Physics-Informed Graph Neural Networks for Water Distribution Systems

Inaam Ashraf, Janine Strotherm, Luca Hermes, Barbara Hammer

Center for Cognitive Interaction Technology, Bielefeld University, Inspiration 1, 33619 Bielefeld, Germany
mashraf@techfak.uni-bielefeld.de, jstrotherm@techfak.uni-bielefeld.de, lhermes@techfak.uni-bielefeld.de,
bhammer@techfak.uni-bielefeld.de

Abstract

Water distribution systems (WDS) are an integral part of critical infrastructure which is pivotal to urban development. As 70% of the world's population will likely live in urban environments in 2050, efficient simulation and planning tools for WDS play a crucial role in reaching UN's sustainable developmental goal (SDG) 6 – "Clean water and sanitation for all". In this realm, we propose a novel and efficient machine learning emulator, more precisely, a physics-informed deep learning (DL) model, for hydraulic state estimation in WDS. Using a recursive approach, our model only needs a few graph convolutional neural network (GCN) layers and employs an innovative algorithm based on message passing. Unlike conventional machine learning tasks, the model uses hydraulic principles to infer two additional hydraulic state features in the process of reconstructing the available ground truth feature in an unsupervised manner. To the best of our knowledge, this is the first DL approach to emulate the popular hydraulic simulator EPANET, utilizing no additional information. Like most DL models and unlike the hydraulic simulator, our model demonstrates vastly faster emulation times that do not increase drastically with the size of the WDS. Moreover, we achieve high accuracy on the ground truth and very similar results compared to the hydraulic simulator as demonstrated through experiments on five real-world WDS datasets.

Introduction

As it is expected that 70% of world's population will live in urban areas in 2050 (Ritchie and Roser 2018), urban development presents a number of complex challenges. In the light of deep uncertainties caused, among others, by climate change and migration, building smart cities requires flexible and efficient short and long-term planning, monitoring and expansion of its critical infrastructure, i.e., its transportation systems, electricity distribution systems, and water distribution systems (WDS). Reliable WDS, in particular, constitute one critical demand of UN's SDG 6. In this realm, AI technologies carry great promises since AI can be used for intelligent planning, monitoring and control of these systems (Eichenberger and et al. 2022; Omitaomu and Niu 2021; Kammoun, Kammoun, and Abid 2022). Currently, there exist first approaches of both classical AI tools and Deep Learning (DL) to solve various tasks in critical infrastructure systems (Dick et al. 2019). In this contribution, we center on one important aspect related to planning and control of WDS: Since WDS cannot be experimented with in practice, the efficient emulation of realistic large-scale WDS constitutes a crucial prerequisite for planning and optimization based on such digital twins. We investigate how to derive an efficient emulation of WDS based on DL.

Currently, hydraulic simulators play a key role in planning and expansion of WDS. The EPANET simulator (Rossman et al. 2020) is one of the most popular hydraulic simulator for WDS. However, it requires considerable resources for extended time simulations that scale non-linearly with the size of the WDS, making WDS optimization a slow process. Since critical infrastructure systems such as WDS are naturally structured as graphs, Graph Neural Networks (GNNs) carry great promises as an alternative, faster, modeling tool.

A WDS consists of N_n junctions and N_e links connecting these junctions. Junctions can be reservoirs, tanks or water consumers, while links can be pipes, valves or pumps. The first step in planning a WDS is to define the structure, i.e., the number of junctions and links. Next, one can estimate the hydraulic state at every junction and link, given a set of features. These features are the head (see section) at the reservoirs and the water demand of every consumer. Based on this, one needs to estimate the head at *every* junction and the flow through *every* link. EPANET iteratively solves N_n differential equations (Rossman et al. 2020), leading to slow simulations for extended time or larger systems.

