Interpretable Compression of Fluid Flows Using Graph Neural Networks

Shivam Barwey[1] and Romit Maulik[2]

[1]Argonne Leadership Computing Facility, Argonne National Laboratory, USA

[2]Mathematics and Computer Science Division, Argonne National Laboratory, USA

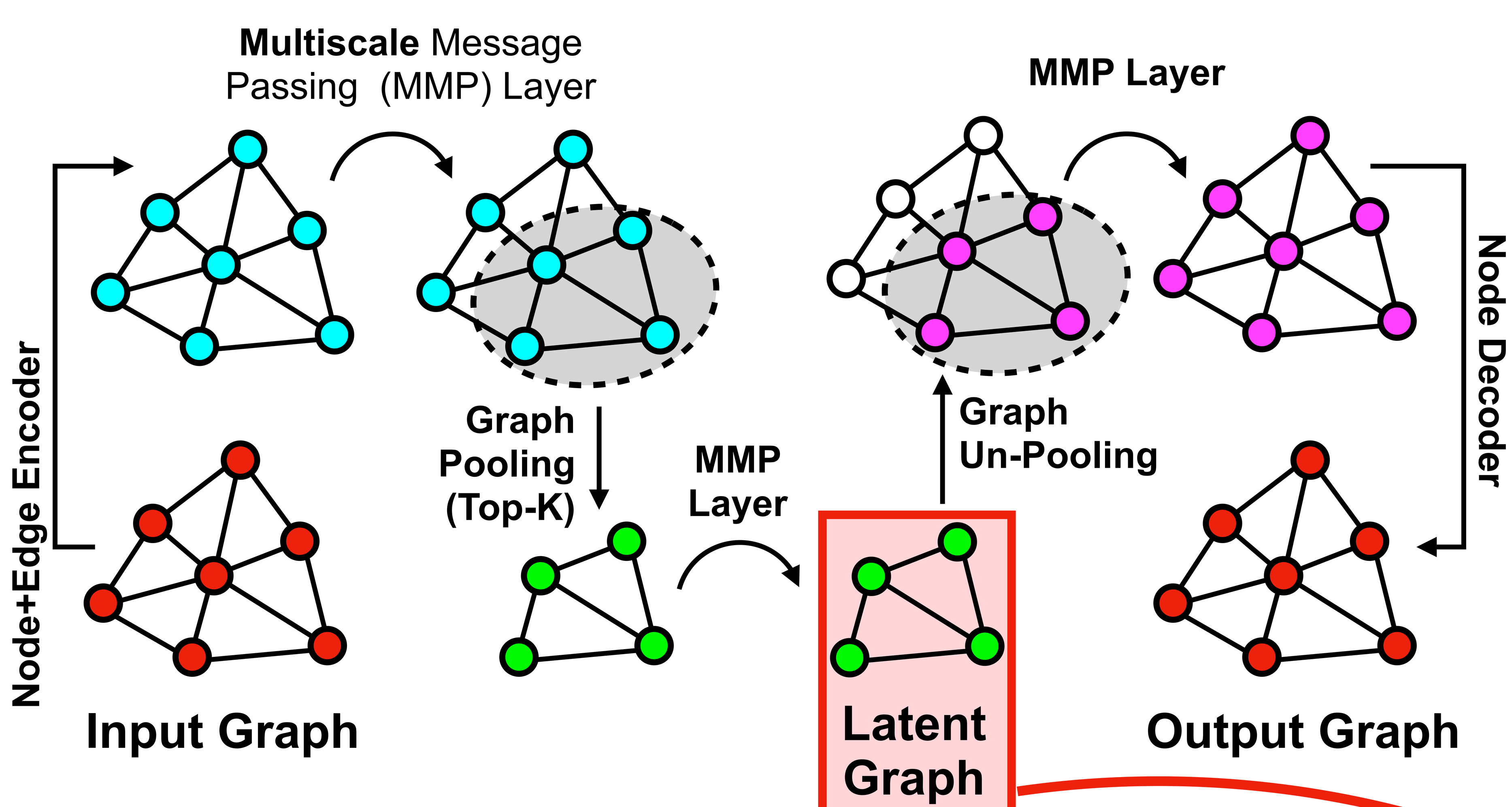
Motivation and Contributions

Neural network (NN) based reduced-order models (ROMs) via autoencoding have been shown to drastically accelerate traditional computational fluid dynamics (CFD) simulations for rapid design optimization and prediction of fluid flows. However, to extend these models to practical engineering problems, two key limitations must be addressed: latent space interpretability and compatibility with unstructured meshes. This is accomplished here with the development of a novel graph neural network (GNN) autoencoding architecture with demonstrations on complex fluid flow applications. **The specific contributions are as follows:**

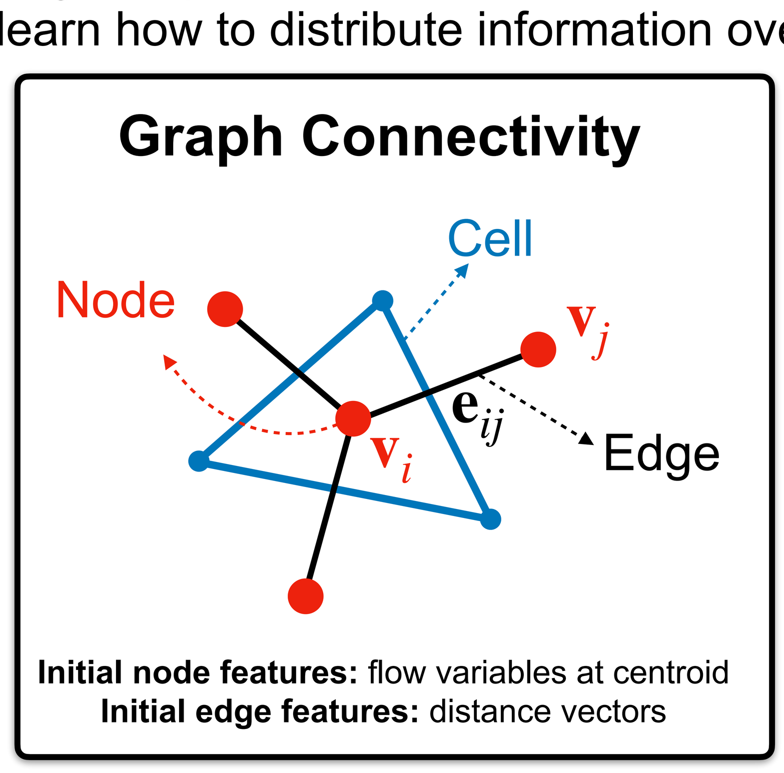
- To address the goal of interpretability, the GNN autoencoder achieves compression through an adaptive graph reduction procedure. The reduction step amounts to flowfield-conditioned node sampling, and produces latent graphs that (a) are visualizable in physical space, and (b) have connectivities that evolve in time with unsteady flow features.
- To address the goal of unstructured mesh compatibility, the autoencoding architecture utilizes a series of multi-scale message passing (MMP) layers, each of which models information exchange among node neighborhoods at various lengthscales.

Methodology

GNN architecture: The architecture consists of an encoder and decoder, utilizing a combination of multiscale message passing layers, graph pooling layers, and graph unpooling layers. Graph pooling operation produces a **interpretable latent graph.**



Graph generation and message passing: An input graph is extracted from a CFD mesh using a finite volume interpretation: nodes represent values at cell centers, and edges represent connections that intersect faces (flux paths). Message passing layers learn how to distribute information over nodes using the edge features.



