Compressible and Inviscid Fluid Flow Modeling Using Graph Neural Networks

Boscariol Jacopo, MSc Mechanical Engineering
Supervisors: Prof. Olga Fink, Vinay Sharma
Intelligent Maintenance and Operations Systems Laboratory, EPFL, Switzerland

***Abstract*—High-fidelity simulations of compressible fluid flows are computationally prohibitive for real-time applications. While Graph Neural Networks (GNNs) have emerged as promising fast-inference surrogates, standard architectures often struggle to maintain physical consistency, leading to significant mass and energy drift over long rollouts. This limitation is particularly acute in compressible regimes characterized by discontinuities and shockwaves. This work introduces the Flux Conservative GNN, a physics-informed architecture designed for the Inviscid Euler equations. By integrating a Finite Volume Method (FVM) backbone directly into the message-passing framework, the proposed architecture enforces exact flux conservation and numerical causality via a dynamic Courant-Friedrichs-Lewy (CFL) condition. Furthermore, SE(2)-equivariant edge operations and a recurrent spatio-temporal edge memory are incorporated to capture rotationally invariant dynamics and temporal dependences. Evaluated on a diverse dataset of compressible gases, this approach demonstrates superior stability and robust out-of-distribution generalization, achieving these results with minimal training epochs and significantly less data than standard baselines.**

I. Introduction

A. Motivation

Modeling complex fluid dynamics is fundamental to engineering applications ranging from aerospace to energy systems. The governing laws are highly non-linear and sensitive to initial conditions. Traditional numerical solvers based on the Finite Volume Method (FVM) are robust but computationally expensive, often requiring fine meshes and small time-steps to resolve features like shockwaves. This computational cost creates a bottleneck for applications requiring real-time feedback, such as design optimization.

B. Data-Driven Surrogates and Limitations

In recent years, Geometric Deep Learning has offered a promising alternative. A purely data-driven example is MeshGraphNets [1], which demonstrated impressive capability in simulating incompressible flows. Brandstetter et al. [2] further formalized the link between Message Passing and numerical PDE solvers, suggesting that GNNs can learn to approximate differential operators. Parallel advancements in *equivariant* architectures, such as EGNNs [3], have further improved data efficiency by guaranteeing that learned dynamics respect physical symmetries like rotation and translation.

However, applying such models to *compressible* inviscid flows (Euler equations) presents unique challenges. In compressible regimes, where density, energy, and momentum are tightly coupled, the lack of strict conservation leads to non-physical accumulation of errors (drift) over long simulation rollouts. Furthermore, without explicit numerical stabilization, these models suffer from numerical instabilities, making long-term predictions un-physical. Hybrid approaches have attempted to bridge this gap. For instance, Kochkov et al. [4] integrated neural networks within a standard CFD solver to correct discretization errors on coarse grids, maintaining conservation.

C. Contributions

To address these limitations, this report proposes a hybrid architecture that embeds the rigorous structure of FVM solvers into the architecture of a GNN. The specific contributions are as follows:

1) **Baseline Evaluation:** MeshGraphNets was implemented and evaluated on an incompressible cylinder flow dataset to identify critical limitations in long-term stability and error accumulation, establishing a baseline for improvement.
2) **Flux Conservative GNN:** A novel architecture is developed to act as a learnable flux function. Unlike standard GNNs that predict state updates directly, this model predicts cell fluxes, guaranteeing conservation of mass, momentum, and energy by design. Key features include:
   - SE(2) Equivariant Edge Operations: By projecting momentum onto local edge normal/tangent basis, the model learns rotationally invariant physics and incorporates advection.
   - Spatio-Temporal Edge Memory: Inspired by the DynamiCAL framework [5], recurrent edge states are implemented, improving convergence.
   - Stability Mechanisms: Learned artificial viscosity is implemented to capture shocks without oscillation, alongside a dynamic CFL condition to guarantee numerical causality.
3) **Robust Training Strategy:** Stability is enhanced through a 5-step unrolled autoregressive training, noise injection, and grid resolution coarsening to minimize computational costs.

Results on the "The Well" dataset [6] indicate that the Flux Conservative GNN significantly outperforms baselines in stability and generalization, accurately simulating unseen gases with varying thermodynamic properties. The codes can be found here: Link.

II. Models and Methods

A. Problem Formulation and Datasets

This work evaluates surrogate models on two distinct 2D regimes of fluid dynamics: incompressible flow past a cylinder and compressible inviscid gas dynamics. Each regime is governed by a specific set of conservation laws and represented by a distinct dataset.

1) Incompressible Cylinder Flow: To establish a baseline for standard GNN performance, the "Cylinder Flow" dataset from Pfaff et al. [1] is adopted. This problem models the flow of water past an infinite cylinder, governed by the incompressible Navier-Stokes equations:

$$\begin{aligned} &\frac{\partial u_i}{\partial x_i} = 0, \\ &\rho\left(\frac{\partial u_i}{\partial t} + u_j\frac{\partial u_i}{\partial x_j}\right) = -\frac{\partial p}{\partial x_i} + \mu\frac{\partial^2 u_i}{\partial x_j \partial x_j}, \end{aligned} \tag{1}$$

where u_i denotes the velocity component in the i-th direction (with $i, j \in \{1, 2\}$), p is pressure, ρ is the constant fluid density, and μ is the dynamic viscosity. The dataset consists of 1000 trajectories with 600 time-steps each (6 seconds total) simulated on irregular 2D triangular meshes (1885 average number of nodes).

2) Compressible Euler Gas Dynamics: The primary contribution of this work targets the more challenging regime of compressible, inviscid flow. The adopted dataset is from "The Well" [6], and consists of approximately 5TB of 2D simulations with periodic boundary conditions. Unlike the incompressible case, density is a variable, and energy is coupled to momentum. Thus, the dynamics are governed by the compressible Euler equations [7], which express the conservation of mass, momentum and energy (with spatial indices $i, j \in \{1, 2\}$):

$$\begin{aligned} &\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u_j)}{\partial x_j} = 0, \\ &\frac{\partial(\rho u_i)}{\partial t} + \frac{\partial(\rho u_i u_j + p\delta_{ij})}{\partial x_j} = 0, \\ &\frac{\partial E}{\partial t} + \frac{\partial((E+p)u_j)}{\partial x_j} = 0. \end{aligned} \tag{2}$$

The system evolves the state vector of conserved variables $\mathbf{U} = [\rho, \rho u_1, \rho u_2, E]^\top$. Here, δ_{ij} is the Kronecker delta, and the term $(\rho u_i u_j + p\delta_{ij})$ represents the momentum flux tensor, accounting for both advective transport and isotropic pressure. The system is closed by the following equation of state:

$$p = (\gamma - 1)\left(E - \frac{1}{2}\rho u_k u_k\right), \tag{3}$$

where γ is the adiabatic constant and $\frac{1}{2}\rho u_k u_k$ ($k \in \{1, 2\}$) represents the kinetic energy density.

