Fluid Minds: Advancing GNNs for Adaptive and Accurate Simulation of Fluid Dynamics (GitHub)

Teo Dimov
Department of Mathematics
Yale University
teo.dimov@yale.edu

Elder G. Veliz
Department of Statistics & Data Science
Yale University
elder.veliz@yale.edu

1 Introduction

Graph Neural Networks (GNNs) have emerged as a promising approach for simulating complex physical systems, particularly in fluid dynamics where traditional computational methods can be prohibitively expensive. Recent work by Sanchez-Gonzalez et al. [2020] demonstrated that GNN-based simulators can effectively model particle-based physical systems through learned message-passing mechanisms. Yet, questions remain regarding optimal architectural choices and training strategies for diverse fluid types.

Our work explores two complementary approaches to improve GNN-based fluid simulation. First, we investigate network structure, introducing **adaptive sampling** and **attention** mechanisms to better capture complex particle interactions. Second, we investigate the impact of **single-step** vs. **multi-step** loss functions on a wide range of fluid behaviors. [1] Specifically, we perform a systematic comparison of Graph Attention Networks (GAT) versus Interaction Networks (IN) across three fluid datasets—Water, Sand, and Goop—using both single-step and multi-step losses.

Experimental evaluation with lean training schema are each run three times, with recorded mean and standard deviation. We compare these metrics across five distinct model setups, between our interaction network (base), GAT model, Enhanced GAT model, interaction network with adaptive sampling, and GAT with adaptive sampling. In these lean training environments, we observe that more complex model architectures don't necessarily yield superior results, with our base interaction net performing the best in MSE measures. We address the limitations of this experiment and future directions to make it more rigorous.

By

2 Related Works

Sanchez-Gonzalez et al. [2020] introduced a novel approach to modeling complex physical systems through Graph Network-Based Simulators (GNS). Their IN architecture consists of:

- An **encoder** that transforms state representations into a latent graph,
- A **processor** that updates intermediate latent graphs sequentially, and
- A **decoder** that extracts dynamic information (e.g., accelerations) from the final graph.

Their approach models particles as nodes connected by edges within a defined "connectivity radius," enabling learned message passing interactions within local neighborhoods. The model employs noise-corrupted input velocities during training to mitigate error accumulation in long rollouts. Despite using only one-step predictions for training, their model maintains physically plausible trajectories over thousands of time steps—though compounding errors degrade accuracy over time.

[1] See Elder Veliz's report, which focuses on the implementation of single vs multi step loss.

Subsequent works addressed this limitation with various strategies. Klimesch et al. [2022] advocated for an alternative **multi-step loss** training scheme to reduce error drift in long-term predictions, finding that simply injecting noise may not fully address compounding errors. Their work also highlighted a crucial limitation: GNNs often rely on problem-specific correlations rather than learning true fluid dynamics, suggesting the need for stronger physical inductive biases or multi-step objectives.

Other researchers have incorporated sophisticated sampling and attention strategies to handle complex fluid scenarios. Liu et al. [2022] presented an **adaptive sampling** mechanism to identify relevant neighbor nodes and an **attention** module to focus on high-gradient regions, reporting speed-ups over conventional computation fluid dynamics solvers.

While attention-based GNN variants such as GAT (Veličković et al., 2018) show promise in many domains, it remains contested whether or how they most benefit particle-based fluid simulation, which requires stable, physically consistent rollouts over many timesteps.

Overall, existing literature underscores two main open challenges: *(i)* robust long-term prediction despite compounding errors, and *(ii)* capturing complex flow phenomena with minimal prior knowledge. This study builds upon prior findings by unifying **attention-based architectures** with **multi-step loss** training in fluid simulation and presenting empirical insights into the interplay of these factors.

3 Methodology

This work's core methodological advance is the integration of a **adaptive sampling and advanced attention** into both GAT and IN architectures, aimed at improving long-term rollouts for water, sand, and goop simulations.

