Graph Neural Networks for State Estimation in Water Distribution Systems: Application of Supervised and Semisupervised Learning

Lu Xing, S.M.ASCE[1]; and Lina Sela, Ph.D., A.M.ASCE[2]

Abstract: Emerging trends of resilient and reliable water infrastructure advocate for the development of efficient state estimation (SE) techniques in water distribution systems (WDSs). SE refers to estimating the flows and heads in the WDS at unmonitored locations based on measurements collected from limited monitoring locations. Current physics-based SE methods typically require more exhaustive than readily available information about the WDS and are computationally demanding to attain real-time SE fully. Using neural networks for SE is a promising avenue because neural networks are more adaptable to the availability of sensory data and can shift most of the computation efforts to the offline training phase. Once trained, the inference is more computationally efficient compared to the physics-based SE methods. This work proposes a graph neural network (GNN) model for SE in WDSs. Unlike traditional neural networks, GNNs are more suitable for the SE problem for two main reasons: (1) given a limited number of monitoring locations, the SE problem inherently requires a semisupervised learning method, and (2) GNNs enable learning from the graph structure of a WDS, thus providing a mechanism to incorporate the functional relationships between the monitored and unmonitored locations and incorporate the physical laws during the training process. To evaluate the performance of GNNs for SE, we tested supervised and semisupervised approaches, investigated the impact of GNN architecture choices on its performance, and examined model performance under different levels of noise in the training data. The results demonstrate that GNNs are promising for SE for their ability to learn from graph structure with a limited amount of information while exhibiting robustness to noise. This study contributes toward advancing real-time GNN-based SE in WDSs. Future research is needed to incorporate various hydraulic devices and investigate the scalability of GNNs to large-scale WDSs. **DOI: 10.1061/(ASCE)WR.1943-5452.0001550.** © *2022 American Society of Civil Engineers.*

Introduction

Advances in sensing technology have made it possible to deploy various sensors, including pressure and flow, to measure and monitor the state of water distribution systems (WDSs). However, considering the size and complexity of WDSs, as well as budget and resource constraints of water utilities, it is practically impossible to monitor every location in the system. Assimilating sensory data provides a promising avenue to complement physics-based models with data-driven components, such that the limited sensory data can be utilized to infer the hydraulic states in the entire system. State estimation (SE) is defined as inferring the states at all locations given measurements at a limited number of locations (Kang and Lansey 2010). In the context of WDSs, system states include the collection of heads at junctions and flows along pipes. The field measurements in a WDS typically include flow in pipes and pumps, pressure heads at junctions and storage tanks, and consumer demands (Tshehla et al. 2017; Wang et al. 2022).

SE has been traditionally formulated as an inverse optimization problem, in which the objective is to find the best match between the model predictions and measurements, as constrained by the network hydraulics, i.e., mass balance and energy conservations (Tshehla et al. 2017). Previous works differ in the choice of cost functions, formulation of constraints, and the choice of optimization algorithms (Andersen and Powell 2000; Andersen et al. 2001; Nagar and Powell 2002; Kumar et al. 2008; Preis et al. 2011; Díaz et al. 2016). However, the inverse optimization formulation typically requires more exhaustive than readily available information to improve solution quality and uniqueness. Additionally, the inverse optimization formulation results in complex nonlinear and sometimes nonconvex systems of equations, which require excessive computational resources to solve (Kumar et al. 2008; Wang et al. 2022).

Speeding up SE using neural networks (NNs), which are more flexible to data availability, is a promising avenue, especially for real-time estimation because most of the computational effort can be accomplished during the offline training stage. Once the models are trained, the SE task is reduced to a series of matrix-vector multiplications, which can be achieved very efficiently. In the context of WDSs, different NN architectures have been proposed and applied to various problems. For example, fully connected NNs were proposed to detect and identify pipe bursts from transient pressure waves (Mounce and Machell 2006; Bohorquez et al. 2021); convolutional neural network (CNN) models were trained as time-series classifiers to distinguish leak from normal signals collected by piezoelectric accelerometers (Kang et al. 2018; Guo et al. 2021); a temporal graph convolutional network was applied to perform head and flow time-series predictions at key facilities to detect cyber-physical attacks (Tsiami and Makropoulos 2021); and seven different NNs were applied to estimate missing pipe information, i.e., materials and diameters, in wastewater networks (Belghaddar et al. 2021).

[1]Postdoctoral Fellow, Dept. of Civil, Architectural, and Environmental Engineering, Univ. of Texas at Austin, Austin, TX 78712. ORCID: https://orcid.org/0000-0002-8881-5487

[2]Assistant Professor, Dept. of Civil, Architectural, and Environmental Engineering, Univ. of Texas at Austin, Austin, TX 78712 (corresponding author). ORCID: https://orcid.org/0000-0002-5834-8451. Email: linasela@utexas.edu

Note. This manuscript was submitted on April 23, 2021; approved on January 10, 2022; published online on March 14, 2022. Discussion period open until August 14, 2022; separate discussions must be submitted for individual papers. This paper is part of the ***Journal of Water Resources Planning and Management***, © ASCE, ISSN 0733-9496.

However, most of these methods were designed for anomaly detection and have not been investigated for SE purposes. We identify three main challenges associated with applying traditional NNs to the SE problem:

- The majority of the previous works rely on the supervised learning approach, which requires labeled data for all samples; however, for the SE problem, data is expected to be available only at a limited number of locations, indicating that SE inherently requires a semisupervised learning approach.
- The supervised learning formulation is purely data-driven and typically ignores the basic physical laws that govern the hydraulics in WDSs, thus failing to satisfy the underlying physics and potentially yielding inconsistent solutions.
- The traditional NNs can only operate on Euclidean data, typically denoted as $\mathbb{R}^n$, where n stands for the dimension, such as time-series ($\mathbb{R}^2$) and images ($\mathbb{R}^3$).

Nevertheless, information in WDSs, which includes network topology as well as attributes of nodes and pipes, inherently persist in graph domains, which are non-Euclidean and cannot be mapped neatly into $\mathbb{R}^n$. The complexity of graph-structured data poses significant challenges to the traditional NNs defined in Euclidean domains, including input representation of non-Euclidean data, the interdependency of nodes connected by edges, and the definition of information aggregation, i.e., convolution and pooling operators, in CNN models.

Recently, graph neural networks (GNNs), the generalization of traditional NNs from the Euclidean domain to the graph domain, have received increasing attention (Zhou et al. 2020). In this study, the GNN is identified as a promising approach for SE for two reasons. First, unlike traditional NNs, GNNs can not only learn from node attributes but also from the connectivity between the nodes, as represented by the network topology, thus making GNNs more appropriate for learning tasks in WDSs. Second, the topology information, as embedded in the graph-structured data, provides a mechanism for incorporating the functional relationships between the monitored and unmonitored locations. The latter is an essential input to formulate the loss that directly penalizes a violation of physical laws. Therefore, we will expect better predictions at the unmonitored locations for which measurements are not provided.

The concept of GNNs was first developed to extend existing NNs for processing the data represented in graph domains (Scarselli et al. 2009). Various GNN structures have been proposed that differ in how information is exchanged and aggregated (Battaglia et al. 2018; Zhou et al. 2020). The commonality of the different information exchange mechanisms can be abstracted as a forward path containing two phases: a message-passing phase, which updates the latent node states based on messages/information from neighbor nodes, and a readout phase, which decodes the latent states to the output feature space (Gilmer et al. 2017; Isufi et al. 2020). Popular GNN variants include gated graph neural networks (Li et al. 2015), graph convolutional networks (Zhang et al. 2019b), graph attention networks (Veličković et al. 2017), and graph recurrent neural networks (Hajiramezanali et al. 2019). Due to their ability to learn on graphs, which can be used to denote different systems, GNNs have been widely applied to various fields, including social network prediction (Fan et al. 2019), recommender systems (Ying et al. 2018; Yin et al. 2019), traffic flow prediction (Wang et al. 2020), particle-based simulation (Sanchez-Gonzalez et al. 2020), and power grid modeling (Donon et al. 2020a, b; Liao et al. 2021).

In addition to innovations in GNN architectures, advanced training methods have also been developed to improve the performance and capabilities of NNs. Traditionally, data-driven approaches were adopted to train a surrogate model in a supervised manner, such that the trained model can imitate solutions provided by some simulators (Bolz et al. 2019). However, this traditional data-driven supervised learning approach ignores valuable domain knowledge, thus failing to satisfy the underlying physical laws and potentially yielding inconsistent solutions. Therefore, there is a pressing need for integrating physical laws and domain knowledge in the model training process, which can, in turn, provide physically consistent solutions and avoid overfitting (Karniadakis et al. 2021). To this end, physics-informed machine learning has been proposed to incorporate domain knowledge into the training process by customizing the loss function to minimize the violation of the governing differential equations (Raissi et al. 2019). This new learning paradigm has been further developed and applied to provide simulator-free solutions in various fields, including modeling and prediction of a power grid flow (Donon et al. 2020a, b), surface fracture (Goswami et al. 2020; Shukla et al. 2020), turbulence (Wu et al. 2018; Wang et al. 2017), and climate (Kashinath et al. 2021). Taking advantage of these advances, we design the GNN loss function to penalize discrepancies between measurements and model predictions and the violation of the governing hydraulic equations.

Motivated by the need for real-time SE, the limited availability of measurements, and graph-structured data, the objectives of this study are as follows: (1) present a GNN model for SE in WDSs, (2) investigate the impact of the model architecture and noisy measurements on prediction quality, and (3) encourage reproducibility and usability of the research. Specifically, this paper is organized as follows. We first present the GNN architecture and formulate two learning approaches: (1) a *supervised* scheme that is trained using complete information provided by a hydraulic solver using many simulations of different network topologies and demands, and (2) a *semisupervised* approach that receives only a limited amount of measurements at given locations and simultaneously integrates the physical laws of mass and energy conservation. We then describe in detail the formulation of the loss function and the training, testing, and evaluation processes. The supervised learning approach is used as a benchmark to test the GNN architecture, while the main goal is to explore the new semisupervised approach. The performance of the two learning methods is then evaluated using several quantitative and qualitative metrics against the results obtained from a hydraulic solver, EPANET (Rossman et al. 2020). The results demonstrate that the proposed GNN architecture can learn to estimate the hydraulic state and that the semisupervised learning approach enables learning with fewer measurements while improving the performance of the trained model. Furthermore, we investigate the impact of various architecture choices on model performance and test the robustness of the model to measurement noise. We conclude the paper and propose several potential future extensions to overcome some of the current limitations. Finally, to encourage research reproducibility, we make all data, codes, and models available in the GitHub repository (Xing and Sela 2021).

Methodology

The objective of this work is to solve the SE problem, which is defined as estimating the heads at all nodes and flows in all pipes in the WDS given information about network layout, pipe characteristics, nodal demands, water levels at the reservoirs, and head measurements at limited locations. In this section, we introduce a novel GNN model to estimate states in WDSs, i.e., flow ($\boldsymbol{Q}$) in each pipe and head ($\boldsymbol{H}$) at each junction, and explain the main steps required to set up the GNN model. The main steps include the following: (1) defining inputs and outputs of the GNN model, (2) setting up the GNN architecture by defining the encoder,

processor, and decoder, (3) formulating the supervised and semi-supervised loss functions that are used to train the model, and (4) testing model performance. Additionally, we evaluated the impact of various key architecture choices on the GNN performance and tested the robustness of the model to different levels of noise in training data.