The original dataset provides high-resolution trajectories (512×512 grid) for 10 distinct gases with varying γ. Each simulation lasts 1.5 seconds distributed on 100 time-steps. To make training more efficient, the data was coarsened to a 128×128 resolution via average pooling.

B. MeshGraphNets

To benchmark performance on the incompressible regime, the MeshGraphNets architecture [1] is adopted.

The computational domain is discretized as a directed graph $G = (V, E)$, where nodes V correspond to the mesh vertices (spatial coordinates) and edges E represent the mesh connectivity. The input node features x_i consist of the current velocity vector $\mathbf{u}_i$ and a one-hot vector $\mathbf{b}_i$ encoding the boundary type (fluid, wall or inflow). Edge features e_{ij} encode the relative displacement $\mathbf{r}_{ij} = \mathbf{x}_i - \mathbf{x}_j$ and its Euclidean norm $\|\mathbf{r}_{ij}\|$.

The model follows the Encoder-Processor-Decoder framework:

1) **Encoder:** Multi-Layer Perceptrons (MLPs) project node inputs x_i and edge inputs e_{ij} into latent embeddings h_i^0 and ϵ_{ij}^0.
2) **Processor:** At each layer l, edge and node latents are updated with residuals:

$$\begin{aligned} \epsilon_{ij}^{l+1} &= \phi^e(h_i^l, h_j^l, \epsilon_{ij}^l) + \epsilon_{ij}^l, \\ h_i^{l+1} &= \phi^v\left(h_i^l, \sum_{j \in \mathcal{N}(i)} \epsilon_{ij}^{l+1}\right) + h_i^l, \end{aligned} \tag{4}$$

   where ϕ^e and ϕ^v are learned MLPs and $\mathcal{N}(i)$ denotes the neighborhood of node i.
3) **Decoder:** A final MLP decodes the processed node embeddings h_i^L to the velocity update.

Training minimizes the Mean Squared Error (MSE) between the predicted and ground-truth velocity updates. Note that the conservation of mass and momentum is not structurally enforced. Consequently, errors in the predicted update accumulate autoregressively, leading to significant drift in long rollouts, as demonstrated in Section III.

C. Flux Conservative GNN

To address the limitations of standard message passing in compressible regimes, this work introduces the Flux Conservative GNN. This architecture learns the flux function within a Finite Volume Method (FVM) framework, strictly enforcing conservation laws.

1) Finite Volume Backbone: Unlike MeshGraphNets, which predicts node updates directly, the proposed model predicts the numerical flux $\mathbf{F}_{ij}$ across the interface between

cells i and j. The state update for node i is computed by explicitly summing these fluxes:

$$\mathbf{U}_i^{t+1} = \mathbf{U}_i^t - \frac{\Delta t}{V_i} \sum_{j \in \mathcal{N}(i)} A_{ij} \mathbf{F}_{ij}, \tag{5}$$

where V_i is the cell volume (area in 2D) and A_{ij} is the interface area (edge length). By construction, any flux leaving cell i enters cell j ($\mathbf{F}_{ij} = -\mathbf{F}_{ji}$), thus ensuring global conservativeness.

2) SE(2)-Equivariant Edge Embeddings and Advection: To ensure the learned physics is invariant to the mesh orientation in space, the model employs SE(2)-equivariant operations by passing only the edge distance r_{ij} to the edge encoder: $\epsilon_{ij} = \phi_{\text{edge}}(r_{ij})$. Then, the relative velocity and momentum vectors are projected onto a local orthonormal basis formed by the edge normal $\mathbf{n}_{ij}$ and tangent $\mathbf{t}_{ij}$:

$$\chi_{ij} = \left[\rho \mathbf{u}_i \cdot \mathbf{n}_{ij}, \rho \mathbf{u}_i \cdot \mathbf{t}_{ij},\ \rho \mathbf{u}_j \cdot \mathbf{n}_{ij}, \rho \mathbf{u}_j \cdot \mathbf{t}_{ij} \right]^\top. \tag{6}$$

This projection allows the network to incorporate the needed advection information.

3) Edge Memory and Shock Detection: The proposed architecture includes a recurrent *edge memory*, inspired by the DynamiCAL framework [5]. This memory state is present across the L message-passing layers within a single time-step, emulating a sub-time stepping process. At each layer l, the message m_{ij}^l is updated using both spatial neighbors and its own previous state:

$$\begin{aligned} m_{ij}^{curr} &= \phi_{\text{msg}}\left(h_i^l, h_j^l, v_{\text{avg}}^l, v_{\text{diff}}^l, \epsilon_{ij}, \chi_{ij}\right), \\ m_{ij}^{l+1} &= \text{LayerNorm}\left(m_{ij}^l + m_{ij}^{curr}\right). \end{aligned} \tag{7}$$

In Equation 7 $v_{\text{avg}}^l = h_i + h_j$ and $v_{\text{diff}}^l = (h_i - h_j)^2$ are two symmetric combinations of node embeddings h_i and h_j that both enforce node permutation symmetry and detect local high gradients regions, i.e. shockwaves.

Finally, the inviscid numerical flux $\mathbf{F}_{ij}^{\text{inviscid}}$ is constructed by decoding the message state via MLPs for mass, momentum and energy, which predict the flux amplitudes along the local basis vectors $\mathbf{n}_{ij}$ and $\mathbf{t}_{ij}$.

