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

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

Teo Dimov
Department of Mathematics
Yale University
teo.dimov@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[1]. Second, we investigate the impact of **single-step** vs. **multi-step** loss functions on a wide range of fluid behaviors. 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 **twelve distinct models** (2 architectures $\times$ 2 loss types $\times$ 3 fluid datasets), each run **3 times with different seeds**, capturing mean $\pm$ standard deviation performance indicate that while attention (as implemented in our GAT) does not consistently outperform Interaction Networks for these fluid systems—often even causing unintended clumping—multi-step loss (here set to 2-step due to computational constraints) generally achieves superior long-range stability in position predictions. Even when single-step loss yields slightly lower one-step MSE, the multi-step models produce more accurate trajectories over an entire rollout.

By systematically comparing short-term vs. long-term accuracy, we demonstrate that multi-step training can be critical for realistic simulation of fluids where error accumulation is a key challenge.

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 Teo Dimov's report, which focuses on the implementation of adaptive sampling and attention.

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 require 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 **multi-step loss** strategy into both GAT and IN architectures, aimed at improving long-term rollouts for water, sand, and goop simulations.

Single-Step Loss. Following Sanchez-Gonzalez et al. [2020], the single-step objective minimizes the MSE between predicted and ground-truth accelerations for just the next timestep:

$$\mathcal{L}_{\text{single}} = \|\ddot{\mathbf{p}}^{t}_{\text{GNS}} - \ddot{\mathbf{p}}^{t}_{\text{GT}}\|^2 \tag{1}$$

where $\ddot{\mathbf{p}}^{t}_{\text{GNS}}$ and $\ddot{\mathbf{p}}^{t}_{\text{GT}}$ denote the predicted and ground-truth accelerations at time t, respectively. While effective for short-term predictions, this approach inherently lacks robustness for long-term simulations. The iterative nature of rollouts requires the model to use its outputs as inputs, exposing it to its own inaccuracies. Without explicit training to handle such recursive errors, the model is ill-prepared to mitigate them, leading to divergence from physically plausible trajectories over time.

To partially address this issue, Sanchez-Gonzalez et al. [2020] introduced artificial noise during training to simulate the imperfections encountered during rollouts. However, this strategy only indirectly addresses error accumulation and relies on approximating the model's error distribution, which may not fully capture the long-term dynamics.

Multi-Step Loss. Instead of focusing only on one-step accuracy, a multi-step loss function explicitly incorporates errors from successive timesteps, training the model to handle its own predictions as inputs. It is defined as:

$$\mathcal{L}_{\text{multi}} = \frac{1}{n}\left(\|\ddot{\mathbf{p}}_{\text{GNS}}(\mathbf{p}^{t-1}_{\text{GT}}) - \ddot{\mathbf{p}}^{t}_{\text{GT}}\|^2 + \sum_{i=1}^{n}\|\ddot{\mathbf{p}}^{t+i}_{\text{GNS}}(\mathbf{p}^{t+i-1}_{\text{GNS}}) - \ddot{\mathbf{p}}^{t+i}_{\text{GT}}\|^2\right) \tag{2}$$

where n is the number of *additional* timesteps included in the loss. This formulation introduced by Klimesch et al. [2022] offers several advantages: By using the model's own predictions as inputs during training, it directly addresses recursive errors, teaching the GNN to mitigate inaccuracies over time. It explicitly trains the model to maintain consistency across extended time horizons, thus reducing the impact of error propagation. Importantly, this method aligns the model with real-world dynamics by simulating scenarios where its outputs determine future states, thereby embedding physical principles into the learning process and reducing reliance on artificial constraints or noise.

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 Veličković et al. [2018].

For both, we adopt 10 message-passing layers, a latent dimensionality of 128, and an MLP-based encoder-decoder (details in Appendix Network Architectures).

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[2]

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 by Li [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.

Each model was 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.

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 30 evaluations.

