Towards a Fully Distributed Multivariable Hydrological Model using Graph Neural Networks

A thesis report

submitted in partial fulfillment of the requirements
for the degree of Master of Science in Civil Engineering by

Peter Isaäc Jonathan Nelemans

to be defended publicly on Monday 26th of February 2024 at 11:00.

Delft University of Technology | Faculty of Civil Engineering and Geosciences

Student number		4656539	
Thesis committee	dr.ir.	R. Taormina	TU Delft
	dr.ir.	M. Hrachowitz	TU Delft
	ir.	R. Bentivoglio	TU Delft
	dr.ir.	A. Meshgi	Deltares
	dr.ir.	J. Buitink	Deltares
	ir.	R. Dahm	Deltares

Preface

This is the report of my thesis for the Master Water Management, track Water Resources Engineering, at the Faculty of Civil Engineering, Delft University of Technology. I did my thesis at Deltares, Delft, in the department of Catchment and Urban Hydrology, within the unit of Inland Water Systems. The topic of my thesis was fully distributed multivariable hydrological modelling with Graph Neural Networks.

This thesis marks the end of my master, and with that, the end of my time as a student. It's been an absolutely amazing 6.5 years; I've met many, many different people, learned an incredible amount of interesting stuff, and done so much amazing things. I am eternally grateful to those who made it possible.

I would like to express my deepest gratitude to Riccardo, who made this entire project possible. You always had time for me, supported me, guided me, and looked out for me. You went far above and beyond what could be expected of a thesis supervisor, and I will never forget it. From the bottom of my heart I would also thank Roberto. You were always available to answer my many, many questions, help me with my code, explain GNNs, or help me to get DelftBlue running. You're support, both on a technical and personal level, was invaluable. I would also like to thank Markus, not only for the supervision and the thought-provoking feedback, but also for sparking my interest in hydrology and hydrological modelling in the first place. Your energetic and engaging lectures is what drew me into this world; something I would have never considered just a few years ago.

I would also sincerely like to thank my supervisors at Deltares: Joost, Ali, and Ruben. We've had many meetings, and you were always willing to listen to me, help me with any issues, provide me with input and new ideas, and answer my many questions. Thank you for helping me make this transition from student to working life; I'm looking forward to being your colleague. A thank you is also in order to all other colleagues and fellow graduate students at Deltares, with of course a special shoutout to Sergio, with whom I could share this journey through the world of GNNs.

Then there are also a great deal of people who weren't directly involved in this thesis, but to whom I owe more than I can describe in a hundred thesis reports. My parents, who have been and always will be my greatest example, and to whom I owe everything that I have accomplished and ever hope to accomplish. My siblings, Arnout, Suzanne, and Marnix, with whom I share countless cherished memories from everything we have experienced together. The friends I've made over the years and who've been there through thick and thin: Tom, Robbert, Jos, Mark, Timon, Kees, Volta2Share, het Schip, Frikadelleneters, Vrøgd, YERP, and many, many more. All my teachers, who not only had an invaluable contribution to my education, but who also helped me grow as a person. And last, but certainly not least, the light of my life, my wife Saskia. The kindest and sweetest person to ever exist, and whose constant love and support is what has kept me going.

Peter Nelemans
Den Hoorn, 18th of February 2024

Towards a fully distributed multivariable hydrological deep learning model with graph neural networks

Peter I.J. Nelemans[1, 2], **Roberto Bentivoglio**[1], **Joost Buitink**[2], **Ali Meshgi**[2], **Markus Hrachowitz**[1], **Ruben Dahm**[2], and **Riccardo Taormina**[1]

[1]Department of Water Management, Faculty of Civil Engineering and Geosciences, Delft University of Technology, Delft, The Netherlands
[2]Department of Catchment and Urban Hydrology, Unit of Inland Water Systems, Deltares, Delft, The Netherlands

Correspondence: Peter I.J. Nelemans (Peter.Nelemans@deltares.nl)

Abstract. Fully distributed hydrological models take into account the spatial variability of a catchment, and allow for assessing its hydrological response at virtually any location. However, these models can be time-consuming when it comes to model runtime and calibration, especially for large-scale catchments. Meanwhile, deep learning models have shown great potential in the field of hydrological modelling, but a multivariable, fully distributed hydrological deep learning model is still lacking. To address the aforementioned challenges associated with fully distributed models and deep learning models, we explore the possibility of developing a fully distributed multivariable deep learning model by using Graph Neural Networks (GNN), an extension of deep learning methods to non-Euclidean topologies. We develop a surrogate model of wflow_sbm, a fully distributed, physics-based hydrological model, by exploiting the similarities between its underlying functioning and GNNs. The GNN model uses the same input as wflow_sbm: gridded static parameters based on physical characteristics of the catchment and gridded dynamic meteorological forcings. The GNN model is trained to approximate wflow_sbm outputs, consisting of multiple gridded hydrological variables such as streamflow, actual evapotranspiration, subsurface flow, saturated and unsaturated groundwater storage, snow storage, and runoff. Our results show that the GNN model accurately predicts multiple hydrological variables in unseen catchments (median KGE=0.76), and can serve as an emulator of wflow_sbm with a shorter runtime. We furthermore demonstrate how the GNN model can function up to a prediction horizon of a full year, using physical system states to account for system memory, as well as a curriculum learning strategy combined with a multi-step ahead loss function during training. Overall, this study contributes to the field of fully distributed modelling using a deep learning approach.

1 Introduction

Hydrological models play a pivotal role in understanding and managing water resources by simulating the complex interactions within hydrological systems, aiding in forecasting, planning, and decision-making processes. To do so, these models use data on (meteorological) forcings and the physical characteristics of the system as input, and predict the hydrological response. Hydrological models can be lumped, semi-distributed, or fully distributed. Lumped models consider the catchment as a single unit, assuming all input and output to be uniformly distributed (Beven, 2012). Semi-distributed models divide the catchment into multiple subcatchments based on relevant characteristic (Refsgaard, 1996). Fully distributed models go a step further and discretize the catchment into a grid, enabling a detailed representation of a catchment's spatial variability (Chen, 2019). Furthermore, they allow for an assessment of the hydrological response of a catchment not only at the outlet, but at every grid cell within the simulated region. (Francés et al., 2007).

1.1 Limitations of fully distributed models

Because of their more detailed representation of the catchment compared to lumped and semi-distributed models, fully distributed generally have more parameters (Beven, 1989). As a result, the runtime if often also longer and calibration is more difficult (Khakbaz et al., 2012). Especially for large-scale catchments, the runtime can be problematic if one wants to model many different scenarios or longer time periods (Gichamo et al., 2020).

Nevertheless, fully distributed hydrological models are favored for their versatility and applicability across various domains, such as flood prediction (e.g. Garrote and Bras, 1995; Liu et al., 2005; Blöschl et al., 2008; Mendoza et al., 2012; Chen et al., 2017), drought prediction (e.g. Liuzzo et al., 2009; Mishra and Singh, 2011; Zhao et al., 2016; Ameli and Creed, 2019), groundwater resources management (e.g. Seibert et al., 1997; Kollet and Maxwell, 2006; Seo et al., 2018; Dai et al., 2021), sediment management (e.g. Hui et al., 2014; Giardino et al., 2018), assessing evaporation, transpiration, and soil moisture (e.g. Barling et al., 1994; Chen et al., 2005; López et al., 2017; Wanders et al., 2014), water quality management (e.g. Richards et al., 1996; Rode et al., 2010; Gao and Lo, 2015) and snow modelling (Dunn and Colohan, 1999; Bhatti et al., 2016; Dong, 2018; van Verseveld et al., 2022). Given the wide range of applications for fully distributed, and given their long runtimes, the need exists for a rapid implementation of a fully distributed model.

1.2 Fully distributed multivariable modelling with deep learning

Deep learning (DL) models, a type of data-driven model based on artificial neural networks, have gained increasing interests in hydrological modelling (Shen, 2018; Sit et al., 2020). Studies investigating the use of deep learning models within hydrology have shown promising results (Xu and Liang, 2021). Various types of deep learning models, such as Long-Short-Term Memory (LSTM) models, Convolutional Neural Networks (CNN), Transformers, Attention Models, Generative Adversarial Networks (GAN), and Graph Neural Networks (GNN) have successfully been used for predicting different hydrological variables, such as water quality, streamflow, water temperature, groundwater, floods, and soil moisture (Tripathy and Mishra, 2024; Sit et al., 2022). Despite all the advances made in the field, so far no fully distributed multivariable deep learning model has been developed.

Multivariable models provide several advantages over single-variable models. Firstly, a models capacity to accurately predict multiple variables is a confident indicator that the model captures the occurring processes in the catchment well. In contrast, a single-variable model may predict its one variable with great accuracy, but it may do so for the wrong reasons (Beldring, 2002). Secondly, multivariable deep learning models may be less of a black box compared to their single-variable counterparts. Contrary to physics-based models and conceptual models, deep learning models yet provide little understanding of the catchment and their parameters and equations are difficult to interpret (Reichstein et al., 2019). In recent years, efforts have been made to improve the interpretability of deep learning models. Such efforts include investigating model states (Kratzert et al., 2019), differentiable models (Bindas et al., 2022), and using neural ODE models (Höge et al., 2022). Developing a multivariable deep learning model can contribute to this effort. The internal parameters and equations of a multivariable deep learning model may yet have litte physical meaning. However, if multiple storages and fluxes within the catchment can be assessed, the behaviour of the catchment can be understood more easily.

Given the potential of DL models, the use of fully distributed in a wide range of applications, and the advantages of multivariable models over single-variable models, there is an opportunity to develop such a fully distributed multivariable deep learning model.

1.3 Fully distributed modelling with GNNs

This paper addresses the aforementioned issues associated with fully distributed models and deep learning models with graph neural networks (GNN). They are a type of deep learning technique capable of operating on non-Euclidean data (Zhou et al., 2020). A graph consists of nodes, which are connected to each other by edges. Each node has properties known as the node embeddings. In a GNN, nodes can exchange information with each other and subsequently update their embedding via a process called message-passing. We hypothesize that GNNs are suitable for hydrological modelling, because their functioning offers the possibility to design a model that mimics the behaviour of catchments. Different parts of the catchment can be represented by different nodes, and these different parts are connected to each other via the river network, just as the nodes are connected to each other by edges. Furthermore, in a catchment an event occurring in one region can influence also other regions of the same catchments, i.e. snow melt in an upstream part of the catchment may lead to increased discharge downstream. Similarly, the embeddings of one node can be influenced by another node via message-passing. The capacity of a GNN to update the embeddings of all nodes renders it a highly suitable candidate for fully distributed modeling. Furthermore, the fact it can update multiple embeddings at a single node, means it is capable of predicting multiple variables. We therefore hypothesize that GNNs are a viable candidate for a fully distributed, multivariable, hydrological deep learning model. We furthermore hypothesize GNNs can achieve a significant speedup compared to traditional fully distributed models, as GNNs have been shown to be fast emulators (e.g. Bentivoglio et al., 2023; Choi and Kumar, 2024).

Additionally, as a GNN can exploit the inductive bias of a graph to generalize to unseen graphs (Yang et al., 2023), they should be well-suited for generalizing to unseen catchments. This spatial transferability implicates the model can be utilized outside the geographic domain it was originally trained on. It is particularly relevant for prediction in ungauged catchments (Hrachowitz et al., 2013), and furthermore considered a more stringent test than temporal transferability (Klemeš, 1986; Merz and Blöschl, 2004; Parajka et al., 2005), even though the latter is more common (Gao et al., 2016). Overall, spatial transferability is seen as a more confident indicator of accurate representation of the hydrological processes at play (Blöschl et al., 2011; Gupta et al., 2014). This is especially relevant for deep learning models, given their black-box nature and the fact that little to no physics is usually embedded (Li et al., 2022).

1.4 Related studies on fully distributed modelling with GNNs

GNNs have already been used successfully for fully distributed hydrological modelling. For instance, Sun et al. (2022) presented a GNN-based model for fully distributed operational streamflow forecasting, employing a pre-training phase on a physics-based model followed by fine-tuning with observational data. Similarly, Jia et al. (2021b) proposed a fully distributed physics-guided recurrent graph model capable of predicting both streamflow and water temperature, integrating pre-training on a physics-guided model alongside observational data assimilation. Xiang and Demir (2022) presented a physics-informed fully distributed model based on graph neural networks to predict runoff. (Liu et al., 2022) developed a graph deep neural network to predict the lumped streamflow seven days ahead. Additionally, Bai and Tahmasebi (2023) tilized a GNN to forecast groundwater levels across multiple wells in Canada. While these initial works show promising results for GNNs, none of the developed models were tested for generalization to unseen catchments.

We would also like to highlight two papers that did not employ GNNs, but that are worth mentioning nonetheless. (Maxwell et al., 2021) trained different types of CNNs on an integrated hydrologic model to fully distributed the pressure head in a fully distributed fashion. Although the CNNs predicted only a single variable in a 25 by 25 meter synthetic catchment, they were capable of producing accurate results in unseen catchment. (Tran et al., 2021) proposed a predictive recurrent neural network, which was also trained on an integrated hydrological model. They showed their proposed model is capable of multivariable, fully distributed hydrological modelling, but did not test on unseen catchments. Hence, we draw the conclusion that, to the best of our knowledge, there exists no fully distributed multivariable hydrological model capable of generalizing to unseen catchments.

The objective of this research is to explore the potential of GNNs for the task of fully distributed multivariable hydrological modelling. This objective can be subdivided into three key facets. First, we aim to assess the performance capabilities of GNNs. Second, we investigate their runtime efficiency. And third, we seek to identify suitable model architectures and training configurations. To address these questions, we develop a GNN-based surrogate model of wflow_sbm, a physics-based fully distributed hydrological model developed at Deltares (van Verseveld et al., 2023). By training the GNN model on a physics-based model, we mitigate some challenges arising from limited observation data availability. Furthermore, we hypothesize that by training the GNN model on a physics-based model, it can learn the embedded physics, and is better capable of generalizing to unseen catchments compared to models directly trained on observations (Jia et al., 2021a). The GNN is trained to approximate ten gridded output variables from wflow_sbm, covering different hydrological processes such as streamflow, snow pack, subsurface flow, saturated and unsaturated water storage, runoff, and actual evapotranspiration. The model is tested in unseen catchments, with performance assessed through a diverse ensemble of metrics. The remainder of this paper is structured as follows. Section 2 details the theory behind graphs and graph neural networks. Section 3 illustrates how to use GNNs specifically for the task of hydrological modelling. Section 4 describes the experimental setup. In Section 5 the results are shown. Finally, Section 6 discusses the results and suggests a pathway for future work.

2 Theoretical background

This section gives a brief introduction into graph theory and Graph Neural Networks, including the relevant mathematical notations. We refer to the work of Sanchez-Lengeling et al. (2021) for a more general and complete introduction into graph theory and GNNs, and to the work of Daigavane et al. (2021) for a complete overview of the different types of GNNs. Next, we introduce the fully distributed hydrological modeling framework wflow_sbm. The section concludes with a discussion on deep learning models and catchment memory.

2.1 Graph theory

A graph $\mathcal{G}$ is a mathematical structure that consists of nodes, connected to each other by edges. The set of all N nodes is denoted as $\mathcal{V} = \{v_i\}_{i=1}^{N}$, where v_i is the i-th node. The set of all E edges is denoted as $\mathcal{E} = \{e_{ij}\}$, where edge e_{ij} connects node pair (v_i, v_j), for $v_i \in \mathcal{V}$ and $v_j \in \mathcal{V}$. A graph, with its nodes and edges, is thus denoted as $\mathcal{G}(\mathcal{V},\mathcal{E})$. The neighbourhood $\mathcal{N}$ of node v_i is the set of all nodes directly connected to v_i, so $\mathcal{N} \in \mathcal{V}$. The neighbourhood of node v_i can be denoted as $\mathcal{N}(v_i) = \{v_j \in \mathcal{V} \,|\, (v_i, v_j) \in \mathcal{E}\}$.

Each node has certain properties called node features, which can be expressed by a node feature vector $\mathbf{x}_i \in \mathbb{R}^D$, where D is the number of node features. The features of all N nodes can be combined into node feature matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$, where each column represent all features of a single node, and each row represents a single features for all nodes. Similar to nodes, edges can also have features. Every edge has P features, and the features of edge e_{ij} are denoted as $\mathbf{e}_{ij} \in \mathbb{R}^P$.

The features of all edges can be combined into edge feature matrix $\boldsymbol{\mathcal{E}} \in \mathbb{R}^{E \times P}$.

Connections between nodes can be indicated by the set of edges $\mathcal{E}$, but the adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ can also be used. If an edge exists between node v_i and v_j, element a_{ij} of the matrix is 1; if no edge exists it is 0. If e_{ij} is bidirectional, $a_{ij} = a_{ji} = 1$, but if it is unidirectional $a_{ij} = 1 \neq a_{ji} = 0$. Nodes can also be connected to themselves, in which case the adjacency matrix is denoted as $\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$. In practice, a normalized version of the adjacency matrix is often used, which is defined as

$$\hat{\mathbf{A}} = \widetilde{\mathbf{D}}^{(-1/2)} \widetilde{\mathbf{A}} \widetilde{\mathbf{D}}^{(-1/2)} \tag{1}$$

Here, $\widetilde{\mathbf{D}}$ is the diagonal matrix containing the node degrees of $\widetilde{\mathbf{A}}$. This normalized version of the matrix is used to improve the numerical stability (Kipf and Welling, 2016).

