BOUNDARY-CONDITION COMPLIANCE IN LEARNED FLUID SIMULATORS: BOUNDARY PARTICLES OVER DISTANCE FEATURES AND THE PRIMACY OF GEOMETRIC FIDELITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning-based simulators reduce the cost of particle-based fluid simulation, but autoregressive rollouts accumulate error over hundreds of steps. The failure that matters most is the loss of boundary-condition compliance: once fluid particles leave the domain, the remaining prediction is physically meaningless. This is usually treated as a training problem. We show instead that at a fixed model and a fixed training budget it turns on where the boundary is placed. We represent container walls as static boundary particles of the same type as rigid bodies, where the type acts as a multiplicative weight in message aggregation combined with a learned attention gate. This ports the boundary particles of smoothed particle hydrodynamics to a learned simulator without architectural change, preserving translation invariance. On Water-3D under a matched training budget, it reduces the 400-step domain escape rate from 35.8 ± 8.3 percent for a graph network simulator with clipped wall-distance features to 1.29 ± 0.32 percent. Moving the wall particles onto the uncorrected metadata box, 1 percent of the domain further in, collapses the rollout to 74.0 ± 9.4 percent escape and injects 2.8 times more energy than omitting walls entirely. Single-step validation loss does not track any of this: across three seeds it differs by 7.4 percent between two configurations whose escape rates differ 11-fold, and on the seed where the two losses are within 0.3 percent the escape rates still differ sevenfold. Boundary representation is therefore a design variable rather than an implementation detail, and we report error, conservation drift, and boundary violations jointly across the rollout horizon.

1 INTRODUCTION

Particle-based fluid simulation underpins engineering design, computer graphics, and scientific analysis, yet classical numerical solvers such as smoothed particle hydrodynamics spend most of their cost on neighbor interactions that grow with the particle count (Monaghan, 2005). Learning-based simulators have replaced the time-stepping operator with a graph neural network and have demonstrated large speedups for fluids, granular media, and deformable bodies (Sanchez-Gonzalez et al., 2020; Pfaff et al., 2021; Ummenhofer et al., 2020). This advantage, however, presupposes autoregressive rollout: the model consumes its own predictions for hundreds of steps, and small single-step errors amplify as they accumulate.

Over a long rollout the gravest failure is not a gradual loss of accuracy but a violation of physical law. Figure 1 shows the phenomenon that motivated this work. Among five configurations trained under an identical budget, the three that do not represent the boundary with the correct geometry lose between 36% and 87% of their particles from the domain by step 400, two seconds of physical time. Once fluid leaves the container, energy and momentum statistics become meaningless and the prediction cannot be used downstream. We call the property at issue **boundary-condition compliance**: the fraction of particles that respect the geometry of the container and the obstacles throughout the rollout.

Maintaining this compliance is hard because of a structural conflict. Generalization across scenes requires translation invariance, which is obtained by removing absolute positions from the in-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

put (Sanchez-Gonzalez et al., 2020; Satorras et al., 2021). A wall, however, is absolute geometry, so an invariant model cannot see it. Existing approaches resolve the conflict in three ways. **(i) Implicit features:** the graph network simulator encodes flat container walls as wall-distance features clipped at the connectivity radius. This presupposes a box, does not extend to general geometry, and, as this work shows, falls short of an explicit representation. **(ii) Boundary particles with a dedicated pathway:** continuous-convolution models place static particles on every boundary but process them through a separate network branch (Ummenhofer et al., 2020; Prantl et al., 2022). Models in the graph network simulator family use boundary particles only for obstacles, and consume the particle type solely as an input embedding. **(iii) Post-hoc correction:** Neural SPH identifies wall penetration as a principal failure mode and repairs it at inference time with classical SPH relaxation (Toshev et al., 2024). The learned model itself still fails to comply.

Taken together, these families reveal three requirements that no approach satisfies simultaneously. **(R1)** The boundary must be visible to a translation-invariant model. **(R2)** Walls and rigid obstacles should be handled by a single mechanism, with no boundary-specific parameters. **(R3)** Compliance should be learned, not appended as a correction at inference time.

This work satisfies all three with a deliberately simple treatment. Sampling the container walls as static boundary particles of the same type as rigid bodies lets the boundary geometry enter only through relative edge features (R1). The particle type acts not as an input embedding but as a multiplicative weight and an attention gate inside message aggregation, so one mechanism covers walls and rigid bodies without architectural change (R2). This ports the boundary-particle construction of classical SPH (Akinci et al., 2012) into a learned setting (R3). The type-weighted mechanism was introduced for rigid obstacles by Choi & Park (2026) and extended to container walls by Choi et al. (2026), in both cases on data generated for those studies. It is presented here in full and established where it had not been tested: on a public benchmark, under a matched budget, against the distance-feature representation it is meant to replace.

More importantly, this construction turns the boundary geometry itself into an experimental variable, which leads to the central finding of this paper. **At a fixed model and a fixed training budget, where the boundary is placed decides whether a rollout stays compliant.** On Water-3D, explicit boundary particles hold the 400-step domain escape rate at $1.29 \pm 0.32\%$ against $86.96 \pm 5.68\%$ without them. Placing the very same particles on the uncorrected metadata box, 1% of the domain further in, collapses the rollout to $74.0 \pm 9.4\%$ and injects 2.8 times more energy than omitting walls entirely: a boundary in the wrong place is worse than no boundary on the conservation axis, even though it escapes less. Compliance is therefore not something a model acquires from more training on a boundary it cannot see correctly.

Contributions.

1. **A finding about boundary placement.** At a fixed model and a fixed training budget, moving the boundary particles inside the fluid region is enough to collapse a rollout that is otherwise compliant, and it injects more energy than leaving the boundary out altogether (Section 5).
2. **A unified boundary treatment that holds on a public benchmark.** Walls and rigid bodies are represented by one type-weighted particle mechanism, introduced by Choi & Park (2026); Choi et al. (2026) and established here on Water-3D, where it cuts the 400-step escape rate from $86.96 \pm 5.68\%$ to $1.29 \pm 0.32\%$ at a matched budget and with 2.9 times fewer parameters than the baseline (Sections 4 and 5).
3. **A controlled comparison of boundary representations.** To the authors' knowledge, no prior work compares clipped wall-distance features against explicit boundary particles under the same data, budget, and metrics. We provide that comparison against a specification-verified reimplementation of the graph network simulator (Section 5).
4. **A compliance-centered evaluation protocol.** We formalize the three axes of error, conservation, and boundary violation, fix the escape tolerance by requiring the ground truth to score zero under the same criterion, and show on Water-3D that single-step validation loss is decoupled from 400-step compliance (Sections 3 and 5).