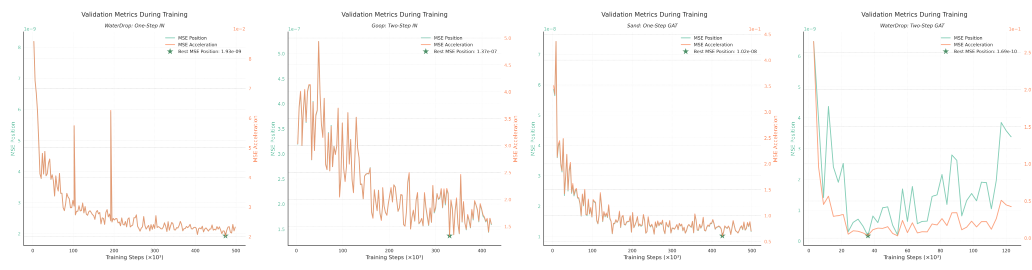

Figure 1: Validation curves for different GNN architectures and loss types over the training process, illustrating their effect on model convergence.

[2]Appendix: Data Summary details the maximum number of particles, edges, trajectories, etc. per dataset.

4.3 Results

Table 1 highlights final test metrics for the 12 model variants over 3 runs. Our primary evaluation comprises four metrics: MSE between predicted and ground-truth *accelerations* and *positions* at the next timestep (`MSE-acc 1` and `MSE-pos 1`, respectively); MSE of positions over the *entire* trajectory rollout (long-horizon accuracy) (`MSE-pos Full`); and Earth Mover's Distance (`EMD`), measuring distributional discrepancy between predicted and ground-truth particle positions across time.

Though each model is optimized to minimize acceleration MSE (following Sanchez-Gonzalez et al. [2020]), we leverage short-term *and* long-term positional metrics to assess overall simulation fidelity.

Table 1: Averaged Metrics for 3 Runs of Each Model on Held-Out Test Set

	Model	**MSE-acc 1**	**MSE-pos 1**	**MSE-pos Full**	**EMD**
WaterDrop	1-step IN	1.83 ± 2.09	$(1.75 \pm 2.00) \times 10^{-7}$	$(6.89 \pm 8.79) \times 10^{3}$	45.2 ± 45.0
	2-step IN	$\mathbf{(7.51 \pm 0.31) \times 10^{-2}}$	$\mathbf{(7.15 \pm 0.30) \times 10^{-9}}$	73.1 ± 63.4	4.22 ± 2.30
	1-step GAT	14.1 ± 7.86	$(1.34 \pm 0.75) \times 10^{-6}$	$(4.16 \pm 3.26) \times 10^{5}$	430 ± 220
	2-step GAT	$(8.94 \pm 2.06) \times 10^{-2}$	$(8.51 \pm 1.95) \times 10^{-9}$	$\mathbf{9.67 \pm 16.6}$	$\mathbf{1.16 \pm 1.63}$
Sand	1-step IN	$\mathbf{1.01 \pm 0.25}$	$\mathbf{(1.65 \pm 0.39) \times 10^{-7}}$	23.6 ± 28.7	2.60 ± 2.36
	2-step IN	1.37 ± 0.95	$(2.36 \pm 1.65) \times 10^{-7}$	$\mathbf{1.19 \pm 1.99}$	$\mathbf{0.48 \pm 0.67}$
	1-step GAT	94.6 ± 141	$(1.66 \pm 2.50) \times 10^{-5}$	$(9.41 \pm 15.9) \times 10^{3}$	37.0 ± 53.7
	2-step GAT	54.1 ± 82.2	$(9.26 \pm 14.2) \times 10^{-6}$	$(3.34 \pm 4.83) \times 10^{2}$	7.71 ± 9.29
Goop	1-step IN	$\mathbf{2.45 \pm 0.06}$	$\mathbf{(2.90 \pm 0.77) \times 10^{-8}}$	1.15 ± 1.88	0.49 ± 0.58
	2-step IN	4.55 ± 0.26	$(5.41 \pm 3.08) \times 10^{-8}$	$\mathbf{0.54 \pm 0.73}$	$\mathbf{0.27 \pm 0.11}$
	1-step GAT	$(2.12 \pm 1.22) \times 10^{2}$	$(2.54 \pm 1.45) \times 10^{-5}$	$(1.95 \pm 2.35) \times 10^{4}$	88.7 ± 63.6
	2-step GAT	9.61 ± 8.57	$(1.13 \pm 1.00) \times 10^{-6}$	49.9 ± 40.4	4.07 ± 2.62

Notes: Mean $\pm$ SD. Best metrics **bolded**. See also Appendix: Model Metrics Boxplots and All Metrics Table.