4) Numerical Stability and Causality: Two critical mechanisms are introduced to stabilize the model in the presence of shocks:

- **Learned Artificial Viscosity:** To prevent the checkerboard instability (typical of inviscid solvers), the network learns a viscosity coefficient $\alpha_{ij} \in [0, 1]$. The final flux is computed as:

$$\mathbf{F}_{ij} = \mathbf{F}_{ij}^{\text{inviscid}} - \alpha_{ij} \left(\mathbf{U}_j - \mathbf{U}_i\right). \tag{8}$$

- **Dynamic CFL Condition:** By default, the time-step used in Equation 5 is $\Delta t = \frac{\Delta t_{\text{GT}}}{L}$, where Δt_{GT} is the simulation physical time-step and L is the number of message passing layers. However, numerical causality requires that information does not propagate faster than the grid allows. Thus, at every step the model computes the maximum wave speed $\lambda_{max} = \|\mathbf{u}\| + c$ (where c is sound speed) and clamps the integration time-step Δt to satisfy the Courant-Friedrichs-Lewy (CFL) condition:

$$\Delta t \leq \text{CFL} \cdot \frac{\Delta x_{\min}}{\max_i(\lambda_{\max,i})}. \tag{9}$$

The CFL number has been fixed to 0.6, following standard FVM solvers values.

An illustration of the model architecture is displayed in Figure 1.

D. Training Strategy

The original 512×512 simulation grid is coarsened to 128×128. This reduces the computational costs and, critically, relaxes the CFL constraint, allowing for larger physically valid time-steps during training. Indeed, from Equation 9, a larger $\Delta x_{\min}$ allows a larger Δt.

Gaussian noise is injected into the input state $\mathbf{U}^t$ during training. This forces the model to learn robust flux features that can recover from small perturbations, preventing drift from small autoregressive errors.

Minimizing one-step error often fails to produce stable autoregressive rollouts. Therefore, the model is trained using a during-training rollout of $K = 5$ steps: the predicted output at step t serves as the input for step $t+1$. The loss function minimizes the cumulative error over the unrolled trajectory:

$$\mathcal{L} = \sum_{k=1}^{5} \gamma^k \|\mathbf{U}^{t+k} - \mathbf{U}_{\text{GT}}^{t+k}\|_2^2, \tag{10}$$

where $\mathbf{U}$ is the predicted state, $\mathbf{U}_{\text{GT}}$ is the ground truth and γ^k the weight of step k. This "training on predictions" approach exposes the network to its own autoregressive errors, teaching it to remain stable over long simulations.

Finally, the Flux Conservative GNN was trained for only 7 epochs on a single NVIDIA Tesla V100-PCIE-32GB GPU.

III. Results

A. MeshGraphNets: Incompressible Cylinder Flow

The MeshGraphNets model was trained on the Cylinder Flow dataset. The Root Mean Squared Error (RMSE) has been computed for one-step predictions, where the model is fed the ground truth at every time step, and for the full autoregressive rollout. As shown in Table I, the model achieves high accuracy on short-term predictions (1-step), which deteriorates over long horizons. Purely data-driven models as MeshGraphNets are known to be data-hungry: Table I also shows how much the error increases when training on half the amount of data. The 50% training data full rollout RMSE is more than two times higher than the full dataset case.

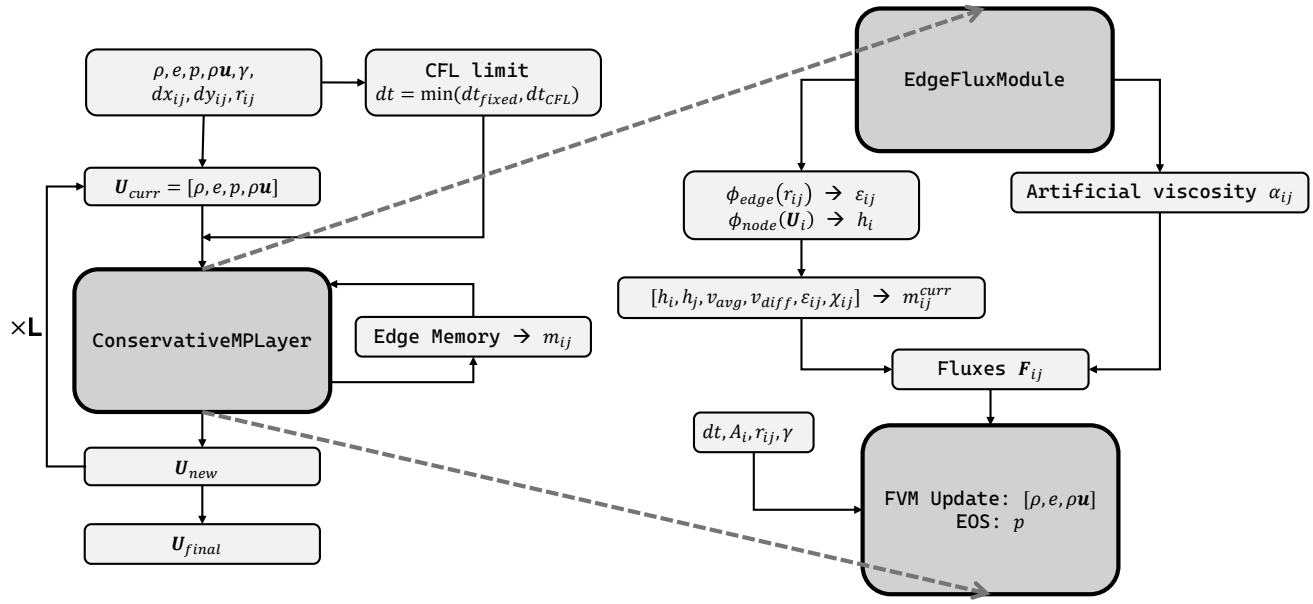

Figure 1: Overview of the Flux Conservative GNN architecture. The model takes the initial state as input, encodes edges and advection (χ_{ij}) using SE(2)-equivariant projections, and evolves a latent edge memory m_{ij} through L message-passing layers. The decoder predicts interface fluxes $\mathbf{F}_{ij}$ (inviscid and viscous) to update the state $\mathbf{U}$ via a Finite Volume scheme.

