

The paper presents a graph neural network model to surrogate the node and edge hydraulic variables in a sewer system.

On a small case study in Virginia, USA, the model shows almost perfect performance across most predicted metrics.

Despite the paper being in a decent state, I have several major concerns regarding the novelty of the paper.

We thank the reviewer for the time and care invested in this review. Your comments have been valuable in prompting us to clarify the presentation of our contributions, to strengthen the empirical support for some components of the model, and to prepare concrete revisions that will improve the manuscript.

That said, we must respectfully disagree with the conclusion that our work lacks novelty. We recognize, however, that aspects of our manuscript did not clearly articulate the scope and implications of our contributions, which may have contributed to this perception. Across the five major concerns, the review frames our contributions as incremental variations of existing approaches. However, as our detailed responses demonstrate, the novelty is not technical tinkering but a re-formulation of the problem itself: developing a rainfall-driven, end-to-end surrogate for stormwater systems that enables applications outside the reach of prior runoff-driven or hydraulics-only models. First, unlike runoff-driven surrogates, our rainfall-driven design unifies hydrologic and hydraulic processes, eliminating reliance on external models and enabling applications that partial surrogates cannot support. Second, our heterogeneous graph formulation is a structural innovation that mirrors the physical distinction between subcatchment nodes and routing nodes, preserving hydrologic integrity and enabling localized interventions such as LID optimization. Third, our physics-guided learning strategy, including the stability loss and pushforward formulation, provides a principled way to balance accuracy and robustness during autoregressive rollout, going beyond existing multi-step losses by explicitly detaching gradients to enforce error recovery.

Taken together, along with the other clarifications and revisions outlined in this response, these elements demonstrate that our framework is not a minor variation of existing models but a distinct step forward. Its novelty lies as much in scope and application as in architecture, creating new system-level capabilities, such as integrated flood modeling and infrastructure optimization, that cannot be achieved with partial surrogates.

With this context, we address each of your comments in detail below. Our responses clarify where our approach differs substantively from prior work, explain the rationale for our design choices, and outline the specific revisions we will make to improve clarity and rigor.

The authors claim that the paper has 5 main novelties, but they are either not novel or the results are not validated enough, as I justify hereafter.

1) The model converts rainfall to hydraulic variables at junctions and conduits. There already exists a paper that models both node and edge variables (Garzon et al., 2024b), despite your paper predicts different variables. The main difference consists in directly taking rainfall as an input rather than the runoff generated by a hydrological model. However, this introduces other issues that are not discussed in the paper, such as including all of the hydrological characteristics of each catchment in the proper training of the model.

The reviewer characterizes our contribution as “taking rainfall as input,” which downplays the substantive distinction between a runoff-driven surrogate and a rainfall-driven surrogate. Prior GNN-based studies emulate only the hydraulic routing given pre-computed runoff; they therefore remain dependent on a separate hydrologic model and do not remove a major computational and operational bottleneck. Our model, in contrast, is end-to-end. That is, it emulates the complete rainfall-to-hydraulics transformation within a single framework. These are fundamentally different problem formulations, and reducing our contribution to a minor input change misrepresents the novelty of our proposed end-to-end approach.

This end-to-end design is important because it enables capabilities unavailable to runoff-driven surrogates: (i) direct coupling with forecast rainfall for real-time operations without invoking a hydrologic pre-processor; (ii) explicit sensitivity to subcatchment attributes (imperviousness, slope, storage, connectivity), which allows evaluation of land-use and LID interventions; and (iii) modular integration with inundation surrogates to support rainfall-to-pipe-to-surface digital twins.

We do not quite understand the reviewer’s point that using rainfall “introduces other issues”. Our framework encodes hydrologic heterogeneity. Static catchment features are explicitly included as node attributes and propagated through the graph, so the model learns rainfall–runoff responses conditioned on local characteristics. This eliminates the need for an external hydrologic model and calibration, increasing portability rather than reducing it.

To make this distinction clear, we will add: (a) expand the related work and methodology sections to clearly contrast runoff-driven surrogates with our end-to-end formulation; and (b) explicitly describe the static hydrologic features included in the model. These clarifications reinforce that our contribution is not a minor variation but an end-to-end surrogate aimed at broader, real-world applications.

2) Use of a heterogeneous GNN: despite using two different types of nodes, i) there is no clear indication of what changes from the GNN perspective other than having different input values, ii) there is no comparison with a baseline to justify the need of heterogeneous nodes, and iii) there is no analysis showing that this representation "enables structured hydrologic representation".

The reviewer questions the value of introducing heterogeneous node types and suggests there is “no clear indication” of what changes beyond input values. This characterization overlooks both the structural and functional importance of the design. Our heterogeneous formulation is not a superficial modification; it enforces a separation between hydrologic and hydraulic processes that a homogeneous architecture cannot capture.