In this paper, we combine a local graph convolutional neural network (GCN) model with a physics informed global algorithm that emulates the hydraulic simulator. Since explicit values for heads or flows are not available, we use hydraulic formulas governing the relationships between demands, heads and flows, instead, as physics-informed learning signals.

Our key contributions are as follows:

- We propose a novel GNN architecture which combines trainable local aspects with global state estimation.
- We use hydraulic principles to infer the required state features rather than example-driven supervised learning.
- We display the accuracy of the model for various WDS benchmarks.

- We show that we can significantly reduce the inference times compared to the hydraulic simulator.
- To the best of our knowledge, this is the first DL approach emulating the simulator using no additional information.

Related Work

Curently, ML technologies have been proposed for specific tasks in WDS such as leakage detection (Fan, Zhang, and Yu 2021), modeling virtual sensors (Magini et al. 2023), or demand prediction (Wu et al. 2023). To the best of our knowledge, the complex task of state estimation has only been dealt with by (Xing and Sela 2022), addressing model hydraulics using GNNs. More specifically, the EPANET simulator (Rossman et al. 2020) is emulated using various information such as sparse pressure sensor measurements as features; this additional information makes the problem significantly easier to solve and it is usually not available in a WDS planning phase. Moreover, the method is demonstrated on an artificial small WDS only.

In deployed WDS, pressure readings from few sensors are available. This leads to the task of pressure estimation at all nodes in a WDS. (Hajgató, Gyires-Tóth, and Paál 2021) used spectral GCNs to achieve encouraging results demonstrated by extensive experiments. Spectral GCNs do not fully utilize the structural information in a graph. Spatial GCNs were used by (Ashraf et al. 2023) to significantly improve the performance. Generally, there exists a variety of different GNN structures, such as recursive graph and tree models (Scarselli et al. 2009; Hammer 2000), spectral GCNs (Bruna et al. 2014; Kipf and Welling 2017; Defferrard, Bresson, and Vandergheynst 2016; Henaff, Bruna, and LeCun 2015; Levie et al. 2018; Li et al. 2018), or spatial GCNs based on the spectral graph kernel (Hamilton, Ying, and Leskovec 2017; Monti et al. 2017; Gao, Wang, and Ji 2018; Niepert, Ahmed, and Kutzkov 2016; Xu et al. 2019; Veličković et al. 2018). Since these local operations are analogous to sending and receiving messages between nodes, spatial GCNs are also called message passing neural networks. In this contribution, we will extend such first advances towards an efficient physics-informed model capable of emulating large-scale WDS based on demands signals only.

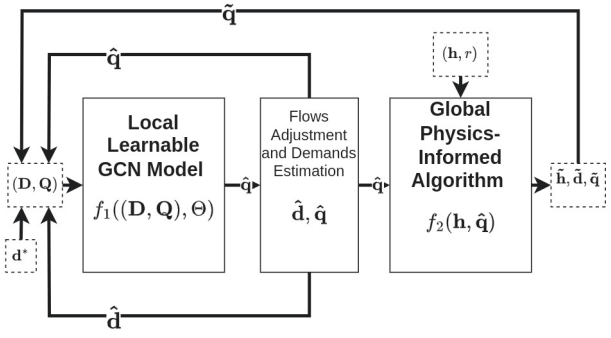

Figure 1: The model architecture: The local GCN model f_1 learns from the global physics-informed algorithm f_2 through multiple iterations.

The majority of ML training schemes are supervised based on example data; moreover, DL often requires a considerable amount of training data for valid generalization. When dealing with complex systems, such information is often not easily available. Hence researchers investigate how to minimize the required amount of training data when using ML models as efficient surrogates for complex systems (Ruff et al. 2023), and some technologies even enable the exchange of training data by general physical principles which can be incorporated into the learning scheme (Raissi, Perdikaris, and Karniadakis 2019). In this contribution, we introduce a novel graph architecture and learning scheme, which enables us to train a surrogate model based on physical principles only.