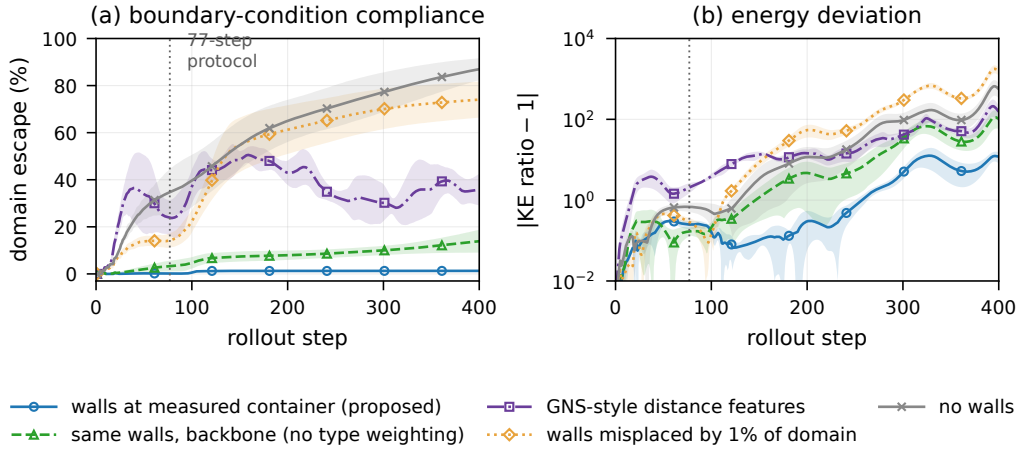

Figure 1: Per-step (a) domain escape rate and (b) kinetic-energy deviation on the Water-3D validation trajectories. Five configurations are trained on identical data with an identical budget; curves are the mean over seeds 42/43/44 and the bands are ± 1 standard deviation. The single-seed corrected-box baseline of Table 1 is not plotted. Only the configuration with explicit boundary particles at the correct position stays compliant over the whole rollout.

2 Related Work

Learned particle simulators. Learned simulators descend from interaction networks and message passing over relational graphs (Battaglia et al., 2016; Gilmer et al., 2017; Battaglia et al., 2018), first turned on physics as learnable engines and particle-based dynamics (Sanchez-Gonzalez et al., 2018; Mrowca et al., 2018). GNS (Sanchez-Gonzalez et al., 2020) set the standard for the field with an encode–process–decode architecture over a graph of particles and proximity edges, trained with noise injection. MeshGraphNets (Pfaff et al., 2021) extends the idea to meshes, and continuous-convolution models (Ummenhofer et al., 2020; Prantl et al., 2022; Winchenbach & Thuerey, 2024) extend it to grid-free convolution. LagrangeBench (Toshev et al., 2023) standardizes SPH datasets and their evaluation.

Boundary handling. Representing a wall with particles is not new in itself. It originates in the boundary particles of classical SPH (Akinci et al., 2012; Ihmsen et al., 2014; Bender & Koschier, 2015), and on the learned side continuous-convolution models unify walls and obstacles as static particles (Ummenhofer et al., 2020), DPI-Net (Li et al., 2019) uses wall particles with one-directional edges, and LagrangeBench (Toshev et al., 2023) ships dummy wall particles and a particle type as part of the dataset specification. Ocean-engineering applications also report adding wall boundary nodes to GNS (Higaki et al., 2025). What differs here is not that the representation exists but *how the type is used*. In earlier work the type is a node input feature or a learned embedding (Sanchez-Gonzalez et al., 2020; Mayr et al., 2023), or it requires a separate network branch (Ummenhofer et al., 2020; Prantl et al., 2022). The model used here keeps the type as a node feature and additionally applies it as a multiplicative weight in message aggregation, which lets one mechanism cover walls and rigid bodies without architectural change.

Rollout stability. Many studies address stability through training and inference techniques: noise injection (Sanchez-Gonzalez et al., 2020), the pushforward loss (Brandstetter et al., 2022), correction against a differentiable solver (Um et al., 2020), coarse learned operators for long horizons (Stachenfeld et al., 2022; Kochkov et al., 2021), and SPH relaxation at inference (Toshev et al., 2024). To our knowledge, none isolates and measures the effect that the choice of boundary *representation*, and its geometric accuracy, has on stability. Neural SPH names wall penetration as a failure mode, but its remedy is classical post-processing rather than learning, and it includes no sensitivity analysis of boundary placement.

3 Background and Evaluation Protocol

3.1 Problem setting

At time t the particle system is described by positions x_i^t, velocities v_i^t, and types $c_i \in \{0, 1\}$, where 0 is fluid and 1 is boundary. Bidirectional edges are created between particle pairs within distance r; node features consist of the last C velocities and the type, and edge features of the relative position and the distance. Absolute coordinates are not part of the input, which secures translation invariance. The model outputs a normalized velocity delta and positions are updated by forward Euler. The positions and velocities of boundary particles are fixed to the given trajectory and enter neither the loss nor any metric.

3.2 Compliance-centered evaluation protocol

Rollouts are usually evaluated with a single position or velocity MSE. This paper reports three axes at every step t, each capturing a different failure mode. Writing F for the set of fluid particles and $\hat{\cdot}$ for a prediction, the *error* axis is

$$\mathcal{E}_v^t = \frac{1}{|F|}\sum_{i\in F}\|\hat{v}_i^t - v_i^t\|^2, \qquad \mathcal{E}_x^t = \frac{1}{|F|}\sum_{i\in F}\|\hat{x}_i^t - x_i^t\|^2, \tag{1}$$

the *conservation* axis is the kinetic-energy ratio deviation and the momentum drift

$$\mathcal{K}^t = \left|\frac{\sum_{i\in F}\|\hat{v}_i^t\|^2}{\sum_{i\in F}\|v_i^t\|^2} - 1\right|, \qquad \mathcal{M}^t = \left\|\frac{1}{|F|}\sum_{i\in F}\left(\hat{v}_i^t - v_i^t\right)\right\|, \tag{2}$$

and the *boundary-condition* axis is the escape rate from the domain Ω_τ, the computational domain dilated by a tolerance τ, together with the penetration rate into the rigid surface $\mathcal{B}$,

$$\mathcal{D}^t = \frac{1}{|F|}\sum_{i\in F}\mathbf{1}\left[\hat{x}_i^t \notin \Omega_\tau\right], \qquad \mathcal{P}^t = \frac{1}{|F|}\sum_{i\in F}\mathbf{1}\left[\, d\big(\hat{x}_i^t, \mathcal{B}\big) < -\delta \,\right] \tag{3}$$

where d is the signed distance to the rigid surface and δ is the penetration depth threshold. Water-3D holds no rigid obstacles, so $\mathcal{P}^t$ belongs to the protocol but is not reported here; the boundary axis is carried by $\mathcal{D}^t$ throughout.