From a modeling standpoint, using distinct node types with distinct feature spaces prevents information dilution. In a homogeneous network, hydrologic descriptors (imperviousness, slope, storage) would have to be artificially assigned to all nodes, forcing meaningless zero-padding for routing-only junctions and outfalls. These non-physical inputs would be propagated through message passing, contaminating the learned functions and blurring the distinction between runoff generation and hydraulic routing. The heterogeneous structure avoids this by restricting hydrologic features to subcatchment nodes, yielding a cleaner and more physically coherent learning task. This is precisely what we mean by “structured hydrologic representation”: the graph is designed to mirror the underlying processes, not just the topology.

From an application perspective, the heterogenous design aligns with how stormwater systems are represented in established models such as SWMM, where only a subset of nodes are linked to

subcatchments. This matters for decision-support use cases. For example, when evaluating LID strategies, changing imperviousness or storage should affect only the relevant subcatchment nodes, not be smeared across the entire graph. A homogeneous architecture would treat these localized interventions as global noise, undermining the model’s utility. Our approach ensures that physical integrity and local interpretability are preserved.

To underscore this point, we will revise both the introduction and methodology sections to make this rationale explicit, clearly framing the heterogeneous formulation as a design that reflects the dual hydrologic–hydraulic processes and preserves localized interpretability for decision-support applications.

3) Physics-guided constraints: penalizing negative values doesn't enforce/constrain them to be positive. There is also no ablation on whether this component actually improves the model's performance.

The central physics-guided element in our framework is the differential consistency loss, which enforces relational hydraulic constraints by preserving depth and inflow differences across connected nodes. This embeds the physics of flow routing directly into the training objective and is particularly well suited to an end-to-end rainfall-to-hydraulics surrogate. This relational constraint goes beyond the localized feasibility checks used in prior GNN-based studies and represents the distinctive physics-guided contribution of our work.

Alongside this, we include a soft penalty on negative values. It is correct that our penalty does not enforce positivity, but that is intentional: we wanted the penalty to serve as a gentle guide toward feasible outputs while maintaining smooth gradients and stable optimization. In contrast, hard enforcement strategies, such as clamping or ReLU activations, guarantee non-negativity but introduce abrupt, non-differentiable operations that disrupt gradient flow, often leading to the well-known dying ReLU problem and destabilizing learning (Lu et al., 2020). Our formulation integrates seamlessly into the loss function, providing stability without sacrificing physical plausibility.

To strengthen clarity, we will revise the manuscript to (a) clearly distinguish between the differential consistency loss (the main physics-guided component) and the soft penalty, (b) explain why soft guidance was chosen over hard enforcement, and (c) provide a comparison showing performance with and without these terms. Together, these clarifications will underscore that our physics-guided formulation is not incidental, but a principled and impactful design choice.

4) Autoregressive forecasting structure: this same approach has been presented in previous papers (e.g., Bentivoglio et al. 2023, Garzon et al. 2024).

Where our contribution lies is in how the autoregressive structure is integrated into our broader framework. In particular, coupling autoregression with our stability-oriented training strategy enables multi-step prediction within an end-to-end rainfall-to-hydraulics surrogate. This design ensures that errors do not accumulate unchecked over long horizons, which is essential for practical deployment in stormwater applications.

To make this clear, we will update the manuscript to frame autoregression not as a standalone novelty but as a necessary component of a stable, application-ready architecture.

5) Pushforward trick: the version presented here is equivalent in terms of equations to the autoregressive approaches mentioned above. The pushforward trick, as described in Brandstetter et al. 2022, is implemented in a different way from the one here in this paper. Even if it was, there is again no analysis on whether this component benefits the training procedure.

We recognize we did not fully describe key implementation details, which created the impression that our approach is equivalent to prior work. Our pushforward method is an adversarial strategy designed to address distribution shift by training the model on its own realistically flawed predictions. At each step, we introduce these predictions back into the sequence and cut the backpropagation gradient from the initial prediction, which forces the model to learn robustness and error recovery rather than simply minimizing one-step errors. This property, often referred to as zero-stability, is central to achieving reliable long-horizon performance. In our implementation, the stability loss is applied across three time steps, which provides stronger regularization and error recovery capacity than a single-step formulation.

This design shares the underlying logic of the pushforward trick introduced by Brandstetter et al. (2022), but our implementation extends it by applying the loss across multiple steps rather than only the immediate next step. That distinction makes the method more effective for stormwater applications where multi-step stability is critical.

It is also important to clarify how our approach differs from curriculum learning strategies. Curriculum methods typically extend the prediction horizon during training, with gradients propagated through the entire trajectory to minimize accumulated error. By contrast, our pushforward strategy introduces perturbations at each step while blocking gradients from flowing through the initial prediction. This ensures the model learns to recover from errors rather than simply avoid them. This principle is fundamentally different from the multi-step-ahead loss of Bentivoglio et al. (2023), which propagates gradients through all time steps, as their own description confirms.

In our revision, we will (a) describe the implementation of the pushforward trick with sufficient technical detail to avoid ambiguity, and (b) include results comparing training with and without the pushforward component to demonstrate its contribution to stability.