If node v_i is connected to node v_j, it can send a message $\mathbf{m}_{ij}$ to v_j. The message depends on the node features of both nodes, and the features of the connecting edge. The message is computed as

$$\mathbf{m}_{ij} = \psi(\mathbf{x}_i, \mathbf{x}_j, \mathbf{e}_{ij}) \tag{2}$$

Based on the messages a node receives, it can update its node features according to

$$\mathbf{h}_i = \sigma\left(\mathbf{x}_i, \bigoplus_{v_j \in \mathcal{N}_i} \mathbf{m}_{ij}\right) \tag{3}$$

where $\mathbf{h}_i$ are the hidden states, which represent the updated node features of node v_i. $\sigma(\cdot)$ is some function used to introduce non-linearity, and can be a non-learnable activation function, as well as a multi-layer perceptron (MLP). $\oplus(\cdot)$ is some function aggregating all messages from the neighbours $v_j \in \mathcal{N}(v_i)$ of node v_i. All nodes simultaneously send messages, receive messages, and compute their hidden states. The process of sending, receiving, and hidden state computation is also called message-passing (Battaglia et al., 2018).

2.2 Graph Neural Networks

A Graph Neural Network is a type of deep learning model that can be applied to graph-structured data. They can be seen as a more general type of Convolutional Neural Network, able to operate in non-Euclidian space (Zhou et al., 2020). A GNN consists of a total of L layers, where each layer is a repetition of the process of message-passing, each time with updated hidden states. By combining Equation (2) and Equation (3), we can write the l-th layer as

$$\mathbf{h}_i^{l+1} = \sigma\left(\mathbf{h}_i^l, \bigoplus_{v_j \in \mathcal{N}_i} \psi(\mathbf{h}_i^l, \mathbf{h}_j^l, \mathbf{e}_{ij})\right) \tag{4}$$

where $\mathbf{h}_i^l$ are the hidden states of node v_i at the l-th layer. The dimension of $\mathbf{h}_i^l$ can be different than that of $\mathbf{h}_i^{l-1}$, and is denoted as D_l. Note that $\mathbf{h}_i^0 \equiv \mathbf{x}_i$. Equation (4) is the same for all nodes in the graph. The number of parameters thus does not increase with more nodes, meaning GNNs do not suffer from the "curse of dimensionality" common to some other types of machine-learning models (Poggio et al., 2017). We can write Equation (4) also at the graph level, like

$$\mathbf{H}^{l+1} = f(\widetilde{\mathbf{A}}, \mathbf{H}^l, \boldsymbol{\mathcal{E}}) \tag{5}$$

where $\mathbf{H}^l \in \mathbb{R}^{N \times D_l}$ is the hidden state matrix at the l-th layer, and $f(\cdot)$ is the message-passing function. Again, at the first layer the node features are used, so $\mathbf{H}^0 \equiv \mathbf{X}$.

The hidden states $\mathbf{h}_i^l$ of node v_i at the l-th layer depend on the hidden states $\mathbf{h}_i^{l-1}$ of its direct neighbours $v_j \in \mathcal{N}(v_i)$ at layer $l-1$. However, these hidden states of its neighbours at layer $l-1$ in turn depend on the hidden states of their neighbours at layer $l-2$. Thus, in a GNN with L layers, the original node features $\mathbf{x}_i$ of node v_i can propagate to nodes that are at most L steps removed from it, also known as the L-hop neighbourhood.

The output of the last layer $\mathbf{H}^L$ is also the output of the GNN. All layers of the GNN combined can be written as

$$\mathbf{H}^L = \Phi(\mathbf{X}, \boldsymbol{\mathcal{E}}, \mathbf{A}) \tag{6}$$

where $\Phi(\cdot)$ denotes a sequence of learnable GNN layers. The GNN can be given some input, and trained to let its output approximate some target. The target vector and prediction vector of node v_i are denoted as $\mathbf{y}_i \equiv \mathbf{h}_i^L \in \mathbb{R}^O$ and $\hat{\mathbf{y}}_i \in \mathbb{R}^O$, respectively. Here, O is the number of output variables. The target matrix and prediction matrix are denoted as $\mathbf{Y} \in \mathbb{R}^{N \times O}$ and $\hat{\mathbf{Y}} \equiv \mathbf{H}^L \in \mathbb{R}^{N \times O}$, respectively. The error between the predictions and the targets is computed according to

$$\mathcal{L} = J(\hat{\mathbf{Y}}, \mathbf{Y}) \tag{7}$$

where $\mathcal{L}$ denotes the loss and J denotes some loss function. The gradients of the loss indicate how the loss would change with small changes to the parameters of the GNN. Through backpropagation, the parameters of the GNN are adjusted as to minimize $\mathcal{L}$.

The general message-passing layer given in Equation (4) knows many modifications (Zhou et al., 2020). Furthermore, many different types of layers can be added before, in-between, or after the message-passing layers, and there are various ways to connect layers with each other. Additionally, different configurations are possible when training a GNN. The hyperparameter search encompasses the identification of the optimal GNN architecture and the refinement of the training configuration.

2.3 wflow_sbm

We train the GNN model on wflow_sbm, a fully distributed, physics-based hydrological model (van Verseveld et al., 2022). It is written in the Julia language and part of the wflow modelling framework. In wflow_sbm, a catchment is divided

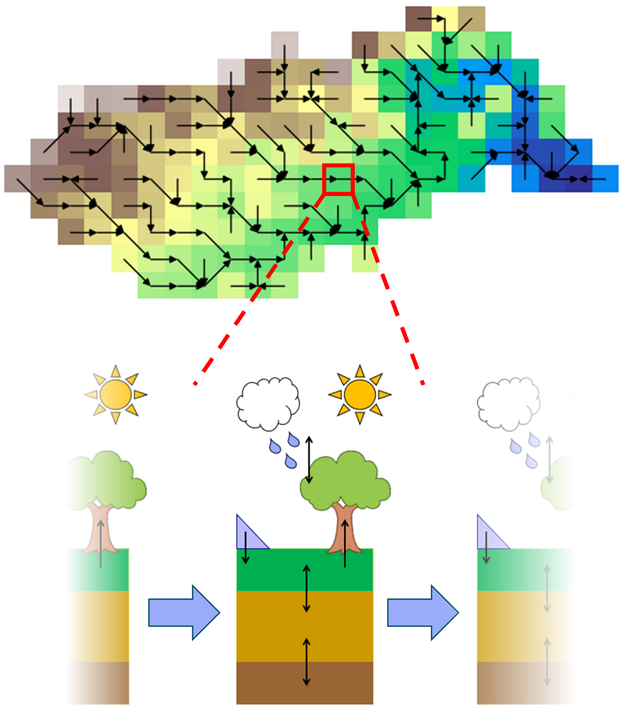


Figure 1. The concept of wflow_sbm visualized for a small caatchment. The catchment is divided into grid cells, and a routing scheme is derived from the elevation of each gridcell.

into grid cells, which are connected to each other via a routing scheme based on the elevation and the slope. Each grid cell contains its own static parameters, representing the physical characteristics in that area, its own forcings, and its own storages and fluxes. Each cell can receive lateral fluxes, such as overland flow, river discharge, and subsurface flow, from upstream cells. Within this lumped grid cell model, the vertical fluxes are then modelled, such as snowfall, rainfall, evaporation, transpiration, infiltration, capillary rise, and snow melt. The different storages within the cell, such as saturated storage, unsaturated storage, land storage, snow storage, and interception storage, are updated, and the cell then generates lateral fluxes for downstream cells. The process of receiving input from upstream cells and forcings, updating storages, and generating output for downstream cells is repeated for each timestep. Figure 1 visualizes the concept of wflow_sbm is visualized in .

2.4 Memory of catchments

The hydrological response of a catchment to forcings often has a strong temporal componenent, and hydrological models need to consider this component. For example, if the soil is completely saturated because of previous rainfall events, a next rainfall event will generate a large amount of runoff, whereas a rainfall event occurring after a period of drought might generate no runoff at all. The state of the catchment thus partially influences its response to forcings, and the GNN model has to take this phenomenon into consideration.

There are two common ways for a hydrological model to account for this temporal component (Gharari and Razavi, 2018). The first method is to include storages into the model, adding them to the model input and thus supplying it with information on the state of the catchment. The storages furthermore provide extra information about the catchment and its behaviour to the user. However, since there is often little observation data available on the storages (McCabe et al., 2017), the model has to be capable of predicting the storages auto-regressively by itself. It is therefore a method mostly used by conceptual and physics-based models, wflow_sbm included. LSTM models use this approach as well in the form of cell states, although these do not explicitly represent a physical component of the catchment (Kratzert et al., 2022).

The second method is to provide the model with a history of forcings, enabling the model to assess the state of the catchment indirectly. Explicitly adding some form of memory to the model is thus avoided (e.g. Sun et al., 2022), which can be challenging especially for data-driven models lacking any physics. However, the lag time between a forcing event and the hydrological response can be more than a year, depending on the processes at play (De Lavenne et al., 2022). This implies that the supplied history should be of the same order of magnitude. That would in turn result in large amounts of input data, which increases model runtime (Vivoni et al., 2011). Furthermore, no additional information on the state and behaviour of the catchment is provided by the model to the user. Nonetheless, many data-driven models use this approach, since they often lack the physics to model catchment storages, and observation data on storages is generally not available (Kashinath et al., 2021).

3 Hydological modelling with GNNs

3.1 GNN architecture

We propose a GNN inspired by wflow_sbm. We discretize the grid cells of wflow_sbm into a graph, using the D8 local drainange direction to define the edges. The physical characteristics of a grid cell are represented by the static node features $\mathbf{x}_{s,i}$ of node v_i. The forcings in a grid cell at time t are represented by the dynamic forcing features $\mathbf{x}^t_{f,i}$ of the corresponding node v_i. The storages and fluxes of a grid cell at time t are represented by the targets $\mathbf{y}^t_i$, and the GNN's predictions of these storages and fluxes are denoted as $\hat{\mathbf{y}}^t_i$. The connections between grid cells are represented by edges, and the characteristics of the connection (its slope and length) are represented by the edge features $\mathbf{e}_{ij}$. As suggested by You et al. (2020), we employ a encoder-processor-decoder architecture; the GNN acts as the processor, and the encoder and decoder are both Multi-Layer Perceptrons (MLP), which consist of multiple linear layers with activation functions inbetween. Figure 2 shows the GNN architecture.

Two encoders are used, one for the node features and one for the edge features. They increase the dimensionality of the input data, allowing for a higher expressivity (Bentivoglio et al., 2023). The edge encoder is defined as

$$\boldsymbol{\mathcal{E}}' = \phi_{\mathbf{e}}(\boldsymbol{\mathcal{E}}) \tag{8}$$

where $\boldsymbol{\mathcal{E}}' \in \mathbb{R}^{P \times D_0}$ is the encoded edge feature matrix and $\phi_{\mathbf{e}}(\cdot)$ is the edge encoder. The node encoder is defined as

$$\mathbf{H}^0 = \phi_{\mathbf{x}}(\mathbf{X}^t) \tag{9}$$

where $\mathbf{H}^0 \in \mathbb{R}^{N \times D_0}$ is the encoded node feature matrix, $\phi_{\mathbf{x}}(\cdot)$ is the node encoder, and $\mathbf{X}^t$ is the node feature matrix at timestep t. The GNN with L layers is defined as

$$\mathbf{H}^L = \Phi(\mathbf{H}^0, \boldsymbol{\mathcal{E}}', \mathbf{A}) \tag{10}$$

We opt for letting the GNN predict the change relative to the previous timestep, instead of having it predict the absolute value. This is inspired by traditional hydrological models; these commonly do no predict the absolute storages of the catchment directly. Instead, they predict the occurring fluxes, and simply update the storages based on these fluxes. The GNN uses a similar approach, by predicting the change instead of the absolute value for all variables, storages and fluxes alike. Deng et al. (2024) achieved improved performance using this method.

The decoder is defined as

$$\hat{\mathbf{Y}}^{t+1} = \hat{\mathbf{Y}}^{t} + \varphi(\mathbf{H}^L) \tag{11}$$

where $\varphi(\cdot)$ is the decoder, transforming the embedded predictions from the GNN-processor to hydrological variables. Combining Equations (8) to (11) we get

$$\hat{\mathbf{Y}}^{t+1} = \hat{\mathbf{Y}}^{t} + \varphi\left(\Phi(\phi_{\mathbf{x}}(\mathbf{X}^t), \phi_{\mathbf{e}}(\boldsymbol{\mathcal{E}}), \mathbf{A})\right) \tag{12}$$

In general, more complex GNNs tend to result in better performance, but at the expense of increased computational costs due to the higher number of parameters. (Li et al., 2019). The number of parameters of a GNN is predominantly determined by the dimensions of the embeddings D_l at each layer l and the total number of layers L. Increasing the number of layers in the GNN enables nodes to indirectly exchange information with more distant nodes within the network. This is particular relevant for hydrological modelling, as water in a catchment can traverse significant distances in a short timeperiod. The number of layers in a GNN should be sufficient to encompass the influence radius that a region can exert within the given timestep. As the model timestep increases, water can potentially flow over longer distances within that timestep, affecting other areas of the catchment. Hence, with coarser temporal resolutions, the GNN requires more layers to adequately capture all relevant processes. Conversely, fewer layers are required as the spatial resolution is reduced. With the same number of hops, information from a node has traversed a greater distance.

3.2 Training strategy

The node feature matrix $\mathbf{X}^t$ at timestep t is defined as

$$\mathbf{X}^t = \begin{bmatrix} \mathbf{X}_a & \mathbf{X}_f^{t-p:t} & \hat{\mathbf{Y}}^{t-q:t} \end{bmatrix} \tag{13}$$

where $\mathbf{X}_a$ is the static feature matrix, $\mathbf{X}_f^t$ is the dynamic forcing matrix at time t, p is the lookback window for the forcings, $\hat{\mathbf{Y}}^t$ is the prediction matrix at timestep t, and q is the lookback window for the storages and fluxes. The GNN model thus accounts for catchment memory both by including storages and by taking a history of forcings as input. However, we limit the length of the forcing history, as to constrain the size of the input data and reduce the computational costs. Therefore, the GNN model is predominately relying on the storages to account for catchment memory. The short history of forcings can nonetheless be utilized by the GNN model to introduce a temporal aspect to the input data and harness knowlegde of previous timesteps (Bentivoglio et al., 2023). For the same reasons, the GNN model is supplied with not just the storages and fluxes of the last timestep, but of multiple previous timesteps. Although the GNN model initially lacks the physics to model the storages, it can approximate the physics as encoded in wflow_sbm during training. Furthermore, the storages provide us with extra information and enable us to compute the water balance. We hypothesize this balance between the two approaches enables the GNN model to reach optimal performance without a significant reduction in running time.

The storages and fluxes used as input by the GNN model are initially given by wflow_sbm. With these inputs the GNN model makes a prediction for the next timestep. These predictions, in turn, serve as inputs for the following timestep. Consequently, the GNN model uses its own output from timestep t as input for timestep $t+1$. Following q timesteps, the input storages and fluxes are exclusively based on the GNN model's output. This auto-regressive process continues until reaching the prediction horizon T. Figure 3 shows the input and output of the GNN model for various timesteps.