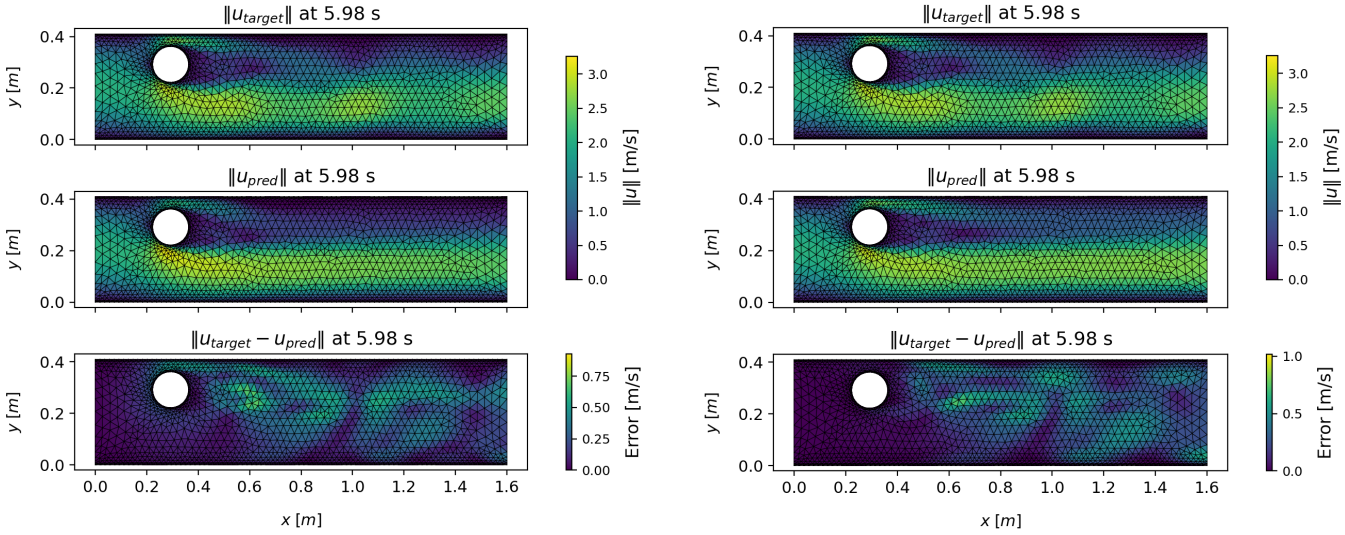

Figure 2: Comparison of velocity magnitude at the last time-step of the simulation. The bottom left panel shows the error map when trained on the full dataset, while the right bottom panel shows the result when trained on only 50% of the trajectory data. Neither of the two is precisely capturing the vortex shedding wake shown in the first panel (target velocity field).

Case	Training Data [%]	Velocity RMSE [m/s]
one-step Prediction	100	$(2.37 \pm 0.12) \times 10^{-3}$
Full Rollout	100	$(40.88 \pm 7.2) \times 10^{-3}$
Full Rollout	50	$(95.84 \pm 11.4) \times 10^{-3}$

Table I: MeshGraphNets velocity RMSE values for the incompressible flow past a 2D cylinder. The full rollout case (600 time-steps, 6 seconds) is reported for both the full and half dataset training.

B. Flux Conservative GNN: Euler Gas Flow

The proposed architecture was trained and evaluated on the compressible Euler gases dataset from [6]. To assess robust generalization, a rigorous stratification strategy was employed based on the thermodynamic properties of the ten available gases.

Given the computational constraints, training was performed *exclusively* on a single gas: Dry Air at $20°C$ ($\gamma = 1.4$). This gas was selected as a representative middle point of the dataset in the feature space defined by the adiabatic

constant γ, mean total energy, and mean momentum. To evaluate zero-shot Out-Of-Distribution (OOD) generalization, the remaining nine gases were clustered into varying levels of difficulty based on their Euclidean distance from the training distribution in this 3D feature space (see Figure 3):

- **In-Distribution (ID):** Gases with thermodynamic properties and flow conditions similar to the training set, though distinct in composition (e.g., CO_2 at $20°C$, $\gamma = 1.3$).
- **OOD Mild:** Gases exhibiting slight thermodynamic deviations (e.g., mixture of H_2 at $100°C$ and Dry Air at $-15°C$, $\gamma = 1.404$).
- **OOD Moderate:** Gases representing a distinct thermodynamic regime (e.g., H_2 at $-76°C$, $\gamma = 1.453$).
- **OOD Strong:** Gases with severe shifts in thermodynamic properties (e.g., Ar at $-180°C$, $\gamma = 1.76$).

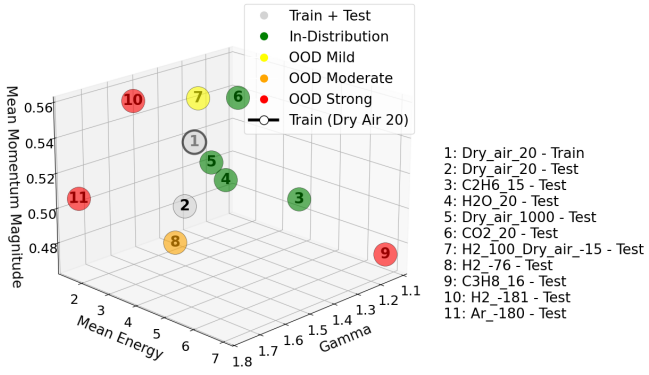

Figure 3: Dataset stratification strategy. The model is trained on a single gas (Dry Air at $20°C$, number 1). The other white circled gas is the Dry Air corresponding test case. The remaining gases are classified into ID, OOD Mild, OOD Moderate, and OOD Strong levels based on their distance in the (γ, Energy, Momentum) space.