One of the novelties of the paper but is not mentioned here is the estimation of flood volumes directly from the predicted hydraulic variables, but it does not justify the paper being published on its own.

We thank the reviewer for this constructive comment. We will highlight this point as we thoroughly and carefully revise and clarify the contributions of our work.

I think that the paper would have to go through a large series of modifications for it to be novel enough to justify its publication.

Because of these concerns, I have to recommend rejection.

General comments:

Introduction:

You mention as main knowledge gap the use of runoff rather than rainfall as a node input.

Despite it being true that you could gain a bit more speed-up from emulating that part as well, you also end up in a new challenge where your model has now to generalize also over different hydrological parameters that can be simply disregarded as an input for the GNN model otherwise.

I didn't see in the experiments any results on how your model would behave for changes in the hydrological characteristics of the node catchments.

This limitation seems to be missing as well from the model limitations later on.

Moreover, there is a paper from Garzon et al. (2024b) that already includes edge-level features.

You should at least compare how you models differ and clarify that there are already examples tackling this gap in the introduction.

We respectfully note that our contribution is not limited to “taking rainfall as an input” but to developing a fully an end-to-end, rainfall-driven surrogate. As explained in our response to your Comment 1 above, this represents a fundamentally different problem formulation from runoff-driven surrogates. Prior models emulate only the hydraulic routing once runoff is pre-computed; our framework directly couples rainfall, hydrologic characteristics, and hydraulic states in a single surrogate.

A key aspect of our design is the inclusion of hydrological parameters (imperviousness, slope, soil type, catchment area, etc.) as node features. This is intentional: these descriptors allow the network to learn how rainfall interacts with physical catchment properties to produce runoff. By encoding this information directly, the model can condition rainfall–runoff transformation on local characteristics, which is essential for eventual transferability across systems with different hydrologic settings. Ignoring these descriptors might reduce inputs, but it would come at the expense of physical realism and the ability to generalize beyond a single study area.

The reviewer suggests that using rainfall inputs introduces a new challenge for transferability, but this issue is not unique to our framework. Any surrogate, whether rainfall-driven or runoff-driven, must be retrained or adapted when applied to a new catchment, since both rely on catchment-specific attributes. Within the study area used in our study, these parameters (e.g., imperviousness, soil type, catchment area) are fixed and do not vary across rainfall events, so variability cannot be tested in this dataset. Their importance emerges only when transferring to new study areas, which is a natural next step beyond this proof-of-concept study. We will revise the manuscript to make this clear and to emphasize that transferability is a broader challenge for surrogate modeling, not a limitation introduced by the end-to-end design.

Finally, with respect to Garzon et al. (2024b), we agree that the introduction can better situate our study in relation to existing work. We will revise the introduction to clarify how our framework differs in scope (end-to-end rainfall-to-hydraulics rather than runoff-to-hydraulics), structure (heterogeneous node design), and application value (supporting decision-relevant tasks such as LID optimization and integration with inundation surrogates).

Section 2:

I don't see the point in having a background section as it currently is.

Consider removing it and integrating it directly in the methodology, since there are already some overlaps in the GNN part and the part on SWMM does not seem to be relevant for the rest of the paper.

We understand the reviewer’s perspective on the structure of the background section. While we believe it is useful to provide context on both GNNs and SWMM, we agree that this material can be streamlined. In the revision, we will integrate the essential details directly into the methodology and remove overlaps to improve flow and conciseness.

Section 3:

Figure 1:

The whole figure needs to be re-designed as it is quite confusing in the current state.

For instance, x_1 , x_4 , etc. are not defined anywhere in the figure;

What should be a vector $\textbf{x}_1$ is written as a scalar;

These vectors (e.g., x_1) seem to have the same input repeated (x_1 present twice, same in the others), though I suppose it has different features;

The message-passing block could be better referenced to the "GN Block" you have above "Aggregation";

You define both $\hat{y}_i$ and $\hat{y}_{ij}$ which seems to indicate the same variable, even if you are referring to two different outputs;

From the figure, it also seems like the decoded node features are given as input to the edge decoder: is it the case? This was not clear in the rest of the paper.

The colouring of the cells inside the figure may seem to help showing that the features are mixing but it also create more confusion, especially after the first mixing.

There is a variable called h_{selected} that appears only in the figure and is unclear.

The caption of the figure should also better explain what is happening in the figure, clarifying the different variables.

I would recommend a simpler design, removing all MLP figures, colored shapes, and case-specific names (x_1 or x_{14}). If you want to leave the latter, consider adding a reference graph with the corresponding node and edge names.

Thank you for the detailed and constructive feedback on Figure 1. We will completely redesign the figure to address all of your comments and improve its clarity. The new design will be a simplified schematic that uses generic, consistent notation, such as clarifying vector components with proper boldface type, and removes confusing elements.