Given the autoregressive nature of the GNN model, training across multiple timesteps poses computational challenges and may compromise stability, particularly for larger values of T. To overcome these limitations, we employ a multi-step-ahead loss function together with a curriculum learning strategy similar to Bentivoglio et al. (2023). The multi-step-ahead loss function evaluates the cumulative error across several successive timesteps, and is defined as

$$\mathcal{L} = \frac{1}{TO}\sum_{t=0}^{T}\sum_{o=1}^{O}\begin{cases} \frac{1}{2}(\mathbf{y}_o^t - \hat{\mathbf{y}}_o^t)^2 & \text{if } \left|(\mathbf{y}_o^t - \hat{\mathbf{y}}_o^t)\right| < \delta \\ \delta((\mathbf{y}_o^t - \hat{\mathbf{y}}_o^t) - \frac{1}{2}\delta) & \text{otherwise} \end{cases} \tag{14}$$

where δ is a threshold parameter. For each timestep t, the output of the GNN model is evaluated against the output of

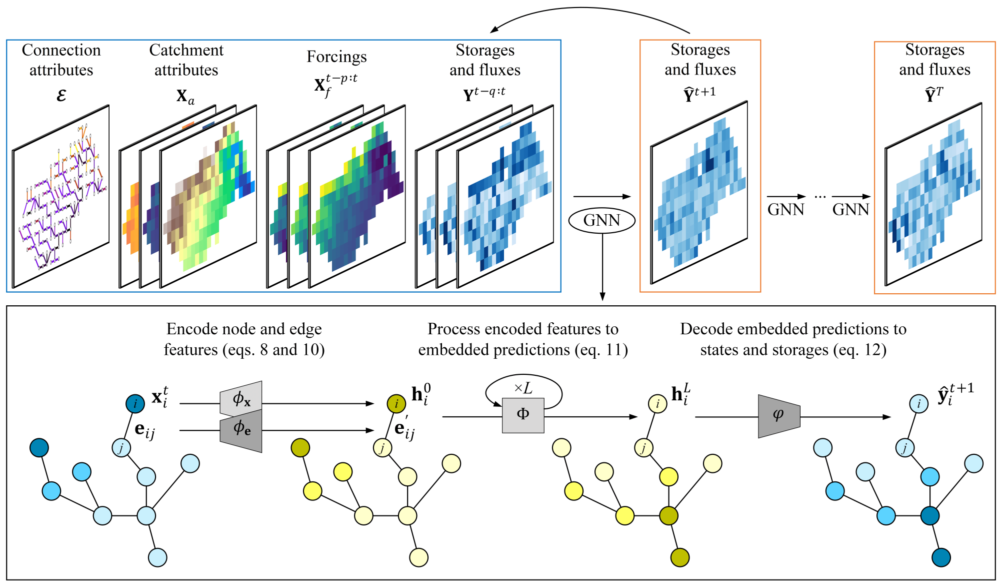


Figure 2. Overview of the encoder-processor-decoder architecture. The physical characteristics of the catchment, the connection attributes, the forcings, and the states and storages are discretized into the static node features $\mathbf{x}_{s,i}$, the edge features $\mathbf{e}_{ij}$, the dynamic node forcings $\mathbf{x}^t_{f,i}$, and the node predictions $\hat{\mathbf{y}}^t_i$, respectively, and used as input into the model (blue box). The model then makes a predictions for the states and storages for the next timestep $t+1$; this process is then repeated autoregressively, using the predicted output as input for the next timestep. The black box shows the encoder-processor-decoder architecture of Equation (12). First, the model inputs are encoded by the MLPs $\phi(\cdot)_\mathbf{x}$ and $\phi(\cdot)_\mathbf{e}$ according to Equations (8) and (9), resulting in a higher dimension embedding of the node features $\mathbf{H}^0$ ($\mathbb{R}^{N\times D} \to \mathbb{R}^{N\times D_0}$) and the edge features $\boldsymbol{\mathcal{E}}'$ ($\mathbb{R}^{E\times P} \to \mathbb{R}^{E\times D_0}$). The encoded input is then processed by the GNN $\Phi(\cdot)$ with L layers according to Equation (10), resulting in a embedded prediction of the change in storages and fluxes $\mathbf{H}^L$. This embedded prediction is then decoded by the decoder according to Equation (11), resulting in the prediction of the states and storages $\hat{\mathbf{Y}}^{t+1}$ at timestep $t+1$. This figure is adapted from Bentivoglio et al. (2023).

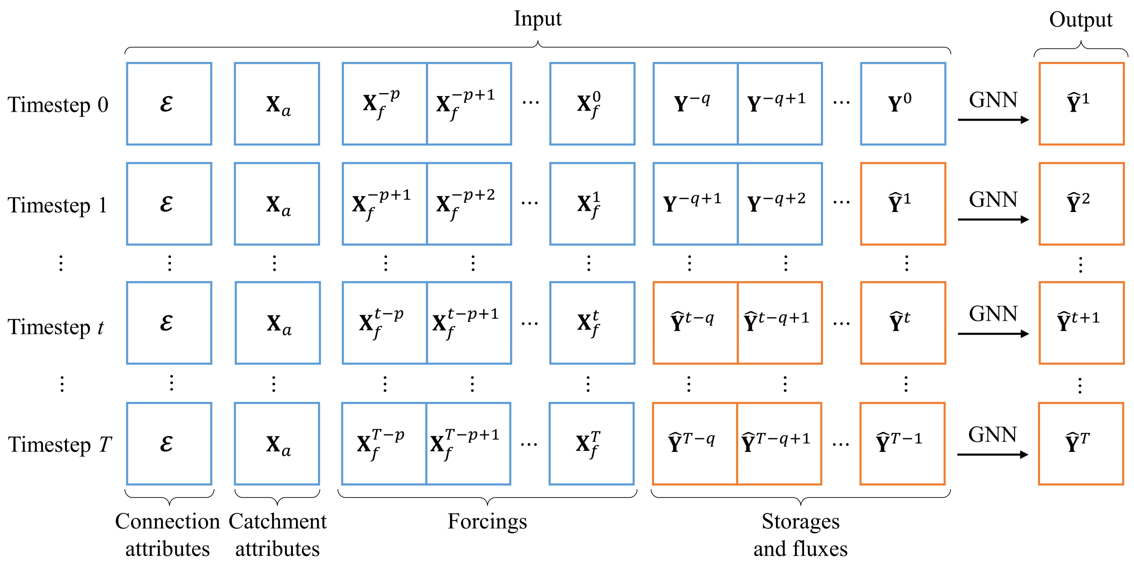


Figure 3. The input and output of the GNN model for several timesteps. As input, the GNN model uses the connection attributes, the static attributes, the forcings with a lookback window p, and the storages and fluxes with a lookback window q. Blue indicates a given, while orange indicates predicted. For timestep 0, the storages and fluxes are all given by wflow_sbm. The predicted storages and fluxes, i.e. the output for the timestep 1, are then used auto-regressively as input for the next timestep. The storages and fluxes at timestep $q+1$ are entirely predicted by the GNN model itself. The model predicts up to a prediction horizon T. This figure was adapted from Bentivoglio et al. (2023).

wflow_sbm using the Hüber loss (Huber, 1964). This loss function was chosen as it is less sensitive to outliers than for example the mean squared error. For every timestep $t > 0$, the GNN model is recursively incorporating its own output into the input of the next timestep, and the loss is averaged over all timesteps T within the prediction horizon. Through this method, the GNN model learns to adjust its own predictions, as well as to produce accurate outputs even when provided with inaccurate inputs, thereby enhancing its robustness and stability (Bentivoglio et al., 2023).

The curriculum training strategy gradually increments the prediction horizon T during training, and is used to improve stability (Wang et al., 2022). Initially, $T = 1$ and the model is trained to predict just a single-day-ahead. An epoch denotes a complete iteration through all training data, and following a pre-determined number of epochs, known as the curriculum epoch, the prediction horizon is extended. By governing the prediction horizon, the GNN model is at first trained to recognize short-term patterns and predict a few timesteps ahead. As the prediction horizon expands, the GNN model is trained to capture longer-term patterns and generate output for a greater number of consecutive timesteps. The curriculum training strategy is adapted from Bentivoglio et al. (2023) and can be seen in Algorithm 1.

Algorithm 1 Curriculum training strategy

```
Initialize:
  $T$ = 1
for epoch = 1 to MaxEpochs do
  $\hat{\mathbf{Y}}^{t+1} = \hat{\mathbf{Y}}^{t} + \varphi\left(\Phi(\phi_{\mathbf{x}}(\mathbf{X}^t), \phi_{\mathbf{e}}(\boldsymbol{\mathcal{E}}), \mathbf{A})\right)$
  $\mathcal{L} = J(\hat{\mathbf{Y}}, \mathbf{Y})$ following Equation (14)
  Update parameters
  if epoch > CurriculumEpoch $\cdot H$ then
    $H = H + 1$
  end if
end for
```

3.3 Training on wflow_sbm

We train the GNN model on wflow_sbm output instead of observation data. Training deep learning models on physics-based models has gained increasing interest in recent years (e.g. Yang et al., 2019; Sun et al., 2019; Yuval and O'Gorman, 2020; Lu et al., 2021; Nonnenmacher and Greenberg, 2021; Jia et al., 2021b, a; Sun et al., 2022), and has been used to overcome several challenges faced by DL models. Firstly, training on physics-guided improves a DL model's ability to generate physically consistent results (Willard et al., 2022). Furthermore, DL models trained on physics-guided models have been shown to better generalize to unseen scenarios (Read et al., 2019). Additionally, physics-guided models can provide the significant amount of training data needed to train DL models, which may not always be available from observations (Kashinath et al., 2021).

By training on wflow_sbm output instead of directly on observation data, the GNN model should thus produce results that are more conform the energy and water balance, and have better spatial transferability. Furthermore. wflow_sbm can provide us with many different hydrological variables in gridded format. These hydrological variables include streamflow, evaporation, snow pack, saturated and unsaturated storage, runoff, and subsurface flow. The GNN requires significant amounts of data of all these variables in gridded format, of the same area, of the same time period, and with the same temporal and spatial resolution. Despite significant advances in the field of remote sensing in the past decade, these may not be available from observation data in the required quantity and quality.

4 Experimental setup

4.1 Study area

We consider 3 different areas in northern England and southern Scotland. The first of the three areas will be used for training the GNN, the second for evaluation, and the third for testing. They make up 70%, 15%, and 15% of the total area, respectively. Figure 4 shows the three areas. Small catchments close to the coasts are omitted from the study area, as the spatial resolution of 30 by 30 arc seconds is too coarse to properly model them.

All areas are dominated by an oceanic climate (Köppen climate classification Cfb) and experience relatively mild winters and mild summers (Mayes and Wheeler, 2002). Precipitation is fairly equally distributed over the year, with the annual precipitation coming to about 1000 - 1300 mm. The landscape is mostly hilly, with most elevations being around 100 - 300 masl, and peaks being around 700 - 800 masl. Table 1 reports various statistics on the climate and landscapes of the areas.

We use data from globally available open-source datasets. Data of the three areas is extracted, projected, clipped, and resampled with HydroMT, v0.9.3 (Eilander et al., 2023a). We subsequently build a wflow_sbm model of each area with HydroMT-wflow (v0.5.0), which also converts the preprocessed data into model parameters (Eilander et al., 2023b). The models are then run with wflow, v0.7.3 (van Verseveld et al., 2023).

We use a temporal resolution of 1 day, and a spatial resolution of 30 arc seconds by 30 arc seconds (approximately 1 km by 1 km at the equator). Each wflow_sbm model is run with input data from 01-01-1988 up until 31-12-2009. We only use wflow_sbm output data from 01-01-2000 onwards, resulting in a spin-up time of 12 years, and 10 years of data available for the GNN model.

As input, wflow_sbm uses gridded catchment attributes and gridded forcing data. We selected ten different variables, which enable us to calculate the water balance at each grid

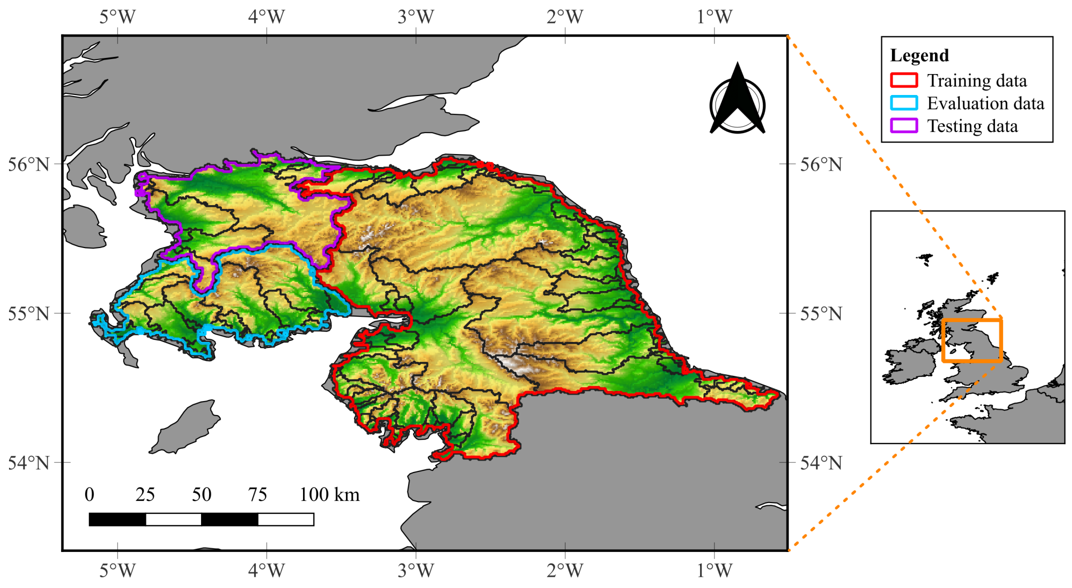


Figure 4. Overview of the three areas used in this study.

Table 1. Statistics on the three areas considered in this study. Max. size denotes the area of the largest catchment in the dataset. The snow fraction is the percentage of precipitation occurring during sub-zero temperatures. The aridity index is the total amount of potential evapotranspiration divided by the total amount of precipitation.

	Training	**Evaluation**	**Testing**
Area [km^2]	24067	5224	5206
No. catchments [-]	104	50	14
Max. size [km^2]	4990	1223	2985
Elevation* [$masl$]	191 ± 134	159 ± 118	178 ± 113
Slope* [$\%$]	4.1 ± 3.7	3.9 ± 0.31	2.9 ± 2.4
Precip.* [$mm\,d^{-1}$]	2.9 ± 4.5	3.7 ± 5.2	3.5 ± 4.9
Snow fraction [%]	0.9	0.4	0.5
Aridity index [-]	0.52	0.41	0.42
Temp. Jan.* [$°C$]	4.3 ± 2.9	5.0 ± 2.8	4.6 ± 2.9
Temp. July* [$°C$]	14.6 ± 2.4	14.4 ± 2.1	14.5 ± 2.2

*Mean and standard deviation are reported

cell. Based on some catchment attributes, we compute connection attributes, to be used as edge features. Table 2 lists all input and output data we used for the GNN model. The GNN model does not make use of all the same inputs as wflow_sbm; some are left out as to limit the computational costs.

Deep learning models tend to perform poorly for scenarios outside the training domain. Therefore, we investigate the occurance of extreme streamflow events at the outlets of the five biggest catchments for each of the three wflow_sbm datasets. Figure 5 shows the results of this investigation. Low flows

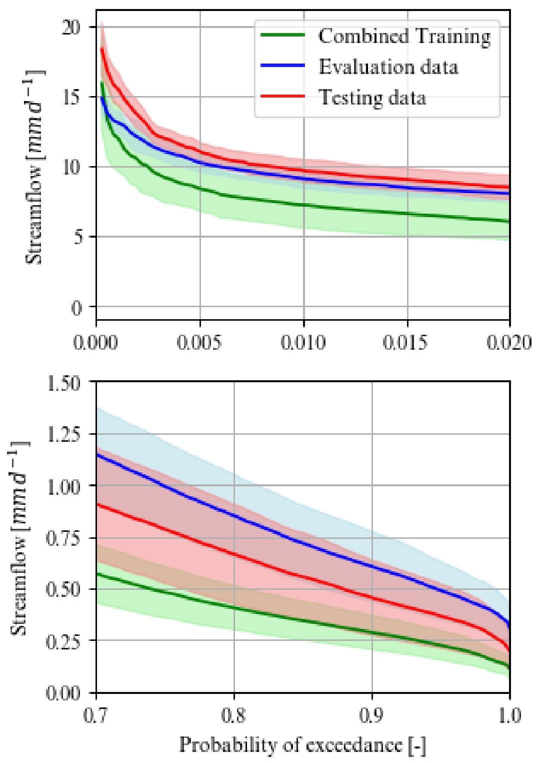


Figure 5. The probability of exceedance curve for streamflow for each dataset with a 95% confidence interval. Each curve is based on the wflow_sbm output at the outlet of the five biggest catchments per dataset. Top image shows the top 2% of flows, i.e. the peak flows. Bottom image shows the bottom 30% of flows, i.e. the low flows.

Table 2. The input and output variables of wflow_sbm we used in the surrogate GNN model

Data type	Variables	Unit
Catchment attributes $\mathbb{R}^{N \times D_a}$	river mask*	-
	river length	m
	river width	m
	river depth	m
	river slope	-
	Manning coefficient river	$s\,m^{-1/3}$
	upstream area	km^2
	Manning coefficient land	$s\,m^{-1/3}$
	grid cell area	km^2
	soil thickness	mm
	rooting depth	mm
	fraction paved surface	-
	fraction open water	-
	porosity	-
	residual water content	-
	extinction coefficient	-
	vertical hydraulic conductivity	$mm\,d^{-1}$
Connection attributes $\mathbb{R}^{E \times P}$	slope	-
	distance	km
Forcings $\mathbb{R}^{N \times D_f \times T}$	temperature	$^{\circ}C$
	precipitation	$mm\,d^{-1}$
	potential evapotranspiration	$mm\,d^{-1}$
Storages and fluxes $\mathbb{R}^{N \times O \times T}$	streamflow	$m^3\,s^{-1}$
	actual evapotranspiration	$mm\,d^{-1}$
	runoff river	$mm\,d^{-1}$
	runoff land	$mm\,d^{-1}$
	solid snow pack	mm
	liquid snow pack	mm
	incoming subsurface flow	$m^3\,d^{-1}$
	outgoing subsurface flow	$m^3\,d^{-1}$
	saturated storage	mm
	unsaturated storage	mm

*wflow_sbm by default considers any cell with an upstream area of 30 km^2 to be a river cell

in the testing data are within the domain of the training and evaluation data. Peak flows in the testing dataset are slightly outside the domain of the training and evaluation data, but the difference is small. Only for the top 0.1%, the difference is outside the 95% confidence interval.