Methodology

WDS systems are characterized by demands $d \in \mathbb{R}_+$, flows $q \in \mathbb{R}$ and heads $h \in \mathbb{R}_+$. The latter is a measure of energy with the relationship $h = p + \epsilon$ to the pressure $p \in \mathbb{R}_+$ and the elevation of the junction $\epsilon \in \mathbb{R}_+$ (Rossman et al. 2020). Before we further explain the water hydraulics in section , we present an overview of our methodology.

Overview

We propose a spatial GCN-based and physics-informed iterative model to emulate the hydraulic simulator EPANET for WDS. As in case of EPANET, our model takes reservoir heads and consumer demands at every junction as inputs and estimates heads at every junction and flows at every link. The strength of our model comes from two main components.

The first component is a GCN-based, learnable function f_1, that makes use of the graph structure of a WDS to update flows based on demands and initial flows. When modelling a WDS as a graph with junctions $V = \{v_1, \dots, v_{N_n}\}$ as nodes and links $E = \{e_{vu} \mid \forall\, v \in V; u \in \mathcal{N}(v)\} = \{e_1, \dots, e_{N_e}\}$ as edges, f_1 takes *demand node features* $\mathbf{D} \in \mathbb{R}^{N_n \times M_n}$ and *flow edge features* $\mathbf{Q} \in \mathbb{R}^{N_e \times M_e}$ as inputs with $M_n = M_e = 2$ as feature dimension per node and edge. f_1 estimates updated *flows* $\hat{\mathbf{q}} = (\hat{q}_e)_{e \in E} \in \mathbb{R}^{N_e}$:

$$\begin{aligned} f_1(\cdot, \Theta):\ \mathbb{R}^{N_n \times M_n} \times \mathbb{R}^{N_e \times M_e} &\longrightarrow \mathbb{R}^{N_e} \\ (\mathbf{D}, \mathbf{Q}) &\longmapsto \hat{\mathbf{q}}, \end{aligned} \tag{1}$$

where the matrices can be rewritten as

$$(\mathbf{D}, \mathbf{Q}) = ((\mathbf{d}_1, \mathbf{d}_2), (\mathbf{q}_1, \mathbf{q}_2)) = ((\mathbf{d}_v^T)_{v \in V}, (\mathbf{q}_e^T)_{e \in E})$$

with $\mathbf{d}_v^T = (d_{v1}, d_{v2})$ and $\mathbf{q}_e^T = (q_{e1}, q_{e2})$ consisting of the ground truth demand d_{v1}, as well as firstly initialized and afterwards to be updated demand d_{v2} and flows q_{e1} and q_{e2} for each node $v \in V$ and edge $e \in E$. We provide further details on initialization in section . Using the updated flows $\hat{\mathbf{q}}$, we also compute associated updated demands $\hat{\mathbf{d}} \in \mathbb{R}^{N_n}$ (cf. eq. (3)).

The second component is a physics-informed function

$$\begin{aligned} f_2:\ \mathbb{R}^{N_n} \times \mathbb{R}^{N_e} &\longrightarrow \mathbb{R}^{N_n} \times \mathbb{R}^{N_e} \\ (\mathbf{h}, \hat{\mathbf{q}}) &\longmapsto (\tilde{\mathbf{h}}, \tilde{\mathbf{d}}, \tilde{\mathbf{q}}) \end{aligned} \tag{2}$$

that makes use of initialized *heads* $\mathbf{h} \in \mathbb{R}^{N_n}$ and the predicted flows $\hat{\mathbf{q}} = f_1(\mathbf{D}, \mathbf{Q}) \in \mathbb{R}^{N_e}$ to compute updated heads $\tilde{\mathbf{h}} \in \mathbb{R}^{N_n}$, flows $\tilde{\mathbf{q}} \in \mathbb{R}^{N_e}$ and demands $\tilde{\mathbf{d}} \in \mathbb{R}^{N_n}$ based on a recursive scheme that takes into account water hydraulics.