You noted the notation $\hat{y}_i$ and $\hat{y}_{ij}$ was unclear; our intent was to define two distinct outputs for node-level and edge-level predictions, respectively. The revised figure will make this separation visually explicit with clearer labeling. Regarding the decoder, you asked if decoded node features are used as input to the edge decoder. This is not the case, and we will make this clearer in the revised figure. Our model conceptually uses separate decoding paths: final node embeddings are mapped to node predictions, and final edge embeddings are mapped to edge predictions. The redesigned figure will clearly illustrate these two distinct pathways to accurately represent the model's architecture.

line 181: you start defining static and dynamic features without having them introduced before.

You also include xy coordinates as inputs: can you expand more on the implications on transferability of this approach to other case studies? (similarly for example to what is done in Garzon et al. 2024)

We will revise Section 3.1 to introduce and clearly define the two categories of input features (static vs. dynamic) before listing them.

Regarding the use of x–y coordinates: these are included to help the network capture spatial relationships and hydraulic gradients between connected nodes, not to tie the model to a specific coordinate system. In practice, the inputs for any new case study would be derived from system schematics (pipe geometry, node elevations, subcatchment properties) and organized in a SWMM-compatible format that preserves connectivity. The x–y information therefore acts as a relational feature that enhances learning, rather than a barrier to transferability.

That said, rigorous testing of cross-watershed transferability was beyond the scope of this proof-of-concept study. We will explicitly acknowledge this limitation in the revised manuscript and expand the discussion to highlight what would be needed for generalization to other systems, such as multi-catchment training across diverse hydrologic and spatial settings.

line 186: is there a reason why you chose 3 time steps as a input history? How is this related to "support multi-step autoregressive prediction"? You can predict autoregressively even without multiple input time steps in theory.

Yes, the autoregression can in principle be performed with a single historical step. However, in practice, the choice of input window size involves a trade-off between providing enough temporal context and avoiding overfitting on long histories. Prior work (Pfaff et al., 2021, Sanchez-Gonzalez et al., 2020) shows that small windows of 2–3 steps are effective for capturing short-term dynamics without introducing spurious long-range correlations.

Our approach aligns with this principle. A single input step provides only the instantaneous state of the system. For stormwater dynamics, this is insufficient because runoff generation and hydraulic response depend not just on current rainfall but also on short-term accumulation and recent flow conditions. By providing three historical steps, the model can capture this trend and momentum, which are essential for correctly representing how rainfall translates into runoff and routing. At the same time, limiting the window to three avoids the risk of overfitting that arises with longer histories.

Our choice of a three-time-step window therefore serves two purposes. First, it provides the minimal temporal context needed to capture accumulation effects and short-term system momentum, which cannot be represented with a single input step. Second, it aligns with our pushforward training strategy: the model predicts at time t using the true, observed data from $t-2$ and $t-1$, then shifts into autoregressive mode where $t+1$, $t+2$, and $t+3$ are generated recursively from its own, previously generated outputs. As illustrated in Figure 4-b, this method deliberately forces the model to learn from its own "noisy" predictions while gradients are detached from the initial step, which stabilizes optimization. The three-step choice is therefore not arbitrary but reflects both physical and methodological considerations.

We will revise the manuscript to make this rationale explicit so the connection between input window length, physical dynamics, and the pushforward strategy is clear.

line 185: how do you calculate the node inflow? is it determined by the predicted edge flows?

The dynamic input features for our model are sourced from SWMM simulations across diverse storm events. Among these features, 'inflow' refers to the Total Inflow, which represents the combined flow into a node from surface runoff (Lateral Inflow), and all connected upstream conduits. We will revise the manuscript to clarify this point.

line 194: with "globally normalized" do you mean that you create a single scaler for all variables or do you have one for each variable?

By “globally normalized” we mean that each variable (e.g., junction depth, conduit flow) is normalized separately using statistics computed across all events in the dataset. This ensures comparability across events while removing scale disparities and stabilizing the learning process. We will revise the manuscript to clarify this point.

Figure 2 and line 198:

It seems that your model predicts in one go the following three time steps, but in the figure's caption it seems like you first predict $t+1$, then $t+2$, and so on. Which of the two is it? And if it's the first, why, again, choosing 3 as a number of time steps?

Figure 3 seems to clarify this issue as it shows that you do the second. Please change the rest of the paper clarifying that your model predicts only one step into the future, meaning that it's not limited to only 3 steps ahead.

Since your model can predict any number of future time steps, why do you limit your predictions to the same size as the input time steps?

In other works, the two characteristics are independent (e.g., Bentivoglio et al. 2023, Garzon et al. 2024).

Yes, our model is a one-step-ahead predictor that is applied recursively to generate multi-step forecasts. Figure 3 correctly illustrates this autoregressive process. We agree that this point was not made clearly enough in the manuscript, and we will revise the methodology to state explicitly that the model predicts one step at a time, not multiple steps in a single forward pass.