Message Passing Block
(Battaglia et al., arXiv 2018)

1) Edge feature update

$\mathbf{e}_{ij} = f^e(\mathbf{e}_{ij}, \mathbf{v}_i, \mathbf{v}_j) \longrightarrow$ Edge-wise MLP

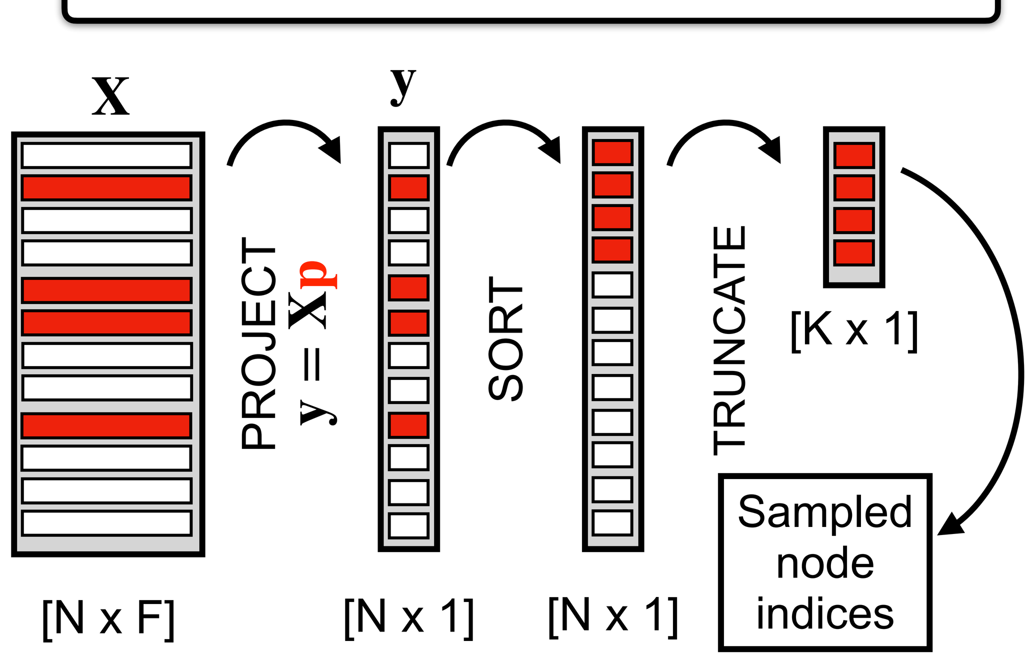
2) Edge aggregation

$\bar{\mathbf{e}}_i = \sum_{j \in N(i)} \mathbf{e}_{ij} \longrightarrow$ Neighborhood summation

3) Node feature update

$\mathbf{v}_i = f^v(\bar{\mathbf{e}}_i, \mathbf{v}_i) \longrightarrow$ Node-wise MLP

Top-K Pooling: A subset of K nodes is sampled from the input graph consisting of N nodes, such that K < N. In a first step, **a learned projection vector** is applied to the N nodes in feature space. In a second step, the nodes are ranked, and only the highest K values are retained. Introduced by Gao and Li (ICML 2019).

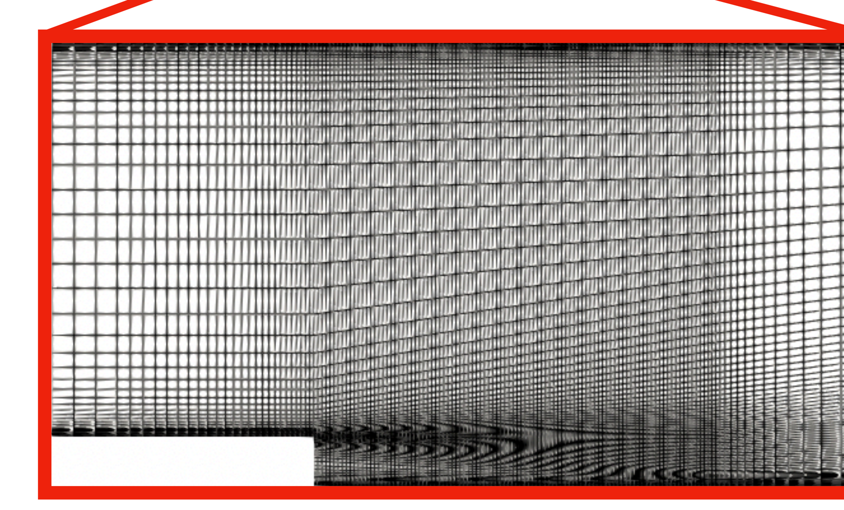


Sampled nodes adapt to input graph (latent node positions can change)