Every metric is restricted to fluid particles. Rigid and wall boundary particles are an external boundary condition driven by the given trajectory, so the system is open and $\mathcal{K}^t$ and $\mathcal{M}^t$ are defined as deviations from the ground truth at the same step rather than as absolute conserved quantities. Each violation metric is reported alongside the value obtained by applying the same criterion to the ground-truth trajectory. That baseline is what makes the choice of τ non-arbitrary: Water-3D ground-truth particles compress up to about 0.009 beyond the metadata box, so at $\tau = 0.01$ the ground-truth escape rate $\mathcal{D}^t$ is exactly 0% at every step, whereas $\tau = 0$ would score the ground truth itself as 7 to 9% false escape.

The three axes are reported together because they can disagree. An over-damped prediction, one that systematically under-predicts motion, attains a low velocity MSE precisely by failing to move, while drifting in position and losing energy. Selecting on velocity MSE alone can therefore pick the model with the largest position error and the largest conservation drift, and Section 5.4 shows the same decoupling between the single-step loss and 400-step compliance.

4 Unified Type-Weighted Boundary Particles

The method in this section follows Choi et al. (2026) and is restated in full so that the experiments can be read without reference to that paper.

4.1 Construction of the boundary particles

Figure 2 summarizes the whole treatment. Each face of the container is sampled on a grid of spacing s to generate static particles, which are assigned type $c = 1$ and zero velocity and appended to every

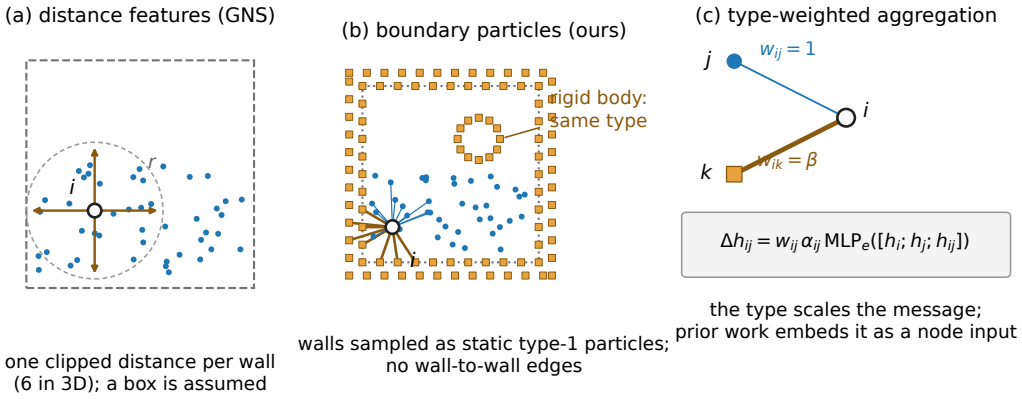

Figure 2: The contrast between boundary representations and where the type is used. (a) GNS assigns each node its distance to the container walls, clipped at the connectivity radius r, which presupposes a box. (b) This work samples the walls as static particles of the same type 1 as rigid bodies, so the boundary geometry enters only through relative edge features; wall-to-wall edges are excluded. (c) Besides entering the node input, the type acts as a multiplicative weight w_{ij} in message aggregation, combined with the learned attention gate α_{ij}.

frame in exactly the same format as rigid boundary particles. Neither the data nor the model distinguishes a wall from a rigid body. Wall geometry is conveyed only through the relative displacement and distance of the edges, so translation invariance is preserved (R1). Wall-to-wall edges contribute nothing to learning and are excluded at graph construction.

4.2 Type-weighted message aggregation

The edge update in each processor layer is

$$\Delta h_{ij} = w_{ij}\,\alpha_{ij}\,\mathrm{MLP}_e([h_i; h_j; h_{ij}]) \tag{4}$$

where α_{ij} is a learned attention gate (Veličković et al., 2018) and w_{ij} is the type weight, set to $w_{ij} = \beta$ on an edge with a boundary particle at either end. The type is also part of the node input, as Section 3.1 states; what is added here is that the same type additionally acts as a **structural weight at the aggregation stage** rather than being consumed by the input embedding alone. The design transfers to message aggregation the classical SPH practice of amplifying the contribution of boundary particles by a correction factor (Akinci et al., 2012): β is physical prior knowledge that is not learned, while α_{ij} is learned from data. Because the mechanism already exists for rigid bodies and is applied to walls unchanged, no architectural change and no wall-specific parameters are required (R2, R3).

The training objective combines a normalized-delta error with weighted position, momentum, and energy terms, and rollout stabilization uses random-walk noise injection, a two-step pushforward loss, and a velocity history, following Sanchez-Gonzalez et al. (2020) and Brandstetter et al. (2022); both are given in full in Appendix A. The same objective and the same stabilization settings are applied to the proposed configuration, the backbone, the misplaced-wall runs, and the no-wall run, so within that group the boundary representation is the only variable; the GNS baseline keeps its own published recipe instead.

5 Experiments

Materials and equipment. The primary dataset is the public benchmark Water-3D (Sanchez-Gonzalez et al., 2020); the first 10 trajectories of the training split and the first 3 of the validation split were taken in file order. Each trajectory holds 801 frames with 4.6k to 11k particles; every particle is labeled water and the container is given only as a metadata bounding box. All runs used a single NVIDIA GeForce RTX 4090 with PyTorch 2.12.1 (CUDA 12.6), and one configuration took

Table 1: Comparison of configurations on Water-3D. Values are the mean ± standard deviation over 3 validation trajectories and seeds 42/43/44, and every configuration uses the same training budget. The escape tolerance is 0.01, at which the ground truth escapes 0%.