Enhanced Graph Attention Module The enhanced Graph Attention Network (GAT) incorporates several key improvements to strengthen performance and stability, including residual connections for better gradient flow and layer normalization to stabilize training. The model also employs attention dropout and feature scaling to prevent overfitting and maintain numerical stability, while multi-head output aggregation enables the network to capture diverse aspects of the graph structure. Additionally, the implementation of adaptive sampling dynamically adjusts the network's receptive field based on gradient information, allowing for more precise node neighborhood selection in regions with significant feature variations. Mathematically, the multi-head attention mechanism with the enhanced concatenation and self-loops can be represented as:

$$\vec{h}_i' = \Big\Vert_{k=1}^{K} \sigma\left(\sum_{j \in \mathcal{N}_i \cup \{i\}} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j\right) \tag{1}$$

$$\alpha_{ij}^k = \frac{\exp(\text{LeakyReLU}(\vec{a}^T[\mathbf{W}^k\vec{h}_i \| \mathbf{W}^k\vec{h}_j]))}{\sum_{l \in \mathcal{N}_i \cup \{i\}} \exp(\text{LeakyReLU}(\vec{a}^T[\mathbf{W}^k\vec{h}_i \| \mathbf{W}^k\vec{h}_l]))} \tag{2}$$

Adaptive Sampling The adaptive sampling enhancement to the Graph Attention Network dynamically adjusts the neighborhood radius for each node based on local gradient information, enabling more precise node selection in regions with high feature variations. By computing gradient magnitudes and using them to scale the sampling radius - where areas with larger gradients get smaller radii for finer-grained sampling - the network can better capture complex local structures and transitions in the flow field. This adaptive approach ensures that computationally expensive fine-grained sampling is focused on regions where it matters most, while maintaining broader sampling in more stable regions. This mechanism can be formally expressed through two key equations that govern the adaptive radius calculation and neighborhood selection:

$$r_i = \frac{r_{base}}{\gamma|\nabla f_i| + \epsilon} \tag{3}$$

$$\mathcal{N}_i = j \mid |(x_j - x_i)^2 + (y_j - y_i)^2| \leq r_i^2 \tag{4}$$

3.1 Implementation

We investigate two GNN variants:

Interaction Network (IN): A message-passing model with explicit edge and node update functions, as in Sanchez-Gonzalez et al. [2020].

Graph Attention Network (GAT): Employs learnable attention weights to highlight the most important neighbors. Our enhanced attention model expands on this framework. For both, we adopt 10 message-passing layers, a latent dimensionality of 128, and an MLP-based encoder-decoder.

Note. While our experiments included an adaptive sampling radius and GAT-based attention for neighbor interactions, attention did not consistently yield performance benefits in these fluid tasks. Instead, certain GAT runs produced "particle clumping" or less stable rollouts, suggesting the direct benefits of attention for fluid simulation remain limited or require further hyperparameter tuning.

4 Experiments

4.1 Data

We evaluate on three fluid datasets from Sanchez-Gonzalez et al. [2020]: WaterDrop, Sand, and Goop, each containing $1{,}060$ trajectories of up to $2{,}000$ particles across $1{,}000$ timesteps. Particles act as graph nodes, with dynamic edges among particles lying within a fixed 0.015 "connectivity radius."

Each dataset is stored in `TFRecord` format with a `metadata.json` file specifying sequence lengths, dimensionalities, simulation bounds, and normalization statistics. This metadata enables consistent graph construction and standardization of physical quantities (e.g., velocity, acceleration).

4.2 Training

Our training setup involved translating the TensorFlow implementation of Sanchez-Gonzalez et al. (2020) into PyTorch, adapting the approach of Li et al. (2024). Raw `TFRecord` files were converted to `pickle` format, and velocity/acceleration normalization was applied to standardize inputs. Particle connectivity was defined via a fixed-radius neighbor search.

Veliz's models were trained for $500{,}000$ gradient steps (~ 10 hours on an `NVIDIA A100`) with a batch size of 2. The learning rate, initialized at 10^{-4}, used a linear warm-up over the first $10{,}000$ steps, then decayed exponentially to 10^{-6}. Gradient clipping with a max norm of 1.0 helped stabilize training. Dimov's experimentation with Adaptive Sampling and the Enhanced Attention module, utilized a leaner training of $100{,}000$ gradient steps.