Problem Formulation

Consider a WDS with n nodes and m pipes; let $N = \{1, \ldots, n\}$ and $M = \{1, \ldots, m\}$ denote the node set and the pipe set, respectively. Within the n nodes, n_d nodes are junctions, i.e., pipe connections or users, denoted as N_d, and n_s nodes are source nodes, i.e., reservoirs or tanks, denoted as N_s, such that $N = N_d \cup N_s$. Each pipe p can then be represented by its start node i and end node j, i.e., $p = (i, j)$. The following settings of the WDS are given as inputs:

- Topology of the network as represented by the network adjacency matrix $A^{n\times n} = (a_{ij})_{i,j\in N}$, where $a_{ij} = 1$ if nodes i and j are connected, and 0 otherwise.
- Demands at all junctions, i.e., q_i, $\forall\ i \in N_d$.
- Heads at reservoirs and tanks, i.e., H_i^*, $\forall\ i \in N_s$.
- Loss coefficients at all pipes, i.e., c_p, $\forall\ p \in M$, which can be calculated

$$c_p = \frac{10.67 \times \text{length}_p}{r_{HW,p}^{1.852} \times \text{diameter}_p^{4.871}} \tag{1}$$

  where $r_{HW,p}$, diameter_p, and length_p = dimensionless Hazen-William coefficient, diameter, and length of pipe p in metric units, respectively.
- Head measurements at some junctions, i.e., H_i^*, $\forall\ i \in N_m$, where N_m denotes the junctions where head measurements are available.

Given all the inputs listed previously, the goal of this work is to design and train GNN models to estimate the states in a WDS, i.e., flow ($\boldsymbol{Q}$) in each pipe and head ($\boldsymbol{H}$) at each junction. The GNN model is a functional map from the inputs, i.e., network characteristics, topology, and measurements at limited locations, to the outputs, i.e., $\boldsymbol{H}$ at all nodes and $\boldsymbol{Q}$ along all pipes. To formulate the problem mathematically, we first present some notations and definitions that will be used throughout the paper. *Input interaction graph*, denoted as $\boldsymbol{G_I} = (\boldsymbol{V}, \boldsymbol{E})$, embeds all inputs regarding the WDS settings, including node related inputs $\boldsymbol{V} = (V_i)_{i\in N}, V_i \in \mathbb{R}^{d_V}$ and pipe related inputs $\boldsymbol{E} = (E_p)_{p\in M}, E_p \in \mathbb{R}^{d_E}$, where d_V and d_E are the dimension of node and pipe inputs, respectively. The *output interaction graph* $\boldsymbol{G_O} = (\boldsymbol{H}, \boldsymbol{Q})$ represents the states of the interaction graph, where $\boldsymbol{H} = (H_i)_{i\in N}$ stands for the heads at all nodes, and $\boldsymbol{Q} = (Q_p)_{p\in M}$ represents the flows in all pipes. In this paper, we only include nodal head ($\boldsymbol{H}$) as the state of the graph because pipe flows ($\boldsymbol{Q}$) can be uniquely determined once all heads are known. The GNN model, $\mathcal{M}_{\boldsymbol{\theta}}$:$\boldsymbol{G_I} \rightarrow \boldsymbol{G_O}$, is then defined as a parameterized function approximator, whose parameters $\boldsymbol{\theta}$ can be optimized for some training objectives, i.e., loss function $L(\boldsymbol{G_I}, \boldsymbol{G_O})$. The goal is thus to learn a GNN model $\mathcal{M}_{\boldsymbol{\theta}}$ that best approximates the mapping $\boldsymbol{G_I} \rightarrow \boldsymbol{G_O}$.

GNN Architecture

The proposed GNN model approximates the hydraulics by interactions among the neighboring junctions, which can be viewed as passing and aggregating information between nodes in the GNN. The overall workflow of the GNN model is described in Fig. 1. It consists of three major steps: (1) the *encoder* transforms input space to an abstract vector space, taking into account the WDS topology; (2) in the *processor* step, the latent state of each node is iteratively updated by passing and aggregating information from their neighbors; and (3) the *decoder* transforms the results back into the output space. The following subsections elaborate the structure in each step.

Encoder

To translate the WDS settings into a format that can be processed by the GNN model, we first encode all inputs into the input interaction graph $\boldsymbol{G_I} = (\boldsymbol{V}, \boldsymbol{E})$, where nodes represent junctions and reservoirs, and edges represent pipes in WDSs. The node inputs are embedded in $\boldsymbol{V} = (V_i)_{i\in M}, V_i \in \mathbb{R}^4$. For node i, the node inputs are represented as $V_i = [I_i^d, q_i, I_i^m, H_i^*]$, where I_i^d is the junction indicator, such that $I_i^d = 1$ if $i \in N_d$ and 0 otherwise; q_i is the demand at node i if $i \in N_d$ and 0 otherwise; I_i^m is the measurement indicator, such that $I_i^m = 1$ if $i \in N_m$ and 0 otherwise; and H_i^* is the head at node i if the head is known and 0 otherwise. Additionally,

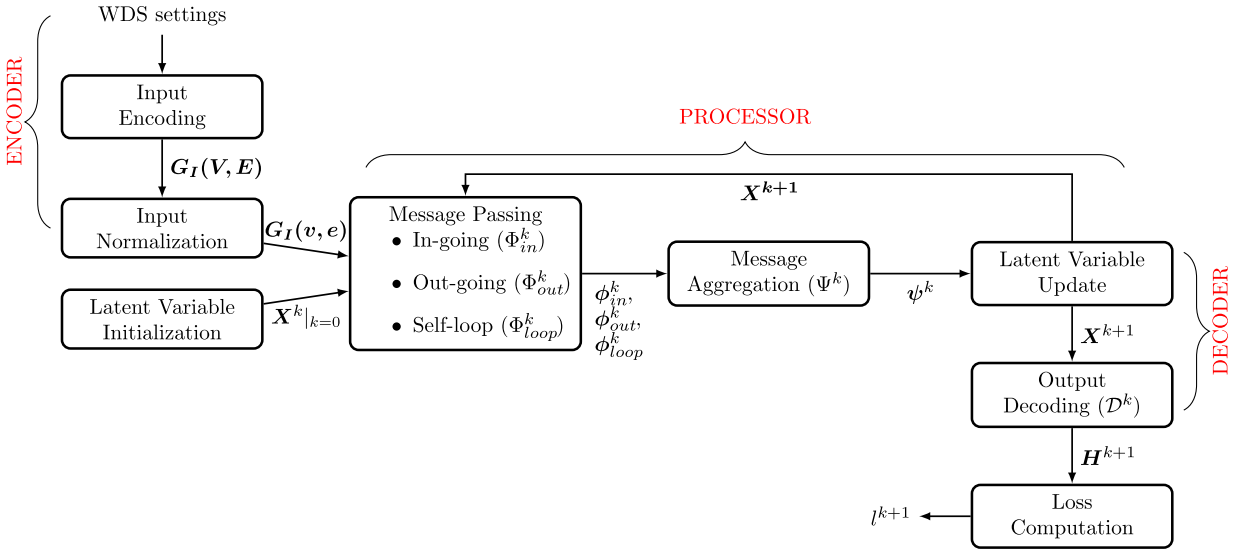

Fig. 1. The overall architecture of the GNN model.

J. Water Resour. Plann. Manage., 2022, 148(5): 04022018

$\boldsymbol{E} = (E_p)_{p\in M}, E_p \in \mathbb{R}^3$ represents pipe inputs, including the start and end nodes, and characteristics of the pipes. For example, for a pipe p whose start and end nodes are i and j, respectively, the pipe input is $E_p = [i, j, c_p]$.

In the input interaction graph, pipe and node inputs have different physical meanings and therefore can exhibit different orders of magnitude. For example, the nodal demands can range from 0 to 10^{-2} m^3/s, while nodal heads are normally 2–4 orders of magnitudes greater. It is critical to scale the input data to improve the training performance of the GNN. We thus create the normalized version of the data, i.e., $\boldsymbol{v} = (v_i)_{i\in N}$ and $\boldsymbol{e} = (e_p)_{p\in M}$, where $p = (i, j)$ is the pipe with start node i and end node j, as follows

$$v_i = \frac{V_i - \mu_V}{\sigma_V}$$
$$e_p = e_{(i,j)} = \frac{E_p - \mu_E}{\sigma_E} \tag{2}$$

where μ_V, σ_V, μ_E, and σ_E = mean and standard deviation of pipe and node inputs, respectively. Subsequently, the initial guesses on the graph states, i.e., nodal heads, are embedded into latent states, i.e., $\boldsymbol{X} \in \mathbb{R}^{d_X}$, where d_X is a hyperparameter of the GNN model, denoting the dimension of the latent states. All latent states $\boldsymbol{X^0}$ are initialized to a zero vector.

Processor

The processor iteratively updates the latent states at each node by passing and aggregating information to and from neighboring nodes using the input interaction graph. The directionality of flows in the WDSs is accounted for in the GNN architecture by learning different message-passing mappings for incoming and outgoing flows. Specifically, in the kth update layer for each node i, messages from incoming, outgoing, and self-loop pipes, i.e., ϕ^k_{in}, ϕ^k_{out}, and ϕ^k_{loop}, are computed based on the latent state at node i, X^k_i, the latent states at its incoming neighbors ($\mathcal{N}_{\text{in},i} = \{j|e_{(i,j)} \in \boldsymbol{E}\}$) and outgoing neighbors ($\mathcal{N}_{\text{out},i} = \{j|e_{(j,i)} \in \boldsymbol{E}\}$), and the normalized inputs of connecting pipes $e_{(i,j)}$, as shown in Eq. (3). It should be noted that the latent variables are denoted as $\boldsymbol{X}$, which is different from the common notation $\boldsymbol{H}$ in GNN literature (Zhou et al. 2020). This choice of notation was made due to the fact that $\boldsymbol{H}$ is used to denote nodal heads following the common practice in water distribution systems research

$$\phi^k_{\text{in},i} = \sum_{j\in\mathcal{N}_{\text{in},i}} \Phi^k_{\text{in},\boldsymbol{\theta}}(X^k_i, e_{(i,j)}, X^k_j)$$
$$\phi^k_{\text{out},i} = \sum_{j\in\mathcal{N}_{\text{out},i}} \Phi^k_{\text{out},\boldsymbol{\theta}}(X^k_j, e_{(j,i)}, X^k_i)$$
$$\phi^k_{\text{loop},i} = \Phi^k_{\text{loop},\boldsymbol{\theta}}(X^k_i, e_{(i,i)}) \tag{3}$$

where $\Phi^k_{\text{in},\boldsymbol{\theta}}$, $\Phi^k_{\text{in},\boldsymbol{\theta}}$, and $\Phi^k_{\text{loop},\boldsymbol{\theta}}$ = trainable mappings, each of which is modeled as a multilayer perception (MLP), which is a class of feedforward artificial neural networks. The MLP consists of an input layer, one or multiple hidden layer(s), and an output layer (Friedman et al. 2001). Each layer transforms the values from the previous layer with a nonlinear activation function and a weighted linear summation, which is parameterized by a set of parameters $\boldsymbol{\theta}$, including weights and intercepts. It should be noted that for each node i, the same MLPs are used for message passing, assuming that the interaction mechanism between different nodes is similar.

