Assessing the performances and transferability of graph neural network metamodels for water distribution systems

Kerimov, Bulat; Bentivoglio, Roberto; Garzón, Alexander; Isufi, Elvin; Tscheikner-Gratl, Franz; Steffelbauer, David Bernhard; Taormina, Riccardo

DOI
10.2166/hydro.2023.031

Publication date
2023

Document Version
Final published version

Published in
Journal of Hydroinformatics

Journal of Hydroinformatics

Journal of Hydroinformatics Vol 25 No 6, 2223 doi: 10.2166/hydro.2023.031

Assessing the performances and transferability of graph neural network metamodels for water distribution systems

Bulat Kerimov [a,*,†], Roberto Bentivoglio [b,†], Alexander Garzón [b], Elvin Isufi [c], Franz Tscheikner-Gratl [a], David Bernhard Steffelbauer [d] and Riccardo Taormina [b]

[a] Department of Civil and Environmental Engineering, Norwegian University of Science and Technology, Trondheim, Norway
[b] Department of Water Management, Faculty of Civil Engineering and Geosciences, Delft University of Technology, Delft, The Netherlands
[c] Department of Intelligent Systems, Faculty of Electrical Engineering, Mathematics, and Computer Science, Delft University of Technology, Delft, The Netherlands
[d] Hydroinformatics Group, KWB – Kompetenzzentrum Wasser, Berlin, Germany
*Corresponding author. E-mail: bulat.kerimov@ntnu.no
[†]Equal contribution.

BK, 0000-0002-1037-0705; RB, 0000-0001-7992-4035; AG, 0000-0002-0012-7293; EI, 0000-0002-1919-260X; FT-G, 0000-0002-2545-6683; DBS, 0000-0003-2137-985X; RT, 0000-0002-1550-504X

ABSTRACT

Metamodels accurately reproduce the output of physics-based hydraulic models with a significant reduction in simulation times. They are widely employed in water distribution system (WDS) analysis since they enable computationally expensive applications in the design, control, and optimisation of water networks. Recent machine-learning-based metamodels grant improved fidelity and speed; however, they are only applicable to the water network they were trained on. To address this issue, we investigate graph neural networks (GNNs) as metamodels for WDSs. GNNs leverage the networked structure of WDS by learning shared coefficients and thus offering the potential of transferability. This work evaluates the suitability of GNNs as metamodels for estimating nodal pressures in steady-state EPANET simulations. We first compare the effectiveness of GNN metamodels against multi-layer perceptrons (MLPs) on several benchmark WDSs. Then, we explore the transferability of GNNs by training them concurrently on multiple WDSs. For each configuration, we calculate model accuracy and speedups with respect to the original numerical model. GNNs perform similarly to MLPs in terms of accuracy and take longer to execute but may still provide substantial speedup. Our preliminary results indicate that GNNs can learn shared representations across networks, although assessing the feasibility of truly general metamodels requires further work.

Key words: artificial intelligence, graph neural network, surrogate model, transfer learning, water distribution system, water network

HIGHLIGHTS

- The accuracy of GNN-based and MLP-based metamodels is comparable on most of the studied water networks.
- The proposed model can be trained on several water networks at once and can learn shared representation between them.
- By learning shared representations, the model achieves comparable performance while requiring fewer training examples.
- GNNs show promising results from transferability, although further study is required.

INTRODUCTION

Water utilities rely on hydrodynamic models to design and control water distribution systems. These physics-based models, such as EPANET (Rossman 2000), compute the state of the system, i.e., the flow rates and pressures at all the pipes and junctions, by solving the underlying equations of mass and energy conservation. The inputs for these computer programs include the layout of the network and the characteristics and settings of components such as pipes, pumps, valves, and reservoirs, among others. Hydrodynamic models provide valuable insight into the functioning of the system. However, the speed of these models is often insufficient for applications such as optimisation (e.g., Bi & Dandy 2014) or criticality assessment (e.g., Meijer *et al.* 2021), especially in large search space problems (Maier *et al.* 2014). One alternative to address this issue is developing surrogate models, also referred to as metamodels.

Metamodels are models that aim to significantly reduce simulation times while still obtaining comparable results to the hydrodynamic model. A main family of metamodels is response surface models (Razavi *et al.* 2012). These surrogate models mimic the input–output relation, i.e., the response surface, of the original physics-based model to obtain results in a fraction of the time while retaining sufficient accuracy. Among the multiple algorithms that can be used for creating these metamodels, artificial neural networks (ANNs) have been increasingly popular due to their high performance and execution speed; previous studies using mainly ANNs have shown remarkable gains in computational time (Broad *et al.* 2005; Martínez *et al.* 2007; Salomons *et al.* 2007; Behzadian *et al.* 2009; Broad *et al.* 2010).

ANNs are models obtained by stacking parametric functions that take an input x and produce an estimated output $\hat{y}$ from a target representation y. The parameters in an ANN are learned, i.e., calibrated, by minimising the difference between the expected and real targets, measured with a loss function. This calibration process is usually performed via backpropagation, i.e., the parameters change based on the value of the loss function. The process of learning these parameters is referred to as training and the data employed during training are called training sets. The model performances are then evaluated on a separate dataset known as a validation dataset, before being employed for testing on unseen data.

Fully connected ANNs, also known as multi-layer perceptrons (MLPs), are arguably the most used metamodels for water distribution systems (WDS). Even though MLPs can approximate any function (Hornik *et al.* 1989), their number of parameters increases exponentially with the size of the input, making them unsuitable for high-dimensional data. This issue is known as the 'curse of dimensionality' and implies the amount of training data required by an MLP increases exponentially with the input's dimensions (Lecun *et al.* 2015). Furthermore, MLPs require a fixed-size input, and consequently, a new model needs to be created when the size of the inputs changes, e.g., by adding new pipes or junctions to a WDS. Thus, they do not overcome a major limitation of traditional metamodels: they are only applicable to the water network they were trained on. As noted by Garzon *et al.* (2022), this implies that new metamodels must be trained with new sets of simulations to account for multiple networks or structural changes in the original system. This characteristic could discourage the use of metamodels or even make them impractical.

