma_\alpha(a,b) = \{P \in \Gamma(a,b) \mid \mathbf{KL}(P, ab^T) \leq \alpha\} = \{P \in \Gamma(a,b) \mid h(P) \geq h(a) + h(b) - \alpha\}. \tag{14}$$

Then they consider the dual problem that arises from Lagrange multiplier:

$$P^\beta = \underset{P \in \Gamma(a,b)}{\operatorname{argmin}} \quad \langle P, M \rangle - \frac{1}{\beta} h(P), \tag{15}$$

This formulation leads to a problem where β adjusts the trade-off between the transport cost and the entropy of the transport plan P. When β increases, the influence of the entropy regularization decreases, making P^β converge closer to the solution of the original Kantorovich problem (11). This implies that a larger β leads to a solution that is more accurate and economically efficient.

By introducing the entropy constraint, the Sinkhorn distance not only regularizes the OT problem but also ensures that the solution is computationally feasible even for large-scale problems. This regularization dramatically improves the numerical stability and convergence speed of the algorithm.

Implementation In the implementation, we set $a = \frac{1}{n_1}\mathbf{1}_{n_1}$, assuming each point in the latent space contributes equally to computations. We set $b = \frac{1}{n_2}\mathbf{1}_{n_2}$, as the sampling mesh $\mathcal{X} = \{x_1, \ldots, x_{n_1}\}$, obtained from CFD simulations, typically offers increased point density in regions with sharp variations. This uniform mass vector on $\mathcal{X}$ ensures a denser representation in these critical areas, improving the accuracy of aerodynamic predictions. For cases with excessively dense regions, we employ voxel downsampling to prevent excessive density variations, thereby maintaining the feasibility of our uniform mass vector. We set $\beta = 1 \times 10^6$ to ensure that the solution is economical without incurring excessive computational costs. For computational support, we utilize the geomloss implementation from the Python Optimal Transport (POT) library [Flamary et al., 2021], which supports GPU acceleration and use lazy tensors to store dense coupling matrix obtained from Sinkhorn method.

Voxel Downsampling To achieve a well-balanced input density function, we employ voxel downsampling as a normalization process that constrains the density function within a predefined range $[0, a]$, where a is determined by the voxel size. This approach mitigates excessive clustering in regions with high point density, such as around the wheels in some some car data, therefore ensuring a more uniform spatial distribution of points.

B.2 OTNO - Monge Map

B.2.1 Transported Mesh

Using the PPMM, detailed in Section B.2.3, we obtain the transported mesh X_j', which is mapped from the latent mesh Ξ_j to the physical space. This results in the same number of points as the latent mesh while representing the distribution of the physical mesh. Since PPMM operates directly on the latent mesh, projecting it towards the physical mesh (as shown in Algorithm 2), the output is the transported mesh itself rather than a mapping that can operate on any function on the latent mesh, and it cannot offer an inverse direction itself. (An OT plan, discretized as a coupling matrix, can handle both.) Therefore, we also compute the indices for encoding and decoding, as outlined in Algorithm 1.

B.2.2 Latent Mesh

Since the latent mesh has the same number of points as the physical mesh, and the 2-dimensional FNO on the latent space requires a square mesh, the number of points in both the latent mesh and the physical mesh should be a perfect square.

Therefore, we first apply voxel downsampling to normalize the mesh density, followed by random sampling to further downsample the mesh so that ensure the number of points is a perfect square.

Finally, we generate the latent mesh to match the square number of points in the downsampled physical mesh. We choose a spherical mesh as the latent mesh, given its suitability for the Projection Pursuit Monge Map (PPMM).