Reference: Barwey, S., Shankar, V. Viswanathan, V., and Maulik, R., 2023. Multiscale Graph Neural Network Autoencoders for Interpretable Scientific Machine Learning. *arXiv preprint arXiv:2302.06186.*

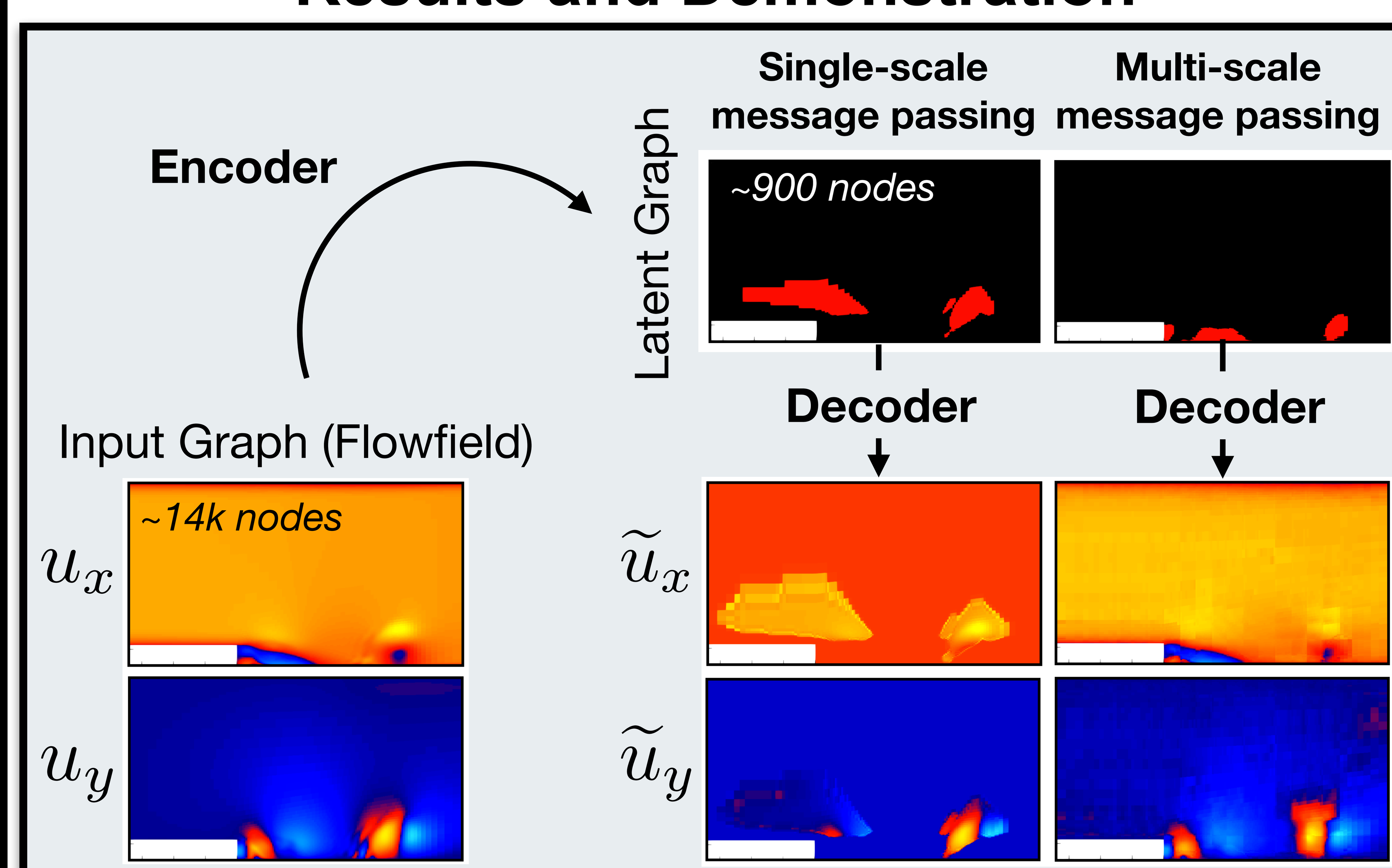
Configuration and Dataset

The GNN is demonstrated on a dataset derived from 2D simulations of unsteady, incompressible flow over a backward-facing step using OpenFOAM at Reynolds numbers in the range of 20,000 to 45,000. Input graph node features correspond to time-evolving streamwise and vertical velocity components.