Components of metamodels can resemble the underlying structure of the problem at hand by including inductive bias, i.e., assumptions or knowledge about the data-generating process, underlying physical processes, or the space of solutions (Battaglia *et al.* 2018). This similitude aids the effectiveness of model transfer by exploiting the connectivity of the nodes and the physical information of the components. For metamodels in WDSs, this information includes connectivity and data such as node elevation, pipe roughness, and length.

Graph neural networks (GNN) are a recent variant of ANN which can perform operations on data that lie on graphs – mathematical objects that describe how entities are connected to each other via nodes and edges. WDSs can be represented by graphs, considering junctions, reservoirs, or storage tanks as nodes and pipes, pumps, or valves as edges. GNNs can then take the information embedded in WDS and apply the same linear and non-linear operations used in MLPs. The main difference is that GNNs can have permutation invariant and equivariant properties, which allows them to consider arbitrarily sized graphs. This inductive bias preserves the additional information embedded in the graph structure which helps in decreasing the number of trainable parameters in the metamodel.

Recently, GNNs found successful applications in water networks. Hajgató *et al.* (2021) and Xing & Sela (2022) used a GNN model to estimate the pressure state of a WDS, based on a few sensors in the network. Bonilla *et al.* (2022) employed a GNN model to predict the pump speed from pressure and flow measurements in the networks. However, no works yet explored the transferability of GNNs for WDSs, i.e., their ability to learn representations and perform predictions across multiple case studies. Furthermore, to the best of our knowledge, no studies on WDSs have compared the performance of GNNs against that of traditional data-driven alternatives, such as MLPs. In this paper, we move the first steps in these directions by developing GNN-based metamodels for estimating nodal pressures on six benchmark WDSs and comparing them against several MLP baselines.

The remainder of the paper is organised as follows. In the methodology, we firstly describe the data generation procedure and the six case studies used in this work. Next, we present the employed metamodels based on MLP and GNN and describe the metrics used for assessing their performances. The section additionally includes the description of the conducted experiments and the adopted setup. The result section presents the comparison between MLP- and GNN-based metamodels for the benchmark WDSs and discusses the results on GNN transferability. The last section concludes the paper.

METHODS

Case studies and data generation

To assess the performances and transferability of GNN-based metamodels, we consider the problem of reconstructing nodal pressures as computed by EPANET steady-state simulations. Table 1 reports the six benchmark datasets employed in this study along with their bibliographic reference, and the number of nodes, pipes, and reservoirs in the system. While the chosen WDSs do not feature hydraulic elements such as pumps and valves, they form a sufficiently diverse ensemble, as shown in the network layouts displayed in Figure 1.

For each of these networks, we employed the WNTR Python package (Klise *et al.* 2018) to generate a dataset of 10,000 samples, divided into training (8,000), validation (1,000), and test (1,000) subsets. Each sample is created by altering all (i) nodal base demands, (ii) pipe diameters, and (iii) pipe roughness coefficients. The altered values are selected from different distributions that reflect commercial ranges for pipe diameters (0–1.5 m at 2.5 cm increments) and Hazen–Williams roughness coefficients (50–150 at 1 unit increments), as well as reasonable base demands with respect to the selected case studies (0–100 L/s with 0.1 increments). To avoid unrealistic configurations leading to very low or very high pressures, the distributions were sampled within a range centred on each original node or pipe characteristics. While the distribution of pipe roughness and nodal-based demands are uniform within these ranges, the selection of pipe diameters is biased towards larger values to reduce the possibility of unfeasible setups (e.g., yielding failed simulation).

The described alterations ensured sufficient variability in the nodal pressures obtained via WNTR pressure-driven simulations. When running the simulations, we kept all other network characteristics constant, including network connectivity, geographical coordinates, elevation, pipe lengths, and boundary conditions (i.e., total head of the reservoirs).

Table 1 | Water distribution systems featured in this study

Name	ID	Reference	# nodes	# pipes	# reservoirs
Fossolo	FOS	Bragalli *et al.* (2012)	37	58	1
BakRyan	BAK	Lee & Lee (2001)	36	58	1
Pescara	PES	Bragalli *et al.* (2012)	71	99	3
Modena	MOD	Bragalli *et al.* (2012)	272	317	4
Marchi Rural	RUR	Marchi *et al.* (2014)	381	476	2
KL	KL	Kang & Lansey (2012)	936	1,274	1

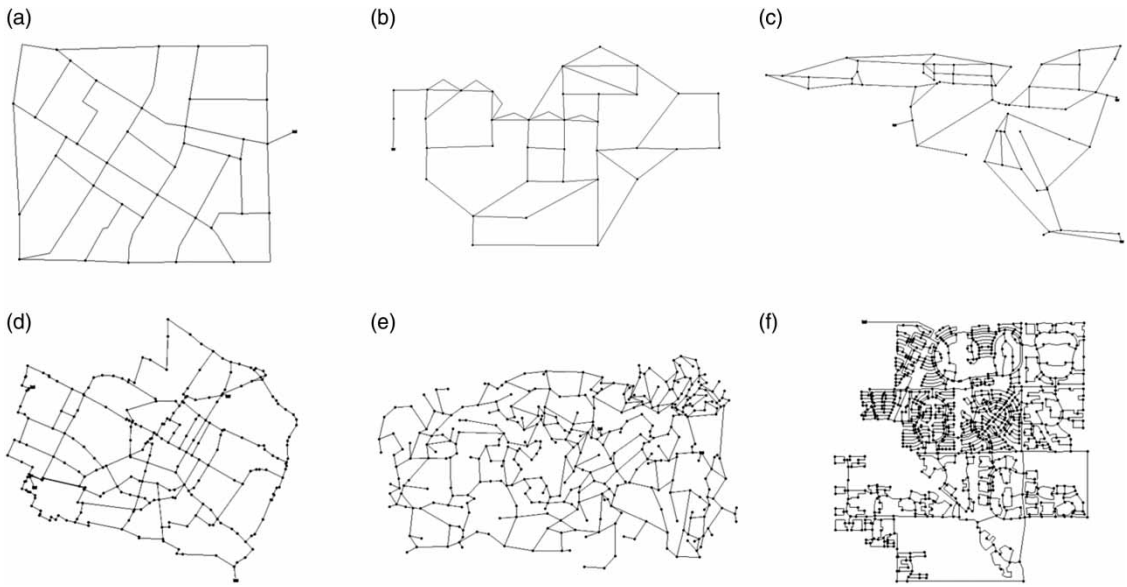

Figure 1 | Layout of selected case studies. (a) FOS, (b) BAK, (c) PES, (d) MOD, (e) RUR, and (f) KL.