B.2.3 Projection pursuit Monge map (PPMM)

Algorithm 2 Projection pursuit Monge map

Input: two matrix $X \in \mathbb{R}^{n\times d}$ and $Y \in \mathbb{R}^{n\times d}$
$k \leftarrow 0$, $X_0 \leftarrow X$
repeat
(a) calculate the most 'informative' projection direction $e_k \in \mathbb{R}^d$ between X_k and Y
(b) find the one-dimensional OT Map $\phi^{(k)}$ that matches $X_k e_k$ to $Y e_k$ (using look-up table)
(c) $X_{k+1} \leftarrow X_k + (\phi^{(k)}(X_k e_k) - X_k e_k^T) e_k^T$ and $k \leftarrow k+1$
until converge
The final mapping is given by $\hat{\phi} : X \rightarrow X_k$

The PPMM proposes an estimation method for large-scale OT maps by combining the concepts of projection pursuit regression and sufficient dimension reduction. As summarized in Algorithm 2, in each iteration, the PPMM applies a one-dimensional OT map along the most "informative" projection direction. The direction e_k is considered the most "informative" in the sense that the projected samples $X_k e_k$ and $Y e_k$ exhibit the greatest "discrepancy." The specific method for calculating this direction is detailed in Algorithm 1 in paper Meng et al. [2019].

C Experiments

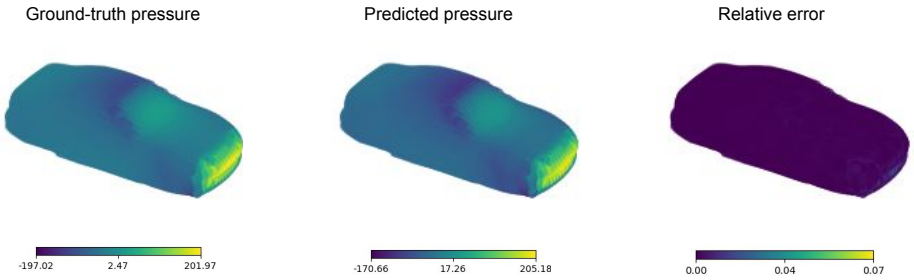
We conducted experiments on three CFD datasets. Two of them are 3D car datasets, where the target prediction is the pressure field or drag coefficient, which depends solely on the car surface—a 2D manifold. The **ShapeNet** dataset includes 611 car designs, each with 3.7k vertices and corresponding average pressure values, following the setup from Li et al. [2023b]. The **DrivAerNet** dataset, sourced from Elrefaie et al. [2024], contains 4k meshes, each with 200k vertices, along with results from CFD simulations that measure the drag coefficient. Additionally, we further evaluate our model on the 2D **FlowBench** dataset [Tali et al., 2024], which features a wider variety of shapes, including three groups, each containing 1k shapes with a resolution of 512×512. Although the PDE solutions are not on the boundary sub-manifold, preventing our model from reducing dimensions by embedding boundary geometries, this dataset provides an excellent opportunity to explore our model's capabilities across diverse shapes.

C.1 ShapeNet-Car Dataset

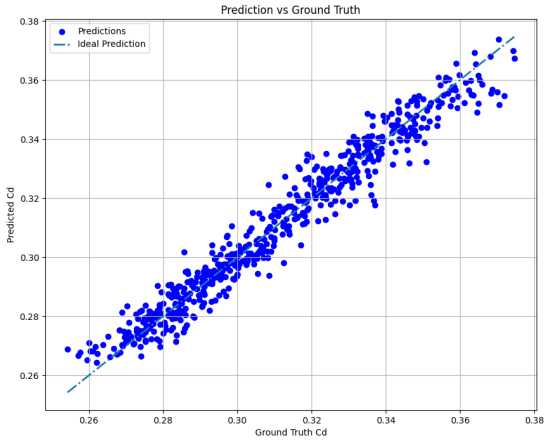
To ensure a fair comparison, we maintained identical experimental settings to those used in GINO paper. We compared the relative error, total time (including data processing and training), and GPU memory usage against key baselines from Li et al. [2023b]. As detailed in Table.1, our method, OTNO, achieved a relative error of **6.70%**, which is a slight improvement over the lowest error reported in the baseline—**7.21%** by GINO. Across all baselines, our method demonstrated reduced total time and lower GPU memory usage. Compared to GINO, our method significantly reduced both time costs and memory expenses by factors of eight and seven, respectively. The visual results for pressure prediction are presented in Fig.3a.