4.2 Performance metrics

To assess the performance of the GNN model, we employ multiple metrics. Firstly, we employ the Kling-Gupta Efficiency (Gupta et al., 2009), which is defined as

$$KGE = 1 - \sqrt{(r-1)^2 + (\alpha - 1)^2 + (\beta - 1)^2} \tag{15}$$

where r is the correlation coefficient, α is a measure for the variability error defined as $\sigma_{\hat{y}}/\sigma_y$, and β indicates the bias, defined as $\mu_{\hat{y}}/\mu_y$. The KGE ranges from $-\infty$ to 1, where 1 is a perfect fit of the predictions to the ground truth. If the predictions fit just as good as the mean, the KGE will be -0.41 ($1-\sqrt{2}$ to be exact), as shown by Knoben et al. (2019). We furthermore make use of the Mean Absolute Error (MAE), defined as

$$MAE = \sum_{t=1}^{T} |y - \hat{y}| \tag{16}$$

The MAE ranges from 0 to inf, where 0 is a perfect score. Additionally, we also employ the Nash-Sutcliffe Efficiency (Nash and Sutcliffe, 1970), which is defined as

$$NSE = 1 - \frac{\sum_{t=1}^{T} \left(y^t - \hat{y}^t\right)^2}{\sum_{t=1}^{T} \left(y^t - \overline{y}\right)^2} \tag{17}$$

Like the KGE, the NSE ranges from $-\infty$ to 1, where 1 is a perfect fit of the predictions to the ground truth. A score of 0 means the predictions fit to the ground truth just as good as the mean. Lastly, we also make use of six metrics focused on streamflow prediction at the outlet, as proposed by Yilmaz et al. (2008). These are the runoff bias (BiasRR), low flow bias (BiasFLV), peak flow bias (BiasFHV), median flow bias (BiasFMM), lag time bias (BiasTLag), and the flow duration curve midsegment slope bias (BiasFMS). All six metrics have their optimum at 0, with negative and positive values indicating a negative and a positive bias in the model, respectively.

4.3 Training setup

We search a wide range of hyperparameters, the full list of which can be found in Appendix A. As part of the hyperparameter search, we try five different types of layers within the GNN processor that are capable of node-level regression. Namely, the graph convolutional operator (Kipf and Welling, 2016), the chebyshev spectral graph convolutional operator (Defferrard et al., 2016), the graph attentional operator (Veličković et al., 2017), the GraphSAGE operator (Hamilton et al., 2017), and the residual gated graph convolutional operator (Bresson and Laurent, 2017). Only the graph attentional operator and the the residual gated graph convolutional operator are capable of handling multidimensional edge features. Other important hyperparameters are the number of layers in the GNN processor, the number of layers in the encoder and decoder, the dimensionality of the hidden states, and the type of connection between layers in the GNN processor.

We train GNN models using the AdamW optimizer (Loshchilov and Hutter, 2017) and an early-stopping algorithm. We tried a wide range of hyperparameters, the full list of which can be found in Appendix A. The maximum prediction was set to 60 days during training and evaluation, and to 365 days during testing. In wflow_sbm, not every variable is predicted at every cell; predictions by the

GNN model of such variables at the corresponding nodes are masked. Streamflow and runoff from river are predicted only at cells with a river. Incoming subsurface flow only occurs at cells that have a connection to an upstream cell. The GNN model's negative predictions are constrained to zero in the post-processing, given that none of the variables can be negative, with the sole exception being net runoff from river.

We scaled all data prior to training; this is needed as deep learning models usually cannot learn on data that isn't all in the same order of magnitude. For scaling we use standardization, according to

$$x_{i,scaled} = \frac{x_i - \mu_x}{\sigma_x} \tag{18}$$

where x is some variable, x_i is a single sample from that variable, and $x_{i,scaled}$ is the scaled sample. We also tried normalization as a form of scaling, but found the GNN model cannot learn on normalized data. Hydrological data is often rich in outliers; hence, normalization would cause the bulk of the data to be restricted to a small domain, limiting the expressivity.

Our scripts are written in Python 3.11 (Van Rossum and Drake Jr, 1995). Data handling was done with the xarray library (Hoyer and Hamman, 2017). We furthermore use PyTorch (Paszke et al., 2019) and PyTorch Geometric (Fey and Lenssen, 2019) for the deep learning part of our work. All figures, with the excepetion of Figure 4 were produced with Matplotlib (Hunter, 2007). QGIS was utlized for producing Figure 4 and selecting the areas used in this study (QGIS Development Team, 2023). Each wflow_sbm model is run with four threads; glaciers, lakes, reservoirs, and masswasting are turned off. We use the default model settings. Input data for the GNN model is acquired using the HydroMT-wflow plugin (Eilander et al., 2023b, a). Hyperparameter tuning was done using Weights & Biases (Biewald, 2020). Training and testing is both done with a batch size of 1. During training this is needed to constrain the computational costs. During testing, this is done as to ensure the GNN model predicts all timesteps sequentially similar to wflow_sbm, thus allowing for a fair comparison of the runtime. We used a NVIDIA A100 80GB GPU (Delft High Performance Computing Centre, 2022) for training and testing the GNN model, and two Intel Xeon Gold 6144 CPU @ 3.50GHz processors for running wflow_sbm.

5 Results

Unless stated otherwise, the performance of the GNN model is always reported with regards to the full ten years of unseen testing data, assuming wflow_sbm output to be the ground truth, using a prediction horizon of 365 days, and for the best performing GNN model only.

5.1 Predicting hydrological variables

5.1.1 Performance per variable

Figure 6 displays the KGE and MAE of the GNN model for streamflow, land runoff, net runoff from river, solid snow pack, and liquid snow pack. Figure 7 displays the KGE and MAE of the GNN model for incoming subsurface flow, outgoing subsurface flow, solid snow pack, liquid snow pack, and actual evapotransporation.

The GNN model accurately predicts streamflow in most parts of the testing area. Poor KGE values are primarily observed in upstream regions due to a small absolute error, leading to a proportionally larger relative error owing to the overall lower flows. During training, the GNN model does not account for relative error, leading to a inclination towards peak flows. This is evidenced by the MAE which shows, contrary to the KGE, poor values in the downstream parts of the catchment. While it is possible to correct this inclination towards peak flows during training, e.g. by utilizing the logarithm of streamflow or incorporating factors such as upstream area or Strahler order, we omit this step to maintain the scope of our research focused on demonstrating the overall performance of the GNN model.

In addition, we evaluate the streamflow predictions of the GNN model using the six metrics proposed by Yilmaz et al. (2008). Table 3 presents the performance scores of the GNN model across the five largest catchments for each of the six metrics. Performance across all six metrics appears consistent across catchment sizes, except for bias in peak flow prediction (BiasFHV). This absolute bias in peak flow prediction appears to increase as catchment size and subsequently discharge volume decrease, underscoring the GNN model's aforementioned inclination towards peak flows during training. The overall small bias in peak flow prediction further reinforces the presence of an inclination towards peak flows during training. Consistent with these observations, the GNN model displays a significantly larger bias in low flow prediction (BiasFLV) compared to peak flow prediction. The emphasis on high flows during training results in substantial under- or overprediction of low flows.

The GNN model exhibits an overall positive bias towards the runoff coefficient (BiasRR), indicating that the GNN model potentially may not be able to close the water balance. Furthermore, the GNN model displays a pronounced positive bias towards the slope of the midsegment of the probability of exceedance curve (BiasFMS), suggesting that the hydrographs it generates are flashier compared to those produced by wflow_sbm. The GNN model does not exhibit any bias with regards to the time lag, which could also orignate from the relative coarse temporal performance of daily timesteps. Median flows are consistently underpredicted across all catchments, albeit with a small bias. Overall, the GNN model demonstrates good performance in predict-

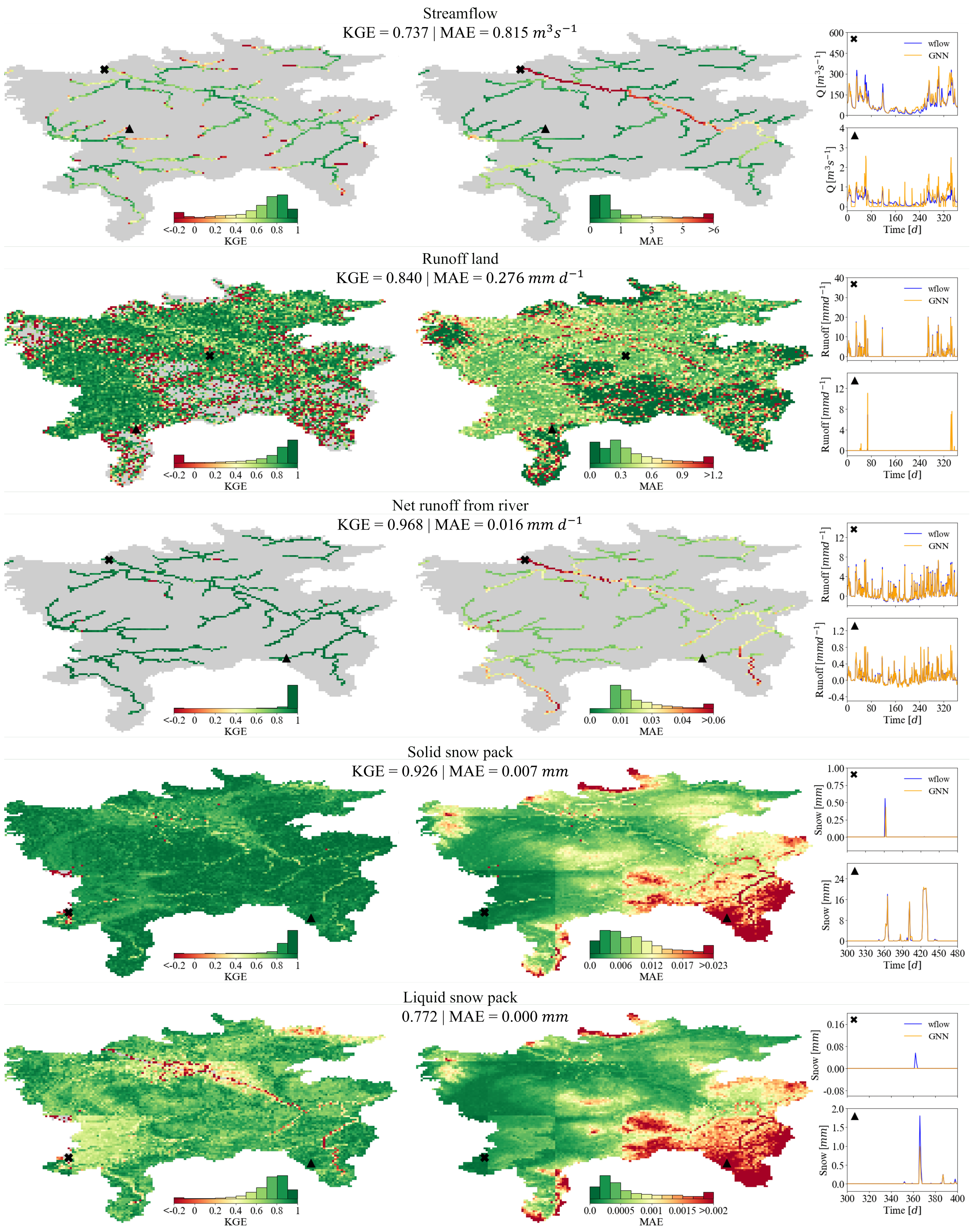


Figure 6. The KGE and MAE at every cell in the testing data for streamflow, land and net river runoff, and solid and liquid snow. Median KGE and MAE are reported above the maps. For each variable, two cells have been selected; the timeseries as given by the GNN model and wflow_sbm are plotted for a certain timeperiod.

.

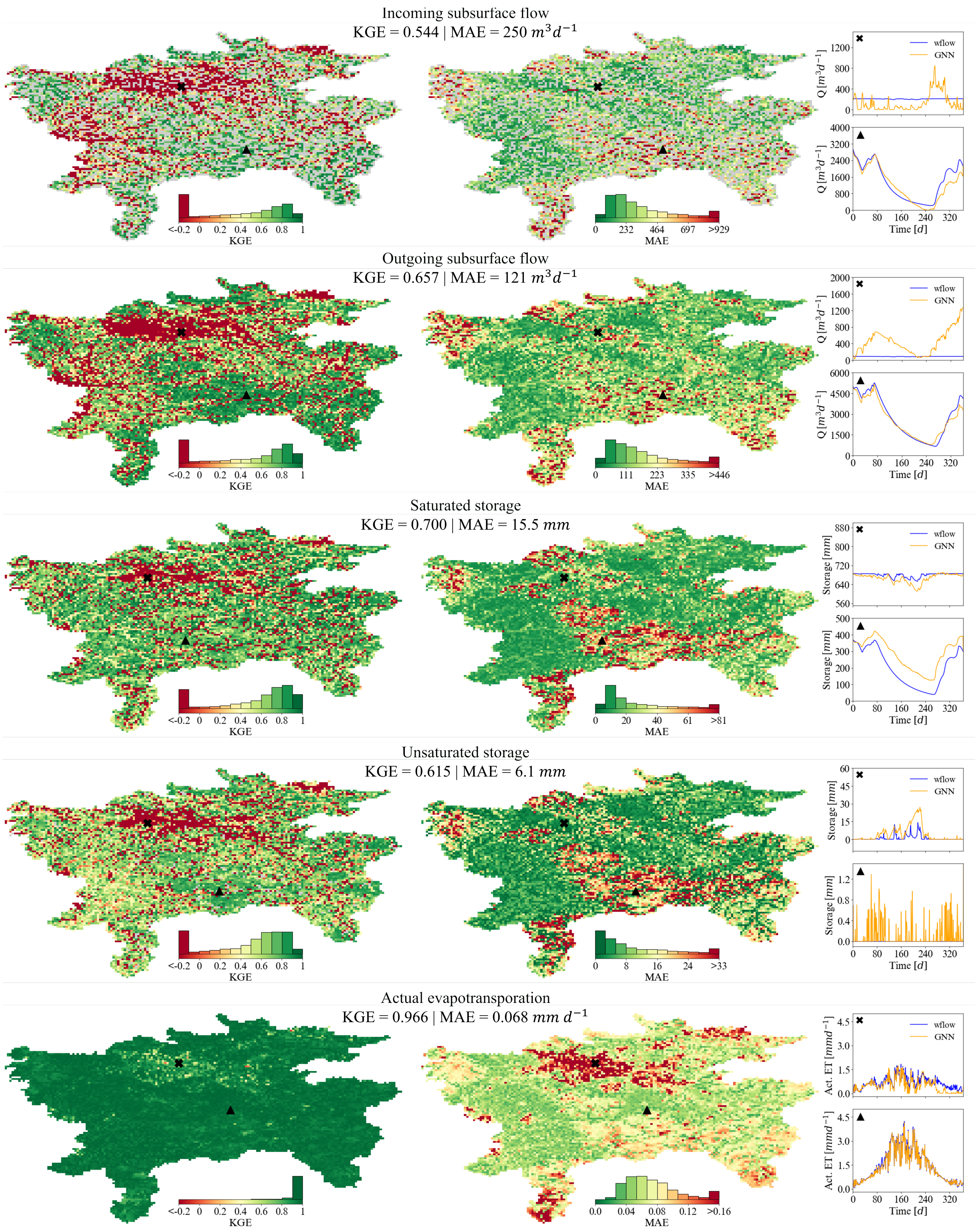

Incoming subsurface flow
KGE = 0.544 | MAE = 250 m^3d^{-1}
Outgoing subsurface flow
KGE = 0.657 | MAE = 121 m^3d^{-1}
Saturated storage
KGE = 0.700 | MAE = 15.5 mm
Unsaturated storage
KGE = 0.615 | MAE = 6.1 mm
Actual evapotransporation
KGE = 0.966 | MAE = 0.068 $mm\,d^{-1}$
KGE
MAE
wflow
GNN
Q $[m^3d^{-1}]$
Storage $[mm]$
Act. ET $[mmd^{-1}]$
Time $[d]$

Figure 7. The KGE and MAE at every cell in the testing data for incoming and outgoing subsurface flow, solid and liquid snow, and actual evapotranspiration. Median KGE and MAE are reported above the maps. For each variable, two cells have been selected; the timeseries as given by the GNN model and wflow_sbm are plotted for a certain timeperiod.

.

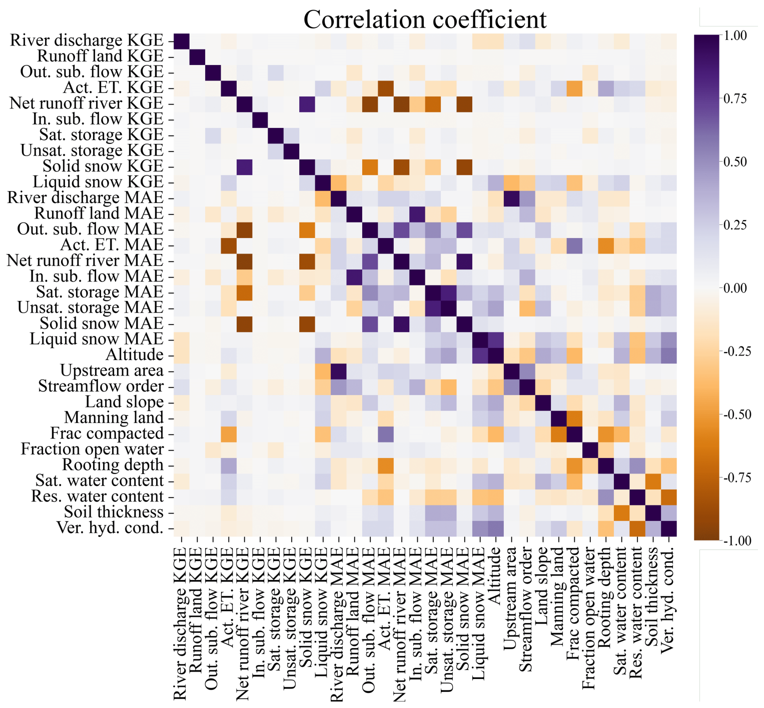


Figure 8. A heatmap showing the Pearson correlation coefficient between the GNN model performance of each output variable, as assessed by both the KGE and the MAE, and an ensemble of catchment parameters.

ing streamflow based on these six metrics, except for low flows.