Subsequently, for each node i, the three messages ϕ^k_{in}, ϕ^k_{out}, and ϕ^k_{loop}, and the normalized node inputs v_i are aggregated using another MLP mapping $\Psi^k_{**\theta}$, as follows

$$\psi^k_i = \Psi^k_{\boldsymbol{\theta}}(X^k_i, v_i, \phi^k_{\text{in},i}, \phi^k_{\text{out},i}, \phi^k_{\text{loop},i}) \tag{4}$$

The aggregated message is then utilized to update the latent variable by adding the output of the previous layer directly to the output of the current layer (He et al. 2016)

$$X^{k+1}_i = X^k_i + \alpha\psi^k_i \tag{5}$$

where α = hyperparameter in the model. Choosing a sufficiently low value of α helps to keep the successive updates at a reasonably low order of magnitude. The update mechanism adds the latent variable from a previous layer to the current latent, which ensures that the higher layer will perform at least as good as the previous layer. At this point, the latent leap from $\boldsymbol{X}^k$ to $\boldsymbol{X}^{k+1}$ has been achieved. This process will then be iterated for $\hat{k}$ number of updates, where $\hat{k}$ is another hyperparameter denoting the number of correction update layers in the model.

Decoder

After each processor update, latent state $\boldsymbol{X}^{k+1}$ is decoded into the meaningful output state, i.e., nodal heads $\boldsymbol{H}^{k+1}$

$$H^{k+1}_i = \mathcal{D}^k_{\boldsymbol{\theta}}(X^{k+1}_i) \tag{6}$$

where $\mathcal{D}_{\boldsymbol{\theta}}$ = MLP mapping parametrized by $\boldsymbol{\theta}$. The same decoder is applied to each node. The output state at the last update layer $\boldsymbol{H}^{\hat{k}}$ is the final output of the model.

Loss Computation

At each update layer k, the loss function $l^k(\boldsymbol{G_I}, \boldsymbol{G_O}(\boldsymbol{H}^k))$ is evaluated. Two different formulations of loss functions are considered in this work: *supervised* and *semisupervised* loss. Supervised loss, l_s, defined in Eq. (7), measures the sum of squared errors between the current state computed by the GNN model $\boldsymbol{H}^k$ and th e reference values $\boldsymbol{H}^*$, e.g., measurements

$$l^k_s = \|\boldsymbol{H}^k - \boldsymbol{H}^*\|_2 \tag{7}$$

The supervised approach trains the GNN by learning only from reference values. On the other hand, the semisupervised approach trains the GNN model by minimizing the violation of physical laws and simultaneously assimilating the reference values, i.e., measurements or simulation results. The semisupervised loss, l_u, can then be decomposed into two parts

$$l^k_u = \beta l^k_m + (1-\beta) l^k_v \tag{8}$$

where $l^k_m = \boldsymbol{I^m}\|\boldsymbol{H}^k - \boldsymbol{H}^*\|_2$ represents the loss related to the sum of squared errors between model outputs and measurements at the limited location; $\boldsymbol{I^m} = [I^m_1, I^m_2, \ldots, I^m_n]$ is the vector of measurement indicators; l^k_v = unsupervised loss that represents the violation of physical laws; and β = weighting factor that allows one to prioritize one loss over the other.

In this study, we measure the unsupervised loss as the flow imbalance at the nodes and the head imbalance at the sources. To define the unsupervised loss, we first revisit the system of equations that governs the steady-state hydraulics in WDSs (Larock et al. 1999). With the Hazen-Williams model, the head loss for a pipe p that connects nodes i and j can be calculated by $h_p = H_j - H_i = c_p\lfloor Q_p\rfloor^{1.852}$, which can then be transformed to a nodal head equation

$$Q_p = \left(\frac{1}{c_p}\lfloor H_j - H_i\rfloor\right)^{0.54} \tag{9}$$

J. Water Resour. Plann. Manage., 2022, 148(5): 04022018

Substituting Eq. (9) into the mass conservation equation for each node yields

$$\sum_{j\in\mathcal{N}_i}\left(\frac{1}{c_p}\lfloor H_j - H_i\rfloor\right)^{0.54} = q_i, \quad \forall\ i \in N_d \tag{10}$$

where the summation is over all pipes entering or leaving the node i. The notation $\lfloor\cdot\rfloor$ is adopted from Boulos et al. (2006, Chap. 5) and accounts for the flow direction. If H_i is greater than H_j, then the flow is from node i to j, and the flow is from node j to i otherwise. Additionally, the head boundary conditions at the sources are specified

$$H_i = H_{s,i}, \quad \forall\ i \in N_s \tag{11}$$

We can then obtain the unsupervised loss, l_v^k, which represents the violation of physical laws, as the sum of squared residuals between model predictions and governing equations Eq. (10) and (11)

$$l_v^k = \sum_{i\in N_d}\left(\sum_{j\in N_i}\left(\frac{1}{c_{ij}}\lfloor H_j^k - H_i^k\rfloor\right)^{0.54} - q_i\right)^2 + \sum_{i\in N_s}(H_i^k - H_{s,i})^2 \tag{12}$$

The first part of the unsupervised loss measures the flow imbalance between inflows and outflows in each node, and the second part measures the imbalance between the estimated and actual head at the sources. In the training process, the loss function will be minimized, which might be more challenging with nondifferentiable functions. Notably, the flow imbalance term in the l_v^k is not differentiable in the case of zero flows, i.e., when $H_i^k = H_j^k$. The ill-conditioning in the presence of zero flows is a known problem in hydraulic simulations in WDSs, which can be avoided by replacing $|H_j^k - H_i^k|$ that is smaller than an arbitrarily small positive number δ by the bound δ (Todini and Pilati 1988). Although other solutions are possible, the bounding strategy is commonly implemented in hydraulic solvers like EPANET (Rossman et al. 2020); hence, this study adopted the same strategy. Additionally, nonlinear optimization methods are sensitive to scaling issues. Hence, because flow imbalance at junctions and head imbalance at sources have different units, scaling should be applied to improve performance. In our case study, the two parts had similar magnitudes; thus, scaling was not implemented.

Therefore, the complete form of semisupervised loss can then be written

$$\begin{aligned} l_u^k = {} & \beta\sum_{i\in N_m}(H_i^k - H_i^*)^2 \\ & + (1-\beta)\sum_{i\in N_d}\left(\sum\nolimits_{j\in N_i}\left(\frac{1}{c_{ij}}\lfloor H_j^k - H_i^k\rfloor\right)^{0.54} - q_i\right)^2 \\ & + (1-\beta)\sum_{i\in N_s}(H_i^k - H_{s,i})^2 \end{aligned} \tag{13}$$

where the first term accounts for the mismatch between model predictions and measurements; the second term represents the flow imbalance at junctions; and the third term accounts for the head imbalance at sources.

To robustify the learning process, all intermediate losses, l_s^k or l_u^k, are taken into account in the total training loss L through a discounted sum

$$L_{s/u}(\boldsymbol{G_I}, \boldsymbol{G_O}(\boldsymbol{H}^{\hat{k}})) = \sum_{k=1}^{\hat{k}}\gamma^{\hat{k}-k} l_{s/u}^k \tag{14}$$

where $L_{s/u}$ = total discounted supervised or semisupervised loss depending on the choice of learning approach; $l_{s/u}$ = supervised or unsupervised loss at update layer k; and $\gamma \in [0, 1]$ = hyperparameter that represents the discount factor, which is chosen empirically, such that more weight is assigned to the accuracy of the later correction update layers, while still allowing gradients to flow to the first few update layers. A lower value of γ means that losses in previous layers are valued less in the total loss. If $\gamma = 0$, the GNN model is concerned only with minimizing loss at the final update layer, and $\gamma = 1$ means that losses in all layers are equally accounted.

Model Training, Testing, and Evaluation

During the training process, a set of T_r different WDSs, defined by T_r different input interaction graphs, is provided to the GNN model. The GNN model performs the forward propagation to compute the output states, i.e., nodal heads $\boldsymbol{H}^k$, and evaluate the loss, as defined in Eqs. (7)–(12), at all correction update layers k. Subsequently, the total loss, as defined in Eq. (14), is computed by taking the weighted sum of the losses evaluated at all update layers. Then, the gradients of the total loss with respect to the parameters $\boldsymbol{\theta}$ in all MLP blocks, i.e., $\Phi_{\text{in},\boldsymbol{\theta}}^k$, $\Phi_{\text{in},\boldsymbol{\theta}}^k$, $\Phi_{\text{loop},\boldsymbol{\theta}}^k$, $\Psi_{\boldsymbol{\theta}}^k$ in the processor, and $\mathcal{D}_{\boldsymbol{\theta}}^k$ in the decoder, are estimated using the back propagation rules. These gradients are then used to update the parameter $\boldsymbol{\theta}$ iteratively, such that all MLP blocks are trained simultaneously to obtain the optimal parameter $\boldsymbol{\theta}$, which minimizes the training loss. Gradient clipping, which involves capping the gradient values to a specific maximum value if the gradient exceeds an expected range, is applied to avoid exploding gradient issues and improve stability and convergence (Zhang et al. 2019a; Mai and Johansson 2021). The training process is repeated on randomly sampled training batches of input interaction graphs until the total loss converges, i.e., no longer decreases.

Once the training process is accomplished, the trained model is then tested using another set of T_e different input interaction graphs. The output states can be obtained through the forward propagation, i.e., a series of matrix-vector multiplications, using the trained set of parameters. Subsequently, the performance of the trained model is evaluated using the test dataset using qualitative and quantitative measures. The qualitative comparison included plotting: (1) GNN predictions versus the reference values obtained from EPANET, (2) the distributions of errors at different nodes to identify nodes that exhibit greater prediction errors, and (3) intermediate losses and predictions to illustrate the convergence process. Quantitative analysis to assess model performance included the following:

- The correlation with respect to heads and flows (Corr_H and Corr_Q) between the GNN and EPANET results are evaluated across all nodes and pipes in all test samples. For example, the head correlation (Corr_H) is computed as follows:

$$\text{Corr}_H = \frac{\sum_{i=1}^{n}\sum_{l=1}^{T_e}(H_{i,j}^{\hat{k}} - \bar{H})(H_{i,j}^* - \overline{H^*})}{\sqrt{\sum_{i=1}^{n}\sum_{l=1}^{T_e}(H_{i,j}^{\hat{k}} - \overline{H^{\hat{k}}})^2(H_{i,j}^* - \overline{H^*})^2}} \tag{15}$$

  where $H_{i,j}$ and $H_{i,j}^*$ = head at ith node in the jth sample obtained by the GNN and EPANET models, respectively; and $\bar{H}$ and $\overline{H^*}$ = overall mean of heads at all nodes across all samples obtained by the GNN and EPANET models, respectively. A higher Corr_H represents a better overall match between GNN estimations and EPANET results.
- The head and flow root mean squared error (RMSE) for each of the test sample results is evaluated, e.g., head RMSE, by

$\sqrt{(1/n)\sum_{i=1}^{n}(H_i^{\hat{k}} - H_i^*)^2}$, which yields a vector of RMSE of length T_e. The mean and standard deviation of the head and flow RMSE vectors are then evaluated and denoted by $\overline{\text{RMSE}_H}$, $\overline{\text{RMSE}_Q}$, $\sigma(\text{RMSE}_H)$, and $\sigma(\text{RMSE}_Q)$, respectively. The $\overline{\text{RMSE}}$ provides insights of the average inference performance of the trained GNN model, while $\sigma(\text{RMSE})$ shows how uncertain the performance is. Better performance is characterized by lower value of $\overline{\text{RMSE}}$ and $\sigma(\text{RMSE})$.