Performance is reported using the Variance-Normalized Root Mean Squared Error (VRMSE). Unlike standard RMSE, this metric is normalized by the standard deviation of the ground truth data, allowing for a scale-independent comparison across different physical quantities:

$$\mathrm{VRMSE}(u, v) = \left(\frac{\langle |u - v|^2 \rangle}{\langle |u - \bar{u}|^2 \rangle + \epsilon} \right)^{1/2}. \qquad (11)$$

1) One-Step Prediction: First, the model's ability to approximate the immediate next state ($t \to t + 1$) was evaluated. Table II compares the proposed Flux Conservative GNN against state-of-the-art baselines from "The Well" benchmark: Fourier Neural Operator (FNO), Temporal FNO (TFNO), U-Net, and ConvNext U-Net. It is crucial to note a fundamental difference in training methodology. The baseline models were trained on the *entire* dataset (all 10 gases), meaning all test cases are effectively In-Distribution (ID) for them. In contrast, the proposed model was trained *only* on Dry Air and must zero-shot generalize to the other regimes. Despite this, Flux Conservative GNN outperforms the corresponding baselines ID cases and remains extremely competitive also in OOD ones.

2) Autoregressive Prediction: The primary advantage of the proposed architecture emerges in long-term rollouts, where errors accumulate autoregressively. Table II presents the averaged VRMSE over medium (steps 6-12) and long (steps 13-30) horizons.

Notably, in the Strong OOD regime, the Flux Conservative GNN remains stable. This demonstrates that the model has successfully learned the underlying Euler flux operator rather than memorizing the training gas statistics. Figures 4 and 5 respectively display the one-step prediction and full autoregressive rollout against the ground truth for the density, energy, momentum and pressure fields.

Model *(Training Data)*	**Dry Air** *(Ref.)*	**ID**	**OOD** *Mild*	**OOD** *Mod.*	**OOD** *Strong*
A. One-step Prediction ($t \to t+1$)					
Baselines (Trained on All Gases)					
FNO			0.4081		
TFNO			0.4163		
U-Net			0.1834		
CNextU-Net			0.1531		
Proposed (Trained on Dry Air Only)					
Flux Cons. GNN	**0.0596**	0.1040	0.0815	0.0807	0.2793
B. Autoregressive Rollout (Steps 6-12)					
Baselines (Trained on All Gases)					
FNO			1.13		
TFNO			1.23		
U-Net			1.02		
CNextU-Net			4.98		
Proposed (Trained on Dry Air Only)					
Flux Cons. GNN	**0.888**	0.946	0.909	0.971	1.03
C. Autoregressive Rollout (Steps 13-30)					
Baselines (Trained on All Gases)					
FNO			1.37		
TFNO			1.52		
U-Net			1.63		
CNextU-Net			> 10		
Proposed (Trained on Dry Air Only)					
Flux Cons. GNN	**1.28**	1.29	1.29	1.34	1.39

Table II: Performance Comparison (VRMSE) across one-step Prediction and Autoregressive Rollouts (Lower is Better). The "**Dry Air** *(Ref.)*" column represents the strict reference case (same gas as training). The subsequent columns show performance on increasingly distinct gases. In the Dry Air test case, which corresponds to the baselines' ID training, Flux Conservative GNN outperforms the other models significantly. Even in the most severe OOD case (Strong) and long horizons (Panel C), the proposed model remains competitive and stable while baselines degrade.

IV. Discussion

The results on the incompressible cylinder flow highlight a structural weakness of purely data-driven architectures like MeshGraphNets. While the model achieves low error on

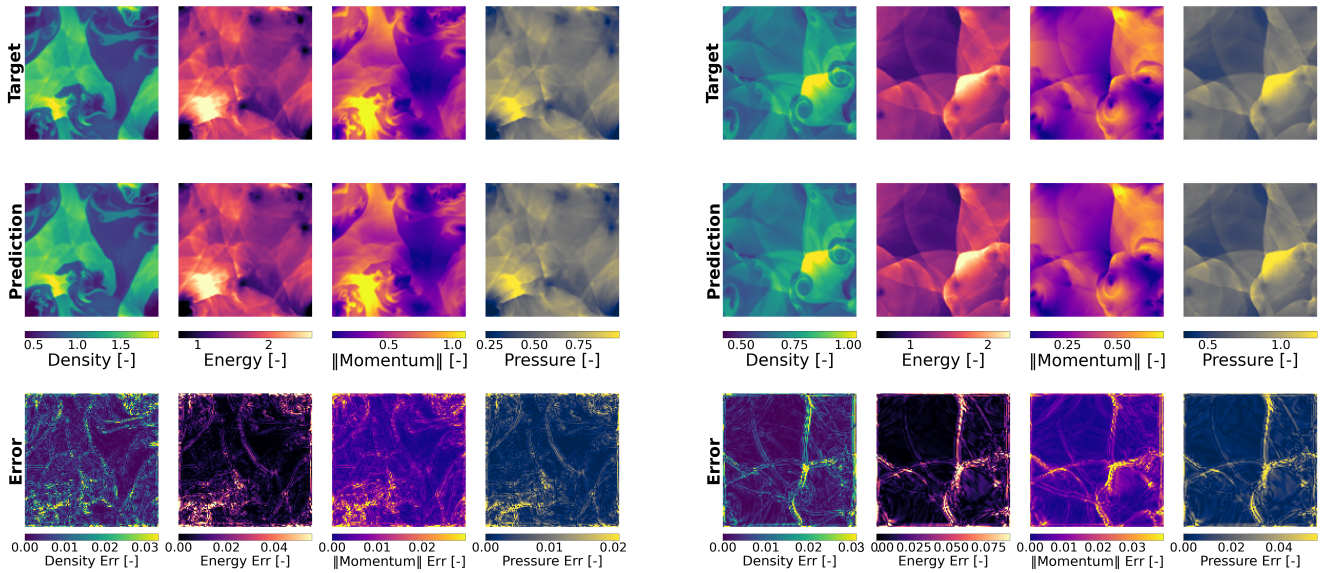

Figure 4: One-step prediction comparison between Flux Conservative GNN results (first row) and ground truth (second row) for density, energy, momentum magnitude and pressure. The third row reports the difference between the model predictions and the true flow fields. On the left, Dry Air at $20^{\circ}C$ is reported (training gas test set), while on the right H_2 at $-181^{\circ}C$ is shown (strong OOD gas). The predictions match nicely the ground truth in both the easy and hard test cases. Notice that, due to the periodic boundary conditions, whatever leaves one of the edges enters the opposite one.