Table 3. The GNN model performance as assessed by the six metrics proposed by Yilmaz et al. (2008). Each metric considers the bias of the GNN model compared to wflow_sbmc for a certain streamflow signature. The optimal value of each metric is 0%. Each metric is computed for streamflow at the outlet of a catchment. Performance is given for the five biggest catchments within the testing data.

Catchment	**1**	**2**	**3**	**4**	**5**
Area [km^2]	2985	713	579	316	215
BiasFHV [%]	0.2	8.2	-3.1	-3.6	30.7
BiasFLV [%]	-23.8	11.1	-49.7	-6.2	13.5
BiasRR [%]	11.3	2.5	14.3	1.4	-10.9
BiasFMS [%]	21.2	30.8	15.3	-5.9	28.5
BiasTLag [%]	0.0	0.0	0.0	0.0	0.0
BiasFMM [%]	-2.5	-6.3	-5.8	-4.0	-10.1

Land runoff is predicted with high accuracy. In some cells, wflow_sbm predicts no land runoff throughout the entire 10 years of data. Consequently, regardless of the GNN model's prediction in these cells, the KGE cannot be computed there (grey cells). Cells with poor KGE values often cluster near those with no runoff. In such cases of poor performance, occurrences of land runoff are rare, amplifying the sensitivity of the KGE metric to even minor errors by the GNN model. For instance, consider a cell where according to wflow_sbm, no land runoff occurs during the full 10 years, except for a single day where 2 mm of land runoff occurs. In the event that the GNN model predicts 1 mm of land runoff solely for that day, while accurately predicting no land runoff for all other days, the resulting KGE remains notably low at -0.41, notwithstanding the model's precision on all other days. This demonstrates that, by definition, the KGE will be $1-\sqrt{2} \approx -0.41$ if the mean has the same goodness-of-fit as the predicted time-series. Thus, the KGE metric is highly sensitive when applied to time series data with minimal variance, as is the case for some cells with little land runoff. For these cells especially, the MAE should also be studied to get a more complete picture of the performance. In some of these cells, MAE scores are good, contradicting the poor KGE scores. Cells with no land runoff at all, where the KGE cannot be computed, have excellent MAE scores, suggesting that the GNN model accurately predicts the lack of runoff in these cells. High MAE values are primarily observed in downstream river cells, despite these cells often exhibiting decent KGE scores, a phenomenon similarly observed in streamflow predictions.

Net runoff from river is predicted with high accuracy across both upstream and downstream cells, as indicated by consistent KGE scores. However, downstream cells exhibit significantly poorer performance in terms of MAE compared to upstream cells. This discrepancy in performance can be attributed to the difference in absolute volume, as evident from the timeseries plots. Such disparities render the MAE a less suitable metric in this context, as it complicates the direct comparison of cells with each other.

The performance of the GNN model in predicting both the solid snow pack and the liquid snow pack exhibits strong similarity. Both variables yield high KGE scores, although solid snow is predicted particularly well. KGE scores for both variables appear to be lower in low-lying areas where snow occurs less frequently, making the KGE metric more sensitive to errors in these cells. The higher-elevated south-western portion of the testing area, characterized by colder winters and more frequent snowfall, demonstrates better KGE scores. The MAE seems to have an almost perfect inverse relationship with the KGE. Low-lying areas surrounding the coast and rivers display good MAE scores, whereas elevated regions exhibit poorer scores. Both the KGE and the MAE for both variables are subpar in river cells, mirroring the trends observed in streamflow and land runoff. Interestingly, the direct cause of the poor performance in these cells is not immediately apparent, as the presence of a river should not directly influence snow-related processes. The coarser spatial resolution of the forcing data ($0.25° \times 0.25°$) can also be dissected from the performance maps of both solid and liquid snow pack.

Incoming and outgoing subsurface flow are generally predicted with medium accuracy, demonstrating closely aligned patterns between the two. Low KGE scores predominantly occur in cells surrounding downstream parts of the area. Analysis of timeseries plots of such cells reveals the subsurface flow predicted by wflow_sbm is almost constant over time, a process that the GNN model does not seem to capture. The MAE exhibits distinct patterns compared to those observed with the KGE, as areas surrounding the downstream parts no longer stand out. Only the river cells themselves persist with poor MAE scores. Furthermore, poor MAE scores predominantly occur at higher elevations, where subsurface flow tends to be higher overall. The considerable differences in the volume of subsurface flow across the area, akin to net runoff from river, complicate the comparison of MAE scores among cells.

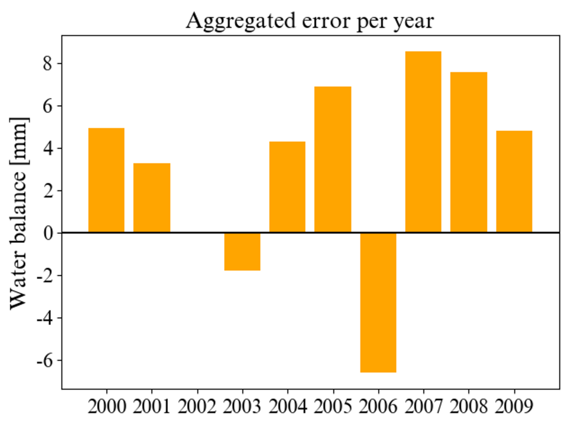
The GNN model demonstrates medium to good performance in predicting both saturated and unsaturated storage, with saturated storage generally exhibiting slightly higher accuracy compared to unsaturated storage. Additionally, the two variables are characterized by very similar underlying patterns. The KGE metric generally indicates poorer performance at river cells, whereas the MAE tends to be poorer in elevated parts of the area. Both saturated and unsaturated have cells with minimal variance, subsequently resulting in poor KGE scores for some of these cells. For some cells wflow_sbm even predicts a completely constant unsaturated storage throughout the ten-year period, making it not possible to compute the KGE in these cells.

Actual evapotranspiration is predicted with excellent accuracy. Some slight loss in performance can be observed around the downstream and high-altitude parts, but the differences are minimal.

Figure 8 presents a heatmap depicting the correlation coefficient between the GNN model's performance for each output variable, as evaluated by both the KGE and the MAE, and an ensemble of catchment parameters. Several noteworthy patterns emerge. For instance, KGE scores for incoming and outgoing subsurface flow demonstrate a degree of correlation, in line with expectations. Similarly, KGE scores on solid and liquid snow pack also exhibit some correlation. Suprisingly, KGE scores on saturated and unsaturated storage do not appear to be correlated. The MAE of different predicted variables demonstrates considerably more correlation among each other compared to the KGE. Notably, saturated and unsaturated storage exhibit a strong correlation, a contrast to their KGE scores. Furthermore, elevation appears to be correlated with several other variables, consistent with previous observations. The performance of actual evapotranspiration shows a negative correlation with the fraction of open water, suggesting that the GNN model may not adequately capture the process of open water evaporation. The MAE of outgoing subsurface flow exhibits a strong correlation with rooting depth, whereas this correlation is not as pronounced between rooting depth and the MAE of incoming subsurface flow. Strong correlations are evident between saturated water content and the MAE of saturated and unsaturated water storage, as well as the MAE of actual evapotranspiration. This correlation aligns with expectations, given the underlying processes involved. Soil thickness and vertical hydraulic conductivity exhibit similar correlation patterns to saturated water content.

5.1.2 Water and energy balance

The ten output variables enable us to compute the water balance per timestep per gridcell. Ideally, the water balance is closed, with the sum of all incoming and outgoing fluxes equaling the change in storage, as is the case for wflow_sbm. Figure 9 depicts both the the aggregated error over all gridcells per year, and the MAE per gridcell per day, based on the predictions of the GNN model. Per gricell, the absolute daily error of the GNN model is on average 1 mm. This is quite a substantial deviation, particularly considering average daily precipitation is approximately 3.5 mm. Furthermore, we observe a strong correlation between elevated precipitation levels and an increase in the water balance error. Aggregating all errors across all grid cells and time steps, the annual water balance error averages only 4 mm. Considering the annual precipitation of 1300 mm, this discrepancy represents a minor deviation. At first sight, the large difference between the relative error of the annual aggregated water balance and the MAE per gridcell per timecell is striking.



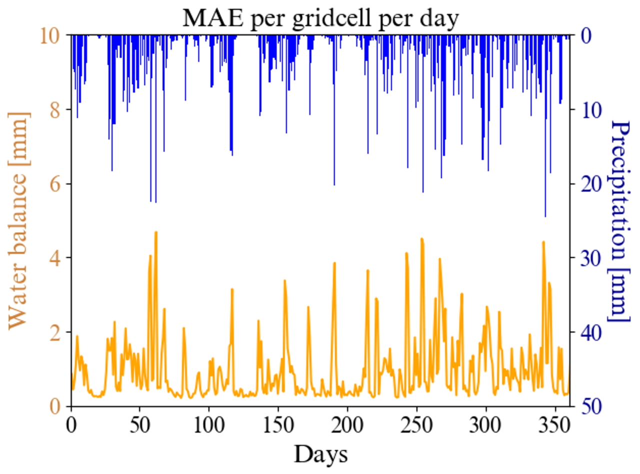


Figure 9. Top: the annual error in the water balance, aggregated over all cells and timesteps. Positive and negative errors can thus cancel each other out. Bottom: the MAE per gridcell per day and the mean daily precipitation.

However, it's important to note that this error does not necessarily indicate an overall imbalance in the GNN model's water budget across all grid cells. Rather, any surplus error in one cell may be compensated by a deficiency in another cell, allowing the model to maintain overall closure of the water balance. This phenomenon can also occur in the temporal dimension. For example, consider a scenario where a gridcell is predicted to receive inflow of water on day $t+1$, but not on day t. As a result, on day t, this cell may exhibit a deficiency in water by a certain amount, while on day $t+1$, it may show a surplus of the same amount. When aggregated over multiple time steps, the cumulative water balance error across all grid cells tends to cancel out, preserving the overall closure of the water balance. Therefore, while the GNN model can exhibit discrepancies in the timing and allocation of water among grid cells on a daily basis, the aggregated water balance can remain balanced over time and over all gridcells.

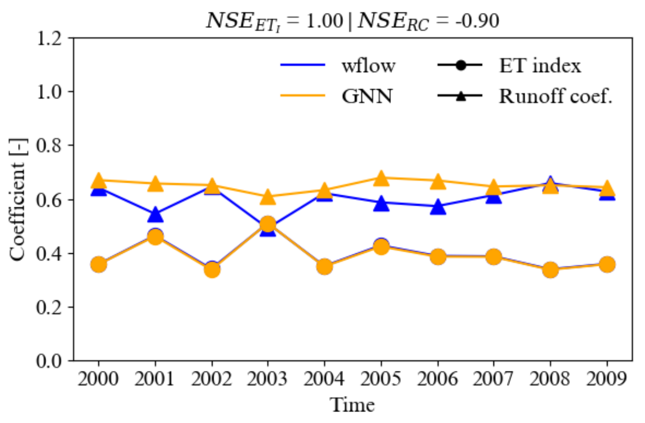


Figure 10. The evapotranspiration index (ET_I) and the runoff coefficient (RC) per year for the entire testing area, as predicted by wflow_sbm and the GNN model.

Figure 10 shows the runoff coefficient and evaporation index per year for the entire testing area. The evaporation index is predicted nearly perfectly, with an NSE of 1.00. This result is not surprising, given the high KGE scores achieved by the GNN model for actual evapotranspiration, as shown in Figure 7. The runoff coefficient is consistently over-predicted, in line with the bias reported in Table 3. Subsequently, the NSE is very poor, although this can also be partially attributed to the limited variance in the timeseries of the runoff coefficient.

Nevertheless, the GNN model exhibits a non-negligible error in predicting the annual runoff coefficient, whereas the error in the aggregated annual water balance is considerably smaller. This suggests that while the GNN model may effectively close the overall annual water balance, it still inaccurately predicts individual components of it.

The energy balance is closed in wflow_sbm, as it is a physics-guided model. Since the GNN model predicts actual evapotranspiration with very high accuracy, it can be inferred that the energy balance of the GNN model is also closed.

5.2 GNN model analysis

5.2.1 Speedup and model complexity

Appendix A details the results of the hyperparameter search. Overall, we find that using the GraphSAGE layer in the GNN processor yields optimal results. Moreover, employing a single-layer encoder and decoder, alongside a GNN-processor comprising 4-6 layers, each with a hidden dimension of 128, results in the best performance. Furthermore, we find that a lookback window of two timesteps for forcings is adequate for achieving good performance with the GNN. Increasing this window size p yields only minimal improvements in performance, while significantly increasing the model's parameter count. Similarly, a lookback window of two days for previous predictions suffices, with

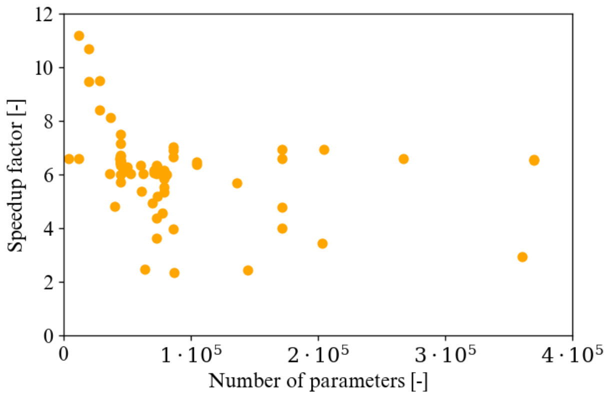


Figure 11. The number of parameters of an ensemble of GNN models and the subsequent speedup on a GPU.

.

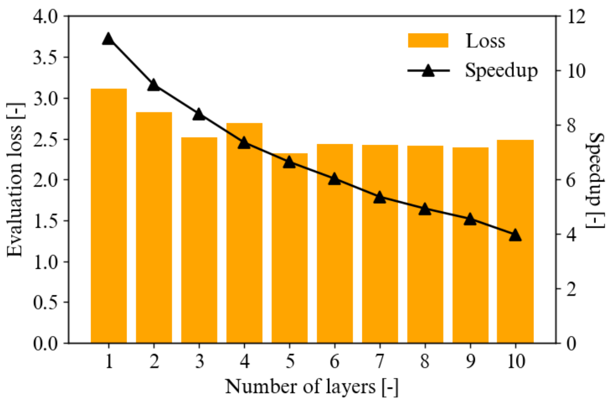


Figure 12. The Huber evaluation loss and the speedup on a GPU of GNN models with an increasing number of layers in the GNN processor. With the exception of the number of layers in the GNN processor, all other hyperparameters are left the same.

.

marginal gains in model performance observed when extending this window size q. Consequently, we consider both the forcings' lookback window p and the previous predictions' lookback window q to achieve their optimal value at two timesteps, balancing performance gains with parameter count and model runtime considerations.

On a GPU, the best GNN model is 6.65 times faster than wflow_sbm, while on a CPU it is twice as slow. Figure 11 illustrates the relationship between speedup and the number of parameters across an ensemble of GNN models. In general, we observe a trend where the speedup decreases as the number of parameters increases. However, notable outliers are evident, indicating that additional factors influence computational cost beyond just parameter count.

We also explore the relationship between the number of layers, model performance, and model runtime. Figure 12 illustrates the relationship between the number of layers within the GNN processor, the resulting speedup, and the Huber evaluation loss. As anticipated, the speedup of the GNN model declines as the number of layers in the GNN processor increases. Each additional layer introduces more parameters, thus increasing the computational cost of the GNN model. Surprisingly, the performance of the GNN model does not show significant improvement with an increase in layers. Although a slight decrease in loss is evident up to five layers, model performance begins to decline and then stabilize for six or more layers. Notably, it is intriguing to observe that a single-layer GNN model already achieves commendably low loss. This contradicts the conventional wisdom suggesting that more complex and deeper GNN models inherently yield superior performance.

5.2.2 Prediction over longer horizons

The GNN model initially relies on ground truth data as input (see also Figure 2). However, as it progresses in making predictions over multiple timesteps, it recursively uses its own output as input. Consequently, with a longer prediction horizon, the model increasingly relies on its own predictions rather than ground truth data. This transition can present a challenge for the GNN model, as errors may accumulate over time, potentially leading to a decrease in performance. To examine the impact of error accumulation on the GNN model's performance, we test the GNN model with increasing prediction horizons, consistently utilizing the same model trained with a prediction horizon of 60 days.

Figure 13 shows the evolution of the median KGE for each predicted variable as the prediction horizon increases during testing. For an initial prediction horizon of 90 days during testing, the GNN model demonstrates high performance, achieving a KGE exceeding 0.75 for all variables. However, as the prediction horizon is extended, the performance of the GNN model diminishes for most variables. The decline in model performance varies significantly across different variables. Incoming and outgoing subsurface flow, saturated and unsaturated storage, and runoff from land experience a rapid decline, although this decline appears to plateau to some extent after a prediction horizon of five years. In contrast, solid and liquid snow pack, actual evapotranspiration, and net runoff from river exhibit no decline in performance. Streamflow experiences only a minor decline.