It becomes obvious that during computation of f_1 and f_2, we consider several demands as well as flows. As we do not have labels for this task, these estimations replace prediction and label pairs: They might differ in the beginning, but converge towards the same values during training of the function $f := f_2 \circ f_1(\cdot, \Theta)$, relying on the several parameters Θ of the GCN layers (cf. theorem 0.2, 0.3 and section).

A visualization of the model architecture is given in fig. 1. In the following sections, we introduce the components and the functionality of the overall resulting model in detail.

Local Learnable GCN-Model

As a first step of the computation of f_1, the node and edge features are embedded in a latent space with dimension M_l by applying fully connected neural network layers α and β:

$$\mathbb{R}^{N_n \times M_n} \times \mathbb{R}^{N_e \times M_e} \longrightarrow \mathbb{R}^{N_n \times M_l} \times \mathbb{R}^{N_e \times M_l}$$
$$((\mathbf{d}_v^T)_{v \in V}, (\mathbf{q}_e^T)_{e \in E}) \longmapsto ((\mathbf{g}_v^T)_{v \in V}, (\mathbf{z}_e^T)_{e \in E}),$$

where for $v \in V$ and $e \in E$, respectively, we define

$$\mathbf{g}_v := \alpha(\mathrm{SeLU}(\mathbf{d}_v)) \text{ and } \mathbf{z}_e := \beta(\mathrm{SeLU}(\mathbf{q}_e)).$$

Hereby, the SeLU activation function is used for inducing self-normalizing properties (Klambauer et al. 2017).

Afterwards, we conduct the standard three-step process of message generation, message aggregation and feature update (Li et al. 2020). This process is repeated for I GCN layers, starting with $\mathbf{g}_v^{(0)} = \mathbf{g}_v$ and $\mathbf{z}_e^{(0)} = \mathbf{z}_e$ for all $v \in V$ and $e \in E$, respectively: For the i-th layer for $i = 0, ..., I-1$, for **message generation**, the latent node and edge features are fed to a Multi-Layer Perceptron (MLP) $\gamma^{(i)}$:

$$\mathbb{R}^{N_n \times M_l} \times \mathbb{R}^{N_e \times M_l} \longrightarrow \mathbb{R}^{N_e \times M_l}$$
$$((\mathbf{g}_v^{(i)})^T)_{v \in V}, ((\mathbf{z}_e^{(i)})^T)_{e \in E}) \longmapsto ((\mathbf{m}_e^{(i)})^T)_{e \in E},$$

where for $e = e_{vu} \in E$, we define

$$\mathbf{m}_e^{(i)} := \gamma^{(i)}(\mathrm{SeLU}(\mathbf{g}_u^{(i)} \parallel \mathbf{g}_v^{(i)} \parallel \mathbf{z}_e^{(i)})) \text{ with } u \in \mathcal{N}(v)$$

and where $\cdot \parallel \cdot$ denotes vector concatenation. For **message aggregation**, max aggregation is used:

$$\mathbb{R}^{N_e \times M_l} \longrightarrow \mathbb{R}^{N_n \times M_l} \times \mathbb{R}^{N_e \times M_l}$$
$$((\mathbf{m}_e^{(i)})^T)_{e \in E} \longmapsto (((\mathbf{m}_v^{(i)})^T)_{v \in V}, ((\mathbf{m}_e^{(i)})^T)_{e \in E}),$$

where for $v \in V$, we define

$$\mathbf{m}_v^{(i)} := \max_{u \in \mathcal{N}(v)} \mathbf{m}_{e_{vu}}^{(i)}.$$

For the **feature update**, the node messages are fed to another MLP $\eta^{(i)}$ whereas the edge features are updated as the edge messages:

$$\mathbb{R}^{N_n \times M_l} \times \mathbb{R}^{N_e \times M_l} \to \mathbb{R}^{N_n \times M_l} \times \mathbb{R}^{N_e \times M_l}$$
$$(((\mathbf{m}_v^{(i)})^T)_{v \in V}, ((\mathbf{m}_e^{(i)})^T)_{e \in E}) \mapsto ((\mathbf{g}_v^{(i+1)})^T)_{v \in V}, ((\mathbf{z}_e^{(i+1)})^T)_{e \in E}),$$

where for $e = e_{vu} \in E$, we define

$$\mathbf{g}_v^{(i+1)} := \eta^{(i)}(\mathbf{m}_v^{(i)}) \text{ and } \mathbf{z}_e^{(i+1)} := \mathbf{m}_e^{(i)}.$$

After the last GCN layer I, new flows are estimated using another MLP λ:

$$\mathbb{R}^{N_n \times M_l} \times \mathbb{R}^{N_e \times M_l} \longrightarrow \mathbb{R}^{N_e}$$
$$((\mathbf{g}_v^{(I)})^T)_{v \in V}, ((\mathbf{z}_e^{(I)})^T)_{e \in E}) \longmapsto (\hat{q}_e)_{e \in E},$$

where we define with slight abuse of notation[1]

$$\hat{q}_e := q_{e1} + \lambda(\mathrm{SeLU}(\mathbf{g}_u^{(I)} \parallel \mathbf{g}_v^{(I)} \parallel \mathbf{z}_e^{(I)}) \text{ with } u \in \mathcal{N}(v)$$

for $e = e_{vu} \in E$. As $f_1((\mathbf{D}, \mathbf{Q}), \Theta) := \hat{\mathbf{q}} = (\hat{q}_e)_{e \in E}$, this is the last step of the learnable GCN-based model f_1, determined by all parameters Θ of all networks $\alpha, \beta, \gamma^{(i)}, \eta^{(i)}$ and λ and for all $i = 0, ..., I-1$. While the structure of these networks is standard, their dynamics are special: We iteratively update the outputs of the overall model $f_2 \circ f_1$ before updating the parameters Θ, as we will discuss in section .

Water Hydraulics

Up to this point, we use GCNs to estimate flows $f_1((\mathbf{D}, \mathbf{Q}), \Theta) = \hat{\mathbf{q}}$ based on input demands $\mathbf{D}$ and input flows $\mathbf{Q}$. Next, we explain how we change these flows such that they obey the principles of water hydraulics.

On the one hand, for two neighboring nodes $v, u \in V$, the flow $q_{e_{vu}}$ from node v to u has to be equal to the negative of the flow $q_{e_{uv}}$ from u to v. Neural networks are not inherently aware of this property. Therefore, after computing flows in both directions by using bidirectional edges in our graph, we discard half of the flows in $\hat{\mathbf{q}}$ and replace each of those with the negative of the remaining corresponding flow.[2]

On the other hand, *true* demands $\mathbf{d}^* = (d_v^*)_{v \in V}$, *true* heads $\mathbf{h}^* = (h_v^*)_{v \in V}$ and *true* flows $\mathbf{q}^* = (q_e^*)_{e \in E}$ are subject to further hydraulic principles: Firstly, due to flow preservation, the sum of incoming and outcoming flows e_{vu} between a node $v \in V$ and its neighbors $u \in \mathcal{N}(v)$ is related to its demand d_v by

$$\textstyle\sum_{u \in \mathcal{N}(v)} q_{e_{vu}}^* = -d_v^*. \tag{3}$$

Hereby, we use the convention that the flow $q_{e_{vu}}$ from v to u has a positive sign if it corresponds to an *outflow* (including demands) and a negative sign if it corresponds to an *inflow* (Rossman et al. 2020). This can be seen by the second property, which states that for any two neighboring nodes $v, u \in V$, pressure heads are related to flows by

$$h_v^* - h_u^* = r_{e_{vu}} \operatorname{sgn}(q_{e_{vu}}^*) |q_{e_{vu}}^*|^x, \tag{4}$$

[1] Note that the dependence of the first component $\mathbf{q}_1 \in \mathbb{R}^{N_e}$ of the input flows $\mathbf{Q} \in \mathbb{R}^{N_e \times M_e}$ added to the output of the MLP λ can easily be integrated in all of the previous functions by adding a second and third factor to each of the tuples and triples, respectively, and using the identity on $\mathbb{R}^{N_e}$ to make $\mathbf{q}_1$ as an input of the latter function. We have only waived this step to save space.