	Escape rate (%)		At step 400		
Boundary representation	step 77	step 400	$\lvert\mathrm{KE}-1\rvert$	vel MSE	pos MSE
Corrected walls + type weight	$\mathbf{0.17 \pm 0.09}$	$\mathbf{1.29 \pm 0.32}$	$\mathbf{11.9 \pm 4.6}$	$\mathbf{1.3\times10^{-5}}$	0.30
Same walls + backbone	3.31 ± 2.82	13.89 ± 5.96	103 ± 54	5.3×10^{-5}	2.37
GNS distances, metadata box	23.92 ± 9.15	35.81 ± 8.32	155 ± 84	4.8×10^{-5}	$\mathbf{0.19}$
GNS distances, corrected box†	24.38	63.79	262	6.3×10^{-5}	0.64
Walls on the metadata box	14.23 ± 3.78	74.04 ± 9.41	1551 ± 424	5.3×10^{-4}	18.5
No walls	34.72 ± 12.05	86.96 ± 5.68	559 ± 277	1.4×10^{-4}	4.92

Escape rate is the fraction of fluid particles outside the domain dilated by the tolerance; $\lvert\mathrm{KE}-1\rvert$ is the deviation of the kinetic-energy ratio from the ground truth; vel MSE and pos MSE are mean squared errors against the ground truth in model units. Every metric is computed over fluid particles only. † single seed (42); every other row is the three-seed mean. The row with walls on the metadata box uses a single wall layer of 10,966 particles, against two layers and 24,316 particles in the other wall rows, so it differs from them in sampling as well as in placement.

1.6 to 8.0 hours to train. The conversion procedure, the training objective, and the full environment are given in Appendix A, where Figure 5 also draws the pipeline.

Method. Every configuration was trained on the same data, with the same budget (epochs and gradient steps, 100 epochs) and the same seeds (42/43/44), and results are reported as the mean ± standard deviation over the three seeds. Adam was used and the gradient norm was clipped at 1.0. Graph size varies between trajectories, so gradient accumulation replaces true mini-batching as the effective-batch control; it was fixed at 1. Normalization statistics were computed from the training split alone. The GNS baseline is a reimplementation whose specification (hidden size 128, 10 GN blocks, velocity history of 5, clipped wall-distance features, acceleration target, random-walk noise) was checked against the original paper and the official code. It was trained with the same 80k-step budget as the proposed configuration (800 samples over 100 epochs) rather than the 20M-step run of the original paper, because the comparison targets the boundary representation at a matched budget, not absolute performance. Because the metadata box the distance features reference lies inside the real container, we also trained the baseline with that box pushed out by the same 0.010 used for the wall particles, at the same budget and seed; escape is judged against the metadata box in every row. Table 2 summarizes the two architectures; the proposed model has 2.9 times *fewer* parameters (0.56M against 1.59M), so the gaps are not explained by capacity.

The experiments answer four questions. **Q1** Do explicit boundary particles beat implicit distance features? **Q2** How much does the geometric accuracy of the boundary placement matter? **Q3** Where does the architecture contribute? **Q4** Do single-step metrics predict rollout stability?

5.1 Q1: BOUNDARY PARTICLES AGAINST DISTANCE FEATURES

Table 1 compares the six configurations. At a matched budget the proposed treatment reached a 400-step escape rate of $1.29 \pm 0.32\%$, 28 times lower than the $35.81 \pm 8.32\%$ of the GNS-style distance features. The advantage held for the energy deviation (11.9 against 155) and for the velocity error, and the same ordering appeared at step 77, where the escape rates were 0.17% and 23.92%. Position MSE at step 400 was lower for GNS (0.19) than for the proposed model (0.30), but by that step 36% of the GNS particles had already left the domain, so the two numbers are not comparable on their own. Figure 4 shows the same contrast in the trajectories.

That comparison could in principle be about geometry rather than representation, because the distance features are computed against the metadata box, which sits inside the real container, while the boundary particles are placed outside it. We therefore trained the baseline again with the reference box pushed out by the same 0.010, at the same budget and seed. It does not close the gap: the corrected baseline escapes 63.79% at step 400 and 24.38% at step 77, against 38.69% and 16.36% for the same seed on the metadata box. The two numbers straddle the spread this baseline shows

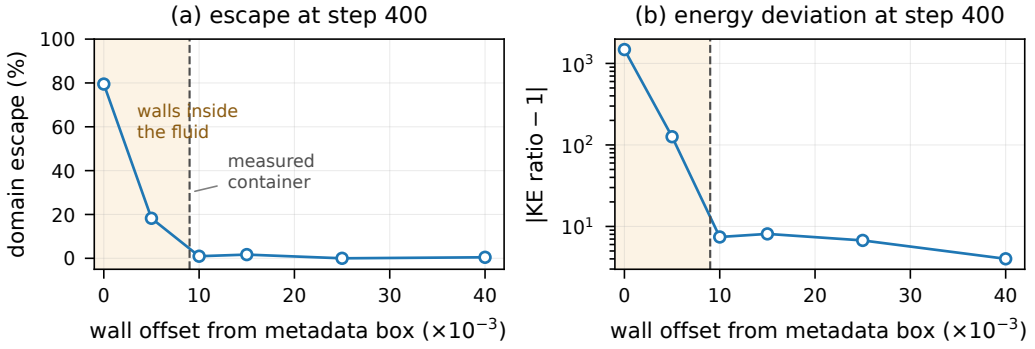

Figure 3: Wall-offset sweep. The horizontal axis is the outward displacement from the metadata bounding box, the shaded band is the range in which the wall lies inside the fluid region, and the dashed line marks the measured container. The rollout collapses while the wall is inside the fluid and is far better once it moves outside. Wall positions are quantized onto a grid of half the sampling spacing, so offsets 0.005 and 0.010 share the same wall planes; the difference between those two points measures run-to-run variation rather than placement. Every point was trained with the same budget and with the single seed 42, so the sweep carries no error bars; the three-seed means are in Table 1.

anyway across seeds and trajectories, 15.5 to 61.5% (Appendix B), so the honest reading is that correcting the reference geometry leaves the distance features where they were. What separates the configurations is the representation, not the box it is measured from.

5.2 Q2: Geometric fidelity of the boundary placement

The bounding box declared in the Water-3D metadata sits inside the real container. Ground-truth particles compress up to about 0.009 beyond it, 1.2% of the domain, so placing walls on the metadata box leaves fluid *outside* the wall in every frame. Figure 3 gives the result of sweeping this offset. Each offset requires its own training run, so the sweep used the single seed 42 and carries no error bars; its numbers do not match the three-seed means of Table 1, and offset 0.010, the proposed setting, appears here as the seed-42 value of 0.95%.

One offset stands apart. At 0.000 the innermost wall plane lands at 0.875, which is 0.007 inside the furthest ground-truth particle, and the 400-step escape rate is 79.5% with an energy deviation of 1480. Every offset that puts the plane outside the fluid escapes far less: 18.3%, 0.95%, 1.69%, 0.00%, and 0.46%, with energy deviations of 126, 7.4, 8.1, 6.7, and 4.0.