One potential explanation for the sustained performance of certain variables over longer prediction horizons could be their minimal auto-correlation. For instance, actual evapotranspiration is primarily influenced by factors such as potential evapotranspiration and unsaturated storage, with less reliance on the evaporation of preceding days. Likewise, solid and liquid snow pack likely have minimal auto-correlation over longer time periods as well, due to the short-lived nature of snowpack in the testing region. Consequently, errors

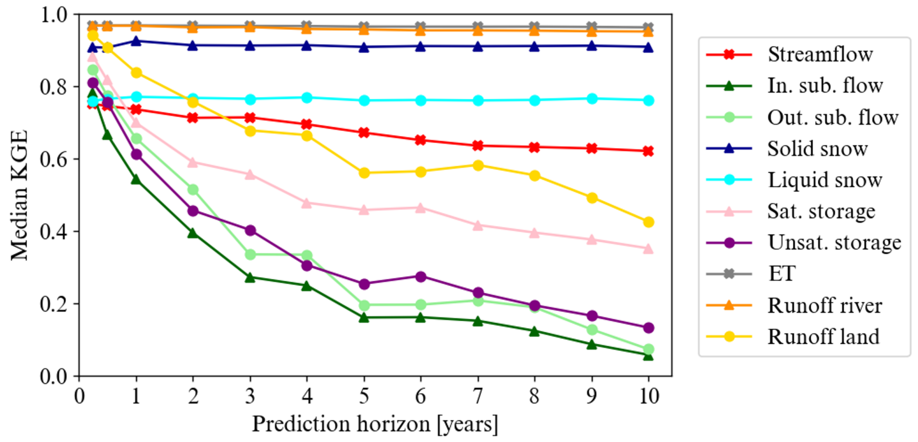


Figure 13. The evolution of the median KGE per variable across all cells in the testing data, as the prediction horizon is increased. The model was trained using a prediction horizon of 60 days.

in the GNN's predictions for these variables do not accumulate, as the model may not heavily depend on prior predictions to forecast subsequent timesteps. Variables exhibiting a more pronounced decline may possess higher levels of autocorrelation, rendering them more susceptible to the influence of errors in preceding predictions.

6 Concluding remarks research

6.1 Summary and conclusion

In this paper, we explored the potential of graph neural networks (GNNs) for the task of fully distributed multivariable hydrological modelling. We developed a surrogate GNN model of wflow_sbm, a physics-based fully distributed hydrological model. The GNN model is trained to approximate wflow_sbm outputs, taking the same inputs, i.e. catchment attribute data and forcing data. The GNN model autoregressively predicts several catchment storages and fluxes. We test the GNN model in previously unseen catchments located in northern England and southern Scotland, using a wide range of metrics to assess performance. Our results revealed substantial variations in model performance across different variables, accompanied by pronounced spatial patterns. Median KGE across all variables and gridcells was 0.76. Streamflow prediction at the outlets of the five biggest catchment was assessed using several metrics, uncovering significant variations in performance among different hydrological signatures, with a mean absolute bias of circa 10%. An inquiry of the water balance revealed substantial daily errors on gridcell level, yet only minor annual errors when aggregated over the entire area. While the GNN model tends to overpredict the runoff coefficient, it accurately models the evaporation index, indicating that it effectively closes the energy balance. Furthermore, the GNN model achieved a runtime speedup of 6.65 compared to wflow_sbm on a GPU. Adding more layers increased the runtime without significant improvement in performance. Moreover, expanding the prediction horizon during testing resulted in a reduction in model performance, albeit with noticeable variations across variables.

The main objective of our paper was to explore the potential of graph neural networks (GNNs) for the task of fully distributed multivariable hydrological modelling. We subdivided this objective into three key facets, namely 1) assessing the performance capabilities of GNNs, 2) investigating their runtime efficiency, and 3) identifying suitable model architectures and training configurations. Our key contributions address these facets as follows:

1. We demonstrate GNNs achieve high performance for the task of fully distributed multivariable hydrological modelling. We develop the first deep learning model to do so in unseen catchments.

2. We show that the GNN model can serve as a surrogate for traditional fully distributed models with shorter runtimes, although speedup is yet limited.

3. We demonstrate how the GNN model can function up to a prediction horizon of a full year, using physical system states to account for system memory, as well as a curriculum learning strategy combined with a multi-step ahead loss function during training.

In alignment with our hypothesis, we thus conclude that GNNs are a viable option for the task of fully distributed multivariable hydrological modelling.

6.2 Limitations and future research

6.2.1 GNN as emulator of wflow_sbm

In Section 1 we mentioned the runtime issues associated with traditional fully distributed models, such as wflow_sbm. We demonstrated that the proposed GNN model can serve as an accurate surrogate model of wflow_sbm with a speedup factor of 6.65. The GNN model could therefore be used as a rapid alternative to wflow_sbm in modelling scenarios with significant time constraints. The speedup factor identified in this study is notably lower than those reported in other studies exploring the use of GNNs as emulators (e.g. Bentivoglio et al., 2023; Choi and Kumar, 2024). Future work could focus on improving the computational runtime of the proposed GNN model.

Moreover, the GNN model could prove particularly advantageous in scenarios requiring multiple iterations with slight variations in inputs, such as ensemble forecasting and model calibration. In ensemble forecasting, this is done to account for uncertainty and provide a range of possible outcomes, whereas in model calibration, this is done to determine the optimal parameter set. Multiple sets of inputs can be combined into a single batch, enabling the GNN model to run them simultaneously with minimal additional runtime (You et al., 2017). For example, 128 distinct parameter sets could be combined into a single batch that the GNN model can process in a single run. Coupled with the existing speed up, this could potentially yield a computational speedup exceeding two orders of magnitude compared to wflow_sbm. As an emulator of wflow_sbm, the GNN model could significantly enhance wflow_sbm model calibration and ensemble forecasting, by enabling a significantly broader and more detailed range of inputs within the same time frame. Future work could focus on exploring the promising prospects of this method.

The study area of our research is located in northern England and southern Scotland, and is characterized by a temperate oceanic climate and subsequent mild winters and mild summers. The catchments of such regions tend to be easier to model compared to catchments with an arid or semi-arid climate (Wheater, 2008) or snow-dominated catchments (Klemes, 1989). Future research could thus explore training the GNN model on catchments from different climatic zones, as well as investigate the performance of the GNN model in these zones. Furthermore, the study area contains only small to mid-sized catchments, with the largest catchment spanning slightly under 5000 km^2. For instance, this catchment is still 37 times smaller than the Rhine catchment, which covers an area of 187.000 km^2. Subsequent research could thus explore the functionality and performance of the GNN model in larger catchments.

We used a spin-up period to provide the wflow_sbm model and subsequently the GNN model with initial states. We did not investigate the behaviour of the GNN model compared to wflow_sbm when no initial states are provided, nor did we investigate the capability of the GNN model to spin up independently from wflow_sbm. Subsequent research could explore this gap in our study.

The proposed GNN model was trained on a wflow_sbm model with a temporal resolution of a single day and a spatial resolution of 30 by 30 arc seconds. The spatial and temporal resolution of a wflow_sbm can be adjusted as needed, but we expect the GNN model to perform poorly on different spatio-temporal resolutions, as it was not trained on that. Forthcoming studies could investigate methods to incorporate predictions at different temporal resolutions, i.e. similar to Gauch et al. (2021), and different spatial resolutions.

We trained the GNN model to approximate ten different hydrological outputs from wflow_sbm, but wflow_sbm predicts many more variables, such as capillary rise, overland flow, percolation, and water ponding depth. Our flexible workflow can easily be adapted to include these additional outputs. Additionally, wflow_sbm can model glaciers, lakes, reservoirs, floodplains, and sediment transport. Future research could investigate the feasibility and potential of including these additional processes into the GNN model.

6.2.2 Incorporation of observation data

Throughout this study, we considered wflow_sbm output to be the ground truth, and we did not compare the predictions of either the GNN model or wflow_sbm to observation data. Regardless of the performance of wflow_sbm, there will be discrepancies between its predictions and the observation data. These mismatches between wflow_sbm and observation data, as well as between the GNN model and wflow_sbm, strongly imply that the disparity between the GNN model and observational data will be substantial. The integration of observation data into the GNN model is therefore an imperative future research direction.

To incorperate observation data, the GNN model could first be pre-trained on wflow_sbm outputs, followed by fine-tuning on observation data. Other studies have shown that fine-tuning a DL model after it has been pre-trained on a physics-guided model can significantly improve its performance (e.g. Sun et al., 2022; Jia et al., 2021b). The fine-tuning step can correct for biases and errors in the physics-guided models (Andersson et al., 2021). We omitted this step, as we wanted to limit the scope of this research to exploring how GNNs can be used for fully distributed multivariable hydrological modelling, instead of aiming for optimal GNN model performance.

Recent advancements and efforts in remote sensing technology are leading to a growing availability of the observation data required for the aforementioned fine-tuning process. For instance, satellite data can be used to derive actual evapotranspiration, saturated storage, unsaturated storage, and snow pack (Dembélé et al., 2020; Frei et al., 2012). Streamflow data is often available from gauges, but can also be derived from satellite data (Hulsman et al., 2020). Translating these remote sensing observations into model variables can pose challenges, as there may be underlying assumptions and margins of uncertainty in the observational data. During the fine-tuning phase of the training, wflow_sbm can be utilized to bridge gaps in cases where this is an issue, or if observation data is lacking altogether.

We recognize that developing a high-performing fully distributed multivariable GNN model capable of generalizing to unseen catchments and trained solely on observation data currently presents a significant challenge. Training the GNN model on a physics-guided model may for now be necessary to address issues related to the paucity of observation data (Kashinath et al., 2021), enhance its capacity to generate physically consistent results (Willard et al., 2022), and to improve its capability to generalize to unseen catchments (Read et al., 2019). Given the advancements in both deep learning and remote sensing, this could be an opportunity holding promise for future exploration.

In addition to incorporating observation data into the GNN model, it is also worthwhile to compare the model's predictions with observational data. This would enable a more comprehensive evaluation of the GNN model's performance as a hydrological model, while also providing a benchmark for comparing the GNN model to other deep learning and traditional models. Furthermore, since observation data provides, to a certain extent, a tangible ground truth, there exists an opportunity to benchmark alternative routing schemes. We did not consider alternative routing schemes, instead opting to mirror the default routing scheme of wflow_sbm. Future research could investigate alternative, possibly more sophisticated routing schemes (Cortés-Salazar et al., 2023).

6.2.3 Enhancing GNN architecture

The focus of our study was on exploring the potential of graph neural networks within hydrological modeling, rather than striving for an optimal GNN performance. Nonetheless, our preliminary findings demonstrate promising model performance even with a moderately limited hyperparameter search. It is plausible that a more thorough and comprehensive exploration of hyperparameters could yield a superior model architecture and training configuration, making it an intriguing direction for future research.

The dichotomy between land and river cells within a catchment, along with the varying speeds at which processes occur in these respective areas, underscores the need for a tailored GNN model capable of addressing these differences. In this context, the addition of pooling layers, which aggregate information over multiple nodes, emerges as a potential area for future work. Moreover, to further enhance information flow between indirectly connected river nodes, synthetic edges could be introduced, effectively creating a supplementary network pathway. Furthermore, there is potential for research to explore the potenrial of heterogeneous graph neural networks (HGNN Zhang et al., 2019). These models could leverage distinct node types for land and river cells, as well as different edge types for connections between land cells, between land and river cells, and between river cells.

Many papers on deep learning in hydrology benchmark their proposed model against several common deep learning models. We did not do this in this research, as we did not propose any new model. We simply applied already existing GNNs to show a proof of concept: GNNs are suitable for multivariable, fully distributed modelling with spatial transferability. However, we do think future could explore benchmarking GNNs for this task against other deep learning models. It could be interesting to see if other deep learning models are capable of achieving similar performance, especially since we think the graph is what makes GNNs so powerful for hydrological modelling.

Appendix A: Hyperparameter search

The hyperparameters we considered during our search can be seen in Table A1. The parameters resulting in the optimal performance are indicated in bold. These depend strongly on the task at hand, and even the data used. Thus, the results of our hyperparameter search are not directly transferable to other GNN tasks. Furthermore, the results are also co-dependent and must therefore be seen as a set.

We briefly elaborate on hyperparameters that have not been previously mentioned. The connection between layers in the GNN processor determines how output from some layer l is used as input by some layer $l+i$, where $i \in \mathbb{Z}^+$ is the size of the connection and $i+l \leq L$. In traditional feed-forward neural networks, layers are connected sequentially, and the output of layer l is used directly as input by layer $l+1$ (and i is thus always 1). With residual connections, the input of layer l is summed with the input of layer $l+i$ He et al. (2015). With dense connections, the input of layer l is concatenated with the input of layer $l+i$ (Huang et al., 2016). Both residual and dense connections allow for easier convergence during training, which is especially relevant for deeper GNNs (Balduzzi et al., 2017).

The encoder and decoder layers are MLP's, which contain linear layers. These linear layers apply to each of their inputs a weight and optionally a bias. Whether this bias was included is one of the hyperparameters. Dropout layers randomly mask elements of the input, resulting in less overfitting (Hinton et al., 2012). The probability of an element being randomly masked is one of the hyperparameters. Graphs can be directed or undirected. In a directed GNN, edges only allow one-way message-passing, whereas messages can travel both ways of an edge in an undirected GNN. Whether our GNN is directed or undirected is one of the hyperparameters.

The learning rate determines the step size when GNN model parameters are adjusted during backpropagation

(Murphy, 2012). A larger learning rate can cause the GNN model to converge quicker to the optimal solution, but can also lead to overshooting or oscillations around it. A smaller learning rate may lead to slower convergence but can also result in a more stable and accurate final solution. The learning rate is therefore an important hyperparameter. At the start of the training, the model parameters are still far from the optimal solution, and a larger learning rate can be employed. As training progresses and the optimal solution is approached, a smaller learning rate is desirable. To achieve this, a learning rate scheduler algorithm can be employed. These algorithms adjust the learning rate during the training. Which algorithm is employed during training is one of the hyperparameters. We found the StepLR algorithm to be the best scheduler, which multiplies the learning rate by some factor γ every fixed number of epochs. This multiplication factor γ is one of the hyperparameters. How many epochs are between each adjustment is known as the step size, and we choose to let this step size be dependent on the curriculum epoch.

Code and data availability. The code repository is available at github.com/PeterNelemans/GNN-Wflow-paper-repo (Nelemans and Bentivoglio, 2023). The Wflow modelling framework, including wflow_sbm, is available at github.com/Deltares/wflow.jl (van Verseveld et al., 2023). The HydroMT-Wflow plugin, used to construct the wflow_sbm model and to preprocess the data, is available at https://github.com/Deltares/hydromt_wflow (Eilander et al., 2023b). The data used by the wflow_sbm model and subsequently the GNN model is all freely available online. LAI data was provided by MODIS (Myneni et al., 2021). Hydrography data was provided by MERIT Hydro (Yamazaki et al., 2019). Meteorological data was provided by ERA5 (Hersbach et al., 2020). River width and backfull discharge is based on the dataset from Lin et al. (2020). Soil data was provided by SoilGrid (Poggio et al., 2021).

Competing interests. The authors declare that they have no conflict of interests.

Acknowledgements. The authors would like to thank AIdroLab and Deltares for making parts of this research possible.

Table A1. The hyperparameters considered during our hyperparameter search. Best option is indicated in bold.

Hyperparameter	**Range/options**
Lookback window forcings p	1 - 15 (**2**)
Lookback window storages and fluxes	1 - 15 (**2**)
Number of encoder layers	0 - 3 (**1**)
Number of GNN layers L	2 - 10 (**5**)
Number of decoder layers	0 - 3 (**1**)
GNN-layer	GCN, GAT, **SAGE**, Cheb, RGGC
Dimension of embedded features	32, 64, **128**
Connection between layers in processor	Sequential, Dense connection, **Residual connection**
Size of dense or residual connection	1 - 3 (**2**)
Bias in encoder and decoder	True, **False**
Layer normalization	True, **False**
Dropout	0 - 0.4 (**0**)
Activation function encoder and decoder	ReLU, **PReLU**, MLP tanh, ELU, GELU, sigmoid, LeakyReLU
Activation function GNN processor	ReLU, **PReLU**, MLP tanh, ELU, GELU, sigmoid, LeakyReLU
Undirected	True, **False**
Curriculum epoch	5, 10, **15**, 20, 25, 30
Learning rate	0.0001 - 0.01 (**0.002**)
Weight decay	10^{-6} - 10^{-4} ($\mathbf{2 \cdot 10^{-6}}$)
Scheduler algorithm	**StepLR**, LinearLR, ReduceOnPlateau, ExponentialLR, None
Multiplication factor γ	0.5 - 1 (**0.8**)
Step size relative to curriculum epoch	-10, -5, 0, **5**, 10

References

Ameli, A. A. and Creed, I. F.: Does Wetland Location Matter When Managing Wetlands for Watershed-Scale Flood and Drought Resilience?, JAWRA Journal of the American Water Resources Association, 55, 529–542, https://doi.org/10.1111/1752-1688.12737, 2019.

