Reviewer #3

This paper presents an interesting study that applies graph neural networks (GNNs) to model discharge over a river network. The methodology is sound, the paper is well written, and the structure is clear. This could be a very important contribution to the rainfall–runoff literature.

Recommendation: minor revision. Most of my comments are about clarity and reporting. However, a few items ***could become major*** if the justification is weak or if the implementation is not as intended: (a) whether the baseline LSTM comparison is fair (retrained vs "extracted"), (b) whether a 180-day input window is sufficient for the target processes and basin scale, and (c) whether the forcing selection (especially using soil moisture as an input) is appropriate and clearly framed/justified.

We sincerely thank Reviewer #3 for the careful reading of the manuscript and for the constructive and insightful comments. Below, we address each comment in detail.

General comments:

1. The title can be more specific. For example: "A GNN routing module is all you need for **routing** LSTM rainfall–runoff models …" or "… for **accounting for routing** in deep learning-based rainfall–runoff models."

   A GNN Routing Module Is All You Need for LSTM Rainfall–Runoff Models

We thank the reviewer for this helpful suggestion. We considered making the title more specific; however, we chose to retain the current wording to avoid an overly long and narrowly scoped title. The proposed examples would explicitly reference particular model components (e.g., LSTM), whereas the primary intent of the title is to highlight the broader message of the study: the importance of explicitly representing routing in deep learning–based rainfall–runoff modelling.

2. Please expand the literature review on existing GNN applications in hydrology in the Introduction (around lines 68–70). The current text mainly states that prior work does not use GNNs to explicitly model routing.

We thank the reviewer for this suggestion. We have expanded the literature review as follows:

"Several recent studies have explored GNNs in R-R modeling, treating them as spatiotemporal modules within DL frameworks and highlighting their potential. These models typically combine GNNs with LSTMs or other recurrent architectures to capture both spatial and temporal dynamics. For example, Sun et al. (2022) utilized GNNs to capture physics-based connectivity, demonstrating that graph-based data fusion can serve as an effective surrogate for process-based models. Similarly, Deng et al. (2023) addressed the non-Euclidean structure of river networks using spatiotemporal graph convolutions to capture upstream-downstream correlations. Beyond surface water, Gai et al. (2023) applied GNNs to simulate spring discharge by modeling the complex subsurface connectivity of karst systems. More recently, Wang et al. (2025) showed that optimizing graph topologies, specifically transforming tree-like networks into dense graphs can accelerate flood warnings by capturing long-range dependencies. These models typically combine GNNs with LSTMs or other recurrent architectures to capture spatiotemporal dynamics, with a primary focus

on improving representations of spatial variability in inputs or learning latent inter-basin correlations, rather than explicitly modeling the flow-routing process along river networks."

3. The model uses three forcings: precipitation, soil moisture, and air temperature (lines 100–102). Why is soil moisture treated as a meteorological forcing (or as a state / reanalysis product)? Please justify this choice, and explain why other variables (e.g., wind, radiation) are not included, as they are commonly used in other LSTM studies.

We treat soil moisture as an input because it acts as the primary physical control on runoff generation, particularly for flood events. In the context of the Upper Danube, runoff is heavily influenced by the antecedent wetness condition of the catchment. By explicitly providing the model with the soil moisture state (derived from the ERA5-Land reanalysis), we allow the LSTM to directly map the basin's saturation level to discharge. This approach aligns with the growing availability of high-resolution soil moisture products, particularly from satellite observations, whose spatial and temporal capabilities are rapidly expanding.
Our decision to exclude wind speed and solar radiation is based on the fact that their hydrological signal is already implicitly encoded within the ERA5-Land soil moisture variable we utilize. Moreover, the primary objective of this study is to compare the proposed LSTM–GNN framework with a baseline LSTM using identical input information. The inclusion of additional dynamic variables will be explored in future work.

Following sentences have revised:
"Three daily meteorological and hydrological variables provided in the LamaH-CE dataset including precipitation, soil moisture (fraction of water in topsoil layer 0 to 100 cm depth), and 2 m air temperature are derived from the ERA5-Land reanalysis (Muñoz-Sabater et al, 2021) and serve as the dynamic inputs for the R-R model."

4. The introduction to the different GNN architectures should be explained in more detail, since this is a key selling point. Only listing names and references may not be sufficient for many readers. A short, verbal description of each architecture (or a simple summary in the Supporting Information) would help.