WaterDrop. Both multi-step IN and GAT significantly outperform their 1-step counterparts. Specifically, the `2-step IN` achieves the lowest `MSE-acc 1` and `MSE-pos 1`, indicating robust short-term accuracy. Concurrently, the `2-step GAT` excels in long-term metrics, attaining the lowest `MSE-pos Full` and `EMD`. Both architectures exhibit performance gains after multi-step loss training. In fact, improvements from multi-step loss are statistically significant across all metrics, whereas the differences between GNN types are not (see Appendix Statistical Validation).

Water, being a fluid and less granular material, benefits uniformly from multi-step training as it involves smooth and continuous interactions well-captured by both GNN architectures when trained for long-term consistency. The absence of significant differences between GNN types implies that both architectures are equally capable of modeling the behavior of water particles. Consequently, the primary driver of performance gains in the WaterDrop simulations is the multi-step loss function, which effectively mitigates error accumulation to improve long-term predictions.

Sand and Goop. INs demonstrate superior performance over GATs for both Sand and Goop, especially in short-term predictive accuracy as measured by `MSE-acc 1` and `MSE-pos 1` metrics. This advantage likely stems from IN's structured message-passing architecture, which captures the localized particle interactions inherent in these materials.

Multi-step loss training yielded improvements in long-term trajectory prediction, although only marginally for Sand. The `2-step IN` models achieved the lowest `MSE-pos Full` and `EMD` metrics, while `2-step GAT` models generally showed strong error reduction compared to their single-step variants. This suggests multi-step training can address error accumulation in extended rollouts—critical for granular and viscous materials where minor deviations can compound rapidly.

Statistical analysis confirms the architectural distinction between IN and GAT significantly influences model performance across all metrics, whereas the choice of loss type does not yield statistically significant differences. This aligns with theoretical predictions, as the structured message-passing in INs naturally accommodates the localized interactions characteristic of granular and viscous materials. While GAT models' attention mechanisms offer theoretical advantages, qualitative analysis revealed a problematic tendency toward particle clustering in rollouts. The GAT architecture's vulnerability to error propagation becomes particularly evident in single-step training scenarios, though this limitation is partially mitigated through multi-step training approaches.

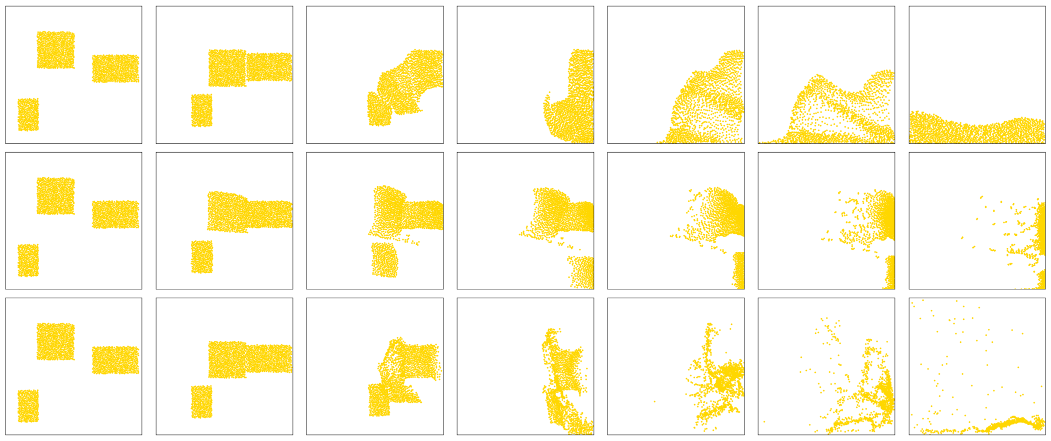
Figure 2: Example Ground Truth (top), 1-step IN (middle), and 2-step IN predictions (bottom) over 7 frames. Each column represents a specific frame in the sequence.

Taken together, these results underscore that:

1. **One-step models** often dominate on near-term (`MSE-acc 1`, `MSE-pos 1`) metrics, having been trained explicitly for immediate accuracy.
2. **Multi-step training** consistently improves extended rollouts (`MSE-pos Full`, `EMD`), mitigating error accumulation. This is especially pronounced in `GAT` models, which appear more sensitive to compounding inaccuracies under single-step training.
3. **Material properties** also matter. For instance, WaterDrop exhibits clearer benefits from multi-step training across all metrics, while Sand and Goop show more pronounced differences with respect to architectural choices.