The confusion in Figure 2 arises from how the pushforward training strategy was visualized. The three-step window shown there reflects the training configuration used to implement the pushforward trick, not the model's inherent prediction capability. We will revise the figures and captions so they consistently illustrate the autoregressive structure and avoid this ambiguity.

Regarding the prediction horizon: in our implementation, the number of past input steps and the forecast horizon are independent hyperparameters. In practice, we found a horizon of three steps yielded the best validation performance during hyperparameter tuning, but the framework is not restricted to this configuration. The model is flexible to accommodate different input histories and forecast horizons depending on the application.

line 197: maybe add a reference to Fig. 2, otherwise it was unclear to me how you combine static and dynamic features.

Thanks for your comment. We will clarify more on that and also reference Figure 2.

Figure 3:

Part a is could be a bit clearer: you can for example clarify what are the inputs and outputs since so far they might look the same.

You should also better show which inputs are taken from ground-truth simulations and which ones are predicted by your model, as the update in the red box seems to show that you use ground-truth data as input.

This figure seems to also indicate that there are no overlaps between training windows from $t-p$ to $t+p$. This decreases by a factor p the number of training samples, making the training faster but potentially less effective. Please add some justification for this choice.

Part b: same comment regarding the coloring of ground-truth data as before.

Thank you for your detailed feedback on Figure 3. We will make sure to clarify the figures in the revised version to make this process unambiguous.

Regarding the training windows, our training methodology does employ overlapping windows. We use a sliding window approach to generate training samples, which maximizes the use of the available simulation data. We will explicitly clarify this in the manuscript and ensure the revised figures accurately reflect this process.

Section 3.2:

While it is true that there are no hydraulic paper that consider the pushforward trick, there are other papers that you cite that deal with the same problem using directly a multi-step-ahead loss that generalizes the pushforward trick to multiple time steps ahead (Bentivoglio et al. 2023, Garzon et al. 2024).

Indeed, Eq. 9 is identical to that of Bentivoglio et al. 2023 and Garzon et al. 2024, so please clarify your novelty claims.

Moreover, in the original paper from Brandstetter et al. 2022, the gradients were cut after the first time step, but it seems you are not doing that.

The reviewer is correct that Brandstetter et al. (2022) cut gradients after the first step, and our implementation follows the same principle. During the autoregressive rollout in training, gradients are detached after the initial prediction so that the model is trained to recover from its own imperfect outputs rather than simply backpropagating through a perfect trajectory. This was not made explicit in the manuscript, and we will revise the methods section to state this clearly.

While Equation 9 is formally identical to the loss functions in Bentivoglio et al. (2023) and Garzon et al. (2024), it is crucial to distinguish between their multi-step-ahead loss and our pushforward approach. A multi-step loss (e.g., Bentivoglio et al., 2023; Garzón et al., 2024) propagates gradients through the entire horizon to minimize accumulated error, teaching trajectory perfection. The pushforward trick instead introduces distribution shift deliberately by feeding back noisy predictions while cutting gradients after the first step, thereby enforcing robustness and error recovery (zero-stability). Although both aim to address long-horizon stability, they rely on different principles.

We will revise the manuscript to make these distinctions explicit so that the novelty of our formulation is clear.

Eq 9 and 10: you are missing the subscript $_v$ on both variables y . Also mention at some point that these are mean squared errors.

The reviewer is correct that the subscripts were missing; we will add the “ $_v$ ” to the y variables in both equations and explicitly state that they represent mean squared errors.

Eq. 10: if you are always predicting your outputs based on ground-truth data, is this equivalent to a one-step-ahead loss accumulated over multiple time steps?

The review is correct that Equation 10 can be interpreted as a one-step-ahead loss accumulated over multiple steps. We believe this question arises because such a formulation resembles curriculum or multi-step loss accumulation strategies used in prior work. Our framework, however, differs in how the two loss terms are structured and how gradients are handled.

Real loss (L_{real} , Fig. 4a): At each step, the model predicts the next state using ground-truth observations as inputs (e.g., depth and inflow at $t-1$ and t to predict $t+1$). This is repeated recursively for three future steps ($t+1$, $t+2$, $t+3$). At each step, the prediction is compared directly against the corresponding ground-truth state, and the mean squared errors across all three steps are averaged to define L_{real} . This ensures the network learns to reproduce the true dynamics under perfect input conditions.

Stability loss ($L_{stability}$, Fig. 4b): In this phase, we deliberately introduce noisy inputs to mimic inference-time conditions. First, the model uses observed states at $t-2$ and $t-1$ to predict the state at t . After this first step, the model’s own predictions are recursively fed back as inputs: for example, the pair consisting of $t-1$ and the predicted t ($t-1$, $\hat{t}$) is used to predict $t+1$; then, the pair consisting of $(\hat{t}, \hat{t}+1)$, both model predictions, is used to predict $t+2$; and so on. At each rollout step, the predicted state is compared directly against the corresponding ground-truth state, and the mean squared errors across the rollout horizon are averaged to define $L_{stability}$. Importantly, gradients are detached after the first step, so the model does not backpropagate through the entire trajectory. This forces the model to learn how to recover from its own imperfect outputs, rather than only minimizing accumulated error through a perfect rollout.