Following text added for Supporting Information:

GNN Architectures for River Routing

To explicitly model basin-scale runoff routing, we evaluate four distinct Graph Neural Network (GNN) architectures. Each architecture defines a different mechanism for aggregating hydrological information from upstream subbasins and propagating it downstream along the river network. Let $h_i^{(l)} \in \mathbb{R}^d$ denote the latent feature embedding of subbasin i at GNN layer l, $\mathcal{N}(i)$ the set of upstream neighbors of node i, and $\sigma(\cdot)$ a nonlinear activation function (e.g., ReLU). All four architectures share the same LSTM-generated initial node embeddings $h_i^{(0)}$ (Section 3.1.1) and differ only in how they propagate information through the river network graph during the routing phase.

1. Graph Convolutional Network (GCN)

The Graph Convolutional Network (GCN; Kipf & Welling, 2016) approximates a spectral convolution on the river network graph. Information is propagated by averaging feature representations from upstream neighbors, weighted by the graph's normalized connectivity

structure. The update rule is given by:

$$h_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}(i)\cup\{i\}} \frac{1}{\sqrt{\tilde{d}_i\tilde{d}_j}} W^{(l)} h_j^{(l)}\right),$$

where $W^{(l)}$is a learnable weight matrix and $\tilde{d}_i$denotes the degree of node i, including self-loops. From a hydrological perspective, GCN assumes that the influence of upstream subbasins is determined primarily by the network topology, treating all tributaries as having a structurally fixed contribution weight normalized by node degree. This symmetric normalization ensures that nodes with many upstream connections (e.g., downstream confluences) do not receive disproportionately large signals, simulating a simplified routing scenario where flow from each tributary contributes proportionally based on network structure rather than hydrological state.

2. Graph Attention Network (GAT)

The Graph Attention Network (GAT; Veličković et al., 2017) extends GCN by introducing a learnable attention mechanism that assigns adaptive weights to upstream neighbors. This is particularly relevant for river routing, where tributaries may contribute unequally depending on their current hydrological conditions. The node update is defined as:

$$h_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}(i)\cup\{i\}} \alpha_{ij}^{(l)}\ W^{(l)} h_j^{(l)}\right),$$

where the attention coefficient α_{ij}represents the learned importance of upstream subbasin $j$to subbasin i. These coefficients are computed as:

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(a^{\top}[Wh_i \parallel Wh_j]\right)\right)}{\sum_{k\in\mathcal{N}(i)\cup\{i\}}\exp\left(\text{LeakyReLU}(a^{\top}[Wh_i \parallel Wh_k])\right)},$$

where $\parallel$denotes vector concatenation and a is a learnable attention vector. Hydrologically, GAT allows the model to dynamically emphasize dominant upstream signals (e.g., a major tributary experiencing a flood event) while down-weighting less influential contributions, providing a flexible representation of flow routing.

3. GraphSAGE

GraphSAGE (Hamilton et al., 2017) is an inductive GNN framework that explicitly separates neighbor aggregation from node update. It summarizes upstream information using a chosen aggregation function and then combines this summary with the local node state. The update rule is:

$$h_i^{(l+1)} = \sigma\left(W^{(l)} \cdot \text{CONCAT}\left(h_i^{(l)}, \text{AGG}\left(\{h_j^{(l)} \mid j \in \mathcal{N}(i)\}\right)\right)\right),$$

where AGG denotes an aggregation operator (e.g., mean pooling). In the context of river routing, GraphSAGE first aggregates upstream runoff information to represent total incoming flow conditions and then fuses this information with the local subbasin state, enabling flexible and scalable routing representations.

4. Chebyshev Spectral GCN (ChebNet)

The Chebyshev Spectral Graph Convolutional Network (ChebNet; Defferrard et al., 2016) extends spectral graph convolutions by approximating filters using Chebyshev polynomials of the graph Laplacian, avoiding costly eigen decomposition. The update rule is given by:

$$h^{(l+1)} = \sigma\left(\sum_{k=0}^{K-1} \theta_k^{(l)}\, T_k(\tilde{L})\, h^{(l)}\right),$$

where $T_k(\cdot)$denotes the Chebyshev polynomial of order k, $\tilde{L}$is the scaled normalized Laplacian of the river network, and $\theta_k^{(l)}$are learnable coefficients. By controlling the polynomial order K, ChebNet explicitly defines the spatial receptive field of routing, allowing information to propagate across multiple upstream subbasins within a single layer. This makes ChebNet particularly suited for capturing long-range upstream dependencies in large river basins.