ANN-based metamodels

MLPs are formed by sequences of linear operations between the input data and a parameter matrix, followed by non-linear activation functions (e.g., ReLU, sigmoid, tanh). The propagation rule for a generic MLP layer is

$$y_{l+1} = \sigma(W_l y_l + b_l) \tag{1}$$

where y_{l+1} is the layer output, $\sigma(\cdot)$ is an activation function, W_l is a trainable weight matrix, y_l is the layer input, and b_l is the bias term. Equation (1) is repeated for a certain number of layers, resulting in a fully connected network (see Figure 2). The number of inputs is given by the problem's variables, i.e., node demand, pipe diameters, and pipe roughness coefficient, while the outputs are the pressure estimate at each network's junction. Considering a generic WDS with N nodes and E pipes, the input X and output Y have thus dimensions $X \in \mathbb{R}^{N+2E}$ and $Y \in \mathbb{R}^{N}$, respectively.

Contrary to GNNs that can learn shared representations within the same WDS and across WDSs by exploiting topological and 'static' features, adding constant inputs to the MLP (e.g., elevations, pipe lengths) hinders its training process. This is carried out by means of gradient descent algorithms minimising a *loss function*; for the nodal regression problem entailed in our metamodelling approach, the loss is chosen as the mean squared error (MSE) between the pressure values of WNTR simulations and those predicted by the MLP averaged for the entire training dataset.

GNN-based metamodels

A WDS can be represented as an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \ldots, N\}$ is the set of nodes (e.g., junctions) and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges (e.g., pipes). For each node $i \in \mathcal{V}$, we can define its neighbourhood as $\mathcal{N}_i = \{j \text{ if } (i,j) \in \mathcal{E}\}$ as a set of nodes that are directly connected to a given node with an edge. Variables defined on the nodes, such as node demands, can be encoded as node signals and multiple variables can be represented by the node matrix $X \in \mathbb{R}^{N \times F}$, where F represents the number of node features. A GNN works by propagating those node features from one node to another via its neighbouring nodes. This propagation can be repeated, for a single layer, for as many K-hop neighbourhoods, as shown in Figure 3. The bigger this value, the wider the reach of information sharing throughout the graph. For mathematical convenience, we define the *graph shift operator* as a matrix $S \in \mathbb{R}^{N \times N}$, where $S_{ij} = S_{ij}$ *if* $(i,j) \in \mathcal{E}$ and $S_{ij} = 0$ otherwise. Thus, the expression for a graph convolutional layer is given by

$$Y_{l+1} = \sigma\left(\sum_{k}^{K} S Y_l H_l^k \right) \tag{2}$$

where Y_{l+1} is the layer output node matrix, σ is an activation function, Y_l is the layer input node matrix, H_l^k is a shared trainable matrix, and K is the K-hop neighbourhood (Gama *et al.* 2020).

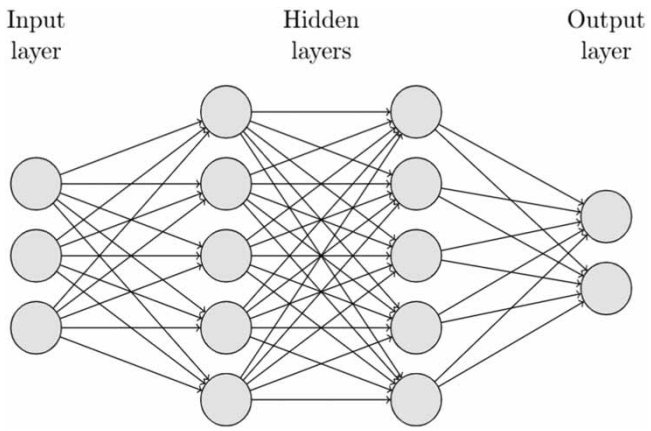

Figure 2 | A multi-layer perceptron (MLP) with two hidden layers. Every layer is connected to the following one by weights, represented by directed arrows.

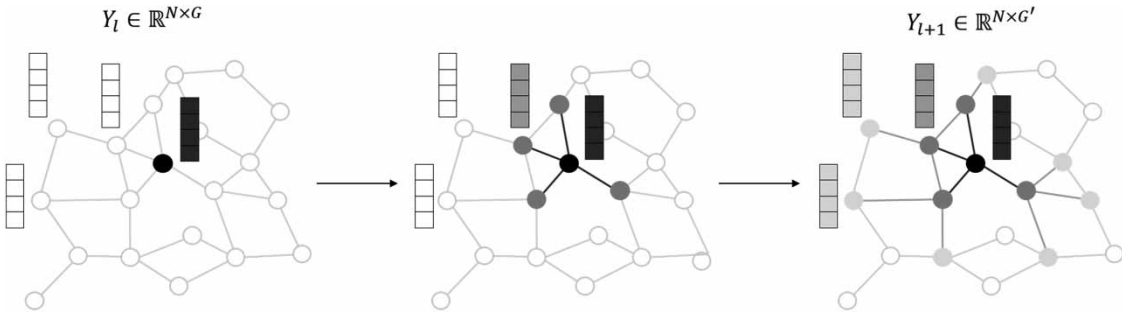

Figure 3 | A graph neural network (GNN) layer with a 2-hop neighbourhood. The figures from left to right indicate how the node signal in the black node propagates throughout the network. The same reasoning is applied to every other node in the graph.