- The 10th, 50th, and 90th percentile of the total supervised loss for each sample, as defined in Eq. (7), are also reported to show the magnitude of loss after convergence is achieved.
- We investigated the impact of GNN architecture choices on its performance by systematically varying the different architectural choices, including the dimension of the latent variables (d^X), number of correction update layers ($\hat{k}$), number of hidden layers in each MLP block, discount factor (γ), update coefficient (α), and shared versus unshared decoders and processors for different update layers.
- To test the model robustness against noise, we added different levels of noise to the head reference values ($\boldsymbol{H}^*$) in the training data and examined model performance.

Experiments and Results

This section details the experimental protocol used to validate the proposed approach, including the process of dataset preparation and the choice of hyperparameters. The results obtained from supervised and semisupervised training approaches with different numbers of monitoring locations, varying architecture choices, and different levels of training noise are presented and compared.

Dataset Preparation

We evaluated the proposed model on a benchmark WDS, comprising 51 junctions, 65 pipes, and 1 reservoir. Its topology is depicted in Fig. 2, and the detailed information can be found in the study by Alperovits and Shamir (1977). To create a more realistic scenario, this WDS hydraulic model was modified to perform an extended period simulation (EPS) of 1 week with a 1-h reporting time-step (169 time steps). Each junction is assigned one of five demand patterns that represent temporal and spatial demand variability in the weekly simulation. We generate the training set based on this single WDS but alter nodal demands and locally modify network topology. Specifically, with changes in demands at each time-step, which are encoded in $\boldsymbol{V}$, the EPS model configuration yields 169 different input interaction graphs $\boldsymbol{G_I}$ for a single topology. Additionally, in order to improve model training, we increased the diversity of network topology, which is encoded in $\boldsymbol{E}$, by randomly disconnecting two pipes while preserving the connectivity between each node in the network and source. A total of 256 networks with different topologies are then generated, each of which shares the same demand pattern assignments among the junctions. Thus, the dataset includes 256×169 samples, each of which is one network with one set of demand loads. Subsequently, the dataset is split 60%/20%/20% (25,958/8, 653/8, 653) for training, validation, and testing, respectively.

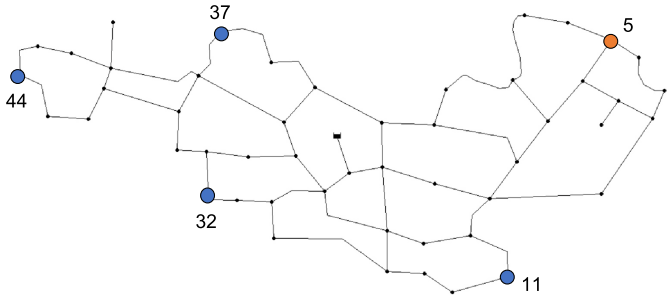

Fig. 2. Layout of the benchmark WDS and locations of measurement nodes as represented by the circles.

For supervised learning, reference nodal heads for all the junctions are required. We use the steady-state hydraulic solver EPANET (Rossman et al. 2020) to compute the nodal heads (given the topology, pipe, and node characteristics). The EPANET simulation results are used as the reference values ($\boldsymbol{H}^*$), which are encoded in the node inputs $\boldsymbol{V}$, where all measurement indicators equal 1, i.e., $I_i^m = 1$, $\forall i$.

In a more realistic scenario, we do not expect measurements at all nodes to be available; instead, we can expect to have pressure information at several key locations in the network. For semisupervised learning, nodal heads at only a few junctions are required. Thus, the EPANET simulation results at some chosen junctions are supplied and encoded into the node inputs $\boldsymbol{V}$ as defined in the methodology section.

Supervised Learning

The GNN model is first trained using the supervised approach. The latent dimension was set to $d_X = 20$, and the number of correction update layers was set to $\hat{k} = 20$. Each MLP block, $\Phi_{\text{in},\boldsymbol{\theta}}^k$, $\Phi_{\text{in},\boldsymbol{\theta}}^k$, $\Phi_{\text{loop},\boldsymbol{\theta}}^k$, $\Psi_{\boldsymbol{\theta}}^k$, and $\mathcal{D}_{\boldsymbol{\theta}}^k$, exhibits one hidden layer and a leaky-ReLU activation layer. Other hyperparameters were chosen as follows: $\alpha = 0.01$ and $\gamma = 0.9$. The Adam optimizer (Kingma and Ba 2014) is adopted to minimize the supervised loss with the learning rate $l_r = 0.001$. Training is completed within 70,000 iterations with batch size 500, which took around 24 h on a Windows machine with Intel(R) Core(TM) i7-7700 CPU@3.60GHz.

After the training, heads at all nodes are obtained, which are then used to compute the pipe flow with $Q_p = (\lfloor H_j^{\hat{k}} - H_i^{\hat{k}} \rfloor / c_p)^{0.54}$, where nodes i and j are the start and end node of pipe p. The performance of the GNN model is then evaluated on the test dataset. The inference time, i.e., the computational time for a single run with a trained GNN, is 0.0067 s. The comparisons of the heads and flows between the results obtained from the EPANET solver and the supervised GNN model are shown in Figs. S1(a and b) of the Supplemental Materials, respectively. The x-axis represents EPANET results, and the y-axis shows supervised GNN results. The perfect alignment between the two models is characterized by the diagonal lines, while the EPANET-GNN result pairs are represented by the markers. Fig. S1(a) shows that the head predictions from the supervised GNN model match well with the results obtained from EPANET simulations. The correlation between the results from the two models is approximately 99.98%. Similarly, the comparison between flow results is presented in Fig. S1(b). Although the general trend shows good agreement with a 97.31% correlation coefficient, it can be noticed that misalignment is more significant when the flows obtained from EPANET simulations are closer to zero. The misalignment near zero flows can be explained by the fact that based on Eq. (9), the derivative of the flow rate with respect to the head difference is steep near zero flow; hence, small changes in the head difference can lead to a relatively great difference in flows. Considering the stochastic nature of GNN training, small head differences are inevitable, thus resulting in the more obvious misalignment near zero flow. The first column of Table 1 summarizes additional metrics evaluating the performance of the supervised GNN model,

Table 1. Performance comparison of the GNN models trained with supervised or semisupervised approaches as evaluated using the test set

Model	Supervised GNN	Semisupervised GNN w/1 measurement	Semisupervised GNN w/5 measurements
$Corr_H$	**99.98%**	71.53%	99.96%
$Corr_Q$	97.31%	90.0%	**99.73%**
$\overline{\text{RMSE}_H}$	2.06×10^{-2}	3.08×10^{-1}	$\mathbf{1.14 \times 10^{-2}}$
$\sigma(\text{RMSE}_H)$	2.42×10^{-3}	1.96×10^{-1}	6.32×10^{-3}
$\overline{\text{RMSE}_Q}$	1.08×10^{-2}	1.92×10^{-1}	$\mathbf{3.63 \times 10^{-3}}$
$\sigma(\text{RMSE}_Q)$	2.42×10^{-3}	7.80×10^{-3}	$\mathbf{1.06 \times 10^{-3}}$
Loss 10th percentile	2.44×10^{-5}	2.66×10^{-5}	$\mathbf{1.31 \times 10^{-5}}$
Loss 50th percentile	5.47×10^{-5}	4.63×10^{-5}	$\mathbf{2.91 \times 10^{-5}}$
Loss 90th percentile	1.19×10^{-4}	8.39×10^{-5}	$\mathbf{7.01 \times 10^{-5}}$

Note: Bold numbers represent the best performing metrics.

including correlation and the RMSE with respect to EPANET results, as well as percentiles of the loss.

To further examine the supervised GNN model performance, we then compare head and flow results obtained from the supervised GNN model and EPANET solver at each node and pipe during the one-week simulation, as presented in Figs. 3(a and b), respectively. In Fig. 3(a), the x-axis represents the node number, and the y-axis stands for the absolute head difference between the two models at a specific node i, i.e., $|H_i^k - H_i^*|$. The absolute head difference at each node is displayed by a boxplot, in which the two ends of the box show the lower and upper quartile (Q1 and Q3), and the line across the box characterizes the median. The upper whisker extends to the last datum less than $\text{Q3} + 1.5(\text{Q3} - \text{Q1})$, and the lower whisker extends to the first datum greater than $\text{Q1} - 1.5(\text{Q3} - \text{Q1})$. Beyond the whiskers, data are considered outliers and are plotted as individual points. It can be observed that the head differences are evenly distributed across different nodes with the median around 0.01 m. This is expected as the heads at all nodes resulting from EPANET simulations are supplied for model training, and the loss function minimizes the head mismatch between the GNN predictions and the EPANET results at all nodes.

Similarly, Fig. 3(b) presents the flow difference at each pipe. Unlike the head differences, the flow difference varies significantly across different pipes. For example, the differences in Pipes 64 and 65 are much larger than those in other pipes. After closer investigation, we notice that the EPANET flow results in these two pipes are zero, e.g., dead-end pipes, or very close to zero. Thus, this observation reinforces that greater flow discrepancies are expected when flows are zero, which can be explained by the fact that flows are not directly involved in the supervised GNN training process; instead, they are computed after the heads are predicted by the training GNN model.

Fig. S2 in the Supplemental Materials illustrates the intermediate losses and the head predictions across the 20 correction update steps, in which the center Fig. 6(a) shows the progress of supervised loss at each update step, and the side Figs. 6(b–g) show the comparisons of head results between the EPANET and GNN model. As expected, the loss decreases as the correction update progresses. Also, the head predictions from the GNN model gradually