C.2 DrivAerNet-Car Dataset

Herein, we follow all the experimental settings from DrivAerNet paper [Elrefaie et al., 2024] and compare our model to RegDGCNN, as proposed in Elrefaie et al. [2024], and the state-of-the-art neural operator model, GINO. Note that we do not use pressure data for training; instead, we only use the drag coefficient (Cd). The visual result for Cd prediction is presented in Fig.3b. For OTNO, we employ voxel downsampling with a size of 0.05, corresponding to approximately 60k samples (see ablation in D.2.2).



(a) Results of pressure on ShapeNet Car dataset.



(b) Results of drag coefficient on DrivAerNet Car dataset

Figure 3: Results visualization for OTNO on Car Datasets

As detailed in Table.2, OTNO(Plan) demonstrates significantly superior accuracy and reduced computational costs compared to RegDGCNN, halving the MSE, speeding up computations by a factor of 5, and reducing GPU memory usage by a factor of 24. Compared to GINO, OTNO(Plan) achieves a slightly lower MSE, a marginally higher R2 score, and notably reduces total computation time and memory usage by factors of 4 and 5, respectively.

Given that PPMM is designed for large-scale OT maps, we evaluate OTNO(Map) on this large-scale dataset to assess its performance. Unfortunately, both the error and time cost are worse than OTNO(Plan). However, the memory cost is significantly lower. This is primarily because OTNO(Plan) expands the latent space, while OTNO(Map) does not.

C.3 FlowBench Dataset

To assess the performance of instance-dependent deformation in our model across diverse geometries, we conducted further evaluations using the FlowBench dataset [Tali et al., 2024].

1. M1: *Global metrics*: The errors of in velocity and pressure fields over the entire domain.
2. M2: *Boundary layer metrics*: The errors in velocity and pressure fields over a narrow region around the object. We define the boundary layer by considering the solution conditioned on the Signed Distance Field ($0 \leq SDF \leq 0.2$).

The complete results of training using the M1 metric (global) and M2 metric (boundary) are presented in Table 4, and Table 5, respectively. The visualization of results are as shown in Fig. 4 and Fig. 5.

The results demonstrate that OTNO significantly outperforms in accuracy under both the M1 (global) and M2 (boundary) metrics, particularly for M1. Furthermore. And OTNO achieves notably better accuracy across all three groups, especially for G2 (harmonics). However, cost reduction is not observed in the FlowBench dataset. It is important to note that the cost reductions seen in car datasets stem from the sub-manifold method, which employs optimal transport to generate 2D representations and perform computations in 2D latent space instead of 3D. In the FlowBench dataset, however, the solutions are not restricted to the boundary sub-manifold. Even for the M2 metric, although the relevant data is close to the boundary with a width of 0.2, it does not reduce to a 1D line. As a result, computations cannot be confined to the sub-manifold, and the associated cost reduction benefits are consequently absent.

We do not present Geo-FNO results on this dataset as the relative L2 errors consistently exceed 60%. Our analysis suggests that Geo-FNO, which uses an end-to-end approach to learn a shared deformation map and the latent operator, is less suited for the diverse shapes present in the FlowBench dataset. In contrast, our OTNO model, which solves the OT plan/map for each shape separately, exhibits superior performance on diverse geometries.

The experimental settings are as follows. The inputs include Reynolds number, geometry mask (a binary representation of shape), and signed distance function (SDF); outputs are velocity in the x and y directions and pressure. Since our model is designed for geometry operator learning, we opt to test it on the Easy case, which uses a standard 80-20 random split, unlike the Hard case, which splits the dataset according to Reynolds number. We evaluate our model across three groups within the dataset: G1, which consists of parametric shapes generated using Non-Uniform Rational B-Splines (NURBS) curves; G2, which consists of parametric shapes generated using spherical harmonics; and G3, which consists of non-parametric shapes sampled from the grayscale dataset in SkelNetOn. For each group, the diversity of shapes is significantly greater than that of car designs in the aforementioned car dataset.