The forward pass predicted the next position using 5 prior positions for 1-step loss or recursively generated predictions for $n+1$ timesteps in multi-step loss. Loss functions compared predicted accelerations with ground-truth for 1-step loss and cumulative prediction errors for multi-step loss. The `Adam` optimizer was used for updates. Artificial noise simulated rollout errors during 1-step loss training. Validation loss, computed every $3{,}000$ steps, monitored performance. Each model was trained on MSE loss for acceleration and evaluated on particle position accuracy. Early stopping triggered if the validation loss showed no improvement over 10 evaluations.

4.3 Results

Table 1 depicts test metrics between Dimov's model architectures, progressing from the foundational Interaction Network (Base) through Graph Attention Networks (GAT), to our Enhanced GAT variant and novel Adaptive Sampling implementations. We observe slightly stronger performance across our interaction base models, with no definitive improvement with Adaptive Sampling when accounting for standard error. Our graph attention module performs the worst across these leaner training cycles.

Table 1: Averaged Metrics for Different Model Architectures on Held-Out Water Test Set

Model	MSE-acc 1	MSE-pos 1
Interaction Network (Base)	$\mathbf{(3.26 \pm 0.65) \times 10^{-2}}$	$\mathbf{(2.90 \pm 2.11) \times 10^{-9}}$
Graph Attention	$(1.52 \pm 0.88) \times 10^{-1}$	$(4.81 \pm 2.97) \times 10^{-9}$
Enhanced Graph Attention	$(7.08 \pm 2.19) \times 10^{-2}$	$(5.33 \pm 1.98) \times 10^{-9}$
Interaction Base w/ Adaptive Sampling	$(3.86 \pm 1.22) \times 10^{-2}$	$(3.68 \pm 1.93) \times 10^{-9}$
Graph Attention w/ Adaptive Sampling	$(5.54 \pm 1.17) \times 10^{-2}$	$(5.28 \pm 3.66) \times 10^{-9}$

Notes: Mean $\pm$ SD. Best metrics **bolded**. Results of training on 100,000 steps.

The Interaction Network (Base) model shows the strongest performance on on both MSE metrics. This suggests that the simpler base architecture is actually more effective at capturing the essential dynamics of the water system compared to more complex variants. The Graph Attention models, both standard and enhanced versions, show notably weaker performance on MSE-acc 1, though they maintain similar orders of magnitude for MSE-pos 1 metrics. It is also possible these results would diverge further down the line with more extensive training, which was difficult given the enormous magnitude of our dataset.

The relatively poor performance of the Graph Attention modules could be attributed to several factors. First, the attention mechanism may be introducing unnecessary complexity for what appears to be a relatively straightforward physical system. Additionally, attention models typically only display significant performance improvements with enormous training scales. With only $100,000$ training steps, the attention-based models might not have had sufficient time to fully optimize their more numerous parameters. The addition of Adaptive Sampling doesn't seem to provide consistent improvements across architectures, suggesting that the base interaction network already captures the relevant spatial relationships effectively without requiring sophisticated sampling strategies. Furthermore, use of the Patience parameter, which provides early stopping when no improvement is yielded over several thousand steps, likely brings inconsistency into these results. Nevertheless, it is a technique used widely across deep learning, so we included it in our implementation.