Andersson, T. R., Hosking, J. S., Pérez-Ortiz, M., Paige, B., Elliott, A., Russell, C., Law, S., Jones, D. C., Wilkinson, J., Phillips, T., Byrne, J., Tietsche, S., Sarojini, B. B., Blanchard-Wrigglesworth, E., Aksenov, Y., Downie, R., and Shuckburgh, E.: Seasonal Arctic sea ice forecasting with probabilistic deep learning, Nature Communications 2021 12:1, 12, 1–12, https://doi.org/10.1038/s41467-021-25257-4, 2021.

Bai, T. and Tahmasebi, P.: Graph neural network for groundwater level forecasting, Journal of Hydrology, 616, 128 792, https://doi.org/10.1016/J.JHYDROL.2022.128792, 2023.

Balduzzi, D., Frean, M., Leary, L., Lewis, J. P., Ma, K. W. D., and McWilliams, B.: The Shattered Gradients Problem: If resnets are the answer, then what is the question?, 34th International Conference on Machine Learning, ICML 2017, 1, 536–549, https://arxiv.org/abs/1702.08591v2, 2017.

Barling, R. D., Moore, I. D., and Grayson, R. B.: A quasi-dynamic wetness index for characterizing the spatial distribution of zones of surface saturation and soil water content, Water Resources Research, 30, 1029–1044, https://doi.org/10.1029/93WR03346, 1994.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gulcehre, C., Song, F., Ballard, A., Gilmer, J., Dahl, G., Vaswani, A., Allen, K., Nash, C., Langston, V., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M., Vinyals, O., Li, Y., and Pascanu, R.: Relational inductive biases, deep learning, and graph networks, https://arxiv.org/abs/1806.01261v3, 2018.

Beldring, S.: Multi-criteria validation of a precipitation–runoff model, Journal of Hydrology, 257, 189–211, https://doi.org/10.1016/S0022-1694(01)00541-8, 2002.

Bentivoglio, R., Isufi, E., Jonkman, S. N., and Taormina, R.: Rapid spatio-Temporal flood modelling via hydraulics-based graph neural networks, Hydrology and Earth System Sciences, 27, 4227–4246, https://doi.org/10.5194/HESS-27-4227-2023, 2023.

Beven, K.: Changing ideas in hydrology — The case of physically-based models, Journal of Hydrology, 105, 157–172, https://doi.org/10.1016/0022-1694(89)90101-7, 1989.

Beven, K.: Rainfall-Runoff Modelling: The Primer: Second Edition, Rainfall-Runoff Modelling: The Primer: Second Edition, pp. 1–457, https://doi.org/10.1002/9781119951001, 2012.

Bhatti, A. M., Koike, T., and Shrestha, M.: Climate change impact assessment on mountain snow hydrology by water and energy budget-based distributed hydrological model, Journal of Hydrology, 543, 523–541, https://doi.org/10.1016/J.JHYDROL.2016.10.025, 2016.

Biewald, L.: Experiment Tracking with Weights and Biases, https://www.wandb.com/, 2020.

Bindas, T., Tsai, W.-P., Liu, J., Rahmani, F., Feng, D., Bian, Y., Lawson, K., and Shen, C.: Improving large-basin streamflow simulation using a modular, differentiable, learnable graph model for routing, Authorea Preprints, https://doi.org/10.1002/essoar.10512512.1, 2022.

Blöschl, G., Reszler, C., and Komma, J.: A spatially distributed flash flood forecasting model, Environmental Modelling & Software, 23, 464–478, https://doi.org/10.1016/J.ENVSOFT.2007.06.010, 2008.

Blöschl, G., Sivapalan, M., Wagener, T., Viglione, A., and Savenije, H.: Runoff prediction in ungauged basins: Synthesis across processes, places and scales, Cambridge University Press, Cambridge, ISBN 9781139235761, https://doi.org/10.1017/CBO9781139235761, 2011.

Bresson, X. and Laurent, T.: Residual Gated Graph ConvNets, https://arxiv.org/abs/1711.07553v2, 2017.

Chen, J. M., Chen, X., Ju, W., and Geng, X.: Distributed hydrological model for mapping evapotranspiration using remote sensing inputs, Journal of Hydrology, 305, 15–39, https://doi.org/10.1016/J.JHYDROL.2004.08.029, 2005.

Chen, Y.: Distributed Hydrological Models, in: Handbook of Hydrometeorological Ensemble Forecasting, edited by Duan, Q., Pappenberger, F., Thielen, J., Wood, A., Cloke, H. L., and Schaake, J. C., pp. 413–436, Springer Berlin Heidelberg, Berlin, Heidelberg, ISBN 9783642399251, https://doi.org/10.1007/978-3-642-39925-1_23, 2019.

Chen, Y., Li, J., Wang, H., Qin, J., and Dong, L.: Large-watershed flood forecasting with high-resolution distributed hydrological model, Hydrology and Earth System Sciences, 21, 735–749, https://doi.org/10.5194/HESS-21-735-2017, 2017.

Choi, Y. and Kumar, K.: Graph Neural Network-based surrogate model for granular flows, Computers and Geotechnics, 166, 106 015, https://doi.org/10.1016/J.COMPGEO.2023.106015, 2024.

Cortés-Salazar, N., Vásquez, N., Mizukami, N., Mendoza, P. A., and Vargas, X.: To what extent does river routing matter in hydrological modeling?, Hydrology and Earth System Sciences, 27, 3505–3524, https://doi.org/10.5194/HESS-27-3505-2023, 2023.

Dai, X., Xie, Y., Simmons, C. T., Berg, S., Dong, Y., Yang, J., Love, A. J., Wang, C., and Wu, J.: Understanding topography-driven groundwater flow using fully-coupled surface-water and groundwater modeling, Journal of Hydrology, 594, 125 950, https://doi.org/10.1016/J.JHYDROL.2020.125950, 2021.

Daigavane, A., Ravindran, B., and Aggarwal, G.: Understanding Convolutions on Graphs, Distill, 6, e32, https://doi.org/10.23915/DISTILL.00032, 2021.

De Lavenne, A., Andréassian, V., Crochemore, L., Lindström, G., and Arheimer, B.: Quantifying multi-year hydrological memory with Catchment Forgetting Curves, Hydrology and Earth System Sciences, 26, 2715–2732, https://doi.org/10.5194/HESS-26-2715-2022, 2022.

Defferrard, M., Bresson, X., and Vandergheynst, P.: Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering, Advances in Neural Information Processing Systems, pp. 3844–3852, https://arxiv.org/abs/1606.09375v3, 2016.

Delft High Performance Computing Centre: DelftBlue Supercomputer (Phase 1), https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase1, 2022.

Dembélé, M., Hrachowitz, M., Savenije, H. H., Mariéthoz, G., and Schaefli, B.: Improving the Predictive Skill of a Distributed Hydrological Model by Calibration on Spatial Patterns With Multiple Satellite Data Sets, Water Resources Research, 56, e2019WR026 085, https://doi.org/10.1029/2019WR026085, 2020.

Deng, J., Couasnon, A., Dahm, R., Hrachowitz, M., van Heeringen, K.-J., Korving, H., Weerts, A., and Taormina, R.: Operational low-flow forecasting using LSTMs, Frontiers in Water, 5, 1332 678, https://doi.org/10.3389/FRWA.2023.1332678, 2024.

Dong, C.: Remote sensing, hydrological modeling and in situ observations in snow cover research: A review, Journal of Hydrology, 561, 573–583, https://doi.org/10.1016/J.JHYDROL.2018.04.027, 2018.

Dunn, S. M. and Colohan, R. J.: Developing the snow component of a distributed hydrological model: a step-wise approach based on multi-objective analysis, Journal of Hydrology, 223, 1–16, https://doi.org/10.1016/S0022-1694(99)00095-5, 1999.

Eilander, D., Boisgontier, H., Bouaziz, L. J. E., Buitink, J., Couasnon, A., Dalmijn, B., Hegnauer, M., de Jong, T., Loos, S., Marth, I., and van Verseveld, W.: HydroMT: Automated and reproducible model building and analysis, https://doi.org/10.5281/ZENODO.7827521, 2023a.

Eilander, D., Boisgontier, H., van Verseveld, W., Bouaziz, L., Hegnauer, M., Buitink, J., and Dalmijn, B.: hydroMT-wflow, https://doi.org/10.5281/zenodo.10185533, 2023b.

Fey, M. and Lenssen, J. E.: Fast Graph Representation Learning with PyTorch Geometric, 2019.

Francés, F., Vélez, J. I., and Vélez, J. J.: Split-parameter structure for the automatic calibration of distributed hydrological models, Journal of Hydrology, 332, 226–240, https://doi.org/10.1016/J.JHYDROL.2006.06.032, 2007.

Frei, A., Tedesco, M., Lee, S., Foster, J., Hall, D. K., Kelly, R., and Robinson, D. A.: A review of global satellite-derived snow products, Advances in Space Research, 50, 1007–1029, https://doi.org/10.1016/J.ASR.2011.12.021, 2012.

Gao, H., Hrachowitz, M., Sriwongsitanon, N., Fenicia, F., Gharari, S., and Savenije, H. H.: Accounting for the influence of vegetation and landscape improves model transferability in a tropical savannah region, Water Resources Research, 52, 7999–8022, https://doi.org/10.1002/2016WR019574, 2016.

Gao, L. and Lo, D.: A review of hydrological/water-quality models, Frontiers of Agricultural Science and Engineering, 1, 267–276, https://doi.org/10.15302/J-FASE-2014041, 2015.

Garrote, L. and Bras, R. L.: A distributed model for real-time flood forecasting using digital elevation models, Journal of Hydrology, 167, 279–306, https://doi.org/10.1016/0022-1694(94)02592-Y, 1995.

Gauch, M., Kratzert, F., Klotz, D., Nearing, G., Lin, J., and Hochreiter, S.: Rainfall-runoff prediction at multiple timescales with a single Long Short-Term Memory network, Hydrology and Earth System Sciences, 25, 2045–2062, https://doi.org/10.5194/hess-25-2045-2021, 2021.

Gharari, S. and Razavi, S.: A review and synthesis of hysteresis in hydrology and hydrological modeling: Memory, path-dependency, or missing physics?, Journal of Hydrology, 566, 500–519, https://doi.org/10.1016/J.JHYDROL.2018.06.037, 2018.

Giardino, A., Schrijvershof, R., Nederhoff, C. M., de Vroeg, H., Brière, C., Tonnon, P. K., Caires, S., Walstra, D. J., Sosa, J., van Verseveld, W., Schellekens, J., and Sloff, C. J.: A quantitative assessment of human interventions and climate change on the West African sediment budget, Ocean & Coastal Management, 156, 249–265, https://doi.org/10.1016/J.OCECOAMAN.2017.11.008, 2018.

Gichamo, T. Z., Sazib, N. S., Tarboton, D. G., and Dash, P.: HydroDS: Data services in support of physically based, distributed hydrological models, Environmental Modelling & Software, 125, 104 623, https://doi.org/10.1016/J.ENVSOFT.2020.104623, 2020.

Gupta, H. V., Kling, H., Yilmaz, K. K., Martinez, G. F., and Kling, H.: Decomposition of the mean squared error and NSE performance criteria: Implications for improving hydrological modelling, Journal of Hydrology, 377, 80–91, https://doi.org/10.1016/j.jhydrol.2009.08.003, 2009.

Gupta, H. V., Perrin, C., Blöschl, G., Montanari, A., Kumar, R., Clark, M., and Andréassian, V.: Large-sample hydrology: A need to balance depth with breadth, Hydrology and Earth System Sciences, 18, 463–477, https://doi.org/10.5194/HESS-18-463-2014, 2014.

Hamilton, W. L., Ying, R., and Leskovec, J.: Inductive Representation Learning on Large Graphs, Advances in Neural Information Processing Systems, 2017-December, 1025–1035, https://arxiv.org/abs/1706.02216v4, 2017.

He, K., Zhang, X., Ren, S., and Sun, J.: Deep Residual Learning for Image Recognition, Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016-December, 770–778, https://doi.org/10.1109/CVPR.2016.90, 2015.

Hersbach, H., Bell, B., Berrisford, P., Hirahara, S., Horányi, A., Muñoz-Sabater, J., Nicolas, J., Peubey, C., Radu, R., Schepers, D., Simmons, A., Soci, C., Abdalla, S., Abellan, X., Balsamo, G., Bechtold, P., Biavati, G., Bidlot, J., Bonavita, M., De Chiara, G., Dahlgren, P., Dee, D., Diamantakis, M., Dragani, R., Flemming, J., Forbes, R., Fuentes, M., Geer, A., Haimberger, L., Healy, S., Hogan, R. J., Hólm, E., Janisková, M., Keeley, S., Laloyaux, P., Lopez, P., Lupu, C., Radnoti, G., de Rosnay, P., Rozum, I., Vamborg, F., Villaume, S., and Thépaut, J. N.: The ERA5 global reanalysis, Quarterly Journal of the Royal Meteorological Society, 146, 1999–2049, https://doi.org/10.1002/QJ.3803, 2020.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R.: Improving neural networks by preventing co-adaptation of feature detectors, https://arxiv.org/abs/1207.0580v1, 2012.

Höge, M., Scheidegger, A., Baity-Jesi, M., Albert, C., and Fenicia, F.: Improving hydrologic models for predictions and process understanding using neural ODEs, Hydrology and Earth System Sciences, 26, 5085–5102, https://doi.org/10.5194/hess-26-5085-2022, 2022.

Hoyer, S. and Hamman, J.: xarray: N-D labeled arrays and datasets in Python, Journal of Open Research Software, 5, https://doi.org/10.5334/jors.148, 2017.

Hrachowitz, M., Savenije, H. H., Blöschl, G., McDonnell, J. J., Sivapalan, M., Pomeroy, J. W., Arheimer, B., Blume, T., Clark, M. P., Ehret, U., Fenicia, F., Freer, J. E., Gelfan, A., Gupta, H. V., Hughes, D. A., Hut, R. W., Montanari, A., Pande, S., Tetzlaff, D., Troch, P. A., Uhlenbrook, S., Wagener, T., Winsemius, H. C., Woods, R. A., Zehe, E., and Cudennec, C.: A decade of Predictions in Ungauged Basins (PUB)—a review, Hydrological Sciences Journal, 58, 1198–1255, https://doi.org/10.1080/02626667.2013.803183, 2013.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q.: Densely Connected Convolutional Networks, Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, 2017-January, 2261–2269, https://doi.org/10.1109/CVPR.2017.243, 2016.

Huber, P. J.: Robust Estimation of a Location Parameter, https://doi.org/10.1214/aoms/1177703732, 35, 73–101, https://doi.org/10.1214/AOMS/1177703732, 1964.

Hui, W., Yongbo, L., Junzhi, L., and A-Xing, Z.: Representation of Agricultural Best Management Practices in a Fully Distributed Hydrologic Model: A Case Study in the Luoyugou Watershed, https://doi.org/10.5814/j.issn.1674-764X.2014.02.011, 5, 179–184, https://doi.org/10.5814/J.ISSN.1674-764X.2014.02.011, 2014.

Hulsman, P., Winsemius, H. C., Michailovsky, C. I., Savenije, H. H., and Hrachowitz, M.: Using altimetry observations combined with GRACE to select parameter sets of a hydrological model in a data-scarce region, Hydrology and Earth System Sciences, 24, 3331–3359, https://doi.org/10.5194/HESS-24-3331-2020, 2020.

Hunter, J. D.: Matplotlib: A 2D graphics environment, Computing in Science & Engineering, 9, 90–95, https://doi.org/10.1109/MCSE.2007.55, 2007.

Jia, X., Xie, Y., Li, S., Chen, S., Zwart, J., Sadler, J., Appling, A., Oliver, S., and Read, J.: Physics-Guided Machine Learning from Simulation Data: An Application in Modeling Lake and River Systems, Proceedings - IEEE International Conference on Data Mining, ICDM, 2021-December, 270–279, https://doi.org/10.1109/ICDM51629.2021.00037, 2021a.

Jia, X., Zwart, J., Sadler, J., Appling, A., Oliver, S., Markstrom, S., Willard, J., Xu, S., Steinbach, M., Read, J., and Kumar, V.: Physics-guided recurrent graph model for predicting flow and temperature in river networks, Proceedings, pp. 612–620, https://doi.org/10.1137/1.9781611976700.69, 2021b.

Kashinath, K., Mustafa, M., Albert, A., Wu, J. L., Jiang, C., Esmaeilzadeh, S., Azizzadenesheli, K., Wang, R., Chattopadhyay, A., Singh, A., Manepalli, A., Chirila, D., Yu, R., Walters, R., White, B., Xiao, H., Tchelepi, H. A., Marcus, P., Anandkumar, A., Hassanzadeh, P., and Prabhat: Physics-informed machine learning: case studies for weather and climate modelling, Philosophical Transactions of the Royal Society A, 379, https://doi.org/10.1098/RSTA.2020.0093, 2021.

Khakbaz, B., Imam, B., Hsu, K., and Sorooshian, S.: From lumped to distributed via semi-distributed: Calibration strategies for semi-distributed hydrologic models, Journal of Hydrology, 418-419, 61–77, https://doi.org/10.1016/J.JHYDROL.2009.02.021, 2012.

Kipf, T. N. and Welling, M.: Semi-Supervised Classification with Graph Convolutional Networks, 5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings, https://arxiv.org/abs/1609.02907v4, 2016.

Klemeš, V.: Dilettantism in hydrology: Transition or destiny?, Water Resources Research, 22, 177S–188S, https://doi.org/10.1029/WR022I09SP0177S, 1986.