Table 4: Prediction under **M1** Metric (global) on FlowBench Dataset (single A100)

Group	Model	Relative L2	Time per Epoch (sec)	GPU Memory (MB)
G1	FNO	16.27%	43	4548
	GINO	8.62%	371	57870
	OTNO	**3.06%**	578	26324
G2	FNO	56.67%	43	4548
	GINO	43.16%	390	58970
	OTNO	**7.16%**	603	22868
G3	FNO	23.20%	44	4548
	GINO	13.27%	383	73140
	OTNO	**4.02%**	606	26008

Table 5: Prediction under **M2** Metric (boundary) on FlowBench Dataset (single A100)

Group	Model	Relative L2	Time per Epoch (sec)	GPU Memory (MB)
G1	FNO	5.65%	43	4536
	GINO	5.83%	190	67632
	OTNO	**3.91%**	135	7300
G2	FNO	29.37%	43	4534
	GINO	19.74%	177	73792
	OTNO	**14.36%**	124	7012
G3	FNO	10.47%	43	4538
	GINO	10.69%	219	67838
	OTNO	**7.18%**	153	11406

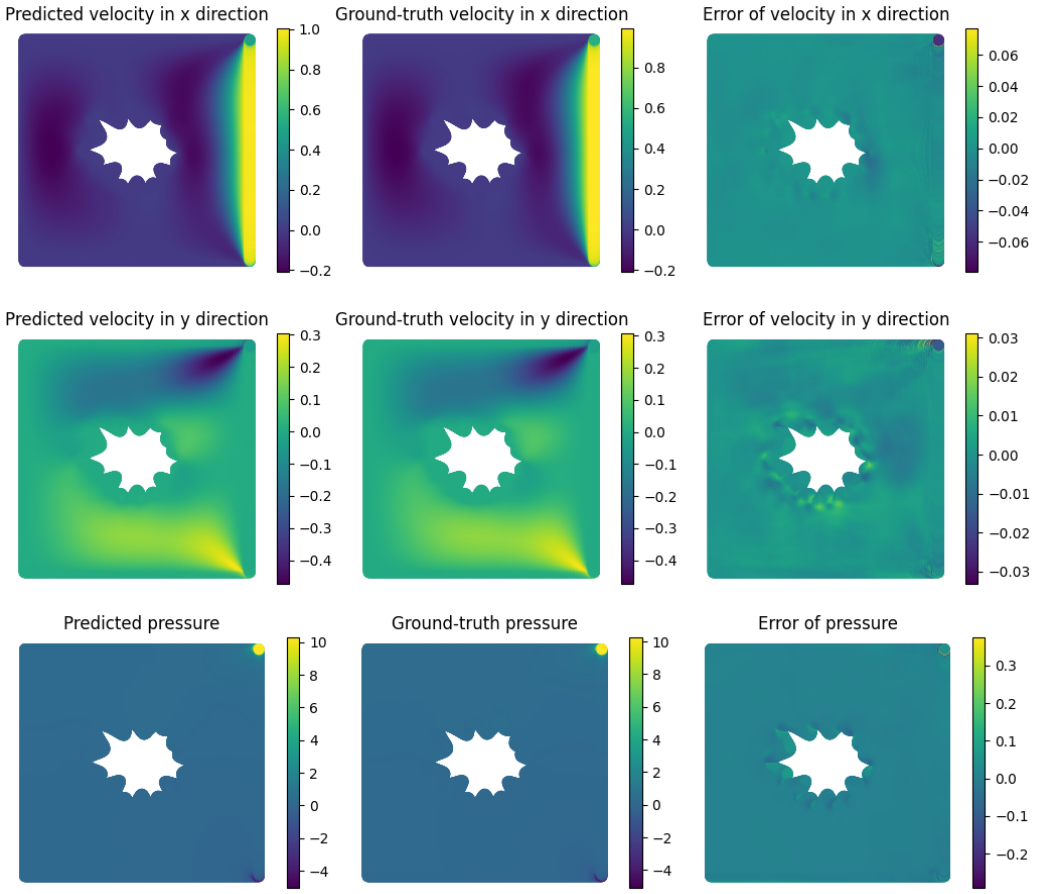


Figure 4: Results Visualization of Flow Dataset under M1 metric (full space)