Nodes and edges can be automatically inferred by the WDS connectivity. Each node has features defined by its elevation and base demand or base head, depending on whether the node is a junction or a reservoir. We included the edge attributes, i.e., diameter, length, and roughness of the pipes, via a preprocessing MLP layer shared by every edge. The obtained output is then aggregated for every node to create a new node embedding $Y_0 \in \mathbb{R}^{N\times G}$, where G is the embedding dimension, as illustrated in Figure 4. As for the GNN layer, we employed the ChebNet graph convolutional model (Defferrard *et al.* 2016), which has been extensively studied in previous works, including some in water networks (Hajgató *et al.* 2021). This considers as graph shift operator S in Equation (2) the Laplacian matrix, defined as $L = I - D^{-1/2}AD^{1/2}$obtained from the adjacency matrix A. Finally, we use an MLP shared on the nodes to convert the final node embeddings into a single pressure prediction for each node. Similarly to the MLP-based metamodel described in the previous section, the GNN is trained by minimising the MSE for the training dataset using stochastic gradient descent.

Metamodel performance

We consider the coefficient of determination R^2 and the root mean squared error (RMSE) as goodness-of-fit metrics. The former is defined by

$$R^2 = 1 - \frac{\sum_i^N (y_i - \hat{y}_i)^2}{\sum_i^N (y_i - \bar{y})^2} \tag{3}$$

where y_i is the target pressure at node i computed by WNTR, $\hat{y}_i$ is the predicted value of the metamodel, and $\bar{y}$ is the mean of the target pressure data for a given dataset of size N. A value of 1.0 indicates a perfect fit, while values of R^2 below 0 indicate that the model is performing worse than simply assuming the dataset mean for each point (i.e., $R^2 = 0$).

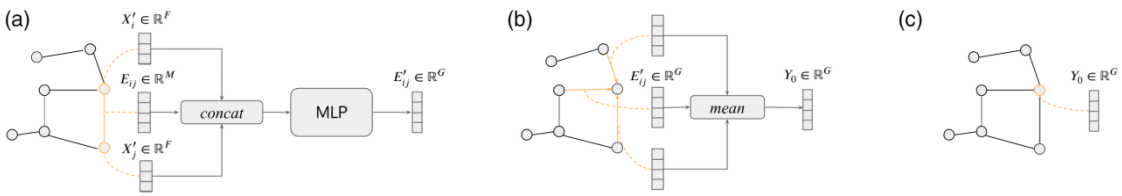

Figure 4 | Procedure to determine the node embeddings Y_0. (a) The edge attributes E_{ij} and node attributes of the incident nodes X_i and X_j are concatenated and then fed to an MLP shared across all edges, to determine an edge embedding E'_{ij}. (b) For every node, the embeddings of incidental edges E'_{ij} are aggregated to determine the final node embedding Y_0. (c) Node embeddings for every node are finally fed to the model.

On the other hand, the RMSE measures the average error in the pressure prediction defined as

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i}^{N}(y_i - \hat{y}_i)^2} \tag{4}$$

Since metamodels are built to overcome the computational costs of physics-based simulators, we evaluate the speedup provided by each metamodel as

$$\text{speedup} = \frac{\bar{T}_{\text{s}}}{\bar{T}_{\text{mm}}} \tag{5}$$

where $\bar{T}_{\text{s}}$ is the average computational time for the numerical simulation, and $\bar{T}_{\text{mm}}$ is the average computational time for the metamodel execution.

Experimental setup

We run two main sets of experiments aimed at (i) assessing the performances of GNN metamodels trained on individual WDSs, and (ii) exploring the advantages of a transferable GNN metamodel trained on datasets featuring samples of all WDSs. We compare the results against that of MLP-based metamodels trained on each WDS separately.

The comparison of MLP- vs GNN-based metamodels on individual WDSs is carried out by training the models on the entire 8,000 samples available for each water network. To facilitate the training process, all variables are scaled using *log* transformations (elevation, pressure, and pipe length) or min–max scaling (all other features). The scaling parameters derived from the training dataset are applied to the validation and test datasets. We decided to substitute the reservoirs' elevation (set to 0 by default in EPANET solvers) with their base hydraulic head and replace this latter feature with a Boolean flag to discriminate reservoirs from junctions. All GNN datasets thus have three features per node (elevation + base head, base demand, node Boolean flag) and three per edge (pipe diameter, pipe length, pipe roughness).

The study on GNN transferability entails three separate training datasets built using samples from all WDSs as shown in Table 2. The first dataset contains 1,024 samples for each WDS, for a total of 6,144 data points. Since the GNN loss function is calculated per node, this dataset is *unbalanced*, as it overrepresents larger networks. To balance the dataset, we undersample the larger WDSs by including a number of data points that is inversely proportional to the number of nodes in the WDS with respect to the smallest systems (FOS, BAK). Using this strategy, we create two extra datasets named *balanced* and *balanced extended* with 5,722 and 22,350 data points, respectively. All training datasets in Table 2 are normalised using the same procedure described before for the individual WDS datasets, but with extreme values computed across all WDSs. Similarly, the derived scaling parameters are then applied to normalize the validation and test datasets, now consisting of data from all WDSs.

Hyperparameter search

We test MLP architectures of different complexity by changing the number of hidden layers, as well as the number of units in each layer. In general, a larger network performs better but requires additional data and computational power. We also employ dropout layers (Srivastava *et al.* 2014) after each fully-connected layer to improve model generalisation. All activation functions are rectified linear units (ReLU). Table 3 shows the hyperparameters' range selected for MLPs, yielding 36 potential combinations. The upper limits of the hyperparameters are selected based on similar works in the water distribution system domain (Martínez *et al.* 2007; Hajgató *et al.* 2021).