The spread within that second group is not a placement effect. The wall generator quantizes positions onto a grid of half the sampling spacing, so offsets 0.005 and 0.010 produce *identical* wall planes and differ only in how far the face grids extend. Their escape rates are 18.3% and 0.95%. A nineteen-fold difference at unchanged geometry sets the scale of run-to-run variation in this single-seed sweep, and it is larger than the differences among the offsets that lie outside the fluid. What the sweep supports is therefore the one-sided condition that **the wall must not be placed inside the fluid region**, and not a resolved transition point or a plateau value. Every point was trained with the same budget.

Misplaced walls also deviated 2.8 times further in kinetic energy than no walls at all, 1551 against 559, while escaping less, 74.04% against 86.96%.

5.3 Q3: Where the architecture contributes

Table 1 also isolates the architecture. Under the same corrected walls, the backbone with the type weighting and the attention gate removed reached a 400-step escape rate of $13.89 \pm 5.96\%$, 11 times that of the proposed model ($1.29 \pm 0.32\%$), and the energy deviation separated as 103 against 11.9. Removing the attention gate also removes its parameters, so the backbone is the smaller model

(391,683 against 557,453); this comparison therefore varies capacity along with the mechanism, unlike the comparison against GNS, where the proposed model is the smaller one. At step 77, by contrast, the gap was small, 3.31% against 0.17%, and the backbone had a marginally lower energy deviation.

5.4 Q4: THE LIMITS OF SINGLE-STEP METRICS

Averaged over the three seeds, the single-step validation loss of the proposed model and of the backbone differ by 7.4% (0.627 against 0.673), yet their 400-step escape rates differ by a factor of 11. The per-seed picture is sharper still: on seed 43 the two losses are within 0.3% of each other (0.647 against 0.649) while the escape rates are 1.58% and 11.17%. Walls on the metadata box differ from corrected walls by about 8% in single-step loss and by a factor of 57 in escape rate.

6 DISCUSSION

The question raised in Section 1 was why learned fluid simulators lose boundary-condition compliance over long rollouts, and whether that loss is a matter of training. The answer this work supports is that it is not. At a matched budget the escape rate changed by more than an order of magnitude with the boundary representation alone, and it changed again by a factor of 57 when the same representation was displaced by 1% of the domain. What governs compliance is how faithfully the boundary geometry is encoded.

What supports the answer. Two results carry it. The offset sweep isolated placement at a fixed model and a fixed budget, so the collapse at offset 0.000, the only setting whose wall plane lies inside the fluid, can only be geometric. Correcting the reference geometry of the distance features, at the same budget and seed, did not recover them, which locates the effect in the representation rather than in the box it is measured from. Capacity does not explain either: the proposed model has 2.9 times fewer parameters than the GNS baseline.

Relation to earlier work. The answer is consistent with classical SPH, where boundary particles exist because a kernel truncated at a wall cannot reproduce the correct pressure response (Akinci et al., 2012), and with Neural SPH, which names wall penetration as a principal failure mode of learned Lagrangian simulators (Toshev et al., 2024). The two diverge on the remedy: Neural SPH repairs the violation at inference with classical relaxation, which leaves the learned model non-compliant. Representing walls as particles is itself not new either; LagrangeBench offers both representations as dataset options without comparing them (Toshev et al., 2023). What has been missing is a measurement of how much the placement of that boundary matters, which is what this paper supplies.

Two unexpected results. Misplaced walls escaped less than no walls at all (74.04% against 86.96%) yet deviated 2.8 times further in kinetic energy (1551 against 559). A wall sitting inside the fluid region confines the particles, which suppresses escape, while exerting a repulsion at positions where no surface exists, which injects energy. The two axes therefore disagree about which configuration is worse, and a protocol reporting only escape would rank the misplaced walls above the absent ones. This is the clearest evidence in the paper that compliance and conservation must be read together.

The second is the weak coupling between single-step loss and rollout behavior. On seed 43 the two losses agree to 0.3% while the escape rates differ sevenfold, so the quantity these models are trained on, and the one most often reported, carries little information about how far a rollout will drift.

Limits of the evidence. The validation covers 3 trajectories and 3 seeds, and the spread between seeds reaches ± 8 to 12 percentage points for the GNS baseline; the gaps between configurations exceed it, but a larger sample would settle the matter. The GNS baseline is a matched-budget reimplementation, so the comparison is between boundary-handling schemes and not against the published performance of GNS. The offset sweep used a single seed, and two of its offsets share the same wall planes yet differ nineteen-fold, so it establishes that a wall inside the fluid is catastrophic but resolves neither the location of the transition nor the size of the effect outside that region.

Boundary particles scale with surface area, a 3.2-fold increase in nodes and 58–67% in per-step cost (Appendix C). The boundary is driven in one direction only. An energy deviation of 11.9 persists at step 400: escape is suppressed, but the late-time energy excess is unresolved.

Why it matters. The boundary belongs among the design decisions of a learned simulator, not among its implementation details. The practical consequence is narrow and testable: sampling the container as boundary particles of the type already used for rigid bodies costs no architectural change and no boundary-specific parameters, and the placement need only stay outside the fluid.

7 Conclusion

Whether a learned fluid simulator stays inside its container over a long rollout turned on where its boundary was placed. At a fixed model and a fixed budget, boundary particles of the rigid-body type held the 400-step escape rate at 1.29% against 86.96% without them, and moving those same particles one percent of the domain further in collapsed it to 74.0% while injecting more energy than having no boundary at all. The boundary is therefore a design decision rather than an implementation detail. The late-time energy excess and the extension to arbitrary geometry remain open.

Ethics statement

This work involves no human or animal subjects and uses only the public Water-3D benchmark; it contains no personal or sensitive information. Because learned fluid simulators may be used in engineering design and safety analysis, we consider it important to report, rather than hide, the intervals where a prediction departs from the ground truth. That is why this paper reports boundary violations and conservation drift alongside a ground-truth baseline instead of a single metric. The proposed treatment lowers the escape rate substantially but leaves an energy deviation at step 400, so it should not be applied to safety-critical use without further validation.

AI use statement

Large language models were used in this work to assist with the literature survey, to assist in writing the data-conversion, training, and evaluation scripts, and to edit the prose of this manuscript. All LLM-assisted code was executed by the authors, and every number reported in this paper was read directly from the logs produced by running that code. The authors checked the metric definitions and the evaluation procedure against the code. The results of the literature survey were verified by the authors against the primary sources. The research idea, the experimental design, and the interpretation of the results are the authors' own, and the authors take responsibility for the final content of this manuscript.

Reproducibility statement

All experiments use the public Water-3D benchmark, and the data conversion, the training and evaluation procedure, and the computing environment are described in Appendix A. Every value in Table 1 is the mean of three independent runs with seeds 42, 43, and 44, evaluated on fixed validation trajectories at a tolerance of 0.01. The code and the trained weights will be released on publication.