D Ablation Studies

D.1 OTNO - Plan (Sinkhorn)

D.1.1 Encoder & Decoder Strategy

Using the Sinkhorn algorithm, we solve the Kantorovich optimal transport problem between the latent mesh $\Xi \in \mathbb{R}^{n_2}$ and the physical mesh $\mathcal{X} \in \mathbb{R}^{n_1}$, resulting in a large, dense coupling matrix $P \in \mathbb{R}^{n_2 \times n_1}$ that approximates the OT plan. The direct way to get transported mesh is as $\mathcal{X}' = P \cdot \mathcal{X}$, we refer to as the Matrix Strategy. However, storing these large, dense matrices for each physical mesh is too costly, and the approximate matrices obtained from the Sinkhorn Method introduce a degree of imprecision in the results. Therefore, this section discusses strategies to implement these approximate matrices more efficiently, aiming to conserve memory and reduce imprecision.

1. **Max vs Mean**
   As the matrix $P \in \mathbb{R}^{n_2 \times n_1}$ represents a discrete probability measure on $\Xi \times \mathcal{X}$, a natural approach to take use of the plan is to transport by the maximum probability, termed as "Max" strategy. Let $\mathcal{X} = (x_1, \ldots, x_{n_1}) \in \mathbb{R}^{n_1 \times d}$. Denote $P = (p_{k,l})_{n_2 \times n_1}$. We calculate the indices of the max element in each row

$$i_k = \arg\min_j \{p_{k,j} : j = 1, \ldots, n_1\}, \text{for } k = 1, \ldots, n_2, \tag{16}$$

   and then use these indices to get a refined transported mesh: $\mathcal{M} = (x_{i_1}, \ldots, x_{i_{n_2}}) \in \mathbb{R}^{n_2 \times d}$. Another approach, perhaps more intuitive, involves finding the element in mesh $\mathcal{X}$ that is closest to the directly transported mesh $\mathcal{X}' = (x'_1, \ldots, x'_{n_2}) \in \mathbb{R}^{n_2 \times d}$. We name this