[2] Since at the beginning, we do not know flow directions, edge directions can be initialized arbitrarily and any of the two half sets of flows can be used.

where $x = 1.852$ and

$$r_{e_{vu}} = 10.667\, l_{e_{vu}} d_{e_{vu}}^{-4.871} c_{e_{vu}}^{-1.852} > 0 \tag{5}$$

is a link-dependent constant based on its length $l_{e_{vu}}$, diameter $d_{e_{vu}}$ and roughness coefficient $c_{e_{vu}}$ (Rossman et al. 2020). Now, as water always (out)flows from a higher head to a lower head, if the head loss $h_v - h_u$ is positive as the head h_v at node v is higher than the head h_u at the neighbor node u, the water outflow q_{vu} needs to be positive and similarly, an inflow q_{vu} needs to be negative.

As we have not utilized any head values in the model f_1 explicitly, during learning, it could choose any set of flows that can satisfy eq. (3) at every node. Yet, such a solution will most likely not obey eq. (4) due to the following lemma:

Lemma 0.1. *Given N_n nodes $v \in V$ and $N_e > N_n$ edges $e \in E$, there is no unique solution to eq.* (3)*.*

Proof. All proofs can be found in the ArXiv version. □

As in practice, $N_e > N_n$ is usually the case (even for the directed graph), we leverage the message passing framework using water hydraulics within the second component f_2 of our overall model $f = f_2 \circ f_1$ to guide the model to a plausible solution, as described in the following section.

Global Physics-Informed Algorithm

The following components of f_2 use eq. (4) to construct the heads $\tilde{\mathbf{h}} \in \mathbb{R}^{N_n}$ and the updated flows $\tilde{\mathbf{q}} \in \mathbb{R}^{N_e}$ as well as the updated demands $\tilde{\mathbf{d}} \in \mathbb{R}^{N_e}$ based on some initialized heads $\mathbf{h} \in \mathbb{R}^{N_n}$ and the outcome $f_1((\mathbf{D}, \mathbf{Q}), \Theta) := \hat{\mathbf{q}} \in \mathbb{R}^{N_e}$ of the first component.

The computation of $f_2(\mathbf{h}, \hat{\mathbf{q}})$ is inspired by the three-step process of message generation, message aggregation and feature update, however, without the usage of trainable parameters. Instead, we define a recursive algorithm that converges to a fix point, as we will show in Theorem 0.2. Thus, starting with $\tilde{h}_v^{(0)} = h_v$ for all $v \in V$, we repeat the following steps J times where $J \in \mathbb{N}$ is the iteration where the algorithm reaches its fixed point: For the j-th iteration for $j = 0, ..., J-1$, for **message generation**, we compute

$$\mathbb{R}^{N_n} \times \mathbb{R}^{N_e} \longrightarrow \mathbb{R}^{N_n} \times \mathbb{R}^{N_e}$$
$$((\tilde{h}_v^{(j)})_{v\in V}, (\hat{q}_e)_{e\in E}) \longmapsto ((\tilde{h}_v^{(j)})_{v\in V}, (m_e^{(j)})_{e\in E}),$$

where for $e = e_{vu} \in E$, we define

$$m_e^{(j)} := \tilde{h}_u^{(j)} - \mathrm{ReLU}\left(-r_e \operatorname{sgn}(\hat{q}_e)|\hat{q}_e|^x\right). \tag{6}$$