References

Nadir Akinci, Markus Ihmsen, Gizem Akinci, Barbara Solenthaler, and Matthias Teschner. Versatile rigid-fluid coupling for incompressible SPH. *ACM Transactions on Graphics*, 31(4):62:1–62:8, 2012. doi: 10.1145/2185520.2185558.

Peter W Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Jan Bender and Dan Koschier. Divergence-free smoothed particle hydrodynamics. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pp. 147–155, 2015. doi: 10.1145/2786784.2786796.

Johannes Brandstetter, Daniel E Worrall, and Max Welling. Message passing neural PDE solvers. In *International Conference on Learning Representations (ICLR)*, 2022.

Junho Choi and Seungmin Park. Hybrid physics-guided graph neural network with an attention gate for accelerating particle-based simulation. In *Proceedings of the KIIS Spring Conference*, volume 36, pp. 36–37, Jeju, Republic of Korea, Apr. 2026.

Junho Choi, Kwangil Kim, and Seungmin Park. Stable prediction of fluid–rigid body interaction simulation using physics-guided graph neural networks. *Journal of Korean Institute of Intelligent Systems*, 2026. in press.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning (ICML)*, pp. 1263–1272, 2017.

Takefumi Higaki, Yuki Tanabe, Hirotada Hashimoto, and Takahito Iida. Step-by-step enhancement of a graph neural network-based surrogate model for Lagrangian fluid simulations with flexible time step sizes. *Applied Ocean Research*, 154:104424, 2025. doi: 10.1016/j.apor.2025.104424.

Markus Ihmsen, Jens Orthmann, Barbara Solenthaler, Andreas Kolb, and Matthias Teschner. SPH fluids in computer graphics. *Eurographics State of the Art Reports*, pp. 21–42, 2014.

Dmitrii Kochkov, Jamie A Smith, Ayya Alieva, Qing Wang, Michael P Brenner, and Stephan Hoyer. Machine learning-accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021. doi: 10.1073/pnas.2101784118.

Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *International Conference on Learning Representations (ICLR)*, 2019.

Andreas Mayr, Sebastian Lehner, Arno Mayrhofer, Christoph Kloss, Sepp Hochreiter, and Johannes Brandstetter. Boundary graph neural networks for 3D simulations. In *AAAI Conference on Artificial Intelligence*, volume 37, pp. 9099–9107, 2023. doi: 10.1609/aaai.v37i8.26092.

Joseph J Monaghan. Smoothed particle hydrodynamics. *Reports on Progress in Physics*, 68(8): 1703–1759, 2005.

Damian Mrowca, Chengxu Zhuang, Elias Wang, Nick Haber, Li Fei-Fei, Joshua B Tenenbaum, and Daniel L K Yamins. Flexible neural representation for physics prediction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations (ICLR)*, 2021.

Lukas Prantl, Benjamin Ummenhofer, Vladlen Koltun, and Nils Thuerey. Guaranteed conservation of momentum for learning particle-based fluid dynamics. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter W Battaglia. Graph networks as learnable physics engines for inference and control. In *International Conference on Machine Learning (ICML)*, pp. 4467–4476, 2018.

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W Battaglia. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning (ICML)*, pp. 8459–8468, 2020.

Víctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) equivariant graph neural networks. In *International Conference on Machine Learning (ICML)*, pp. 9323–9332, 2021.

Table 2: Training setup. Both configurations use the same data, radius, epoch count, noise scale, and seeds. The proposed model has the smaller parameter count.

	Proposed (type-weighted)	GNS baseline
Hidden size / layers	64 / 10	128 / 10
Velocity history C	3	5
Prediction target	normalized velocity delta	normalized acceleration
Boundary representation	type-1 boundary particles	clipped wall distances
Parameters	557,453	1,590,787
Radius r / step Δt	0.035 / 1.0 (GNS convention)	
Noise standard deviation	6.7×10^{-4} (random walk)	
Learning rate	$3 \times 10^{-4} \to 10^{-6}$	$10^{-4} \to 10^{-6}$
Learning-rate schedule	cosine	exponential
Optimizer / gradient clip	Adam / norm 1.0	
Gradient accumulation	1 (variable graph size, no true batching)	
Pushforward probability p	0.3	—
Type weight β	2.0	—
Epochs / frame stride	100 / 10	
Seeds	$42, 43, 44$	
Training time	1.6–8.0 h (one RTX 4090)	

Kimberly Stachenfeld, Drummond B Fielding, Dmitrii Kochkov, Miles Cranmer, Tobias Pfaff, Jonathan Godwin, Can Cui, Shirley Ho, Peter Battaglia, and Alvaro Sanchez-Gonzalez. Learned coarse models for efficient turbulence simulation. In *International Conference on Learning Representations (ICLR)*, 2022.

Artur P Toshev, Gianluca Galletti, Fabian Fritz, Stefan Adami, and Nikolaus A Adams. LagrangeBench: A lagrangian fluid mechanics benchmarking suite. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2023.

Artur P Toshev, Jonas A Erbesdobler, Nikolaus A Adams, and Johannes Brandstetter. Neural SPH: Improved neural modeling of Lagrangian fluid dynamics. In *International Conference on Machine Learning (ICML)*, 2024.

Kiwon Um, Robert Brand, Yun Fei, Philipp Holl, and Nils Thuerey. Solver-in-the-loop: Learning from differentiable physics to interact with iterative PDE-solvers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Benjamin Ummenhofer, Lukas Prantl, Nils Thuerey, and Vladlen Koltun. Lagrangian fluid simulation with continuous convolutions. In *International Conference on Learning Representations (ICLR)*, 2020.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Rene Winchenbach and Nils Thuerey. Symmetric basis convolutions for learning Lagrangian fluid mechanics. In *International Conference on Learning Representations (ICLR)*, 2024.

A Details of the Materials and the Experimental Environment

Conversion of Water-3D. The original release holds positions only, in the SequenceExample TFRecord format. It was decoded with a pure-Python parser that interprets the record lengths and CRCs directly, without a TensorFlow dependency, and then converted into per-trajectory arrays. Velocity is defined as the frame-to-frame position difference, so the step size in model units is $\Delta t = 1$ (the GNS convention). The bounding box given in the metadata is $[0.125,\ 0.875]$ on each axis and the connectivity radius is 0.035. Trajectories were not sampled at random: the first N records of the training and validation files were read in order.