5 Conclusion

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.

E. Li. Learn-to-simulate: Pytorch implementation of learning-to-simulate (icml2020). https://github.com/Emiyalzn/Learn-to-Simulate, 2024. Accessed: 2024-11-27.

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.

P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks, 2018. URL https://arxiv.org/abs/1710.10903.

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

Appendix

Network Architectures

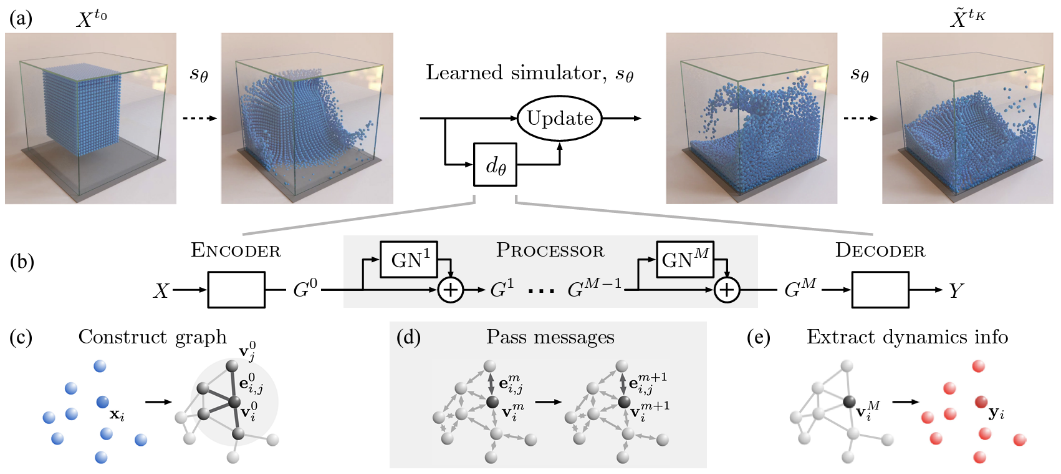

Figure 3: Model architecture used in Sanchez-Gonzalez et al. [2020]

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.

Data Summary

Table 2: Data Statistics from Sanchez-Gonzalez et al. [2020]

Material	WaterDrop	WaterDropSample	Sand	Goop
Maximum Number of Particles	1k	1k	2k	1.9k
Maximum Number of Edges	12k	12k	21k	19k
Trajectory Length	1000	1000	320	400
# Trajectories (Train/Val/Test)	1000/30/30	2/2/2	1000/30/30	1000/30/30