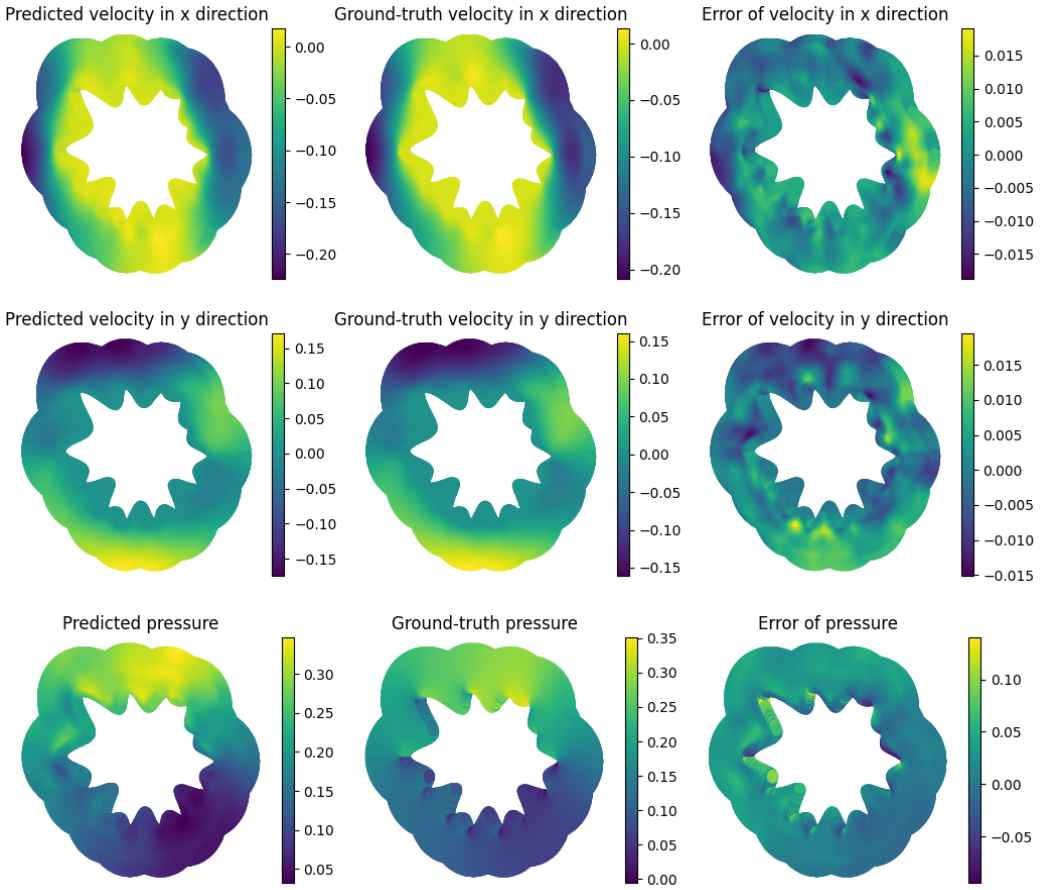


Figure 5: Results Visualization of Flow Dataset under M2 metric (boundary)

approach as "Mean" strategy due to the weight product across rows in $\mathcal{X}' = P \cdot \mathcal{X}$. The specific process is described as follows: first compute the indices

$$i_k = \arg\min_j \{|x'_k - x_j| : j = 1, \ldots, n_1\}, \text{for } k = 1, \ldots, n_2. \tag{17}$$

The refined transported mesh is $\mathcal{M} = (x_{i_1}, \ldots, x_{i_{n_2}}) \in \mathbb{R}^{n_2 \times d}$.
As shown in Table.6, "Mean" strategy has a better performance.

Table 6: Comparison of Max vs. Mean Strategy

Dataset	**Matrix**	**Max**	**Mean**
ShapeNet (Relative L2)	7.21%	7.01%	**6.70%**
DrivAerNet (MSE e-5)	5.6	4.8	**3.4**

2. **Single vs Multiple**
Besides the implementation of OT, integrating OT with FNO poses another significant question. For FNO, the requirement of inputs is just need to be features on a latent grid, suggesting that the encoder's output can combine multiple transported distributions, same for the decoder. Thus, under the "Mean" strategy, we further investigate utilizing indices of the top k closest elements, termed as "Multi-Enc" and "Multi-Dec". We choose $k = 8$ for "Multi-End" and $k = 3$ for "Multi-Dec" setups. The comparative results are presented in Table 7, indicating that "Multi-Enc" and "Multi-Dec" configurations underperform relative to "Single" strategy which utilize the index of the closest point instead of top k closest points.

Table 7: Comparison of Multi vs. Single Encoder/Decoder Strategy

Dataset	Multi Enc	Multi Dec	Single
ShapeNet (Relative L2)	20.55%	72.30%	**6.70%**
DrivAerNet (MSE e-5)	4.2	4.5	**3.4**

D.1.2 Normal Features

Our model, which incorporates latent FNO, learns the operator mapping from T to the latent solution function v, where T represents the transport map or the marginal map of the transport plan, as described in equations (5). The basic representation of the map T in our experiments can be defined as $\mathcal{T} = (\Xi, T(\Xi))$. The previous section discussed how to refine the transported mesh $T(\Xi)$, and in this section, we further explore the addition of normal features to enhance the representation $\mathcal{T}$, leveraging the normal's ability to describe a surface.