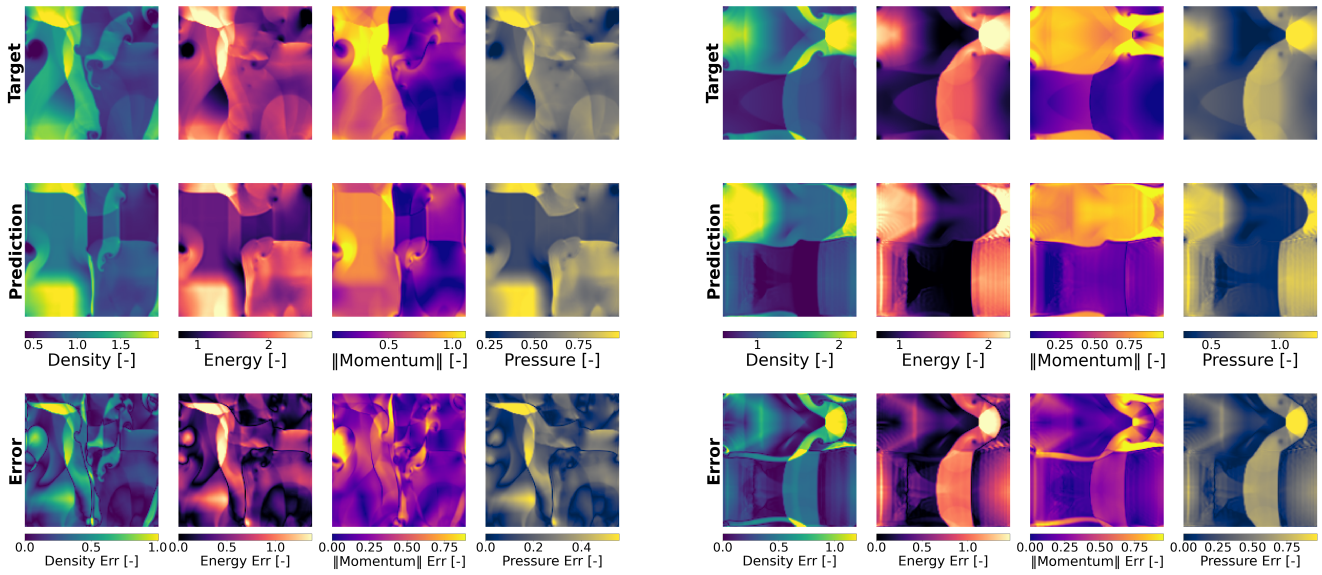

Figure 5: Full autoregressive rollout prediction comparison between Flux Conservative GNN results (first row) and ground truth (second row) for density, energy, momentum magnitude and pressure. The third row reports the difference between the model predictions and the true flow fields. On the left, Dry Air at $20^{\circ}C$ is reported (training gas test set), while on the right H_2 at $-181^{\circ}C$ is shown (strong OOD gas). The prediction starts to degrade for the strong OOD case, but the model is still capturing the non-linearities and shockwaves to some extent.

one-step predictions (2.37×10^{-3}), the rollout error accumulates significantly over 600 steps. Furthermore, Table I shows that when reducing by half the training dataset, the testing error more than doubles, highlighting how data-hungry such models can be. This degradation is a direct consequence of the lack of conservative constraints. Indeed,

in a standard MeshGraphNets the predicted velocity updates are not guaranteed to satisfy $\nabla \cdot \mathbf{u} = 0$. Consequently, the model depends on large-scale data exposure to learn physics purely as a soft constraint, resulting in a weak inductive bias.

The Flux Conservative GNN addresses this problem. By predicting interface fluxes $\mathbf{F}_{ij}$ instead of nodal states and imposing that whatever leaves cell i must also enter the neighbouring cell j ($\mathbf{F}_{ij} = -\mathbf{F}_{ji}$), the mass, energy and momentum are conserved by construction, forcing errors to remain local rather than accumulating globally.

The baselines from [6] (FNO, TFNO, U-Net, CNextU-Net) were trained on the complete dataset (all 10 gases), meaning they have effectively been tested within the training distribution. In contrast, the Flux Conservative GNN was trained *only* on one gas (Dry Air) and evaluated zero-shot on unseen gas groups distributed between In-Distribution and Mild, Moderate, and Strong OOD. Furthermore, the model was trained for just 7 epochs on a single GPU.

Despite these disadvantages, the proposed model appears sample-efficient as it outperforms all baselines in both accuracy and stability (Table II). This suggests that the Flux Conservative GNN learns the underlying differential operator of the Euler equations. The ability to stably simulate the "Strong OOD" group (despite quality degradation), which exhibits significantly different flow fields than Dry Air, confirms that the SE(2)-equivariant flux formulation successfully separates the fluid physics from the specific thermodynamic constants. Furthermore, the introduction of edge memory, shock-detection features and artificial viscosity mitigate error accumulation, leading to stable rollouts even for OOD gases. Finally, imposing a CFL constraint, following typical FVM solvers implementations, ensures numerical causality in the entire domain.

The superior stability of the Flux Conservative GNN comes at the cost of increased architectural complexity and training cost. Unlike standard GNNs that operate on raw graph inputs, the proposed framework requires a strict Finite Volume mesh definition (calculation of normals, cell areas) and a specialized flux computation module. Furthermore, the inclusion of the autoregressive unrolling strategy significantly increases the training time compared to simple one-step baselines. However, the experimental results on the Euler dataset suggest that this additional cost is a necessary condition for successful surrogate modelling of compressible flows of this kind. While purely data-driven models offer easier implementation and faster training, they lack the structural robustness required to handle the typical discontinuities of compressible gas simulations.

A significant limitation of the current formulation is the global coupling of the integration time-step to the minimum grid spacing Δx_{min} in Equation 9. In cases involving non-uniform discretization, such as refined boundary layers around aerodynamic profiles, the global Δt is severely constrained by a small subset of the domain. This leads to over-resolved temporal integration in coarser regions, significantly increasing the total computational cost required to reach a target physical time.

V. Summary

This work benchmarked the limitations of standard Graph Neural Networks in fluid dynamics and proposed an architecturally physics-informed alternative. Evaluations on the incompressible cylinder flow baseline demonstrated that high one-step accuracy does not translate to long-term stability due to the violation of conservation laws. To solve this, the **Flux Conservative GNN** was developed, integrating a Finite Volume backbone, SE(2)-equivariant projections, and explicit shock-capturing mechanisms into the message-passing framework.