Model Metrics Boxplots

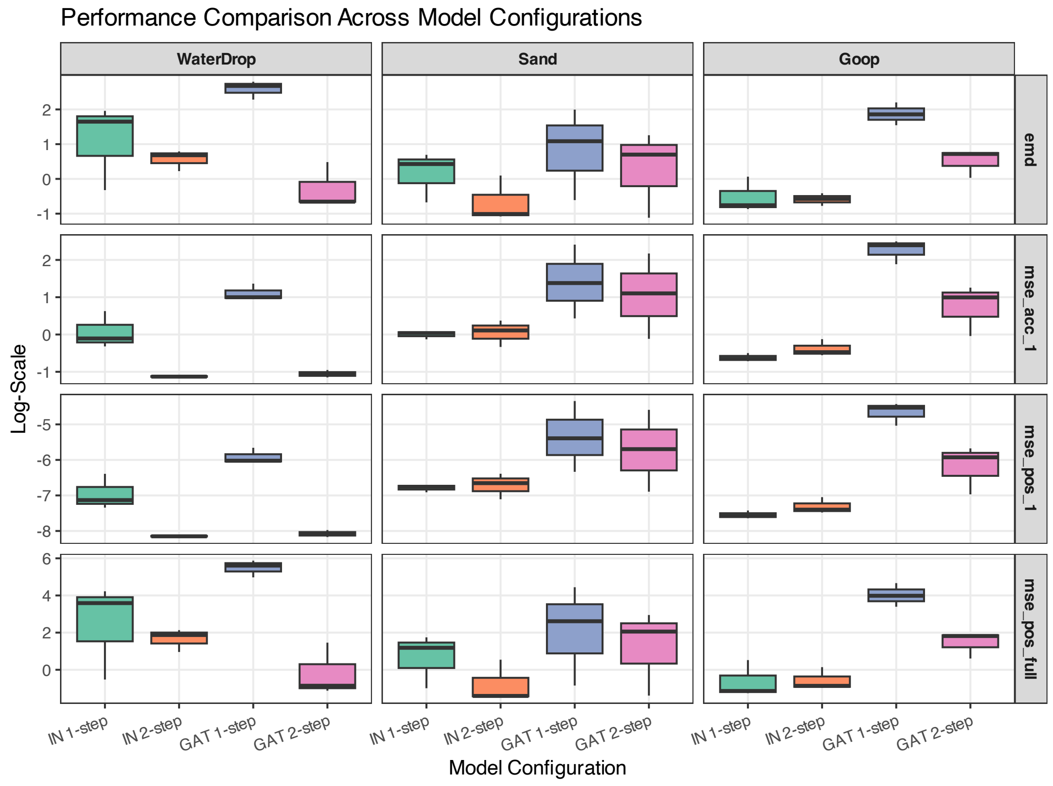

Figure 4: Performance comparison of different model configurations across datasets and metrics.

All Metrics Table

Table 3: Averaged Metrics for 3 Runs of Each Model on Held-Out Test Set

	Metric	**1-step IN**	**2-step IN**	**1-step GAT**	**2-step GAT**
WaterDrop	MSE-acc 1	1.83 ± 2.09	$\mathbf{(7.51 \pm 0.31) \times 10^{-2}}$	14.1 ± 7.86	$(8.94 \pm 2.06) \times 10^{-2}$
	MSE-pos 1	$(1.75 \pm 2.00) \times 10^{-7}$	$\mathbf{(7.15 \pm 0.30) \times 10^{-9}}$	$(1.34 \pm 0.75) \times 10^{-6}$	$(8.51 \pm 1.95) \times 10^{-9}$
	MSE-acc 20	$(7.02 \pm 8.93) \times 10^{10}$	$(7.41 \pm 6.42) \times 10^{8}$	$(4.25 \pm 3.32) \times 10^{12}$	$\mathbf{(9.79 \pm 16.8) \times 10^{7}}$
	MSE-pos 20	$(6.73 \pm 8.58) \times 10^{3}$	71.4 ± 61.9	$(4.06 \pm 3.18) \times 10^{5}$	$\mathbf{9.42 \pm 1.61}$
	MSE-acc Full	$(7.19 \pm 9.14) \times 10^{10}$	$(7.59 \pm 6.58) \times 10^{8}$	$(4.35 \pm 3.40) \times 10^{12}$	$\mathbf{(1.00 \pm 1.72) \times 10^{8}}$
	MSE-pos Full	$(6.89 \pm 8.79) \times 10^{3}$	73.1 ± 63.4	$(4.16 \pm 3.26) \times 10^{5}$	$\mathbf{9.67 \pm 16.6}$
	EMD	45.2 ± 45.0	4.22 ± 2.30	430 ± 220	$\mathbf{1.16 \pm 1.63}$
Sand	MSE-acc 1	$\mathbf{1.01 \pm 0.25}$	1.37 ± 0.95	94.6 ± 141	54.1 ± 82.2
	MSE-pos 1	$\mathbf{(1.65 \pm 0.39) \times 10^{-7}}$	$(2.36 \pm 1.65) \times 10^{-7}$	$(1.66 \pm 2.50) \times 10^{-5}$	$(9.26 \pm 14.2) \times 10^{-6}$
	MSE-acc 20	$(1.36 \pm 1.68) \times 10^{8}$	$\mathbf{(6.65 \pm 1.12) \times 10^{6}}$	$(4.96 \pm 8.39) \times 10^{8}$	$(1.81 \pm 2.61) \times 10^{7}$
	MSE-pos 20	21.9 ± 26.6	$\mathbf{1.13 \pm 1.89}$	$(8.77 \pm 1.4.9) \times 10^{3}$	$(3.06 \pm 4.44) \times 10^{2}$
	MSE-acc Full	$(1.47 \pm 1.81) \times 10^{8}$	$\mathbf{(6.99 \pm 1.17) \times 10^{7}}$	$(5.32 \pm 8.99) \times 10^{10}$	$(1.97 \pm 2.84) \times 10^{9}$
	MSE-pos Full	23.6 ± 28.7	$\mathbf{1.19 \pm 1.99}$	$(9.41 \pm 15.9) \times 10^{3}$	$(3.34 \pm 4.83) \times 10^{2}$
	EMD	2.60 ± 2.36	$\mathbf{0.48 \pm 0.67}$	37.0 ± 53.7	7.71 ± 9.29
Goop	MSE-acc 1	$\mathbf{2.45 \pm 0.06}$	4.55 ± 0.26	$(2.12 \pm 1.22) \times 10^{2}$	9.61 ± 8.57
	MSE-pos 1	$\mathbf{(2.90 \pm 0.77) \times 10^{-8}}$	$(5.41 \pm 3.08) \times 10^{-8}$	$(2.54 \pm 1.45) \times 10^{-5}$	$(1.13 \pm 1.00) \times 10^{-6}$
	MSE-acc 20	$(9.15 \pm 1.49) \times 10^{6}$	$\mathbf{(4.29 \pm 5.72) \times 10^{6}}$	$(1.54 \pm 1.85) \times 10^{11}$	$(4.00 \pm 3.23) \times 10^{8}$
	MSE-pos 20	1.10 ± 1.80	$\mathbf{0.51 \pm 0.68}$	$(1.83 \pm 2.21) \times 10^{4}$	47.1 ± 38.1
	MSE-acc Full	$(9.57 \pm 15.6) \times 10^{6}$	$\mathbf{(4.58 \pm 6.17) \times 10^{6}}$	$(1.64 \pm 1.97) \times 10^{11}$	$(4.24 \pm 3.43) \times 10^{8}$
	MSE-pos Full	1.15 ± 1.88	$\mathbf{0.54 \pm 0.73}$	$(1.95 \pm 2.35) \times 10^{4}$	49.9 ± 40.4
	EMD	0.49 ± 0.58	$\mathbf{0.27 \pm 0.11}$	88.7 ± 63.6	4.07 ± 2.62