We propose three different approaches to integrate normals and compare them with the case where no normal features are added. The three methods are as follows: "Car": Only add car surface normals. "Concatenate": Add the concatenation of latent surface normals and car surface normals as normal features. "Cross Product": Add the cross product of latent surface normals and car surface normals as normal features to capture the deformation information of the transport.

As shown in Table 8, the "Cross Product" method performs the best.

Table 8: Comparison of Different Normal Features

Dataset	Non	Car	Concatenate	Cross Product
ShapeNet (Rel L2)	7.19%	6.82%	6.83%	**6.70%**
DrivAerNet (MSE e-5)	3.4	3.9	3.8	**3.4**

D.1.3 Shape of mesh in latent space

We investigated the effects of different shapes for the latent mesh, i.e., the target shapes for optimal transport. The results presented in Table 9 indicate that the torus provides the best performance due to its alignment with the periodic Fourier function. Although the capped hemisphere and the spherical surface share the same topological structure as the car surface, their performance is suboptimal. Additionally, we experimented with the double sphere method [Mildenberger and Quellmalz, 2023], which unfortunately yielded worse accuracy compared to the sphere and doubled the time costs.

Table 9: Comparison of Different Mesh Shapes in Latent Space

Dataset	Hemisphere	Plane	Double Sphere	Sphere	Torus
ShapeNet (Rel L2)	8.9%	8.67%	7.41%	7.09%	**6.70%**
DrivAerNet (MSE e-5)	4.7	4.4	4.6	4.1	**3.4**

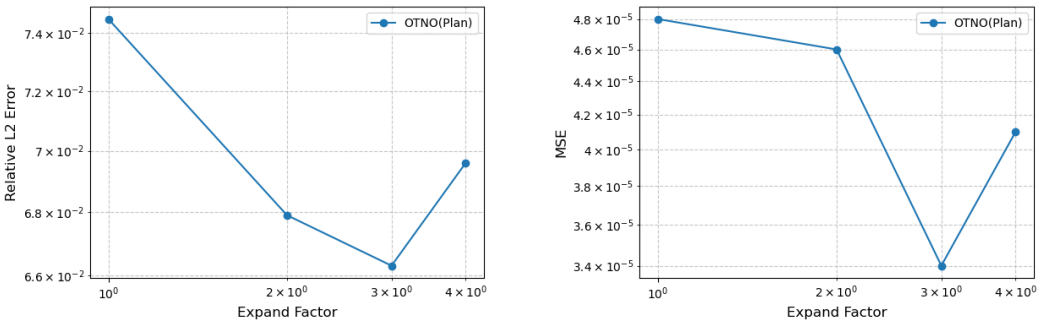
D.1.4 Expand Factor

For OTNO(Plan), we found that expanding the size of the latent mesh within a certain range leads to better results. The ablation experiments for different expansion factors $\alpha = 1, 2, 3, 4$are presented. As shown in Fig. 6, on both the DrivAerNet and ShapeNet datasets, $\alpha = 3$ achieves the best accuracy.

D.2 OTNO - Map (PPMM)

D.2.1 Number of Iterations

The theoretical time complexity of the PPMM is $O(Kn\log(n))$, where K is the number of iterations and n is the number of points. While the original study claims that empirically $K = O(d)$ works



(a) Tests of different expand factor for OTNO(Plan) on ShapeNet dataset

(b) Tests of different expand factor for OTNO(Plan) on DrivAerNet dataset

Figure 6: Ablation Studies of expand factor for OTNO(Plan)

reasonably well, our experience with 3D car datasets tells a different story. In this section, we conduct ablation studies to explore the relationship between the optimal number of iterations and the number of points. We employ voxel downsampling to generate subsets of the car surface mesh, varying in the number of points. The results for different voxel sizes (r) and iteration numbers (K) are presented in Table 10. From these results, we observe that $K \propto r^{-1}$. Given that the car surface mesh is a 2D manifold embedded in 3D space, the number of points n should be inversely proportional to the square of the voxel size r, i.e., $n \propto r^{-2}$. Consequently, $K \propto n^{1/2}$, and the experimental time complexity of PPMM in our model on the car dataset is accordingly $O(n^{3/2}\log(n))$.