Table 2 | Training datasets used for the study on transferability

Name	FOS	BAK	PES	MOD	RUR	KL	Total
Unbalanced	1,024	1,024	1,024	1,024	1,024	1,024	6,144
Balanced	2,048	2,048	1,067	279	199	81	5,722
Balanced extended	8,000	8,000	4,169	1,088	777	316	22,350

Table 3 | Range of hyperparameters for the MLP models

Hyperparameter	Range
Number of hidden layers	1, 2, 3, 4
Hidden layers dimension	64, 128, 256
Dropout rate	0, 0.1, 0.25

Table 4 | Range of hyperparameters for the GNN models

Hyperparameter	Range
Embedding dimension	32, 64
Number of convolutional layers	1, 2, 3
Hidden layers dimension	64, 128
K-hop neighbourhood	3, 6

Table 4 shows the values chosen for the optimisation of the GNN hyperparameters, yielding 24 possible combinations. The hyperparameters include the embedding dimensions of the shared preprocessing MLP, the number of graph convolutional (ChebNet) layers after the preprocessing MLP, the number of output channels of the graph convolutional layers (e.g., number of hidden units), and the max K-hop neighbourhood considered by the GNN. As for the MLP metamodel described before, all activation functions in the GNN are ReLU. No hyperparameter tuning is performed for the transferability experiments, where we use the largest possible GNN with 64 embedding dimensions, 3 ChebNet layers, 128 hidden output channels, and $K = 6$.

We run all the experiments using the Pytorch library (Paszke *et al.* 2019) for MLP models and Pytorch Geometric (Fey & Lenssen 2019) for the GNN models. We used default library weight initialisation methods (Glorot & Bengio 2010) and fixed the random seeds. Each metamodel was trained using the Adam optimisation algorithm with a constant learning rate of 0.001, no weight decay, and default selection of parameters. The training was carried out for 30 epochs with no early stopping and a batch size of 128. In terms of hardware, we employed a Xeon W-10855M @2.8 GHz CPU and a Nvidia Quadro RTX 5000, 16Gb RAM GPU.

RESULTS AND DISCUSSION

In this section, we first compare the MLP and the GNN in terms of performance and execution time one WDS at a time. Then, we assess the transferability of GNNs trained on the combined datasets of Table 2.

Table 5 | Hyperparameters and R^2scores for the best metamodels

Model		FOS	PES	PES	MOD	RUR	KL
MLP	# hidden units	256	128	256	256	256	64
	# hidden layers	4	3	3	2	2	2
	Dropout	0	0.25	0.25	0	0	0.25
	R^2 validation	0.364	0.991	0.570	0.859	0.944	0.472
	R^2 test	0.360	0.993	0.561	0.868	0.929	0.482
GNN	Embedding dimension	32	32	64	32	32	64
	# conv. layers	3	2	3	3	3	3
	# hidden units	64	128	128	128	128	128
	K-hop neigh.	6	3	6	6	6	6
	R^2 validation	0.748	0.991	0.496	0.759	0.924	0.463
	R^2 test	0.815	0.993	0.445	0.763	0.906	0.468

Table 6 | Goodness-of-fit metrics on the test dataset, execution speedups, and number of parameters of the best metamodels

	MLP				GNN			
	R^2	RMSE	Speedup	#parameters, 10^3	R^2	RMSE	Speedup	#parameters, 10^3
FOS	0.379	3.38	879	200	0.815	1.84	71	60
BAK	0.993	0.65	1,393	50	0.993	0.65	56	60
PES	0.561	6.76	1,241	200	0.445	7.60	43	200
MOD	0.868	1.22	2,223	300	0.763	1.63	24	200
Rural	0.929	1.27	2,029	500	0.906	1.47	27	200
KL	0.482	6.11	4,001	300	0.468	6.19	22	200

MLP vs GNN

Table 5 reports the best configurations of the metamodels based on the validation R^2 for each WDS, and after considering all hyperparameter combinations described in the previous section. From the comparison of the test R^2, it emerges that MLPs outperform GNNs in all benchmark datasets apart from BAK, where the performances are almost identical, and FOS, where the GNN largely outperforms the MLP.

Table 6 presents the comparison also in terms of RMSE and computational speedups, along with the total number of parameters of the best metamodels. While the RMSE follows the same trends described for R^2, it better indicates the average error in meters for nodal pressure estimation. As expected, the best GNNs are usually smaller than the MLPs in terms of parameters, especially for the FOS and the RUR case studies where the GNN achieves better or similar performances. Nevertheless, MLPs are faster, granting execution speedups of three orders of magnitude with respect to WNTR simulations. Similarly, their training time is between one to two orders of magnitude smaller than that of GNNs. That said, the GNNs provide substantial speedups of up to 70× that may justify their utilisation. Furthermore, this gap in optimised GPU implementation will likely get smaller as these relatively novel techniques become mainstream.

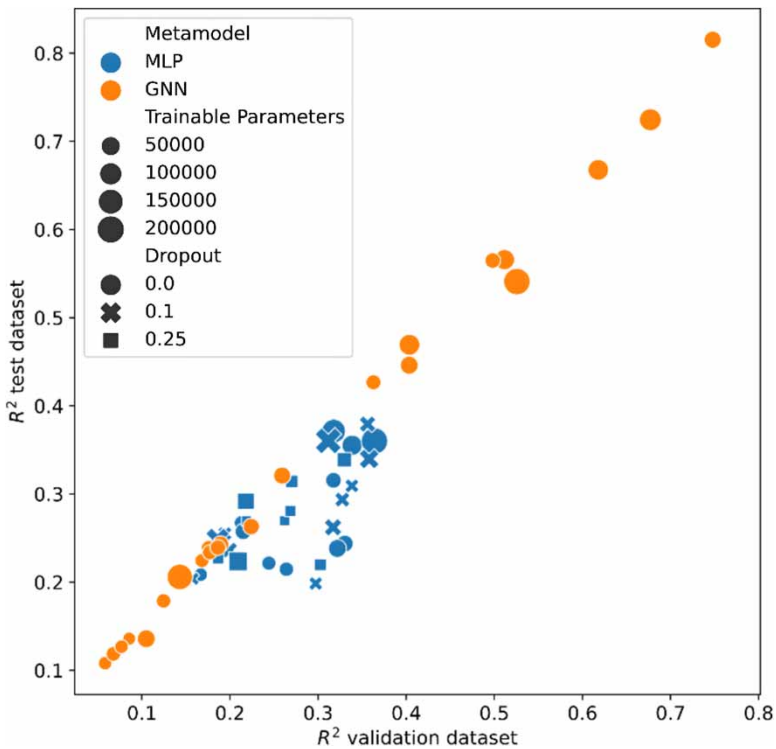

Figure 5 | Scatter plot of MLPs (blue) and GNNs (orange) validation and testing R^2 for the FOS case study.