Together, these two losses balance accuracy and robustness. L_{real} ensures the model aligns closely with true system dynamics under ideal inputs, while $L_{stability}$ explicitly prepares it for autoregressive deployment by training it to handle noisy, imperfect inputs. This gradient-handling strategy distinguishes our pushforward approach from prior multi-step loss accumulation methods (e.g., Bentivoglio et al., 2023), which propagate gradients through the entire horizon to enforce trajectory perfection rather than error recovery.

We will revise the manuscript to clarify this distinction so that the complementary roles of L_{real} and $L_{stability}$ are unambiguous.

Figure 4:

As for the previous figures, it would be to have a legend that clarifies what each color represents, mainly to highlight which outputs are predicted and which ones are ground-truth.

Also, the top figure includes 3 previous input time steps while figure a and b only 2.

It might make the figure easier to understand if you compressed all static and dynamic features that are always ground-truth into a single block of a more transparent shade.

We appreciate this detailed feedback. As clarified in earlier responses, our model consistently uses a two-step input history for its predictions. The discrepancy in Figure 4 reflects the visualization of the pushforward training setup rather than the model’s actual prediction process. The model begins with two observed inputs ($t-2$ and $t-1$) to predict the current state at time t , and then proceeds autoregressively, feeding its own predictions back as inputs.

In the revision, we will update Figure 4 to eliminate this ambiguity. Specifically, we will (a) add a clear legend distinguishing predicted vs. ground-truth values, (b) correct the input history to consistently reflect the two-step design, and (c) compress static and ground-truth dynamic features into a single block to simplify interpretation.

Section 3.3:

This section and 3.2 should be merged for clarity as they both deal with a loss function.

We will revise the manuscript to merge the two sections.

line 239: penalty term: penalising negative values doesn't "ensure that your model respects hydraulic feasibility", it just helps skewing the results to that direction.

We agree with the reviewer's observation that the penalty functions as a soft constraint, guiding predictions toward physical plausibility rather than strictly enforcing feasibility. As noted in our response to Major Comment 3, this was a deliberate design choice to preserve smooth optimization while still biasing the model toward feasible outputs. In the revision, we will update the text to replace "ensure" with a more accurate term such as "promote" or "encourage," so that the manuscript reflects this behavior precisely.

Some other works, like Palmitessa et al. (2022), directly use a ReLU activation to guarantee that there are no negative values. Did you also try out this approach? Does the presence of this loss term improve the results?

We did not implement the hard constraint approach described by Palitessa et al. (2022). This guarantees feasibility, but applying a similar hard clipping (e.g., ReLU) directly to network outputs has a significant drawback: whenever a prediction is negative, the output is forced to zero and the gradient vanishes. The model therefore receives no corrective signal, making it unable to learn how to avoid producing non-physical predictions. In contrast, our penalty loss acts as a soft constraint. It provides a constant, non-zero gradient whenever the prediction is negative, which continuously nudges the network toward feasible values while still allowing the optimization process to converge smoothly. We opted for this approach because it provides a more stable and informative learning signal, which is especially important in an end-to-end rainfall-to-hydraulics surrogate where robustness depends on recursive predictions.

We will clarify this rationale in the revised manuscript so the distinction between hard and soft enforcement strategies is explicit.

line 247: please use the same notation on flows, inflows, etc. throughout the paper. I think using these symbols (also in the rest of Section 3) makes it clearer to identify which variables you are considering.

We will revise the manuscript to ensure consistent notation.

Eq 13: If you decide to keep the penalty term, with a valid justification, please define it before mentioning it in the loss function.

We will add the following formal mathematical definition for the penalty loss, $L_{penalty}$, before it appears in the total loss function:

$$L_{penalty} = \lambda_d \sum_{i \in V} ReLU(-\hat{d}_i) + \lambda_c \sum_{(i,j) \in E} ReLU(-\hat{d}_{ij})$$

where V is the set of nodes, E is the set of edges, $\hat{d}_i$ is the predicted depth at node i , $\hat{d}_{ij}$ is the predicted depth in conduit (i, j) , λ_d and λ_c are weighting coefficients controlling the penalty strength, and $ReLU(\cdot)$ is the Rectified Linear Unit function, which penalizes only non-physical negative depth predictions.

line 250: you seem to imply that in validation the loss is given by the base term, which comprises both ground-truth and predicted inputs.

Is this the case or are you only considering the "stability" term?

Yes, that is correct. During validation, we compute the base loss, which combines the Real Loss (calculated with ground-truth inputs) and the Stability Loss (calculated with predicted inputs). We will revise the manuscript to make this point explicit so there is no ambiguity.

line 260: why do you also measure the Pearson correlation?