Table 10: MSE (e-5) Results on the DrivAerNet Dataset for Different Voxel Sizes and Different Iteration Numbers of PPMM

Voxel Size / Itr	500	1000	2000	4000
0.2	**6.2**	6.6	6.9	8.0
0.1	5.3	**4.6**	5.5	5.4
0.05	4.8	4.2	**3.9**	4.5

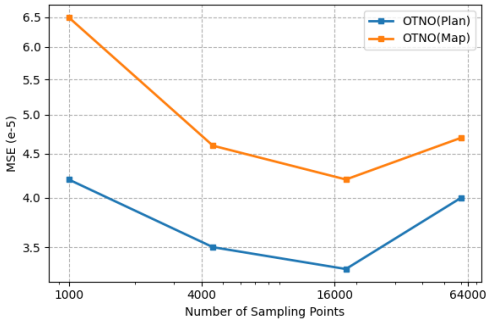
D.2.2 Sampling Mesh Size

It is a challenging problem to efficiently and effectively solve large-scale OT problems. We investigate the Sinkhorn method for large-scale OT plans and the PPMM algorithm for large-scale OT maps, and accordingly build two models, OTNO(Plan) and OTNO(Map). In this section, we conduct experiments on the large-scale DrivAerNet dataset to explore our models' ability to handle large sampling meshes. We use voxel downsampling to reduce the mesh with 200k points from the DrivAerNet dataset into a normalized sampling mesh of varying sizes.

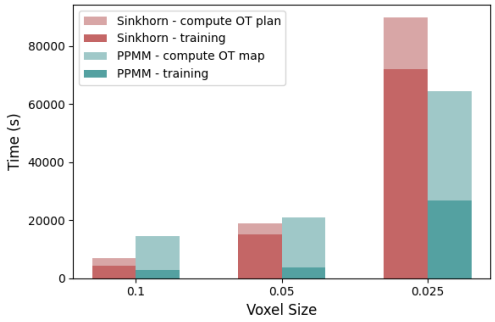
As shown in Fig. 7a, both OTNO(Plan) and OTNO(Map) achieve their lowest accuracy at a voxel size of 0.05, corresponding to approximately 18k points. However, further increasing the sampling mesh size does not lead to improved accuracy. This suggests that our model has a range of limitations when dealing with large-scale problems, primarily due to the difficulty of solving large-scale OT problems with high precision. Regarding the comparison between OTNO(Plan) and OTNO(Map), we find that OTNO(Plan) consistently outperforms OTNO(Map) in terms of accuracy, regardless of the sampling mesh size.

From the experimental results, we observe that the training time of OTNO(Plan) is much larger than OTNO(Map) as shown in Fig. 7b. This is because we expand the latent space by a factor of 3 for OTNO(Plan), and the FNO on the latent space has linear complexity, attributed to the linear FFT. Note that the time complexity of Sinkhorn and PPMM are $O(n^2)$ and $O(n^{3/2}\log(n))$, respectively, both of which are larger than the FNO training complexity of $O(n)$. Therefore, the overall time complexity of OTNO(Plan) and OTNO(Map) relative to the number of points n should be $O(n^2)$ and

$O(n^{3/2}\log(n))$, respectively. These results align with the plots shown in Fig. 7b that the time cost of OTNO(Plan) increases faster than that of OTNO(Map) as the sampling mesh size grows.



(a) Test errors associate with the number of sampling points. We set the voxel size $r = 0.2, 0.1, 0.05, 0.025$, corresponding to about 1k, 4.5k, 18k, 60k points.



(b) Time complexity associate with the number of sampling points. We set the voxel size $r = 0.1, 0.05, 0.025$, corresponding to about 4.5k, 18k, 60k points.

Figure 7: Ablation Studies of Sampling Mesh Size for OTNO(Map) and OTNO(Plan) on DrivAerNet Dataset