While further GNN hyperparameters tuning may narrow the gap in performances for all WDS where MLPs perform best, Figure 5 suggests that MLPs may never reach performances similar to those of the GNNs for the FOS benchmark. Indeed, regardless of the model size and dropout used, the MLPs reach a performance ceiling at around $R^2 = 0.35$ and then 'overfit' the validation dataset, despite the consistency in the distribution of input and output variables across the different splits. This behaviour does not occur for the remaining WDS, indicating that, for some case studies, GNNs may discover hidden representations hardly accessible to MLPs. This initial finding requires further analysis linking model performances to topological and hydraulic features of the WDS.

The better fit of GNNs for the FOS case study is clearly visible in Figure 6(a) that shows the test dataset scatterplot of simulated vs predicted pressures across all nodes for the best-performing models. Figure 6 also shows that, regardless of the case study, the developed metamodels tend to overestimate the simulated pressures. The isolated clusters in each panel (e.g., see the top left corner of Figure 6(a)) usually correspond to individual simulation runs which are more difficult to surrogate. These challenges could be rooted in the sensitivity of the pressure response of WDS to input pipe parameters, such as pipe diameter. Employing a data generation process with systematically smaller pipe diameters could potentially be helpful in the training. In that case, the dataset will exhibit a broader distribution of pressure values. Consequently, the metamodel's sensitivity to the input might increase and may require more parameters to capture more complex relationships.

Transferability of GNNs

Figure 7 reports the R^2 scores of the transferable GNNs trained on the combined datasets (see Table 2) for all WDS test datasets, along with the distribution of R^2 for the MLPs (blue cross) and GNNs (orange cross) trained with 8,000 simulations on the individual benchmark WDS. The comparison between the GNN trained on the *unbalanced* (cyan circle) and *balanced* dataset (green square) seems to confirm that training on the former favours larger networks, such as MOD, RUR, and KL.

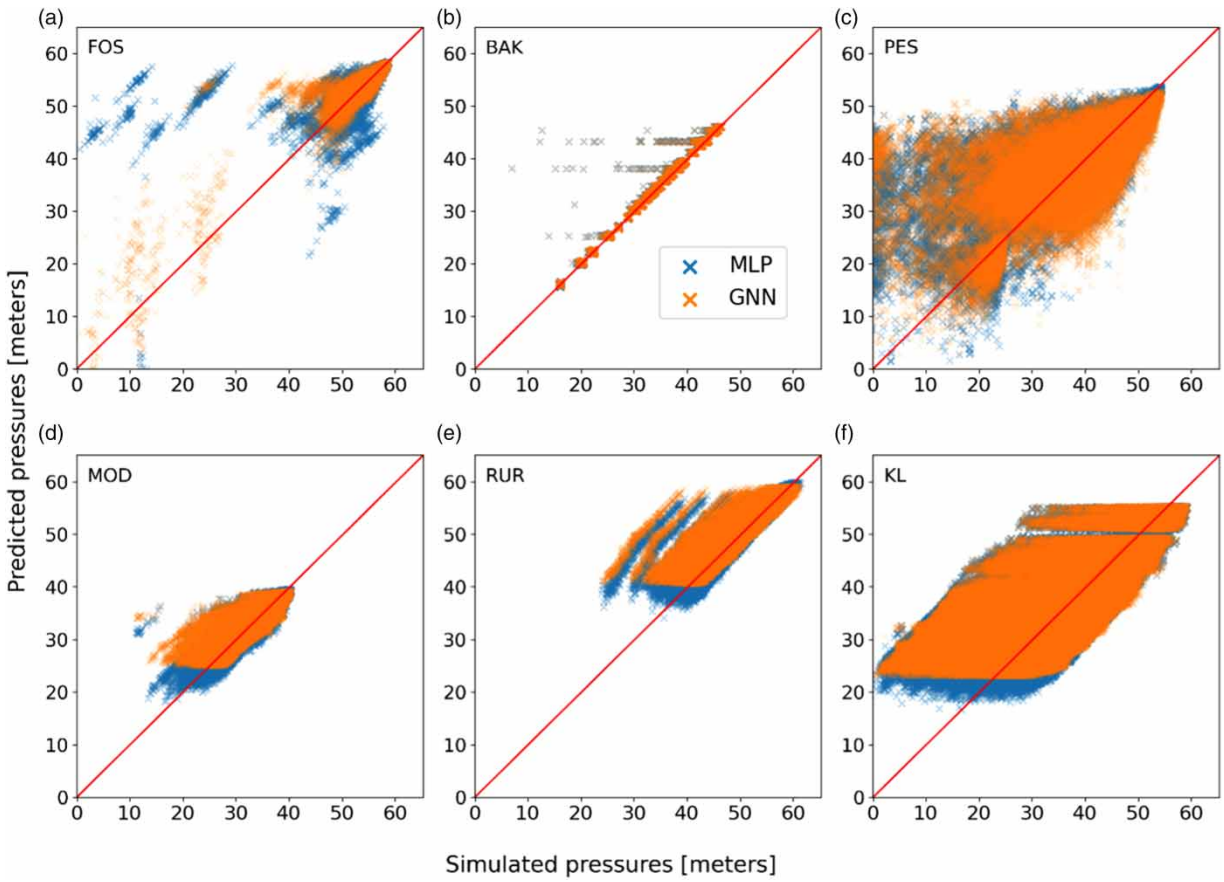

Figure 6 | Scatter plots of MLPs (blue) and GNNs (orange) predictions vs simulated pressures on the test datasets of FOS (a), BAK (b), PES (c), MOD (d), RUR(e), and KL (f) datasets.