We included the Pearson correlation (r) because it measures an aspect of performance that is not captured by RMSE or NSE. While RMSE quantifies absolute error and NSE evaluates the model's accuracy relative to the observed mean, Pearson's r specifically measures the trend agreement and timing of the predictions relative to the observations.

Consider, for example, a case where the model correctly tracks the rise and fall of a flood hydrograph but systematically over- or underestimates flow magnitudes. In such a scenario, RMSE and NSE would penalize the model heavily for magnitude errors, even though the temporal dynamics are captured very well. Pearson's r , by contrast, would still be close to 1, reflecting that the model successfully reproduces the timing and shape of the hydrograph. This highlights the added value of including r , since it provides insight into temporal agreement that cannot be disentangled from magnitude errors using RMSE or NSE alone.

This distinction is also illustrated in Figure 15, where certain locations show high correlation (r) even when NSE is low, emphasizing how r complements RMSE and NSE in evaluating model performance.

Section 4:

line 285: why did you consider this coastal case study if you then have to adapt the real conditions (pipes with sea water) to a simplified version? Doesn't the model work with presence of water in the system?

The coastal case study was selected because it provided a realistic and complex network that could serve as a high-fidelity testbed. In this initial study, we deliberately simplified the SWMM simulation by excluding tidal backflow effects so that the focus remained on the core research challenge: demonstrating that a GNN can learn both rainfall-runoff generation and network routing directly from rainfall inputs.

This simplification should not be interpreted as a limitation of the GNN's ability. The GNN learns the dynamics that are represented in its training data. Extending the framework to include boundary conditions such as tidal influence is straightforward once the end-to-end rainfall-to-hydraulics

surrogate has been established. We will clarify this rationale in the manuscript and note explicitly that incorporating tidal forcing is an important direction for future work.

Figure 5:

It would help to have an elevation map as well to visualize the slope of the sewer system.

We will modify Figure 5 to include an elevation layer to help visualize the system's topography and slope.

It seems that all SWMM nodes are flooded according to Waze. How did you calibrate the SWMM model then based on these observations (lines 288-289)?

To calibrate SWMM against Waze flood observations, we iteratively adjusted pipe sizes in tidally influenced areas using elevation data, achieving a reasonable match with reported flooded nodes. Although the real system frequently floods due to combined rainfall and tide, this baseline model (of resized pipes) was intended as a simplified yet representative test case for GNN development.

It also seems like there are some disconnected parts in your system. Is it an error in the map or do you model separate parts?

Yes, this is an error in the map, and we will correct and modify it in the revised version.

lines 291-293: What does this sensitivity analysis mean? Is this the variability of the static catchment attributes across all SWMM nodes?

Sensitivity analysis of static catchment attributes (e.g., subcatchment width) confirmed that modest perturbations did not meaningfully alter model outcomes (flooded nodes), which is important given the ungauged nature of the watershed and reliance on binary flood validation via Waze. Finally, only individual rainfall events were simulated, so no consecutive or antecedent rainfall conditions were considered.

line 298: "each event was scaled by factors of 1.2, 1.4, or 1.6." How did you choose which ones to scale with which factor? The final number of events is 300 but you start with 85 events.

Our goal in scaling rainfall events was to enrich the dataset with more intense rainfall scenarios and to expand the sample size to 300 events. We began with 85 original events. To create the remaining 215 augmented events, we applied multiplicative scaling factors of 1.2, 1.4, and 1.6. From the full set of 255 possible augmentations ($85 \text{ events} \times 3 \text{ scaling factors}$), we randomly selected 215 unique combinations of (event, scaling factor). The final dataset therefore consists of 85 original events plus 215 augmented events.

Moreover, it would be useful to show the variability in hyetographs between training, validation, and testing, as, based on the basically perfect results, it looks like there might be some overlap between them.

We will add a new figure or table to the manuscript to address this. It will compare the statistical properties of the hyetographs across the training and testing sets to demonstrate their variability and confirm the independence of the test set.

lines 367-372: these are a repetition of the training details that to me mostly add confusion, as it seemed that now your model was predicting only 3 steps ahead into the future (line 368), while in line 373 it seems that you don't use any ground-truth. Please clarify in case it's the latter.

You are correct that the text as written blurs two separate procedures. We will revise the manuscript to make this distinction explicit.

The first paragraph (lines 367-372) describes the training process. The model is trained with a sliding-window approach. It repeatedly receives a 3-step history of ground-truth data and is trained to predict the next 3 ground-truth steps. This teaches the network short-term dynamics under ideal input conditions.

The second paragraph (starting line 373) explains the inference (or testing) phase. Once trained, the model is initialized with the first 3 observed time steps. From there, it operates autoregressively, predicting one step at a time and feeding its own outputs back as inputs. This process can be continued for any forecast horizon. In inference, ground-truth hydraulic states are not used beyond initialization; rainfall remains the only external input at each future step.