Experiments on the Euler gases dataset showed that the proposed architecture achieves superior autoregressive stability compared to state-of-the-art baselines, despite being trained on a single gas (Dry Air) versus the full dataset. This also highlights that incorporating physical knowledge in the model architecture lowers the needed amount of training data to achieve satisfactory performance. Anyway, increasing the number of training epochs or the volume of training data (e.g., training on multiple gases) is expected to further improve the final performance. The model successfully generalizes zero-shot to unseen gas groups with widely varying thermodynamic properties, validating the hypothesis that enforcing strict numerical structure (FVM conservation and CFL causality) is essential for robust, generalizable neural surrogates in computational fluid dynamics.

A. Future Works

Future research could investigate a Convective-Pressure Flux Splitting (CPS) strategy, inspired by the formulation presented in [8]. In 2D, this approach separates the numerical flux into a convective component $\mathbf{F}_c = [\rho u, \rho u^2, \rho uv, Eu]^\top$ $\left(\mathbf{G}_c = [\rho v, \rho v^2, \rho vu, Ev]^\top\right)$ and a pressure component $\mathbf{F}_p = [0, p, 0, pu]^\top$ $\left(\mathbf{G}_p = [0, p, 0, pv]^\top\right)$, effectively isolating mass transport from pressure wave propagation.

The pressure flux propagates isotropically and significantly faster than the convective velocity $\mathbf{u}$, since the wave speed is $\|\mathbf{u}\| + c \gg \|\mathbf{u}\|$, where c is the speed of sound. Therefore, the idea is to process $\mathbf{F}_p$ $(\mathbf{G}_p)$ via a separate Graph Neural Network layer (e.g., a MeshGraphNets) operating on a coarsened mesh or utilizing increased message-passing steps. This multi-scale approach would decouple the slow and fast dynamic scales, potentially improving stability. Furthermore, processing pressure and momentum on distinct graph structures (e.g., distinct meshes) mimics a staggered grid formulation. This approach should naturally suppress checkerboard instabilities, potentially reducing the reliance on the learned artificial viscosity currently required for stabilization.

REFERENCES

[1] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. W. Battaglia, "Learning Mesh-Based Simulation with Graph Networks," in *International Conference on Learning Representations (ICLR)*, 2021. [Online]. Available: https://arxiv.org/abs/2010.03409

[2] J. Brandstetter, D. Worrall, and M. Welling, "Message Passing Neural PDE Solvers," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://arxiv.org/abs/2202.03376

[3] V. G. Satorras, E. Hoogeboom, and M. Welling, "E(n) Equivariant Graph Neural Networks," in *International Conference on Machine Learning (ICML)*. PMLR, 2021, pp. 9323–9332. [Online]. Available: https://arxiv.org/abs/2102.09844

[4] D. Kochkov, J. A. Smith, A. Alieva, Q. Wang, M. P. Brenner, and S. Hoyer, "Machine learning-accelerated computational fluid dynamics," *Proceedings of the National Academy of Sciences*, vol. 118, no. 21, 2021.

[5] V. Sharma and O. Fink, "Dynami-CAL GraphNet: A Physics-Informed Graph Neural Network Conserving Linear and Angular Momentum for Dynamical Systems," *arXiv preprint arXiv:2501.07373*, 2025. [Online]. Available: https://arxiv.org/abs/2501.07373

[6] R. Ohana, M. McCabe, L. Meyer, R. Morel, F. J. Agocs, M. Beneitez, M. Berger, B. Burkhart, S. B. Dalziel, D. B. Fielding, D. Fortunato, J. A. Goldberg, K. Hirashima, Y.-F. Jiang, R. R. Kerswell, S. Maddu, J. Miller, P. Mukhopadhyay, S. S. Nixon, J. Shen, R. Watteaux, B. Régaldo-Saint Blancard, F. Rozet, L. H. Parker, M. Cranmer, and S. Ho, "The Well: a Large-Scale Collection of Diverse Physics Simulations for Machine Learning," *arXiv preprint arXiv:2412.00568*, 2024. [Online]. Available: https://arxiv.org/abs/2412.00568

[7] E. F. Toro, *Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction*, 3rd ed. Springer, 2009.

[8] J. Mandal and V. Sharma, "A genuinely multidimensional convective pressure flux split riemann solver for euler equations," *Journal of Computational Physics*, vol. 297, pp. 669–688, 2015.

APPENDIX

A. Additional Visualizations

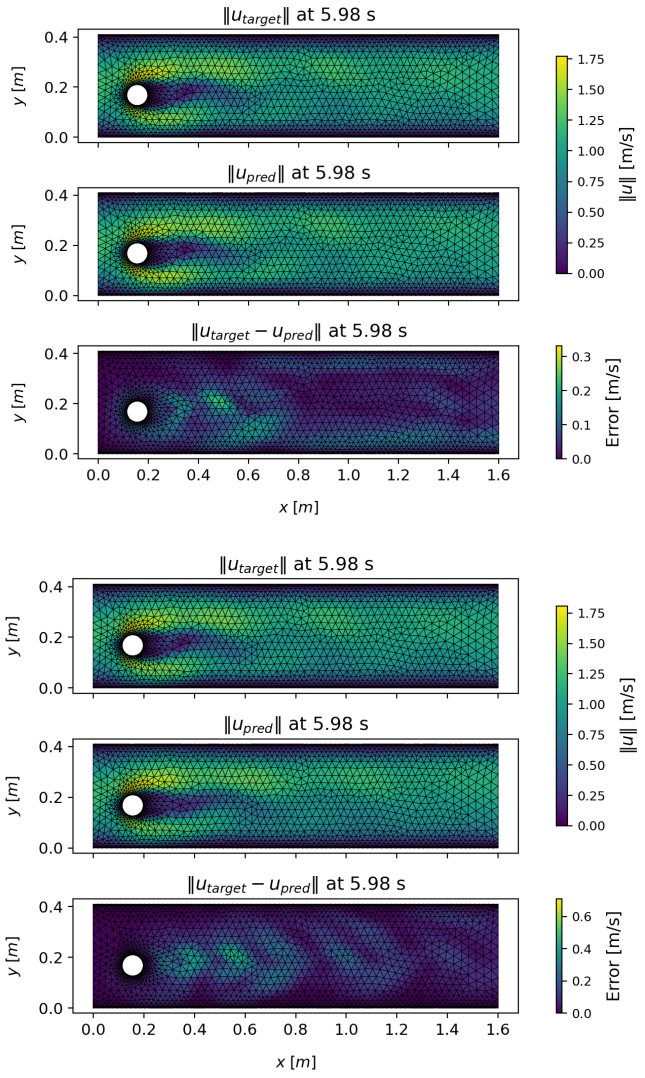

Figure 6: Comparison of velocity magnitude at the last time-step of another test simulation for MeshGraphNets on the incompressible cylinder flow case. The bottom left panel shows the error map when trained on the full dataset, while the right bottom panel shows the result when trained on only 50% of the trajectory data.