Notes: Values are mean $\pm$ SD. Best metrics **bolded**.

Statistical Validation

Table 4: Mann–Whitney U Test p-values for GNN Type and Loss Type Comparisons per Dataset

Dataset	Comparison	MSE-pos-1	MSE-pos-full	MSE-acc-1	EMD
WaterDrop	GNN Type	0.3939	0.8182	0.3939	0.8182
	Loss Type	0.0022*	0.0260*	0.0022*	0.0260*
Sand	GNN Type	0.0260*	0.0931	0.0260*	0.1797
	Loss Type	0.8182	0.3095	0.8182	0.3095
Goop	GNN Type	0.0022*	0.0022*	0.0022*	0.0043*
	Loss Type	0.9372	0.6991	0.9372	0.4848

Notes: * $p < 0.05$ indicates statistical significance. Compares `GAT` vs `IN` and `one-` vs `multi-step` losses.

To further validate these patterns, we performed Mann–Whitney U tests (nonparametric) comparing the effects of *(i)* GNN architecture (`GAT` vs. `IN`) and *(ii)* loss type (`multi-step` vs. `one-step`) on each dataset.

WaterDrop. Loss type matters substantially for all measured metrics ($p < 0.05$)—multi-step yields significantly better 1-step and full-trajectory performance. However, GNN type is not statistically significant here, suggesting both architectures benefit roughly equally from multi-step training here.

Sand. GNN type significance emerges for `MSE-pos 1` and `MSE-acc 1` ($p < 0.05$), but not for longer-horizon metrics. Multi-step vs. one-step differences do not reach significance, though the magnitude shift from using multi-step is qualitatively large for GATs in particular.

Goop. GNN type is significant for all reported metrics ($p < 0.01$), indicating INs strongly outperform GAT overall. However, loss type is *not* statistically significant on these metrics.

It is important to acknowledge that our models were only run three times, resulting in a limited number of observations. This small sample size may reduce the statistical power of our tests and the generalizability of the observed trends. Future experiments with a larger number of runs would provide more robust validation of these findings.