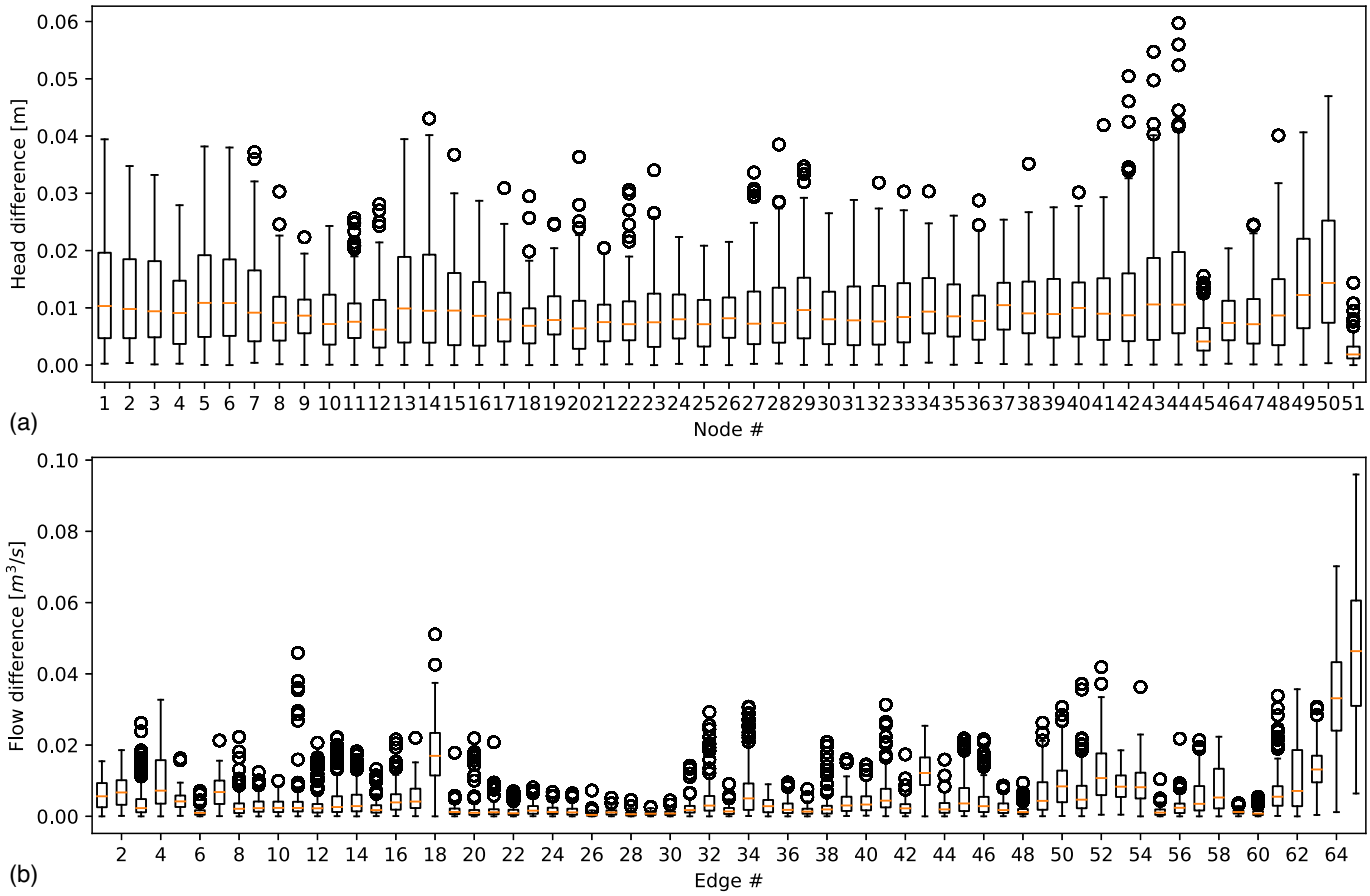

Fig. 3. The differences between the results from the EPANET solver and the supervised GNN model at each junction and pipe: (a) head differences; and (b) flow differences.

approach the EPANET results and achieve good alignment at the final update step.

Semisupervised Learning with One Measurement Location

In this experiment, the GNN model is trained in a semisupervised fashion. In this study, we assume that only one sensor is available in the network, i.e., $|N_m| = 1$, at which the nodal head can be estimated. We chose junction 5, represented by the circle in Fig. 2, as the measurement location, i.e., $I_5^m = 1$. The head results from EPANET at junction 5 are encoded in the node input $\boldsymbol{V}$ and are supplied to the GNN model in the input interaction graph $\boldsymbol{G_I}$, while the EPANET results at all other junctions are invisible to the GNN model. The same hyperparameters and optimizer are used as in the supervised model to enable comparison between the two training approaches.

The performance of the semisupervised GNN model, as evaluated by correlation and RMSE with respect to EPANET results, as well as loss percentiles, is displayed in the third column in Table 1. Compared with the supervised learning approach, as shown in the first column, the performance of this model is considerably worse. For example, the correlations with EPANET results are only 71.53% for heads and 90.0% for flows. Notably, unlike the supervised approach, the flow correlation is better than that of heads. This is because the semisupervised loss as defined in Eq. (13) accounts for the nodal flow imbalance; thus, flows are more directly trained than the heads.

Subsequently, Figs. 4(a and b) compare head and flow results obtained from EPANET solver and the semisupervised GNN model with one head measurement at junction 5 at each node and pipe. It can be noticed from Fig. 4(a) that the head differences vary significantly across different nodes. For example, at junction 5, where head measurements are assumed, the head differences are close to zero, indicating a good match between the EPANET and GNN model. A good match is expected because the mismatch between the measurements and GNN prediction enters directly into the loss function in term l_m^k. Moreover, some nodes other than the measurement node itself, e.g., 1, 2, 3, 6, and 13, also agree well with the EPANET results. However, the head differences at some other nodes, e.g., 24, 34, and 42, get as large as 1.0 m.

To investigate why the head differences between the two models are larger at some nodes while smaller at others, we hypothesized that the error in model predictions increases as we move away from the location where measurements are available. Intuitively, the measurements provide reference head values to compensate for the flow imbalance loss, which essentially regulates the relative relations between nodal heads in the form of $\lfloor H_j - H_i \rfloor$. Without the measurements, GNN needs to build and learn along a long-distance message passing path from each node to the source nodes, in which the known heads provide a reference. If measurements are available, it is then enough to build the message passing path from unmonitored nodes to the nearby monitored nodes. The learning of message passing on a shorter path is easier as it involves fewer function approximations. Thus, more measurements can help improve the GNN training. To validate the hypothesis, we computed

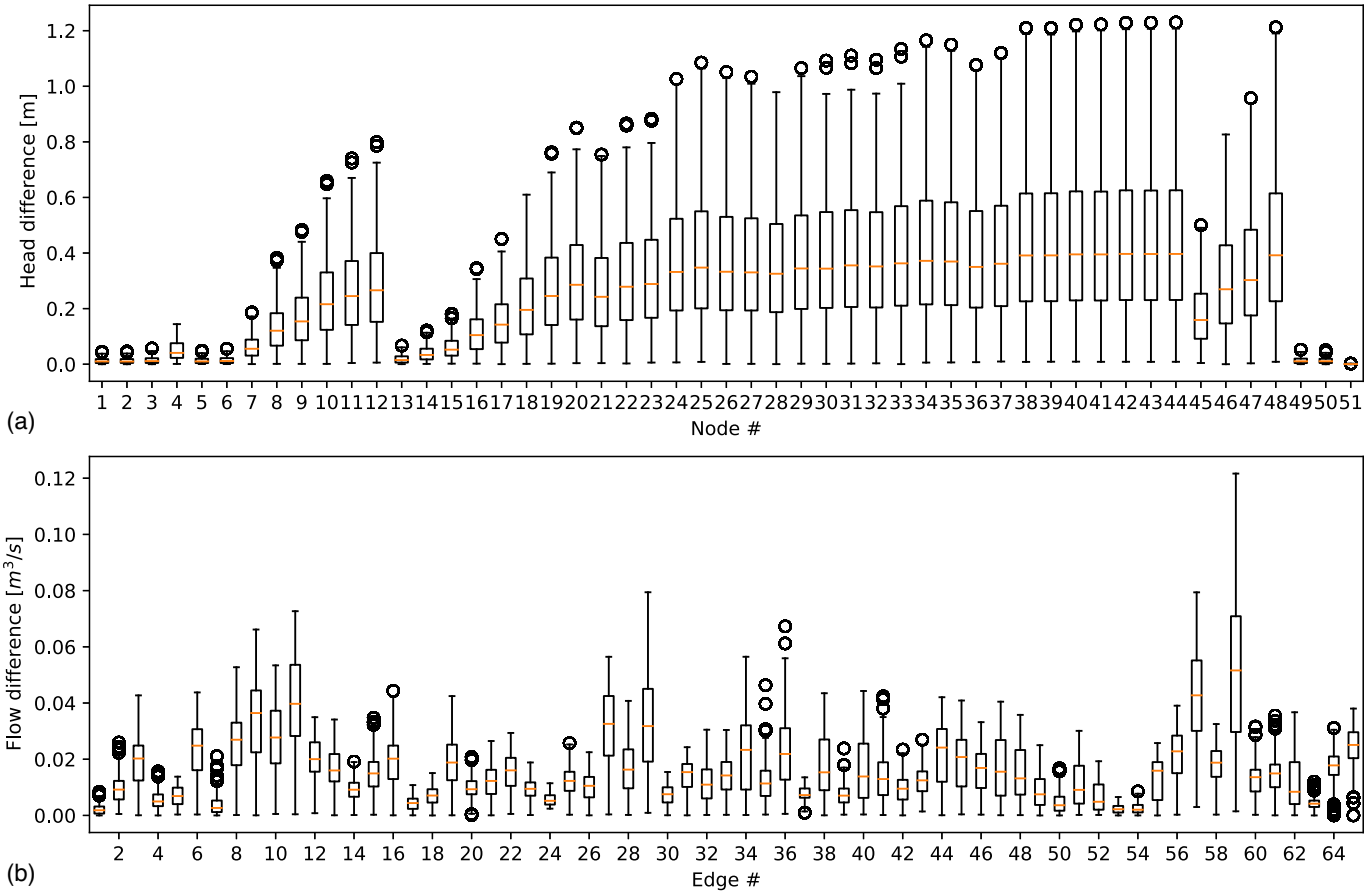

Fig. 4. The differences between the results from the EPANET solver and the semisupervised GNN model trained with one measurement at each junction and pipe: (a) head differences; and (b) flow differences.

J. Water Resour. Plann. Manage., 2022, 148(5): 04022018

the shortest path from each node to the measurement node, i.e., junction 5, using Dijkstra's algorithm (Johnson 1973). We then plotted the relation between head differences and the number of pipes on the shortest path, as illustrated in Fig. S3 in the Supplemental Materials, in which each data point represents a node in the network. The corresponding x value is the number of pipes on the shortest pipe from this node to junction-5. The corresponding y value is the averaged head difference between the GNN and EPANET results on this node across all test samples. Noticeably, the mean head difference increases as the number of pipes on the shortest path increases, i.e., the node gets further away from the measurement node. This implies that one measurement is not enough for the GNN model to be trained properly with the semisupervised approach, and more measurements are required such that all nodes in the network are relatively close to one of the measurement nodes. For example, if 0.2 m in head difference is tolerable, as indicated by the dashed horizontal lines, all nodes should be within a 4-pipe distance from the nearest measurement node, as indicated by the dashed vertical line. Therefore, in the next section, we investigate the performance of the semisupervised GNN model with more measurement locations.

Semisupervised Learning with Five Measurement Locations

In this section, five nodes 5, 11, 32, 37, and 44, as represented by the markers in Fig. 2, are chosen to be the measurement nodes in which the nodal heads are available, i.e., $|N_m| = 5$. These nodes were selected such that every node in the network is within a 4-pipe distance from the nearest measurement node. The head measurements at the five chosen nodes are then embedded into the input interaction graph and included in the GNN training dataset. The same hyperparameters and optimizers are used as in the previous two experiments to enable comparison between the different training approaches.

The last column in Table 1 summarizes the performance of the semisupervised GNN model trained with five measurement nodes. For both flows and heads, correlations between results from the trained GNN model and EPANET are above 99%, which validates this semisupervised training approach. The agreement between the semisupervised GNN model and EPANET can also be observed in Fig. S4(a)in the Supplemental Materials, in which the EPANET-GNN result pairs reside close to the diagonal line characterizing a perfect match. The head correlation of the semisupervised approach is slightly worse than that of the supervised training approach, which can be explained by the fact that the semisupervised training minimizes the violation of node flow balance, while the supervised approach minimizes the mismatch between nodal heads. However, the quality of flow prediction is superior in the semisupervised approach, as evidenced by the higher correlation, the lower RMSE, the smaller losses in all reported percentiles, and the better alignment shown in Fig. S4(b)in the Supplemental Materials, as compared with the results from the supervised approach. Additionally, the flow misalignment near zero is improved compared with the results obtained from the supervised learning approach. The improvement can be explained by the inclusion of flow imbalance in the semisupervised loss, which penalizes the flow misalignment in all pipes, including where flows are zero.