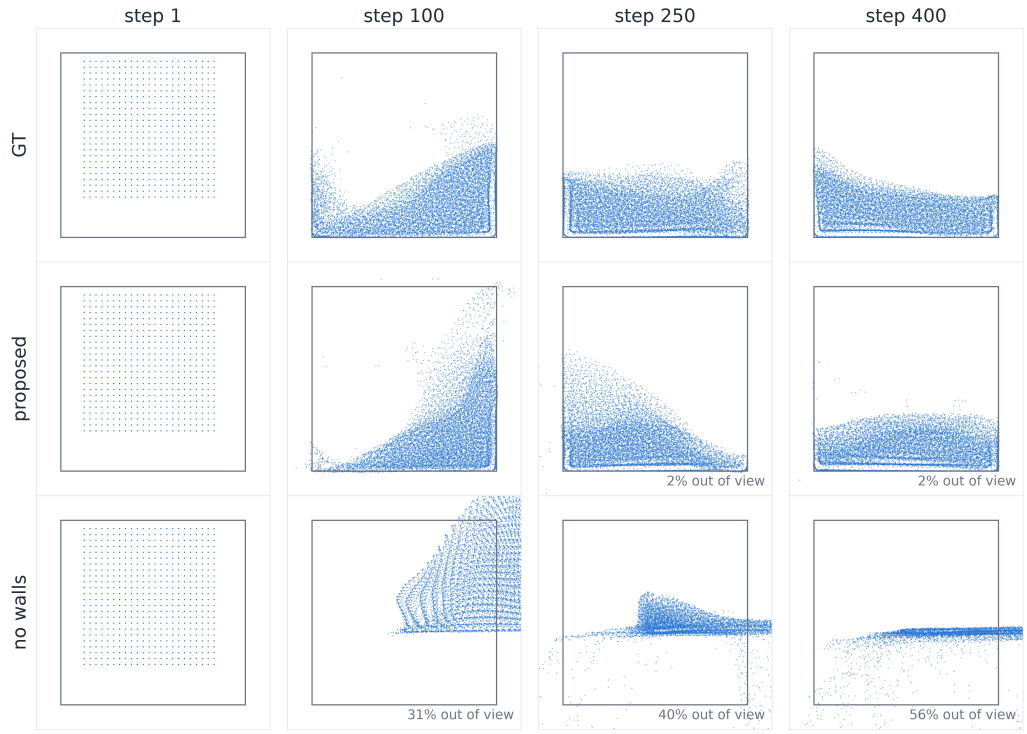

Figure 4: Qualitative comparison on a validation trajectory (x–y projection). Rows are the ground truth, the proposed method, and the configuration without walls; columns are rollout steps. The grey box is the metadata bounding box, and the annotation in each panel is the fraction of particles outside the view. Without walls the fluid receives no reaction force and collapses sideways.

Training objective and stabilization. The loss combines a normalized-delta error as the primary term with three weighted physical terms,

$$\mathcal{L} = \mathcal{L}_\Delta + \lambda_x \mathcal{L}_x + \lambda_p \mathcal{L}_p + \lambda_E \mathcal{L}_E. \tag{5}$$

For the normalized delta $z_i = \sigma^{-1}(v_i^{t+1} - v_i^t)$, the primary term and the position-consistency term are

$$\mathcal{L}_\Delta = \frac{1}{|F|} \sum_{i \in F} \|\hat{z}_i - z_i\|^2, \qquad \mathcal{L}_x = \frac{1}{|F|} \sum_{i \in F} \left\| \hat{v}_i^{t+1} - \frac{x_i^{t+1} - x_i^t}{\Delta t} \right\|^2 \tag{6}$$

and the conservation terms use the same quantities as the evaluation metrics, clipped to prevent divergence,

$$\mathcal{L}_p = \min\left(\frac{\|\bar{\hat{v}} - \bar{v}\|^2}{\frac{1}{|F|}\sum_{i \in F} \|v_i\|^2 + \epsilon},\ 10 \right), \qquad \mathcal{L}_E = \min\left(\frac{|\hat{E}_k - E_k|}{E_k + \epsilon},\ 10 \right) \tag{7}$$

where $\bar{v}$ is the mean fluid velocity and E_k the fluid kinetic energy. Every term applies to fluid particles only; boundary particles are snapped to the ground truth and contribute to none of them. The weights were fixed at $(\lambda_x, \lambda_p, \lambda_E) = (0.1, 0.05, 0.05)$.

For rollout stabilization, random-walk noise is injected into the input state (Sanchez-Gonzalez et al., 2020), a two-step pushforward loss is applied with probability p (Brandstetter et al., 2022), and C velocities of history are included in the node features. This combination of training techniques is established in prior work. It is applied identically to the proposed configuration, the backbone, the misplaced-wall runs, and the no-wall run, so within that group the boundary representation is the only variable. The GNS baseline keeps its own published recipe instead, and Section 5 states what that leaves uncontrolled.

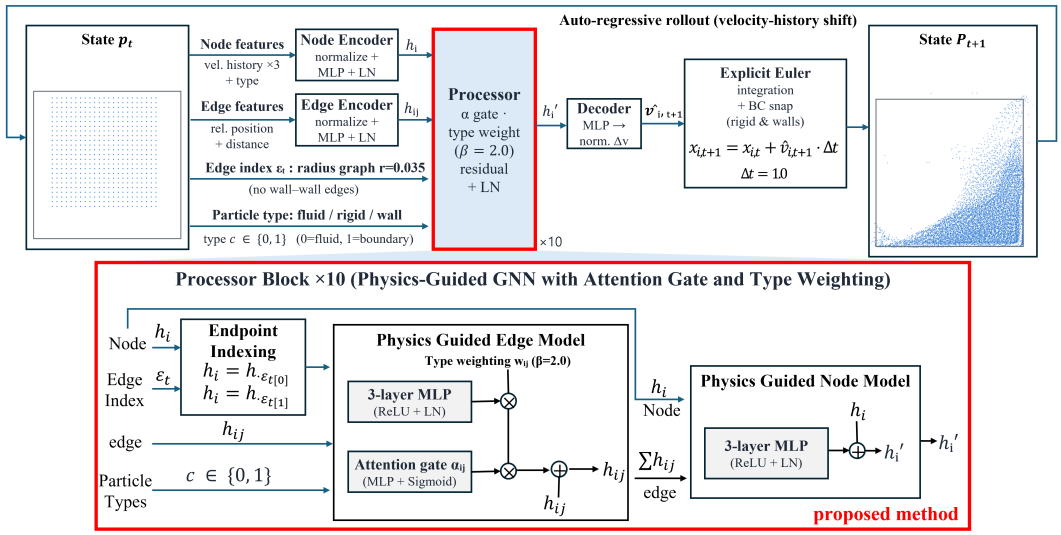

Figure 5: The full pipeline. One step encodes the node and edge features, applies ten processor blocks, decodes a normalized velocity delta, and integrates by forward Euler while the boundary particles are snapped to the given trajectory; the velocity history is then shifted and the step repeats. The lower panel expands one processor block and shows the components the main text leaves implicit: endpoint indexing of the node states onto the edges, 3-layer MLPs with ReLU and LayerNorm in both the edge and the node model, residual connections on both updates, and the point at which the attention gate α_{ij} and the type weight w_{ij} multiply the edge message before it is summed into the node. The states shown are Water-3D validation frames.