5. Baseline comparison: lines 193–194 mention a baseline LSTM. Is this LSTM re-trained / fine-tuned as a standalone model, or directly extracted from the LSTM-GNN framework? If it is extracted without proper re-training, this could bias the comparison, which *could* be a major problem of the paper. Please clarify.

Thank you for raising this critical methodological concern. We can confidently confirm that our comparison is methodologically sound and unbiased.

The baseline LSTM model is independently trained from scratch as a standalone model. It is not extracted from the LSTM-GNN framework without retraining.

Our comparison is fair and methodologically rigorous for the following reasons. First, the baseline LSTM and the LSTM–GNN models were trained in fully independent training runs, with separate model initialization, training loops, and hyperparameter optimization. Second, both models share the same architectural foundation, specifically an identical LSTM configuration and static feature integration. Third, both models were trained using the same training data and identical train–validation–test splits. Fourth, the same optimization procedure was applied to both models, including the use of the MSE loss function and the Adam optimizer. Finally, each model was trained to convergence with weights optimally tuned for its respective architecture: the baseline LSTM was optimized for direct discharge prediction, whereas the LSTM–GNN was optimized for runoff generation followed by explicit GNN-based routing.

Proposed Manuscript Clarification:

To eliminate any ambiguity, we will revise Section 3.1 as follows:

"The best-performing LSTM–GNN configuration is compared against a baseline LSTM model that is independently trained from scratch as a standalone model. The baseline uses the same LSTM architecture and static feature integration as the LSTM component within the LSTM–GNN framework, but replaces the GNN routing module with a direct linear output layer for discharge prediction. Both models are trained independently using identical training data, loss functions, and optimization procedures, ensuring a fair comparison in which the only difference is the presence or absence of explicit spatial routing."

6. Lines 130–131: why is a 180-day sliding window used? This may be short for capturing

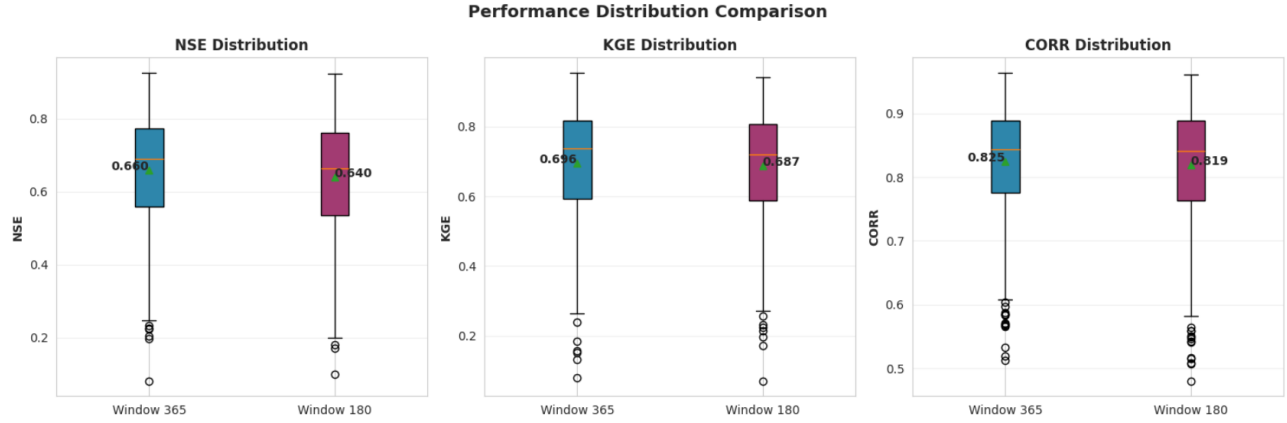
annual-scale dynamics of catchments. Please justify this choice and, if possible, add a sensitivity test (e.g., 365 days or longer).

We appreciate the reviewer's question regarding the choice of the 180-day input window. To address this systematically, we conducted a sensitivity analysis comparing window sizes of 180 and 365 days.

For fair comparison, both models were trained with identical sample sizes (extracted from the 365-day window dataset to ensure no temporal gaps).

Based on these results, we selected the 180-day window as optimal because the 365-day window provides only a marginal improvement in NSE (+0.02) but requires substantially more GPU memory (~2× increase), limiting batch size and training scalability. For operational deployment and future extensions, the 180-day window is more practical.

Please note that the sample counts in this sensitivity analysis differ from those reported in the main paper results. This is because the sensitivity analysis required creating a no-gap dataset from the 365-day window to ensure fair comparison, whereas the main paper uses all available samples from the optimal 180-day window configuration.