Subsequently, the comparison of results at each node and pipe is presented in Fig. 5. The median head differences at all nodes are below 0.02 m, indicating a reliable and consistent performance in terms of head predictions. Moreover, in the majority of pipes, the median flow difference is below 0.005 m^3/s. As an exception, the mismatches in Pipes 18, 63, 64, and 65 are considerably larger than other pipes. A closer examination into the network topology reveals that, as previously noted, these pipes are directly connected or adjacent to dead-ends, suggesting that further study should be performed to improve the performance of GNN models at near dead-end pipes.

Finally, Fig. 6 illustrates the intermediate semisupervised losses and the head predictions across the 20 correction update layers, in which the center Fig. 6(a) shows the progress of semisupervised loss at each update, in which the black line shows the median of losses computed on all test samples, and the shaded area represents the 10th to 90th percentile of the losses. The side of Figs. 6(b–g) show the comparison of the head results between the EPANET and GNN model for one test sample as an example. The vertical dotted lines represent the nodes where measurements are available. As expected, the loss decreases as the correction update progresses. It can also be observed that the disparities between the heads predicted by GNN and EPANET at measurement nodes are reduced during the first few updates, as shown in Figs. 6(b–d). The heads at other nodes then follow as the messages pass through the network topology, and the head predictions from the GNN model gradually approach the EPANET results. At the final update layer, a satisfactory agreement is achieved between the head results predicted by the GNN model and EPANET.

Overall, the presented results demonstrate that incorporating physical laws into the training process of the GNN model can improve the performance of the trained model. Additionally, the semisupervised learning approach only requires measurements at a limited number of locations, i.e., five, in this case, thus enabling solving the practical problem of SE with only a few available sensors.

Investigating GNN Architectural Choices

To investigate how different design choices influence the performance of the trained GNN model, we trained GNNs with different architectural choices, including the dimension of the latent variables (d^X), number of correction update layers ($\hat{k}$), number of hidden layers in each MLP block, discount factor (γ), update coefficient (α), and shared versus unshared decoders for different update layers.

The test was carried out using the semisupervised learning approach assuming five measurement locations. We varied these architecture choices systematically for each dimension while fixing all other dimensions with the default architecture choices. For a fair comparison, we performed 180,000 training iterations for each hyperparameter setting, which ensured convergence for all settings, although the training process converged much earlier for some instances. The impact of the hyperparameters on model performance is evaluated based on head and flow RMSE for the test dataset and is reported in Fig. 7.

Figs. 7(a and b) represent the impact of the latent dimension (d^X) on model performance. Both head and flow RMSEs decrease as the latent dimension increases. The RMSE for head and flow are especially high when the message is exchanged directly on 1-D head vectors, i.e., $d^X = 1$, without encoding into higher dimension latent variables. Intuitively, the higher latent dimension embeds more information and allows proper propagation of information contained in the node and edge inputs, thus enabling better learning. We also empirically observed that a higher latent dimension enables faster training convergence. However, the inference complexity, i.e., the computational time for a single run with a trained GNN, scales quadratically with d^X, which presents a trade-off between accuracy and computation efficiency. Figs. 7(c and d) show that a greater number of message-passing steps, i.e., update layers ($\hat{k}$),

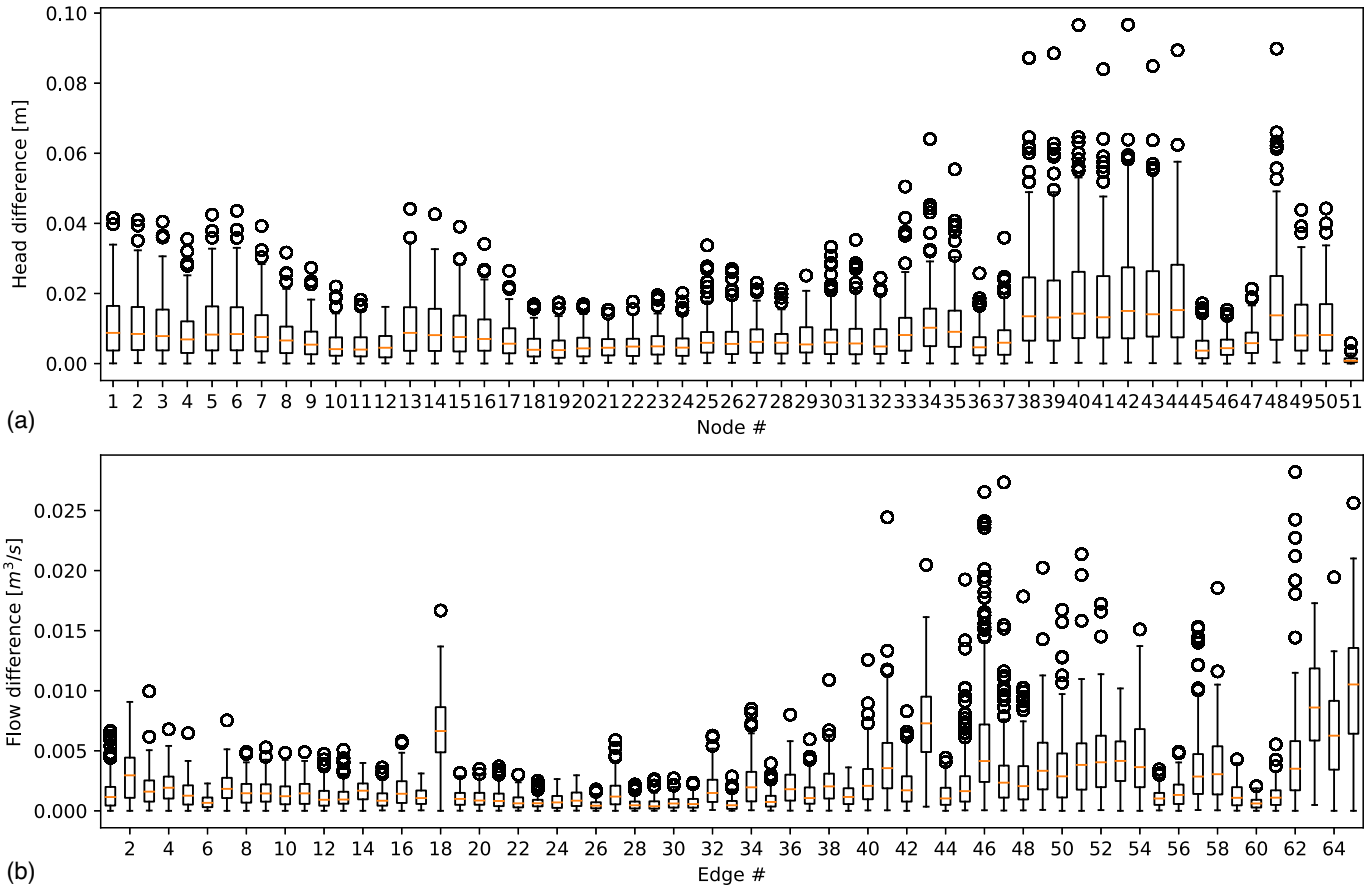

Fig. 5. The differences in the graph state between the results from the EPANET solver and the semisupervised GNN model trained with five measurements: (a) head differences; and (b) flow differences.

yields improved performance in both head and flow accuracy. This is likely because increasing $\hat{k}$ allows computing more complex interactions among nodes. On the other hand, the inference computation time scales linearly with $\hat{k}$; therefore, a smaller $\hat{k}$ is beneficial in terms of computation efficiency. Figs. 7(e and f) demonstrate that model performance improves when the depth of MLP blocks increases from zero (linear layer) to one hidden layer. However, the performance remains similar with two hidden layers. This indicates that one hidden layer is sufficient to model the interaction between the nodes for this dataset. Figs. 7(g and h) illustrate that the model performance is generally robust against the choice of discount factor (γ). However, worse performance is observed when $\gamma = 0$, i.e., only the loss at the final update layer is minimized. Intuitively, when $\gamma > 0$, the losses at all update layers are taken into consideration through the discount factor, so penalizing *risky* moves in the earlier update layers. More specifically, as the latent variables at the final update layers depend upon all the previous ones through the update function, i.e., $X_i^{k+1} = X_i^k + \alpha\psi_i^k$, the discounted loss can robustify the learning process by regularizing the processors and decoders for all update layers. Figs. 7(i and j) show the RMSE w.r.t. changing the update coefficient (α). Results show that model performance does not change significantly and has mixed trends for heads and flows using different update coefficients. Furthermore, we tested the model when the decoder and processor have shared parameters for all update layers and reported the results in Figs. 7(k–n), respectively. It can be observed that a model with shared parameters, assuming functions in Eqs. (3) and (4) will be the same for all k, yields worse performance. The underlying intuition is that shared parameters impose a strong inductive bias that the nature of information exchange is the same for all update layers and make the GNN analogous to a recurrent model. On the other hand, unshared parameters are more analogous to a deep architecture, which incurs $\hat{k}$ times more parameters and thus allows more flexibility in the training process.

Testing GNN Robustness to Noisy Measurements

In this section, we investigated the impact of noisy measurements on the performance of the proposed GNN model in the semisupervised learning approach with five measurement locations. Specifically, we added noises following normal distribution ($N(0, \sigma_n)$) to the head measurements in the training datasets. Two levels of noise were investigated: $\sigma_n = 0.1m$ and $\sigma_n = 1m$. In this study, the reference values obtained from EPANET are all in the range of 396.6–399 m, with an average of 397.8 m and a standard deviation of 0.46 m. Therefore, the measurement noise following $N(0, 1m)$ can be considered as a high level of noise. The performance on the test datasets, as evaluated by the correlation and RMSE with respect to EPANET results as well as loss percentiles, is reported in the last column of Table 2.

Table 2 shows that even with the high level of noise ($\sigma_n = 1m$), GNN can still achieve satisfactory correlation with the reference values, i.e., 99.55% and 92.96% with EPANET head and flow results, respectively. This performance indicates that under the semisupervised framework, GNN can learn noisy measurements well, which are inevitable in real-life applications. It should be