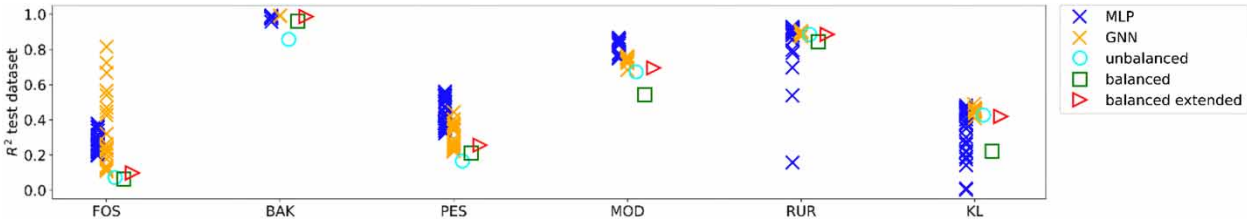

Figure 7 | Test R^2 scores of transferable GNNs for all benchmark WDSs.

While the general GNN seems to retain comparable performances to the individual ones for BAK, MOD, RUR, and KL, the performances drop drastically for PES, and especially, FOS. The problem persists even when considering the GNN trained on the *balanced extended* dataset (red triangle), which features 8,000 simulations for FOS and over 4,000 for PES. Further analysis is thus necessary to understand why the GNN seems to learn transferable features across a subset of the available WDSs, failing to perform on others.

On the other hand, these results also indicate that GNNs may exploit information learned in some case studies for predictions elsewhere. This particularly emerges when considering the performances of the transferable GNN trained on the *balanced extended* dataset on RUR and KL. This GNN shows equal performances to those trained on the *unbalanced* dataset despite having only around 75 and 30% of the training samples for these two WDSs, respectively (see Table 2), and it slightly underperforms the best individual MLPs and GNNs trained individually on 8,000 samples. While the training times of the transferable GNNs grow longer with larger training datasets, the model retains execution speedups in the order of those reported in Table 6 for the individual case studies.

CONCLUSION

In this work, we assessed the performances and transferability properties of GNNs used for metamodelling the pressure response surface of six benchmark WDSs. After generating a large sample of steady-state (snapshot) simulations with WNTR, we first compared the performances of multiple configurations of GNNs trained on individual WDSs against MLPs. The results indicate that, while MLPs tend to slightly outperform GNNs in most case studies, there is partial evidence that GNNs may be inherently better architectures for some WDSs. Despite requiring less trainable parameters to achieve comparable goodness-of-fit, GNNs are substantially slower than MLPs. At the current stage, the direct applicability of both GNNs and MLPs in downstream tasks might be limited for some WDSs. For the other cases, however, they still provide consistent speedups that could justify their use.

We assessed the transferability property of GNNs by training a single model on datasets with samples from all WDSs. Testing results for the larger WDSs suggest that a general GNN may perform comparably to the best MLP and GNN models while requiring substantially less training data for these case studies. These initial findings may indicate that GNNs can indeed learn shared representations across different water networks. However, the performance drop witnessed for two of the six networks implies that substantial efforts are required to design adequate datasets and test the general validity of this approach.

This exploratory study only considered a limited combination of GNN architectures consisting of ChebNet layers with a shared MLP for embedding nodal and edge features. Future studies should assess the effects of other graph layers and GNN paradigms on individual WDS performances (Wu *et al.* 2021) and transferability (Ruiz *et al.* 2020). Additionally, the effect of each hyperparameter could be investigated in more detail. Furthermore, we aim to investigate whether we can achieve better transferability by including more variability in the data generation process for example by resorting to a large number of randomly generated WDSs (Sitzenfrei 2016). This can include more complex techniques of sampling the pipe parameters that will result in broader pressure distribution in the dataset. New research could consider the equivalences that occur across different settings, such as when different pipe parameters in consecutive pipes result in the same headloss. These equivalences can be used in a self-supervised setting (Xie *et al.* 2022). Similarly, we aim to extend this work by considering surrogates of extended-period hydraulic analyses, rather than steady-state simulations.

ACKNOWLEDGEMENTS

The authors acknowledge the financial support through the NTNU Green2050 Centre for Green Shift in the Built Environment and the TU Delft AI Labs programme.

DATA AVAILABILITY STATEMENT

All relevant data are available from an online repository or repositories at https://github.com/rtaormina/GNN_metamodels_wds.

CONFLICT OF INTEREST

The authors declare there is no conflict.

REFERENCES

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, V., Heess, N., Wierstra, D., Kohli, P., Botvinick, M., Vinyals, O., Li, Y. & Pascanu, R. 2018 Relational inductive biases, deep learning, and graph networks. *ArXiv*, 1–40.

Behzadian, K., Kapelan, Z., Savic, D. & Ardeshir, A. 2009 Stochastic sampling design using a multi-objective genetic algorithm and adaptive neural networks. *Environmental Modelling & Software* **24** (4), 530–541. https://doi.org/10.1016/J.ENVSOFT.2008.09.013.

Bi, W. & Dandy, G. C. 2014 Optimization of water distribution systems using online retrained metamodels. *Journal of Water Resources Planning and Management* **140** (11), 04014032. https://doi.org/10.1061/(asce)wr.1943-5452.0000419.

Bonilla, C. A., Zanfei, A., Brentan, B., Montalvo, I. & Izquierdo, J. 2022 A digital twin of a water distribution system by using graph convolutional networks for pump speed-based state estimation. *Water (Switzerland)* **14** (4). https://doi.org/10.3390/w14040514

Bragalli, C., D'Ambrosio, C., Lee, J., Lodi, A. & Toth, P. 2012 On the optimal design of water distribution networks: a practical MINLP approach. *Optimization and Engineering* **13** (2), 219–246. https://doi.org/10.1007/s11081-011-9141-7.

Broad, D. R., Dandy, G. C. & Maier, H. R. 2005 Water distribution system optimization using metamodels. *Journal of Water Resources Planning and Management* **131** (3), 172–180. https://doi.org/10.1061/(asce)0733-9496(2005)131:3(172).