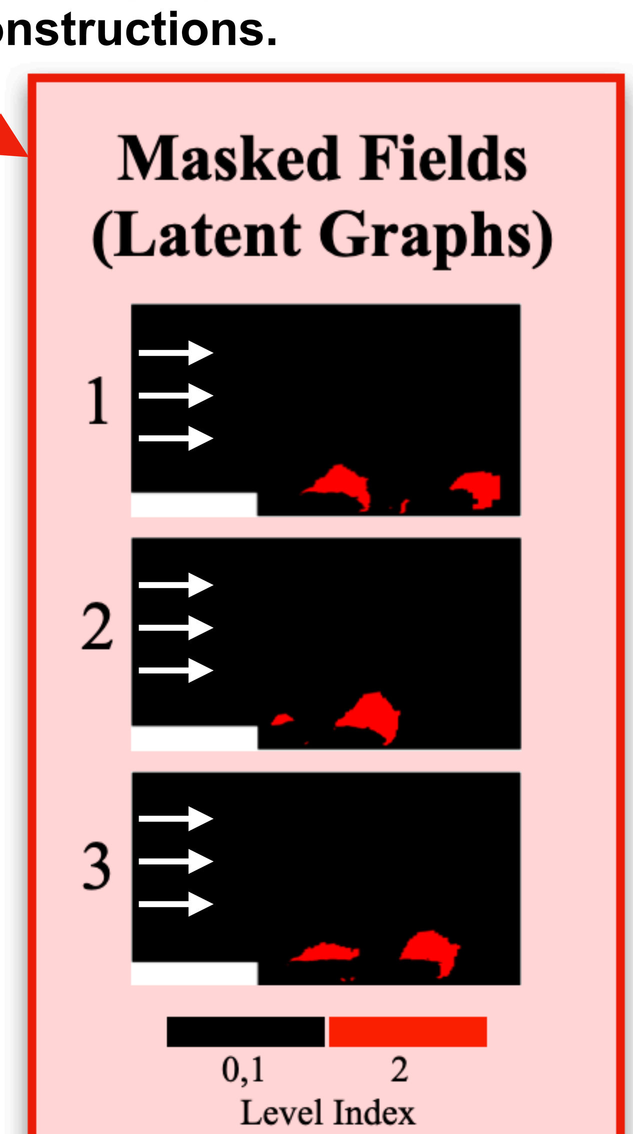


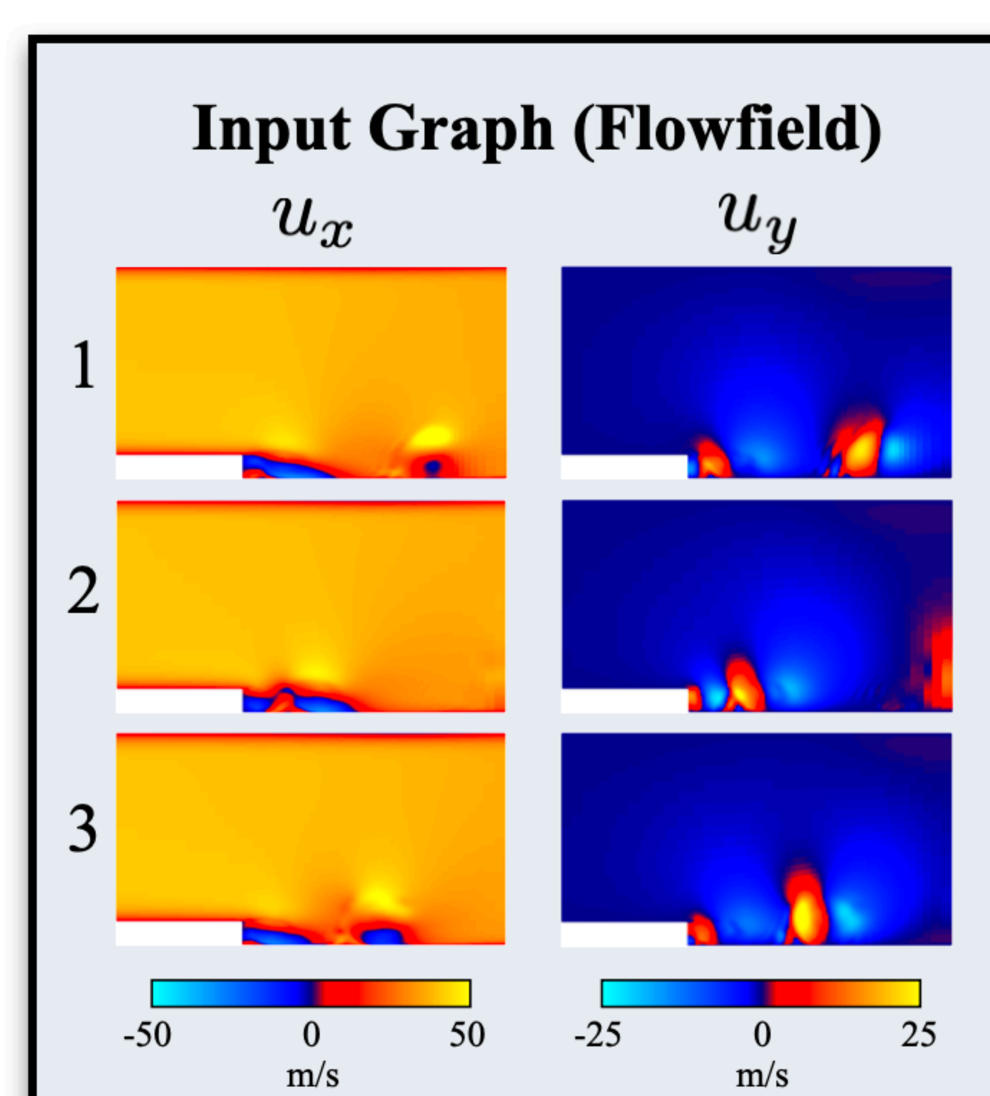
Cropped Domain

Results and Demonstration



The encoder creates a latent graph with reduced degrees-of-freedom using a node sub-sampling process. **Latent graph connectivities evolve in time and generate masked fields that can be visualized in physical space.** In the decoding process, information in the masked region (latent graph) is used to reconstruct the flow on the original graph. **Multiscale GNN operations allow for more accurate reconstructions.**





Next Steps

This work demonstrates a novel pathway for interpretable model development using graph-based autoencoders and adaptive pooling strategies. Next steps include (1) using the latent representations produced by the autoencoder presented here to develop a prognostic ROM, and (2) scaling up GNN evaluations using graph partitioning strategies.

Acknowledgements

The authors acknowledge support from resources of the Argonne Leadership Computing Facility, which is a U.S. Department of Energy Office of Science User Facility operated under contract DE-AC02-06CH11357, and funding support from DOE-ASCR FOA-2493: "Data-intensive scientific machine learning."

U.S. DEPARTMENT OF ENERGY