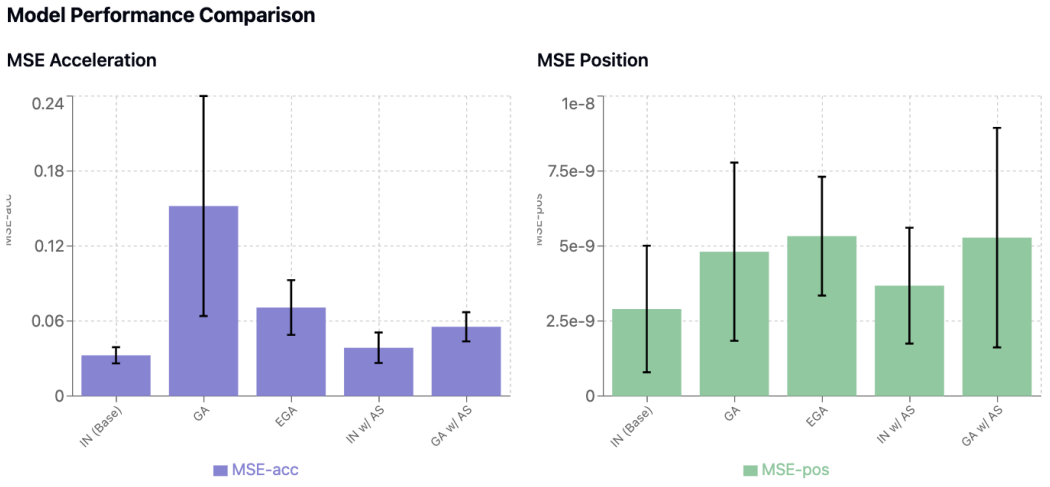

Figure 1: Model Performance Comparison

Further, a paired t-test demonstrates low probabilities that the models' true performance is the same, especially when comparing the Interaction Network with some of the base models. With only n=3 tests however, these results are inconclusive.

Table 2: Paired t-test Results Comparing Interaction Network (Base) with Other Models

Comparison	t-stat (MSE-acc)	p-value (MSE-acc)	t-stat (MSE-pos)	p-value (MSE-pos)
Interaction Network (Base) vs Graph Attention	-13.53	3.05×10^{-24}	-5.24	8.98×10^{-7}
Interaction Network (Base) vs Enhanced Graph Attention	-16.72	1.35×10^{-30}	-8.40	3.34×10^{-13}
Interaction Network (Base) vs Interaction Base w/ Adaptive Sampling	-4.34	3.44×10^{-5}	-2.73	7.55×10^{-3}
Interaction Network (Base) vs Graph Attention w/ Adaptive Sampling	-17.03	3.44×10^{-31}	-5.63	1.66×10^{-7}

Another observation in our low data training regime — as exemplified training on the WaterDropSample dataset, which only contains several trajectories — is a clumping phenomenon. This likely occurs because early on in training, particles are putting more too much weight on their nearest neighbors on not enough on the global system. As we trained models in higher data domains, this effect largely went away.

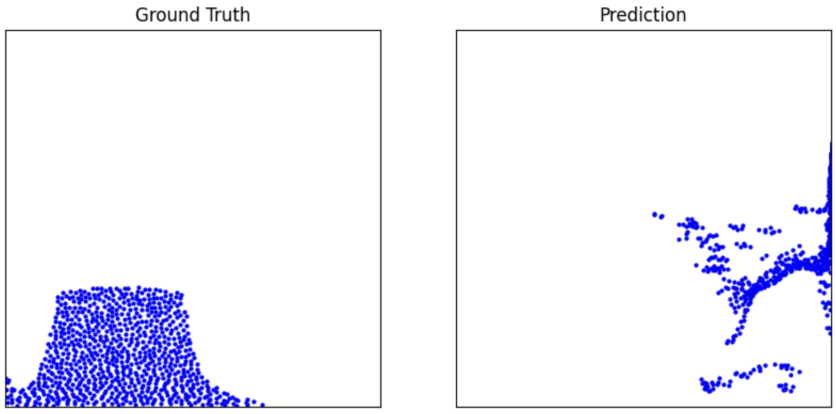

Figure 2: Clumping phenomenon in low data regimes

5 Conclusion

Our work explored methods to improve fluid dynamics simulations using Graph Neural Networks, focusing on adaptive sampling and attention mechanisms. We found that multi-step loss training improves long-term stability in simulations, while attention-based models like GAT did not consistently outperform simpler Interaction Networks. Adaptive sampling provided mixed results, suggesting that existing models already capture key spatial relationships effectively.

Future research could explore the impact of these new architectures into a higher compute training regime, which would likely outline more meaningful differences between the models. Ablation tests could also be run across different thresholds of the adaptive sampling technique to obtain a more robust understanding of how the change affects the simulation result. Finally, given the current difficulties of training such models, future work could entail optimizing the training processes and compute costs of such work, which could unlock more lightweight, powerful physics simulations.

References

Explaining graph neural networks. https://pytorch-geometric.readthedocs.io/en/latest/tutorial/explain.html, 2024. Accessed: 2024-10-04.