Broad, D. R., Maier, H. R. & Dandy, G. C. 2010 Optimal operation of complex water distribution systems using metamodels. *Journal of Water Resources Planning and Management* **136** (4), 433–443. https://doi.org/10.1061/(asce)wr.1943-5452.0000052.

Defferrard, M., Bresson, X. & Vandergheynst, P. 2016 *Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering*. https://doi.org/10.48550/arxiv.1606.09375.

Fey, M. & Lenssen, J. E. 2019 *Fast Graph Representation Learning with PyTorch Geometric*. https://doi.org/10.48550/arxiv.1903.02428.

Gama, F., Isufi, E., Leus, G. & Ribeiro, A. 2020 *Graphs, Convolutions, and Neural Networks. November*.

Garzon, A., Kapelan, Z., Langeveld, J. & Taormina, R. 2022 Machine learning-based surrogate modelling for Urban Water Networks: review and future research directions. *Water Resources Research*. https://doi.org/10.1029/2021WR031808

Glorot, X. & Bengio, Y. 2010 Understanding the Difficulty of Training Deep Feedforward Neural Networks. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research*, Vol. 9, pp. 249–256. Available from: http://proceedings.mlr.press/v9/glorot10a.html

Hajgató, G., Gyires-Tóth, B. & Paál, G. 2021 *Reconstructing Nodal Pressures in Water Distribution Systems with Graph Neural Networks. 2020*.

Hornik, K., Stinchcombe, M. & White, H. 1989 Multilayer feedforward networks are universal approximators. *Neural Networks* **2** (5), 359–366. https://doi.org/10.1016/0893-6080(89)90020-8.

Kang, D. & Lansey, K. 2012 Revisiting optimal water-distribution system design: issues and a heuristic hierarchical approach. *Journal of Water Resources Planning and Management* **138** (3), 208–217. https://doi.org/10.1061/(asce)wr.1943-5452.0000165.

Klise, K. A., Murray, R. & Haxton, T. 2018 An overview of the water network tool for resilience (WNTR). In *1st International WDSA/CCWI 2018 Joint Conference*, Kingston, Ontario, July 23–25 2018.

Lecun, Y., Bengio, Y. & Hinton, G. 2015 Deep learning. *Nature* **521** (7553), 436–444. https://doi.org/10.1038/nature14539.

Lee, S.-C. & Lee, S.-I. 2001 Genetic algorithms for optimal augmentation of water distribution networks. *Journal of Korea Water Resources Association* **34** (5), 567–575.

Maier, H. R., Kapelan, Z., Kasprzyk, J., Kollat, J., Matott, L. S., Cunha, M. C., Dandy, G. C., Gibbs, M. S., Keedwell, E., Marchi, A., Ostfeld, A., Savic, D., Solomatine, D. P., Vrugt, J. A., Zecchin, A. C., Minsker, B. S., Barbour, E. J., Kuczera, G., Pasha, F. & Reed, P. M. 2014 Evolutionary algorithms and other metaheuristics in water resources: current status, research challenges and future directions. *Environmental Modelling and Software* **62**, 271–299. https://doi.org/10.1016/j.envsoft.2014.09.013.

Marchi, A., Dandy, G., Wilkins, A. & Rohrlach, H. 2014 Methodology for comparing evolutionary algorithms for optimization of water distribution systems. *Journal of Water Resources Planning and Management* **140** (1), 22–31. https://doi.org/10.1061/(asce)wr.1943-5452.0000321.

Martínez, F., Hernández, V., Alonso, J. M., Rao, Z. & Alvisi, S. 2007 Optimizing the operation of the Valencia water-distribution network. *Journal of Hydroinformatics* **9** (1), 65–78. https://doi.org/10.2166/hydro.2006.018.

Meijer, D., Post, J., van der Hoek, J. P., Korving, H., Langeveld, J. & Clemens, F. 2021 Identifying critical elements in drinking water distribution networks using graph theory. *Structure and Infrastructure Engineering* **17** (3), 347–360. https://doi.org/10.1080/15732479.2020.1751664.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L. & Chintala, S. 2019 *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. https://doi.org/10.48550/arxiv.1912.01703.

Razavi, S., Tolson, B. A. & Burn, D. H. 2012 Review of surrogate modeling in water resources. *Water Resour. Res* **48**, 7401. https://doi.org/10.1029/2011WR011527.

Rossman, L. a. 2000 *Epanet 2. September*, 104.

Ruiz, L., Chamon, L., Ribeiro, A., 2020 Graphon Neural Networks and the Transferability of Graph Neural Networks. *Advances in Neural Information Processing Systems* **33**, 702–1712.

Salomons, E., Goryashko, A., Shamir, U., Rao, Z. & Alvisi, S. 2007 Optimizing the operation of the Haifa-A water-distribution network. *Journal of Hydroinformatics* **9** (1), 51–64. https://doi.org/10.2166/hydro.2006.017.

Sitzenfrei, R. 2016 A Review on Network Generator Algorithms for Water Supply Modelling and Application Studies. In: *World Environmental and Water Resources Congress*, West Palm Beach, FL, May 22–26 2016, pp. 505–515. https://doi.org/10.1061/9780784479865.053.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. 2014 Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning* **15**, 1929–1958. https://doi.org/10.1016/0370-2693(93)90272-J.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C. & Yu, P. S. 2021 A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* **32** (1), 4–24. https://doi.org/10.1109/TNNLS.2020.2978386.

Xie, Y., Xu, Z., Zhang, J., Wang, Z. & Ji, S. 2022 Self-Supervised Learning of Graph Neural Networks: A Unified Review. *ArXiv*. arXiv:2102.10757. Available from: https://arxiv.org/abs/2102.10757

Xing, L. & Sela, L. 2022 Graph neural networks for state estimation in water distribution systems: application of supervised and semisupervised learning. *Journal of Water Resources Planning and Management* **148** (5), 1–14. https://doi.org/10.1061/(asce)wr.1943-5452.0001550.

First received 14 February 2023; accepted in revised form 29 September 2023. Available online 17 October 2023