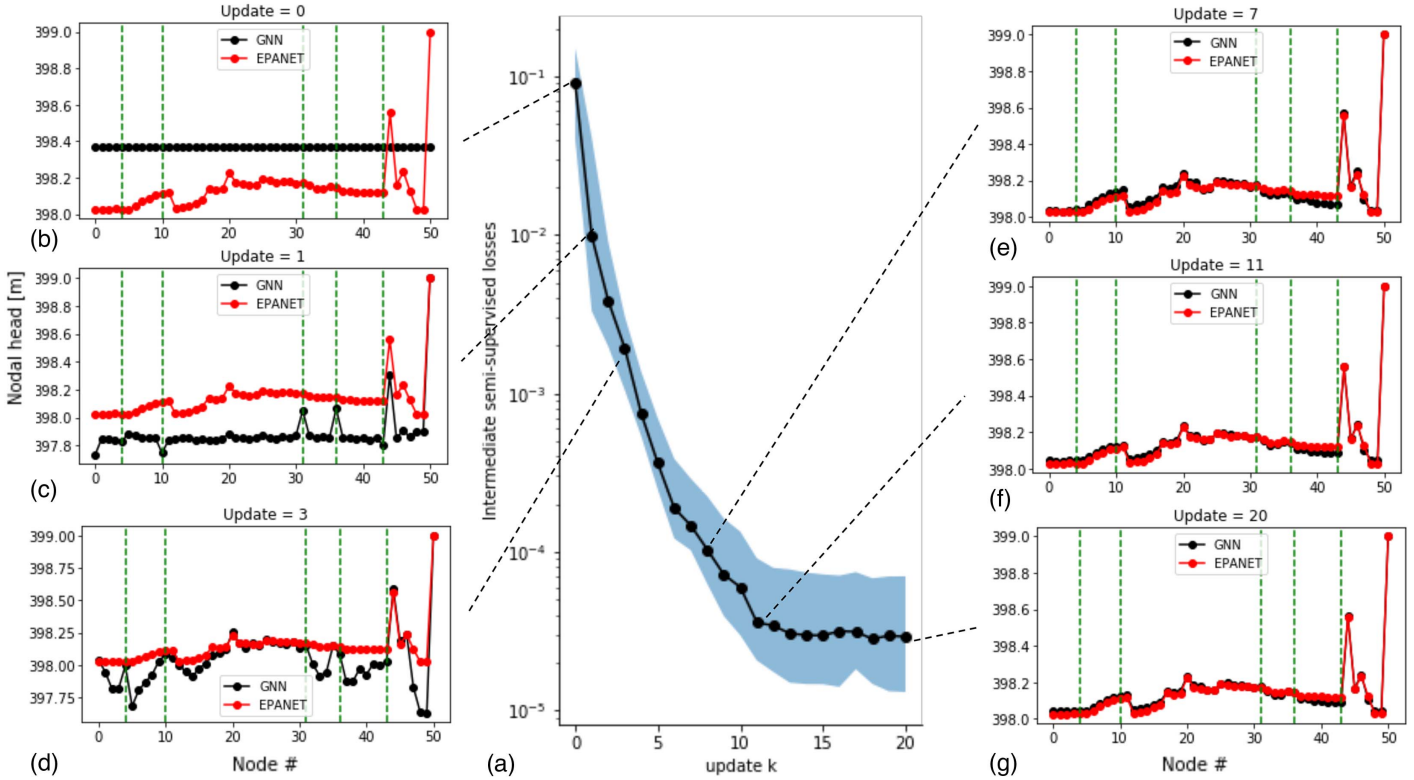

Fig. 6. Intermediate semisupervised losses and head predictions: (a) the semisupervised loss at each update step (the dotted line shows the median of losses computed on all test samples, and the shaded area represents the 10th to 90th percentile of the losses); and (b–g) the nodal head results from EPANET and the GNN model for one test sample as an example.

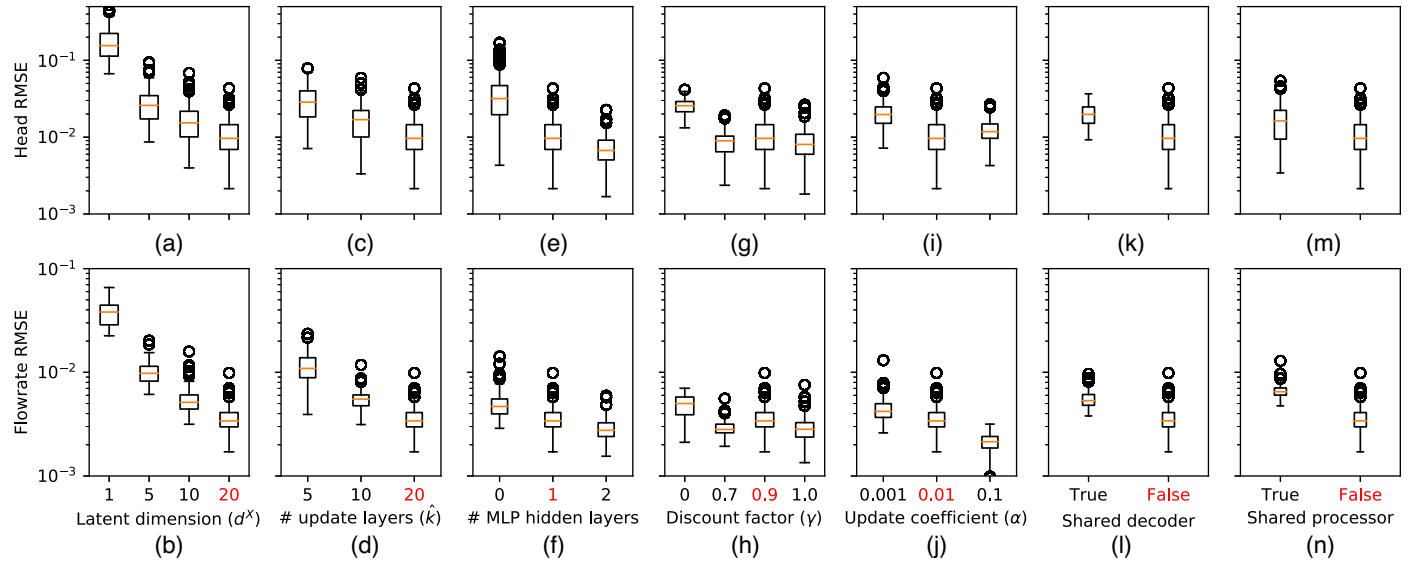

Fig. 7. Effect of different architecture choices compared to the default model on the (a, c, e, g, i, k, and m) head RMSE; and (b, d, f, h, j, l, and n) the flow RMSE.

mentioned that all performance metrics decrease as the noise level increases. This is because the noises added to the measurements in the training set conceal some of the higher-order relations in the mapping from input to output interaction graph. For the same reason, we empirically observed early convergence with the presence of measurement noise.

Conclusions and Future Works

This research proposed a GNN model for hydraulic modeling and SE in WDSs. GNNs not only learn from the node attributes but also from the network topology, thus making them more appropriate for the task of SE with limited measurements. Two training

Table 2. Performance comparison of the GNN models trained with different noise levels as evaluated using the test set with five measurement locations

Model	No noise	Noise ~N(0,0.1m)	Noise ~N(0,1m)
$Corr_H$	**99.96%**	99.88%	99.55%
$Corr_Q$	**99.73%**	98.42%	92.96%
$\overline{\text{RMSE}_H}$	$\mathbf{1.14 \times 10^{-2}}$	2.11×10^{-2}	5.58×10^{-2}
$\sigma(RMSE)_H$	$\mathbf{6.32 \times 10^{-3}}$	1.11×10^{-2}	9.70×10^{-3}
$\overline{\text{RMSE}_Q}$	$\mathbf{3.63 \times 10^{-3}}$	8.94×10^{-3}	2.00×10^{-2}
$\sigma(RMSE)_Q$	$\mathbf{1.06 \times 10^{-3}}$	2.19×10^{-3}	3.92×10^{-3}
Loss 10th percentile	$\mathbf{1.31 \times 10^{-5}}$	6.68×10^{-5}	5.67×10^{-4}
Loss 50th percentile	$\mathbf{2.91 \times 10^{-5}}$	1.48×10^{-4}	7.21×10^{-4}
Loss 90th percentile	$\mathbf{7.01 \times 10^{-5}}$	3.51×10^{-4}	1.14×10^{-3}

Note: Bold numbers represent the best performing metrics.

approaches, i.e., supervised and semisupervised training, were presented. The more traditional supervised approach was tested and demonstrated promising results; however, the supervised approach is not realistic for practical SE applications because it requires knowing the heads at all nodes of the WDSs during the training process, which is a limiting assumption. To overcome this limitation, a semisupervised approach was proposed to minimize the violation of physical laws directly and simultaneously assimilate measurement results at a limited number of locations. The semisupervised training approach provides a novel modeling paradigm in which physical laws and measurement data can be incorporated to improve the training of GNN models in more realistic settings. Although not without limitations, the results demonstrate that semisupervised learning is a promising avenue for data assimilation and SE in WDSs. The incorporation of the physical process enables training with significantly fewer reference values, i.e., measurements or simulation results, and also improves the performance of the trained model. Additionally, our empirical results show that the GNN model is robust to the choice of various hyperparameters and different levels of noise.

Future works should focus on overcoming some of the current limitations by improving the architecture of the GNN models and the training process to obtain models with better performance. To begin with, the current model adopted the least square error metric as the loss function; however, experiments show that the convergence of the training losses is slow and not consistent. Thus, other loss functions, such as least absolute errors and weighted least squared errors, should be investigated (Tshehla et al. 2017). Moreover, more experiments are needed to test the performance of the trained models, especially on larger networks with more complicated components, such as valves and pumps. Additionally, the current models are limited to homogeneous sensors, while at the same time, advances in technologies provide a wide range of data from different types of sensors, such as pressure, flow, and demand meters (Shafiee et al. 2020; Sela and Abbas 2020). Future works should explore the SE problem with heterogeneous information that can enable the integration of information from different sources. Furthermore, future research should investigate the performance of GNNs compared to other data-driven approaches for semisupervised learning on graphs, such as gated graph neural networks (Li et al. 2015), graph convolutional networks (Zhang et al. 2019b), graph attention networks (Veličković et al. 2017), graph recurrent neural networks (Hajiramezanali et al. 2019), and graph signal processing (Ortega et al. 2018). Finally, with the growing rollout of advanced metering and digital twins (Smart Energy International 2018; Shafiee et al. 2018), it is imperative that the performance and limitations of data-driven methods should be compared with model-driven methods, such as those discussed by Tshehla et al. (2017) and Wang et al. (2022).

Data Availability Statement

All the relevant data, code, and training models that support the findings of this study are available from the GitHub repository https://github.com/glorialulu/GNN_StateEstimation_WDS.git.

Acknowledgments

This work was supported by the National Science Foundation under Grant 1943428. The authors would like to acknowledge Balthazar Donon for providing explanations of the GNN implementation for power flows.

Supplemental Materials

Figs. S1–S4 are available online in the ASCE Library (www.ascelibrary.org).

References

Alperovits, E., and U. Shamir. 1977. "Design of optimal water distribution systems." *Water Resour. Res.* 13 (6): 885–900. https://doi.org/10.1029/WR013i006p00885.

Andersen, J., R. S. Powell, and J. Marsh. 2001. "Constrained state estimation with applications in water distribution network monitoring." *Int. J. Syst. Sci.* 32 (6): 807–816. https://doi.org/10.1080/00207720121343.

Andersen, J. H., and R. S. Powell. 2000. "Implicit state-estimation technique for water network monitoring." *Urban Water* 2 (2): 123–130. https://doi.org/10.1016/S1462-0758(00)00050-9.

Battaglia, P. W., et al. 2018. "Relational inductive biases, deep learning, and graph networks." Preprint, submitted October 17, 2018. https://arxiv.org/abs/1806.01261.

Belghaddar, Y., N. Chahinian, A. Seriai, A. Begdouri, R. Abdou, and C. Delenne. 2021. "Graph convolutional networks: Application to database completion of wastewater networks." *Water* 13 (12): 1681. https://doi.org/10.3390/w13121681.

Bohorquez, J., A. R. Simpson, M. F. Lambert, and B. Alexander. 2021. "Merging fluid transient waves and artificial neural networks for burst detection and identification in pipelines." *J. Water Resour. Plann. Manage.* 147 (1): 04020097. https://doi.org/10.1061/(ASCE)WR.1943-5452.0001296.

Bolz, V., J. Rueß, and A. Zell. 2019. "Power flow approximation based on graph convolutional networks." In *Proc., 18th IEEE In. Conf. on Machine Learning and Applications (ICMLA)*, 1679–1686. New York: IEEE.

Boulos, P. F., K. E. Lansey, and B. W. Karney. 2006. *Comprehensive water distribution systems analysis handbook for engineers and planners.* Denver: American Water Works Association.