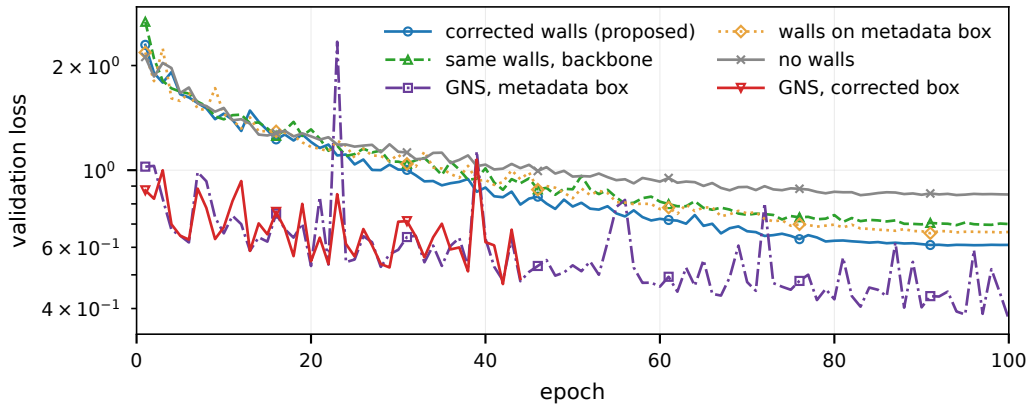

Figure 6: Validation loss over the matched 100-epoch budget, seed 42. Losses are comparable *within* an architecture but not across architectures, because the proposed model regresses a normalized velocity delta and the GNS baseline a normalized acceleration, each against its own statistics; the vertical position of the two GNS curves relative to the others therefore carries no meaning. Two things do read off the plot. The configurations with walls separate from the wall-free run early and stay separated. And the GNS baseline is still improving when the budget ends, gaining 14.5% over its last 25 epochs against 6.1% for the proposed configuration, so the matched budget leaves it short of convergence; this is a limit of the comparison and is stated as such in Section 6.

Choice of the GNS variant. The GNS configuration reported in Table 1 is the variant without self-edges. The variant with self-edges, the default in the official implementation, was worse at the same budget and the same seed, with escape rates of 55.8% at step 77 and 49.7% at step 400, so the better of the two was adopted as the baseline.

Computing environment. Training and evaluation were carried out on a single NVIDIA GeForce RTX 4090 (24 GB, compute capability 8.9). The software environment is Python 3.13, PyTorch 2.12.1 (CUDA 12.6, cuDNN 9.10.2), NumPy 2.4.6, and SciPy 1.18.0. No geometric deep-learning library was used; the scatter sum is implemented directly on `index_add_`. That operation is non-deterministic on the GPU, so runs differ slightly even at the same seed. Training the proposed configuration on Water-3D took about 350 s per epoch and about 8 hours for the full 100 epochs; configurations with fewer particles finished in 1.6 hours.

B Per-Trajectory Spread

Table 1 reports a mean and a standard deviation over three seeds, which hides how much the three validation trajectories differ. Table 3 gives all nine values behind each row.

Table 3: 400-step escape rate (%) for each of the three validation trajectories, under each of the three seeds. Entries are ordered by trajectory within a seed.

Configuration	seed 42	seed 43	seed 44	range
Corrected walls + type weight	2.2 / 0.7 / 0.0	2.7 / 2.0 / 0.0	2.6 / 1.3 / 0.1	0.0–2.7
Same walls + backbone	22.3 / 21.2 / 18.6	17.3 / 10.2 / 6.0	21.1 / 0.4 / 7.8	0.4–22.3
GNS, metadata box	48.2 / 27.5 / 40.3	40.0 / 61.5 / 25.4	15.5 / 43.2 / 20.7	15.5–61.5
Walls on metadata box	70.4 / 96.3 / 69.6	100.0 / 99.6 / 40.9	93.0 / 47.1 / 49.6	40.9–100.0
No walls	92.1 / 80.9 / 70.2	97.2 / 92.3 / 87.8	90.3 / 83.2 / 88.7	70.2–97.2

Two things follow. The proposed configuration and the GNS baseline do not overlap on any of the nine runs: the worst proposed run escapes 2.7% and the best GNS run 15.5%. The proposed configuration and the backbone *do* overlap, since one backbone run reaches 0.4% while one proposed run reaches 2.7%; the architectural effect of Section 5.3 is therefore a shift in distribution rather than a separation, and the three-seed mean in Table 1 overstates how cleanly the two are distinguished. Walls placed on the metadata box lose every particle on one run.

C Computational Cost of the Boundary Particles

Boundary particles enlarge the graph, and Table 4 measures by how much on a Water-3D validation trajectory. Walls are sampled at spacing $r/2$ with a margin of 0.010 and two layers, which is the configuration used throughout the paper, and adds 24,316 static particles to the 11,040 fluid particles. Forward-pass times are the mean of 20 runs after 5 warm-up runs, measured on the RTX 4090 described above with the proposed model.

Table 4: Graph size and forward-pass time with and without wall boundary particles, on one Water-3D validation trajectory. Wall-to-wall edges are excluded in every row that has walls. At frame 0 the fluid has not yet reached the container, so no fluid–wall edges exist and the edge count is unchanged.

Frame	Walls	Nodes	Edges	Forward (ms)
0	no	11,040	185,995	44.7
0	yes	35,356	185,995	47.3
200	no	11,040	133,785	32.5
200	yes	35,356	212,619	54.3
400	no	11,040	152,927	36.9
400	yes	35,356	228,561	58.5

Forward is the time of one forward pass of the proposed model, the mean of 20 runs after 5 warm-up runs, on one NVIDIA GeForce RTX 4090.

The node count grows by a factor of 3.2, which is set by the surface area of the container rather than by the amount of fluid. The edge count grows far less, by 49 to 59% once the fluid is in contact with the walls, because wall-to-wall pairs are dropped at graph construction and only fluid–wall pairs survive. The forward pass costs 58 to 67% more per step in that regime. The cost is therefore

dominated by the extra nodes, and it is the price paid for the order-of-magnitude reduction in escape rate reported in Section 5.