B. Hyperparameters and Architecture Details

The Flux Conservative GNN was implemented in PyTorch using the PyTorch Geometric library. The architecture consists of $L = 12$ message-passing layers with a latent dimension of $d = 64$. All Multi-Layer Perceptrons (MLPs) within the network utilize GELU activations. To ensure robustness, the model was trained using a 5-step unrolled autoregressive loss. A lower number of unrolled steps was found insufficient for long-term stability, while an higher

number was computationally too expensive. Input states were perturbed with Gaussian noise ($\sigma = 0.015$) during training to simulate accumulation errors and improve long-term stability. The momentum conservation equation was weighted by a factor of 5.0 in the loss function to prioritize accurate flow directionality. Optimization was performed using AdamW with gradient clipping.

Parameter	**Value**
Architecture	
Message Passing Layers (L)	12
Node Embedding Dim (d_{node})	64
Edge Embedding Dim (d_{edge})	32
Message MLP Hidden Dim	[128]
Flux Heads MLP Hidden Dim	[64]
Activation Function	GELU
Optimization & Training	
Optimizer	AdamW
Learning Rate	1×10^{-4}
Weight Decay	1×10^{-5}
Batch Size	1
Training Rollout Steps (K)	5
Input Noise Std (σ)	0.015
Gradient Clipping	1.0
CFL Factor	0.6
Loss Weights	
λ_{density}, λ_{energy}, $\lambda_{\text{pressure}}$	1.0
$\lambda_{\text{momentum}}$	5.0

Table III: Hyperparameters and architecture configuration for the Flux Conservative GNN model.

Note that increasing the number of message-passing layers L corresponds to a lower default integration time-step $\Delta t = \frac{\Delta t_{\text{GT}}}{L}$ as discussed in Section II-C. $L = 12$ was found to be an optimal trade-off between computational costs and flux integration accuracy.

Similar details for the MeshGraphNets case can be found in [1].

C. Empirical Design

The final configuration of the Flux Conservative GNN is the result of systematic empirical testing. Two components proved essential for long-term stability:

- **Unrolled Training ($K = 5$):** Training solely on one-step predictions ($K = 1$) yielded low one-step errors but led to rapid rollout error accumulation. Exposing the model to its own predictions during training was necessary to minimize drift. Experiments with $K > 5$ showed marginal gains in stability but significantly higher computational overhead and memory requirements, leading to the selection of $K = 5$ as the optimal trade-off.
- **Artificial Viscosity:** In the absence of the learned viscosity coefficient α_{ij}, the model exhibited high-frequency checkerboard instabilities. While one-step predictions often appeared smooth, these artifacts accumulated during autoregressive rollouts, eventually causing numerical divergence. The viscosity mechanism acts as a necessary dissipative feature for inviscid regimes.

D. Learning Dynamics

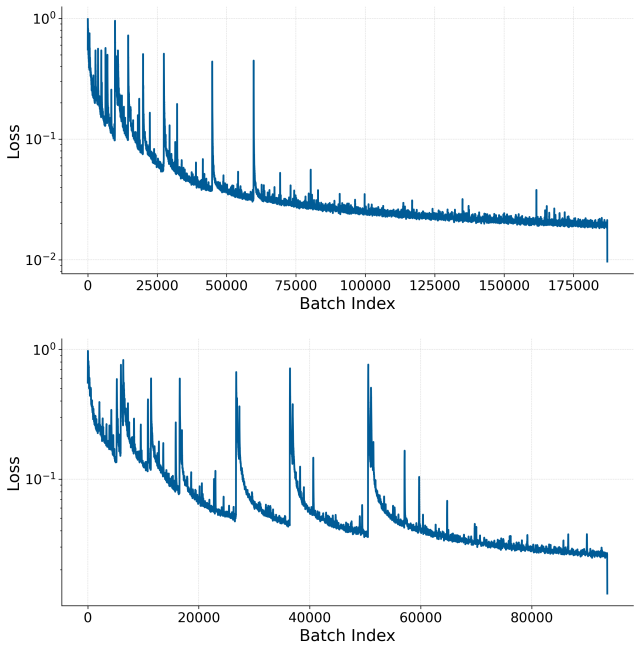

Figure 7: MeshGraphNets on the incompressible cylinder flow case, training learning curves. The curves are displayed in log scale and a smoothing factor has been applied to improve readability. The top figure corresponds to the full data training, while the bottom one refers to the half training data case. Noiser training dynamics are present in the second case, highlighting the data-hungry nature of standard data-driven GNN models.

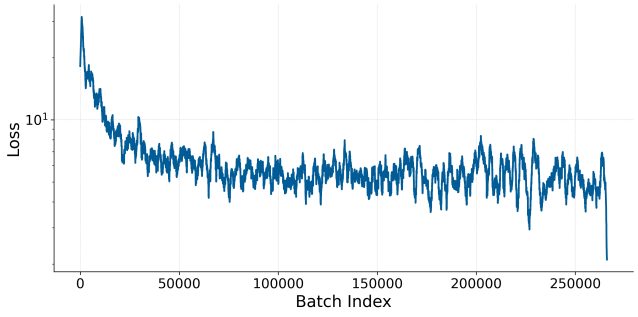

Figure 8: Flux Conservative GNN on the compressible Euler gases flow case, training learning curve (Dry Air, $20^{\circ}C$). The curve is displayed in log scale and a smoothing factor has been applied to improve readability.