Díaz, S., J. González, and R. Mínguez. 2016. "Uncertainty evaluation for constrained state estimation in water distribution systems." *J. Water Resour. Plann. Manage.* 142 (12): 06016004. https://doi.org/10.1061/(ASCE)WR.1943-5452.0000718.

Donon, B., R. Clément, B. Donnot, A. Marot, I. Guyon, and M. Schoenauer. 2020a. "Neural networks for power flow: Graph neural solver." *Electr. Power Syst. Res.* 189 (Dec): 106547. https://doi.org/10.1016/j.epsr.2020.106547.

Donon, B., Z. Liu, W. Liu, I. Guyon, A. Marot, and M. Schoenauer. 2020b. "Deep statistical solvers." *Adv. Neural Inf. Process. Syst.* 33: 7910–7921.

Fan, W., Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin. 2019. "Graph neural networks for social recommendation." In *Proc., World Wide Web*

Conf., 417–426. New York: Association for Computing Machinery. https://doi.org/10.1145/3308558.3313488.
Friedman, J., T. Hastie, and R. Tibshirani. 2001. *The elements of statistical learning, Vol. 1. Springer series in statistics*. New York: Springer.
Gilmer, J., S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. 2017. "Neural message passing for quantum chemistry." In *Proc., Int. Conf. on Machine Learning, PMLR*, 1263–1272. London: Proceedings of Machine Learning Research.
Goswami, S., C. Anitescu, S. Chakraborty, and T. Rabczuk. 2020. "Transfer learning enhanced physics informed neural network for phase-field modeling of fracture." *Theor. Appl. Fract. Mech.* 106 (Apr): 102447. https://doi.org/10.1016/j.tafmec.2019.102447.
Guo, G., X. Yu, S. Liu, Z. Ma, Y. Wu, X. Xu, X. Wang, K. Smith, and X. Wu. 2021. "Leakage detection in water distribution systems based on time–frequency convolutional neural network." *J. Water Resour. Plann. Manage.* 147 (2): 04020101. https://doi.org/10.1061/(ASCE)WR.1943-5452.0001317.
Hajiramezanali, E., A. Hasanzadeh, N. Duffield, K. R. Narayanan, M. Zhou, and X. Qian. 2019. "Variational graph recurrent neural networks." Preprint, submitted April 23, 2020. https://arxiv.org/abs/1908.09710.
He, K., X. Zhang, S. Ren, and J. Sun. 2016. "Deep residual learning for image recognition." In *Proc., IEEE Conf. on Computer Vision and Pattern Recognition*, 770–778. New York: IEEE.
Isufi, E., F. Gama, and A. Ribeiro. 2020. "Edgenets: Edge varying graph neural networks." Preprint, submitted July 27, 2021. https://arxiv.org/abs/2001.07620.
Johnson, D. B. 1973. "A note on Dijkstra's shortest path algorithm." *J. ACM (JACM)* 20 (3): 385–388. https://doi.org/10.1145/321765.321768.
Kang, D., and K. Lansey. 2010. "Optimal meter placement for water distribution system state estimation." *J. Water Resour. Plann. Manage.* 136 (3): 337–347. https://doi.org/10.1061/(ASCE)WR.1943-5452.0000037.
Kang, J., Y.-J. Park, J. Lee, S.-H. Wang, and D.-S. Eom. 2018. "Novel leakage detection by ensemble CNN-SVM and graph-based localization in water distribution systems." *IEEE Trans. Ind. Electron.* 65 (5): 4279–4289. https://doi.org/10.1109/TIE.2017.2764861.
Karniadakis, G. E., I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang. 2021. "Physics-informed machine learning." *Nat. Rev. Phys.* 3 (6): 422–440. https://doi.org/10.1038/s42254-021-00314-5.
Kashinath, K., et al. 2021. "Physics-informed machine learning: Case studies for weather and climate modelling." *Philos. Trans. R. Soc. A* 379 (2194): 20200093. https://doi.org/10.1098/rsta.2020.0093.
Kingma, D. P., and J. Ba. 2014. "Adam: A method for stochastic optimization." Preprint, submitted January 30, 2017. https://arxiv.org/abs/1412.6980.
Kumar, S. M., S. Narasimhan, and S. M. Bhallamudi. 2008. "State estimation in water distribution networks using graph-theoretic reduction strategy." *J. Water Resour. Plann. Manage.* 134 (5): 395–403. https://doi.org/10.1061/(ASCE)0733-9496(2008)134:5(395).
Larock, B. E., R. W. Jeppson, and G. Z. Watters. 1999. *Hydraulics of pipeline systems.* Boca Raton, FL: CRC Press.
Li, Y., D. Tarlow, M. Brockschmidt, and R. Zemel. 2015. "Gated graph sequence neural networks." Preprint, submitted September 22, 2017. https://arxiv.org/abs/1511.05493.
Liao, W., B. Bak-Jensen, J. R. Pillai, Y. Wang, and Y. Wang. 2021. "A review of graph neural networks and their applications in power systems." Preprint, submitted June 12, 2021. https://arxiv.org/abs/2101.10025.
Mai, V. V., and M. Johansson. 2021. "Stability and convergence of stochastic gradient clipping: Beyond Lipschitz continuity and smoothness." Preprint, submitted June 10, 2021. https://arxiv.org/abs/2102.06489.
Mounce, S. R., and J. Machell. 2006. "Burst detection using hydraulic data from water distribution systems with artificial neural networks." *Urban Water J.* 3 (1): 21–31. https://doi.org/10.1080/15730620600578538.
Nagar, A. K., and R. S. Powell. 2002. "LFT/SDP approach to the uncertainty analysis for state estimation of water distribution systems." *IEEE Proc. Control Theory Appl.* 149 (2): 137–142. https://doi.org/10.1049/ip-cta:20020096.
Ortega, A., P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst. 2018. "Graph signal processing: Overview, challenges, and applications." *Proc. IEEE* 106 (5): 808–828. https://doi.org/10.1109/JPROC.2018.2820126.
Preis, A., A. J. Whittle, A. Ostfeld, and L. Perelman. 2011. "Efficient hydraulic state estimation technique using reduced models of urban water networks." *J. Water Resour. Plann. Manage.* 137 (4): 343–351. https://doi.org/10.1061/(ASCE)WR.1943-5452.0000113.
Raissi, M., P. Perdikaris, and G. E. Karniadakis. 2019. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations." *J. Comput. Phys.* 378 (Feb): 686–707. https://doi.org/10.1016/j.jcp.2018.10.045.
Rossman, L., H. Woo, M. Tryby, F. Shang, R. Janke, and T. Haxton. 2020. *EPANET 2.2 user's manual*. Washington, DC: USEPA.
Sanchez-Gonzalez, A., J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. Battaglia. 2020. "Learning to simulate complex physics with graph networks." In *Proc., Int. Conf. on Machine Learning, PMLR*, 8459–8468. London: Proceedings of Machine Learning Research.
Scarselli, F., M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. 2009. "The graph neural network model." *IEEE Trans. Neural Networks* 20 (1): 61–80. https://doi.org/10.1109/TNN.2008.2005605.
Sela, L., and W. Abbas. 2020. *Distributed sensing for monitoring water distribution systems*, 1–9. London: Springer.
Shafiee, M. E., Z. Barker, and A. Rasekh. 2018. "Enhancing water system models by integrating big data." *Sustainable Cities Soc.* 37 (Feb): 485–491. https://doi.org/10.1016/j.scs.2017.11.042.
Shafiee, M. E., A. Rasekh, L. Sela, and A. Preis. 2020. "Streaming smart meter data integration to enable dynamic demand assignment for real-time hydraulic simulation." *J. Water Resour. Plann. Manage.* 146 (6): 06020008. https://doi.org/10.1061/(ASCE)WR.1943-5452.0001221.
Shukla, K., P. C. Di Leoni, J. Blackshire, D. Sparkman, and G. E. Karniadakis. 2020. "Physics-informed neural network for ultrasound nondestructive quantification of surface breaking cracks." *J. Nondestr. Eval.* 39 (3): 1–20. https://doi.org/10.1007/s10921-020-00705-1.
Smart Energy International. 2018. "Global smart water meters market trends." Accessed December 18, 2021. https://www.smart-energy.com/magazine-article/global-smart-water-meters-market-trends/.
Todini, E., and S. Pilati. 1988. "A gradient algorithm for the analysis of pipe networks." In *Computer applications in water supply: Vol. 1—Systems analysis and simulation*, 1–20. Baldock, UK: Research Studies Press.
Tshehla, K. S., Y. Hamam, and A. M. Abu-Mahfouz. 2017. "State estimation in water distribution network: A review." In *Proc., IEEE 15th Int. Conf. on Industrial Informatics (INDIN)*, 1247–1252. New York: IEEE.
Tsiami, L., and C. Makropoulos. 2021. "Cyber—Physical attack detection in water distribution systems with temporal graph convolutional neural networks." *Water* 13 (9): 1247. https://doi.org/10.3390/w13091247.
Veličković, P., G.. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. 2017. "Graph attention networks." Preprint, submitted February 4, 2018. https://arxiv.org/abs/1710.10903.
Wang, J.-X., J.-L. Wu, and H. Xiao. 2017. "Physics-informed machine learning approach for reconstructing Reynolds stress modeling discrepancies based on DNS data." *Phys. Rev. Fluids* 2 (3): 034603. https://doi.org/10.1103/PhysRevFluids.2.034603.
Wang, S., A. F. Taha, N. Gatsis, L. Sela, and M. H. Giacomoni. 2022. "Probabilistic state estimation in water networks." *IEEE Trans. Control Syst. Technol.* 30 (2): 507–519. https://doi.org/10.1109/TCST.2021.3066102.
Wang, X., Y. Ma, Y. Wang, W. Jin, X. Wang, J. Tang, C. Jia, and J. Yu. 2020. "Traffic flow prediction via spatial temporal graph neural network." In *Proc., Web Conf. 2020*, 1082–1092. New York: Association for Computing Machinery. https://doi.org/10.1145/3366423.3380186.
Wu, J.-L., H. Xiao, and E. Paterson. 2018. "Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework." *Phys. Rev. Fluids* 3 (7): 074602. https://doi.org/10.1103/PhysRevFluids.3.074602.
Xing, L., and L. Sela. 2021. "Graph neural networks for state estimation in water distribution systems." Accessed September 5, 2021. https://github.com/glorialulu/GNN_StateEstimation_WDS.git.
Yin, R., K. Li, G. Zhang, and J. Lu. 2019. "A deeper graph neural network for recommender systems." *Knowl.-Based Syst.* 185 (Dec): 105020. https://doi.org/10.1016/j.knosys.2019.105020.

Ying, R., R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec. 2018. "Graph convolutional neural networks for web-scale recommender systems." In *Proc., 24th ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining*, 974–983. New York: Association for Computing Machinery. https://doi.org/10.1145/3219819.3219890.

Zhang, J., T. He, S. Sra, and A. Jadbabaie. 2019a. "Why gradient clipping accelerates training: A theoretical justification for adaptivity." Preprint, submitted February 10, 2020. https://arxiv.org/abs/1905.11881.arXiv preprint arXiv:1905.11881.

Zhang, S., H. Tong, J. Xu, and R. Maciejewski. 2019b. "Graph convolutional networks: A comprehensive review." *Comput. Soc. Networks* 6 (1): 1–23. https://doi.org/10.1186/s40649-019-0069-y.

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