Specific comments:

1. Line 161: please check the notation for concatenation of two vectors.

   Revised.

2. Line 169: "*which can be defined in different ways to investigate the impact of river network representation*", please add more details (what are the different definitions considered?).

We have modified the paragraph in the paper as follows:

"The connectivity is encoded in an adjacency matrix $A \in \mathbb{R}^{n \times n}$*, which can be defined in different ways to investigate the impact of river network representation, including binary connectivity (*$A_{ij} = 1$ *for connected subbasins), inverse distance weighting (*$A_{ij} = 1/d_{ij}$*, where* d_{ij} *is the Euclidean distance), or inverse travel-time weighting. In this study, we adopt a directed inverse travel-time–weighted adjacency, where each entry is defined as* $A_{ij} = 1/travel\_time_{ij}$ *if water flows from subbasin* i *to subbasin* j*, and* $A_{ij} = 0$ *otherwise. Travel time is estimated using time-of-*

concentration calculations based on the Kirpich equation (Kirpich, 1940).”

3. Line 171: please clarify whether $A_{j,i}$ is non-zero whenever $A_{i,j}$ is non-zero (i.e., whether the river connections are directional or symmetric).

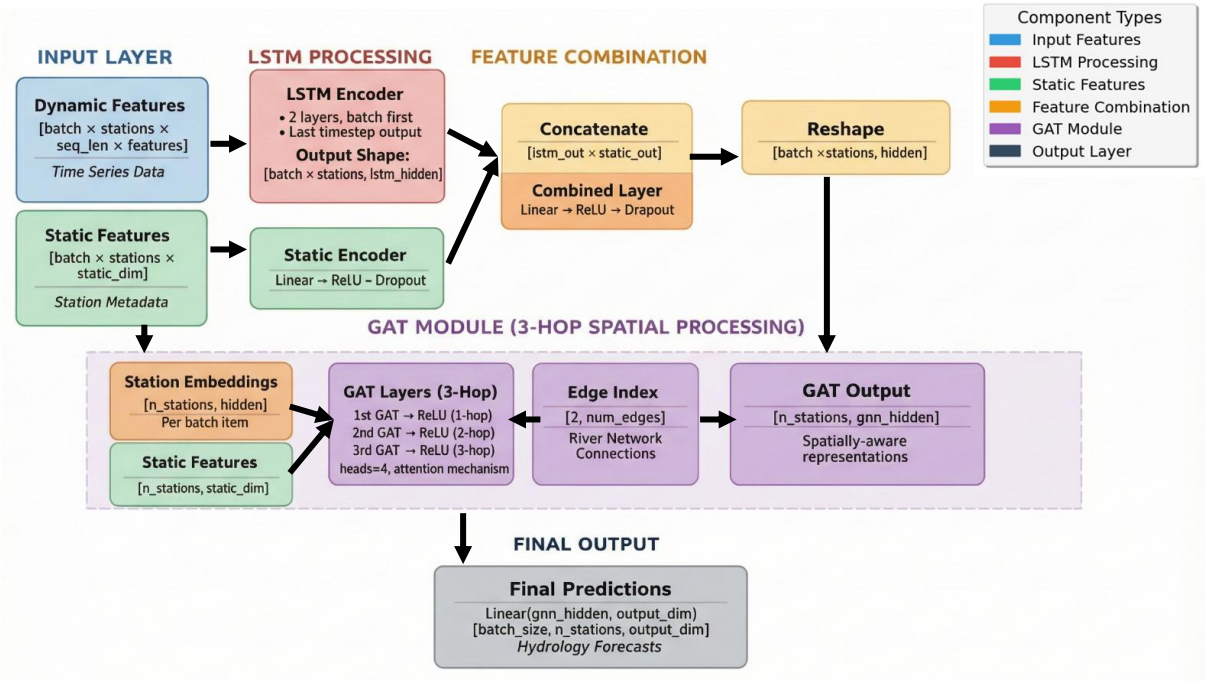
   The adjacency matrix A is directional.

4. Table 1: please improve table formatting; the square-root notation is not clearly visible.

   We have changed the font.

5. Figure 2: please improve aesthetics/readability; arrows and text overlap and are hard to read.

   Revised as follows:



6. Figure 3: in the caption, “m3/s” should use a superscript: m^3/s.

   Revised.

7. Line 240: “NSELST-GAT-NSELSTM” — the mathematical notation is unclear. Please revise for readability.

   Revised.