Figure 3 depicts this distinction, and we will revise the text to more explicitly separate training from inference so there is no ambiguity about how predictions are generated.

In the results section, do you consider dry-period events for the simulations? If so, can you tell how much they affect the NSE values?

We did not explicitly separate wet and dry periods during training or evaluation. Instead, the model was trained on sliding windows with shuffled mini-batches, ensuring that both low-flow and active periods were naturally included. Evaluation was conducted over full events using standard autoregressive rollout.

For context, the regime-splitting approach referenced in prior work relies on a 1 mm rainfall depth threshold over a 9-step window, with rebalanced sampling between wet and dry regimes. Our history length is shorter (3 steps rather than 9), so such explicit stratification was not necessary in our setup.

For clarity, we will note this in the revised manuscript and emphasize that the reported NSE values reflect model performance across entire events, without regime-specific stratification.

Section 5.3:

Why do you test the flooding performance only on a single test scenario?

The flood performance analysis focused on a single, high-intensity storm, which we selected as a representative event to rigorously test the model's flood detection capabilities under challenging conditions. We note, however, that results for multiple rainfall events are already included in other sections of the manuscript. For the sake of brevity, we initially presented only one scenario here, but in the revised version we can add complementary results summarizing flood detection metrics (CSI, precision, and recall) across several additional storm events from our test set to demonstrate that the model's strong performance is consistent.

Figure 13:

It should include a comparison with the ground-truth flooded nodes according to SWMM, which you compare against.

We will modify Figure 13 to include a direct comparison with the ground-truth flooded nodes from the SWMM simulation.

lines 468-469: please add a reference for this claim.

We will revise the manuscript to support this claim with references from the broader GNN literature (e.g., Horie and Mitsume, 2022).

Section 5.3.2:

This section on limitations doesn't address one important limitation that you introduce with your approach, i.e., the range of hydrological parameters that you now should be modelling in place of SWMM.

It would be also useful to add some insights on the transferability of this model to other case studies. This is an important observation, and we will expand the limitations section to discuss it explicitly.

In a stormwater system, hydrological parameters (e.g., subcatchment area, slope, imperviousness) are fixed once the model is calibrated and do not vary across rainfall events within the same catchment. Therefore, in the present study, these parameters remain constant across simulations.

Their importance emerges when considering transferability to other catchments, where hydrologic properties differ. However, this issue is not unique to an end-to-end surrogate. Even runoff-driven surrogates that emulate the hydraulic routing component are not directly transferable without retraining on new catchment data. Transferability is therefore a broader challenge for data-driven surrogates not a limitation created by our end-to-end formulation.

We will revise the manuscript to make this point clear and to emphasize that improving generalizability requires training on data spanning multiple catchments with diverse hydrological parameters. Such a strategy would enable the model to capture transferable relationships between catchment properties and hydraulic response, rather than fitting a single system in isolation.

Section 5:

At some point of the paper, you should point out the computational times needed for the model to train and test, since one of the main drivers of your research is speed.

We will add a new subsection to the "Results" section detailing the computational performance of our model. This will include the total training time, the average inference time per storm, and a direct speed-up factor compared to the SWMM model.

Section 6:

Please remove the whole section as it does not add any relevant information to the paper.

It also resembles a lot the interactive dashboard provided by Garzon et al. (2024) (https://github.com/alextrmo0205/SWMM_GNN_Repository_Paper_version).

We will remove Section 6 from the manuscript.

Other comments:

line 95: there is a reference error

Thank you for raising this issue. We will fix this in the revised manuscript.

line 382: please reference fig 10 and then 11, and not viceversa.

Thank you for raising this issue. We will fix this in the revised manuscript.

Reference:

- Lu, L., Shin, Y., Su, Y., & Karniadakis, G. E. (2019). Dying relu and initialization: Theory and numerical examples. *arXiv preprint arXiv:1903.06733*.
- Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., & Battaglia, P. (2020, October). Learning mesh-based simulation with graph networks. In *International conference on learning representations*.
- Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., & Battaglia, P. (2020, November). Learning to simulate complex physics with graph networks. In *International conference on machine learning* (pp. 8459-8468). PMLR.
- Garzón, A., Kapelan, Z., Langeveld, J., & Taormina, R. (2024). Transferable and data efficient metamodeling of storm water system nodal depths using auto-regressive graph neural networks. *Water Research*, 266, 122396.
- Bentivoglio, R., Isufi, E., Jonkman, S.N. and Taormina, R., 2023. Rapid spatio-temporal flood modelling via hydraulics-based graph neural networks. *Hydrology and Earth System Sciences*, 27(23), pp.4227-4246.
- Brandstetter, J., Worrall, D. and Welling, M., 2022. Message passing neural PDE solvers. *arXiv preprint arXiv:2202.03376*.
- Horie, M., & Mitsume, N. (2022). Physics-embedded neural networks: Graph neural pde solvers with mixed boundary conditions. *Advances in Neural Information Processing Systems*, 35, 23218-23229.