Klemes, V.: The modelling of mountain hydrology: the ultimate challenge, IAHS-AISH publication, 190, 29–43, 1989.

Knoben, W. J., Freer, J. E., and Woods, R. A.: Technical note: Inherent benchmark or not? Comparing Nash-Sutcliffe and Kling-Gupta efficiency scores, Hydrology and Earth System Sciences, 23, 4323–4331, https://doi.org/10.5194/HESS-23-4323-2019, 2019.

Kollet, S. J. and Maxwell, R. M.: Integrated surface–groundwater flow modeling: A free-surface overland flow boundary condition in a parallel groundwater flow model, Advances in Water Resources, 29, 945–958, https://doi.org/10.1016/J.ADVWATRES.2005.08.006, 2006.

Kratzert, F., Herrnegger, M., Klotz, D., Hochreiter, S., and Klambauer, G.: NeuralHydrology – Interpreting LSTMs in Hydrology, in: Explainable AI: Interpreting, Explaining and Visualizing Deep Learning, edited by Samek Wojciech
}and Montavon, G., Andrea, V., Kai, H. L., and Klaus-Robert, M., pp. 347–362, Springer International Publishing, Cham, ISBN 978-3-030-28954-6, https://doi.org/10.1007/978-3-030-28954-6_19, 2019.

Kratzert, F., Gauch, M., Nearing, G., and Klotz, D.: NeuralHydrology — A Python library for Deep Learning research in hydrology, Journal of Open Source Software, 7, 4050, https://doi.org/10.21105/joss.04050, 2022.

Li, G., Muller, M., Thabet, A., and Ghanem, B.: DeepGCNs: Can GCNs Go As Deep As CNNs?, https://sites.google.com/view/deep-gcns, 2019.

Li, K., Huang, G., Wang, S., and Razavi, S.: Development of a physics-informed data-driven model for gaining insights into hydrological processes in irrigated watersheds, Journal of Hydrology, 613, 128 323, https://doi.org/10.1016/J.JHYDROL.2022.128323, 2022.

Lin, P., Pan, M., Allen, G. H., de Frasson, R. P., Zeng, Z., Yamazaki, D., and Wood, E. F.: Global Estimates of Reach-Level Bankfull River Width Leveraging Big Data Geospatial Analysis, Geophysical Research Letters, 47, e2019GL086 405, https://doi.org/10.1029/2019GL086405, 2020.

Liu, Y., Hou, G., Huang, F., Qin, H., Wang, B., and Yi, L.: Directed graph deep neural network for multi-step daily streamflow forecasting, Journal of Hydrology, 607, 127 515, https://doi.org/10.1016/J.JHYDROL.2022.127515, 2022.

Liu, Z., Martina, M. L., and Todini, E.: Flood forecasting using a fully distributed model: application of the TOPKAPI model to the Upper Xixian Catchment, Hydrology and Earth System Sciences, 9, 347–364, https://doi.org/10.5194/HESS-9-347-2005, 2005.

Liuzzo, L., Noto, L. V., Vivoni, E. R., and Loggia, G. L.: Basin-Scale Water Resources Assessment in Oklahoma under Synthetic Climate Change Scenarios Using a Fully Distributed Hydrologic Model, Journal of Hydrologic Engineering, 15, 107–122, https://doi.org/10.1061/(ASCE)HE.1943-5584.0000166, 2009.

López, P. L., Sutanudjaja, E. H., Schellekens, J., Sterk, G., and Bierkens, M. F.: Calibration of a large-scale hydrological model using satellite-based soil moisture and evapotranspiration products, Hydrology and Earth System Sciences, 21, 3125–3144, https://doi.org/10.5194/HESS-21-3125-2017, 2017.

Loshchilov, I. and Hutter, F.: Decoupled Weight Decay Regularization, 7th International Conference on Learning Representations, ICLR 2019, https://arxiv.org/abs/1711.05101v3, 2017.

Lu, D., Konapala, G., Painter, S. L., Kao, S. C., and Gangrade, S.: Streamflow Simulation in Data-Scarce Basins Using Bayesian and Physics-Informed Machine Learning Models, Journal of Hydrometeorology, 22, 1421–1438, https://doi.org/10.1175/JHM-D-20-0082.1, 2021.

Maxwell, R. M., Condon, L. E., and Melchior, P.: A Physics-Informed, Machine Learning Emulator of a 2D Surface Water Model: What Temporal Networks and Simulation-Based Inference Can Help Us Learn about Hydrologic Processes, Water 2021, Vol. 13, Page 3633, 13, 3633, https://doi.org/10.3390/W13243633, 2021.

Mayes, J. and Wheeler, D.: Regional climates of the British Isles, Routledge, 2002.

McCabe, M. F., Rodell, M., Alsdorf, D. E., Miralles, D. G., Uijlenhoet, R., Wagner, W., Lucieer, A., Houborg, R., Verhoest, N. E., Franz, T. E., Shi, J., Gao, H., and Wood, E. F.: The future of Earth observation in hydrology, Hydrology and Earth System Sciences, 21, 3879–3914, https://doi.org/10.5194/HESS-21-3879-2017, 2017.

Mendoza, P. A., McPhee, J., and Vargas, X.: Uncertainty in flood forecasting: A distributed modeling approach in a sparse data catchment, Water Resources Research, 48, https://doi.org/10.1029/2011WR011089, 2012.

Merz, R. and Blöschl, G.: Regionalisation of catchment model parameters, Journal of Hydrology, 287, 95–123, https://doi.org/10.1016/J.JHYDROL.2003.09.028, 2004.

Mishra, A. K. and Singh, V. P.: Drought modeling – A review, Journal of Hydrology, 403, 157–175, https://doi.org/10.1016/J.JHYDROL.2011.03.049, 2011.

Murphy, K. P.: Machine learning: a probabilistic perspective, MIT press, 2012.

Myneni, R., Knyazikhin, Y., and Park, T.: MODIS/Terra Leaf Area Index/FPAR 8-Day L4 Global 500m SIN Grid V061, https://doi.org/10.5067/MODIS/MOD15A2H.061, 2021.

Nash, J. E. and Sutcliffe, J. V.: River flow forecasting through conceptual models part I — A discussion of principles, Journal of Hydrology, 10, 282–290, https://doi.org/10.1016/0022-1694(70)90255-6, 1970.

Nelemans, P. I. and Bentivoglio, R.: GNN-Wflow, https://github.com/PeterNelemans/GNN-Wflow-paper-repo, 2023.

Nonnenmacher, M. and Greenberg, D. S.: Deep Emulators for Differentiation, Forecasting, and Parametrization in Earth Science Simulators, Journal of Advances in Modeling Earth Systems, 13, e2021MS002 554, https://doi.org/10.1029/2021MS002554, 2021.

Parajka, J., Merz, R., and Blöschl, G.: A comparison of regionalisation methods for catchment model parameters, Hydrology and Earth System Sciences, 9, 157–171, https://doi.org/10.5194/HESS-9-157-2005, 2005.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S.: PyTorch: An Imperative Style, High-Performance Deep Learning Library, in: Advances in Neural Information Processing Systems 32, pp. 8024–8035, Curran Associates, Inc., https://doi.org/10.48550/arXiv.1912.01703, 2019.

Poggio, L., De Sousa, L. M., Batjes, N. H., Heuvelink, G. B., Kempen, B., Ribeiro, E., and Rossiter, D.: SoilGrids 2.0: Producing soil information for the globe with quantified spatial uncertainty, SOIL, 7, 217–240, https://doi.org/10.5194/SOIL-7-217-2021, 2021.

Poggio, T., Mhaskar, H., Rosasco, L., Miranda, B., and Liao, Q.: Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review, International Journal of Automation and Computing, 14, 503–519, https://doi.org/10.1007/S11633-017-1054-2/METRICS, 2017.

QGIS Development Team: QGIS Geographic Information System, http://qgis.osgeo.org, 2023.

Read, J. S., Jia, X., Willard, J., Appling, A. P., Zwart, J. A., Oliver, S. K., Karpatne, A., Hansen, G. J., Hanson, P. C., Watkins, W., Steinbach, M., and Kumar, V.: Process-Guided Deep Learning Predictions of Lake Water Temperature, Water Resources Research, 55, 9173–9190, https://doi.org/10.1029/2019WR024922, 2019.

Refsgaard, J. C.: Hydrological Modelling and River Basin Management, Ph.D. thesis, University of Copenhagen, Copenhagen, https://doi.org/10.1007/978-94-009-0257-2_2, 1996.

Reichstein, M., Camps-Valls, G., Stevens, B., Jung, M., Denzler, J., Carvalhais, N., and Prabhat: Deep learning and process understanding for data-driven Earth system science, Nature 2019 566:7743, 566, 195–204, https://doi.org/10.1038/s41586-019-0912-1, 2019.

Richards, K., Sharp, M., Arnold, N., Gurnell, A., Clark, M., Tranter, M., Nienow, P., Brown, G., Willis, I., and Lawson, W.: An integrated approach to modelling hydrology and water quality in glacierized catchments, Hydrological Processes, 10, 479–508, https://doi.org/https://doi.org/10.1002/(SICI)1099-1085(199604)10:4<479::AID-HYP406>3.0.CO;2-D, 1996.

Rode, M., Arhonditsis, G., Balin, D., Kebede, T., Krysanova, V., Van Griensven, A., and Van Der Zee, S. E.: New challenges in integrated water quality modelling, Hydrological Processes, 24, 3447–3461, https://doi.org/10.1002/HYP.7766, 2010.

Sanchez-Lengeling, B., Reif, E., Pearce, A., and Wiltschko, A.: A Gentle Introduction to Graph Neural Networks, Distill, 6, https://doi.org/10.23915/distill.00033, 2021.

Seibert, J., Bishop, K. H., and Nyberg, L.: A test of TOPMODEL's ability to predict spatially distributed groundwater levels, Hydrological Processes, 11, 1131–1144, 1997.

Seo, S. B., Mahinthakumar, G., Sankarasubramanian, A., and Kumar, M.: Conjunctive Management of Surface Water and Groundwater Resources under Drought Conditions Using a Fully Coupled Hydrological Model, Journal of Water Resources Planning and Management, 144, 04018 060, https://doi.org/10.1061/(ASCE)WR.1943-5452.0000978, 2018.

Shen, C.: A Transdisciplinary Review of Deep Learning Research and Its Relevance for Water Resources Scientists, Water Resources Research, 54, 8558–8593, https://doi.org/10.1029/2018WR022643, 2018.

Sit, M., Demiray, B. Z., Xiang, Z., Ewing, G. J., Sermet, Y., and Demir, I.: A comprehensive review of deep learning applications in hydrology and water resources, Water Science and Technology, 82, 2635–2670, https://doi.org/10.2166/WST.2020.369, 2020.

Sit, M., Demiray, B. Z., and Demir, I.: A Systematic Review of Deep Learning Applications in Streamflow Data Augmentation and Forecasting, https://doi.org/10.31223/X5HM08, 2022.

Sun, A. Y., Scanlon, B. R., Zhang, Z., Walling, D., Bhanja, S. N., Mukherjee, A., and Zhong, Z.: Combining Physically Based Modeling and Deep Learning for Fusing GRACE Satellite Data: Can We Learn From Mismatch?, Water Resources Research, 55, 1179–1195, https://doi.org/10.1029/2018WR023333, 2019.

Sun, A. Y., Jiang, P., Yang, Z. L., Xie, Y., and Chen, X.: A graph neural network (GNN) approach to basin-scale river network learning: the role of physics-based connectivity and data fusion, Hydrology and Earth System Sciences, 26, 5163–5184, https://doi.org/10.5194/HESS-26-5163-2022, 2022.

Tran, H., Leonarduzzi, E., De la Fuente, L., Hull, R. B., Bansal, V., Chennault, C., Gentine, P., Melchior, P., Condon, L. E., and Maxwell, R. M.: Development of a Deep Learning Emulator for a Distributed Groundwater–Surface Water Model: ParFlow-ML, Water 2021, Vol. 13, Page 3393, 13, 3393, https://doi.org/10.3390/W13233393, 2021.

Tripathy, K. P. and Mishra, A. K.: Deep learning in hydrology and water resources disciplines: concepts, methods, applications, and research directions, Journal of Hydrology, 628, 130 458, https://doi.org/10.1016/J.JHYDROL.2023.130458, 2024.

Van Rossum, G. and Drake Jr, F. L.: Python tutorial, vol. 620, Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.

van Verseveld, W., Visser, M., Boisgontier, H., Bootsma, H., Bouaziz, L., Buitink, J., Eilander, D., and Hegnauer, M.: Wflow.jl, https://doi.org/10.5281/zenodo.10201793, 2023.

van Verseveld, W. J., Weerts, A. H., Visser, M., Buitink, J., Imhoff, R. O., Boisgontier, H., Bouaziz, L., Eilander, D., Hegnauer, M., ten Velden, C., and Russell, B.: Wflow_sbm v0.6.1, a spatially distributed hydrologic model: from global data to local applications, Geoscientific Model Development Discussions, 2022, 1–52, https://doi.org/10.5194/gmd-2022-182, 2022.

Veličković, P., Casanova, A., Liò, P., Cucurull, G., Romero, A., and Bengio, Y.: Graph Attention Networks, 6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings, https://doi.org/10.1007/978-3-031-01587-8_7, 2017.

Vivoni, E. R., Mascaro, G., Mniszewski, S., Fasel, P., Springer, E. P., Ivanov, V. Y., and Bras, R. L.: Real-world hydrologic assessment of a fully-distributed hydrological model in a parallel computing environment, Journal of Hydrology, 409, 483–496, https://doi.org/10.1016/J.JHYDROL.2011.08.053, 2011.

Wanders, N., Bierkens, M. F., de Jong, S. M., de Roo, A., and Karssenberg, D.: The benefits of using remotely sensed soil moisture in parameter identification of large-scale hydrological models, Water Resources Research, 50, 6874–6891, https://doi.org/10.1002/2013WR014639, 2014.

Wang, Q., Ma, Y., Zhao, K., and Tian, Y.: A Comprehensive Survey of Loss Functions in Machine Learning, Annals of Data Science, 9, 187–212, https://doi.org/10.1007/S40745-020-00253-5/METRICS, 2022.

Wheater, H. S.: Modelling hydrological processes in arid and semi-arid areas: an introduction to the workshop, Hydrol. Model. Arid. Semi-Arid Areas, pp. 1–20, 2008.

Willard, J., Jia, X., Xu, S., Steinbach, M., and Kumar, V.: Integrating Scientific Knowledge with Machine Learning for Engineering and Environmental Systems, ACM Computing Surveys, 55, https://doi.org/10.1145/3514228, 2022.

Xiang, Z. and Demir, I.: Fully distributed rainfall-runoff modeling using spatial-temporal graph neural network, EarthArXiv Preprint, https://doi.org/10.31223/X57P74, 2022.

Xu, T. and Liang, F.: Machine learning for hydrologic sciences: An introductory overview, Wiley Interdisciplinary Reviews: Water, 8, e1533, https://doi.org/10.1002/WAT2.1533, 2021.

Yamazaki, D., Ikeshima, D., Sosa, J., Bates, P. D., Allen, G. H., and Pavelsky, T. M.: MERIT Hydro: A High-Resolution Global Hydrography Map Based on Latest Topography Dataset, Water Resources Research, 55, 5053–5073, https://doi.org/10.1029/2019WR024873, 2019.

Yang, C., Wu, Q., Wang, J., and Yan, J.: Graph Neural Networks are Inherently Good Generalizers: Insights by Bridging GNNs and MLPs, 2023.

Yang, T., Sun, F., Gentine, P., Liu, W., Wang, H., Yin, J., Du, M., and Liu, C.: Evaluation and machine learning improvement of global hydrological model-based flood simulations, Environmental Research Letters, 14, 114 027, https://doi.org/10.1088/1748-9326/AB4D5E, 2019.

Yilmaz, K. K., Gupta, H. V., and Wagener, T.: A process-based diagnostic approach to model evaluation: Application to the NWS distributed hydrologic model, Water Resources Research, 44, 9417, https://doi.org/10.1029/2007WR006716, 2008.

You, J., Ying, Z., and Leskovec, J.: Design Space for Graph Neural Networks, Advances in Neural Information Processing Systems, 33, 17 009–17 021, https://doi.org/https://doi.org/10.48550/arXiv.2011.08843, 2020.

You, Y., Gitman, I., and Ginsburg, B.: Scaling sgd batch size to 32k for imagenet training, arXiv preprint arXiv:1708.03888, 6, 6, 2017.

Yuval, J. and O'Gorman, P. A.: Stable machine-learning parameterization of subgrid processes for climate modeling at a range of resolutions, Nature Communications 2020 11:1, 11, 1–10, https://doi.org/10.1038/s41467-020-17142-3, 2020.

Zhang, C., Song, D., Huang, C., Swami, A., and Chawla, N. V.: Heterogeneous graph neural network, Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 793–803, https://doi.org/10.1145/3292500.3330961, 2019.

Zhao, J., Xu, J., Xie, X., and Lu, H.: Drought monitoring based on TIGGE and distributed hydrological model in Huaihe River Basin, China, Science of The Total Environment, 553, 358–365, https://doi.org/10.1016/J.SCITOTENV.2016.02.115, 2016.

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M.: Graph neural networks: A review of methods and applications, AI Open, 1, 57–81, https://doi.org/10.1016/j.aiopen.2021.01.001, 2020.