J. Kakkad, J. Jannu, K. Sharma, C. Aggarwal, and S. Medya. A survey on explainability of graph neural networks, 2023. URL https://arxiv.org/abs/2306.01958.

J. Klimesch, P. Holl, and N. Thuerey. Simulating liquids with graph networks, 2022. URL https://arxiv.org/abs/2203.07895.

S. Li. Simulating complex physics with graph networks: Step by step. https://medium.com/stanford-cs224w/simulating-complex-physics-with-graph-networks-step-by-step-177354cb9b05, 2022.

Q. Liu, W. Zhu, X. Jia, F. Ma, and Y. Gao. Fluid simulation system based on graph neural network, 2022. URL https://arxiv.org/abs/2202.12619.

A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. W. Battaglia. Learning to simulate complex physics with graph networks, 2020. URL https://arxiv.org/abs/2002.09405.

V. J. Shankar, S. Barwey, R. Maulik, and V. Viswanathan. Practical implications of equivariant and invariant graph neural networks for fluid flow modeling. In *ICLR 2023 Workshop on Physics for Machine Learning*, 2023. URL https://openreview.net/forum?id=3Y6XRCIUT5.

J. You, R. Ying, and J. Leskovec. Design space for graph neural networks, 2021. URL https://arxiv.org/abs/2011.08843.

Appendix

Network Architectures

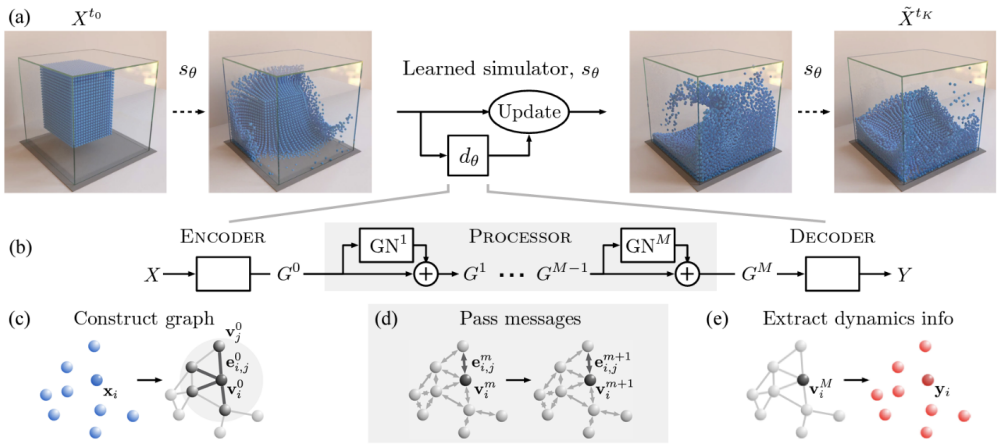

Figure 3: Model architecture used in Sanchez-Gonzalez et al. [2020]

As we used the same base architecture described in Sanchez-Gonzalez et al. [2020], we only briefly summarize it here:

- **Encoder:** An MLP that maps raw node features (particle type, velocity, boundary distances) and raw edge features (relative positions, distance) into a latent space of dimension 128.
- **Processor:** A stack of 10 message-passing blocks. For IN, each block has separate MLPs for edge updates and node updates. For GAT, each block is a GAT layer with 8 heads concatenated, followed by a linear projection back to 128 channels.
- **Decoder:** An MLP that maps the final node embeddings back to acceleration predictions.

Below we briefly summarize the GAT architecture:

- **Encoder:** Identical to the IN case. An MLP maps raw node features (particle-type embeddings, velocities, boundary distances) and raw edge features (relative displacements, distances) into a 128-dimensional latent space.
- **Processor:** Each message-passing block is replaced by a GAT layer with 8 attention heads, whose outputs are concatenated and then linearly projected back to 128 channels. We apply dropout after each attention operation to stabilize training. GAT weights neighbor information via learned attention coefficients rather than fixed MLP-based message aggregation, potentially allowing more expressive modeling of local interactions.
- **Decoder:** As with IN, a final MLP converts the 128-dimensional node embeddings into per-particle acceleration predictions.