For **message aggregation**, max aggregation is used:

$$\mathbb{R}^{N_n} \times \mathbb{R}^{N_e} \longrightarrow \mathbb{R}^{N_n} \times \mathbb{R}^{N_n}$$
$$((\tilde{h}_v^{(j)})_{v\in V}, (m_e^{(j)})_{e\in E}) \longmapsto ((\tilde{h}_v^{(j)})_{v\in V}, (m_v^{(j)})_{v\in V}),$$

where for $v \in V$, we define

$$m_v^{(j)} := \max_{u\in\mathcal{N}(v)} m_{e_{vu}}^{(j)}.$$

Finally, the **feature update** outputs the heads $\tilde{h}^{(j+1)} \in \mathbb{R}^{N_n}$ of the next iteration[3]:

$$\mathbb{R}^{N_n} \times \mathbb{R}^{N_n} \longrightarrow \mathbb{R}^{N_n}$$
$$((\tilde{h}_v^{(j)})_{v\in V}, (m_v^{(j)})_{v\in V}) \longmapsto (\tilde{h}_v^{(j+1)})_{v\in V},$$

where for $v \in V$, we define

$$\tilde{h}_v^{(j+1)} := \max\{\tilde{h}_v^{(j)}, m_v^{(j)}\}. \tag{7}$$

Theorem 0.2. *Let $V_r \subset V$ be a subset of non-connecting nodes of the WDS with associated values $(h_v^*)_{v\in V_r}$, $L_{\max}$ the length of the longest path from any instance $v_r \in V_r$ to any instance $v \in V$ and define $c := \min_{v_r\in V_r} h_{v_r}^* - L_{\max} \cdot \max_{e\in E} \mathrm{ReLU}\left(-r_e \operatorname{sgn}(\hat{q}_e)|\hat{q}_e|^x\right)$. If we initialize $\tilde{\mathbf{h}}^{(0)} = \mathbf{h} \in \mathbb{R}^{N_n}$ according to*

$$h_v := \begin{cases} h_v^* & \text{if } v \in V_r \\ c & \text{if } v \in V \setminus V_r, \end{cases}$$

the recursive algorithm defined by eq. (7) *converges in at most $J = L_{\max}$ steps.*

After this recursive algorithm has converged, the updated demands $\tilde{\mathbf{d}} \in \mathbb{R}^{N_n}$ as well as updated flows $\tilde{\mathbf{q}} \in \mathbb{R}^{N_e}$ are reconstructed from the heads $\tilde{\mathbf{h}} := \tilde{\mathbf{h}}^{(J)} \in \mathbb{R}^{N_n}$ based on the water hydraulics from eq. (3) and (4):

$$\mathbb{R}^{N_n} \longrightarrow \mathbb{R}^{N_n}$$
$$(\tilde{h}_v)_{v\in V} \longmapsto ((\tilde{d}_v)_{v\in V}, (\tilde{q}_e)_{e\in E}),$$

where for $v \in V$ and $e = e_{vu} \in E$, we define

$$\begin{aligned} \tilde{q}_e &:= \operatorname{sgn}(\tilde{h}_v - \tilde{h}_u) \cdot (r_e^{-1}|\tilde{h}_v - \tilde{h}_u|)^{1/x} + \zeta, \\ \tilde{d}_v &:= -\textstyle\sum_{u\in\mathcal{N}(v)} \tilde{q}_{e_{vu}}, \end{aligned} \tag{8}$$

and ζ is a small number that ensures that this equation remains differentiable. This is the last step of the physics-informed component f_2, mapping initialized heads $\mathbf{h}$ and the flows $\hat{\mathbf{q}} = f_1((\mathbf{D}, \mathbf{Q}), \Theta)$ given by the first component f_1 to updated heads, demands and flows $(\tilde{\mathbf{h}}, \tilde{\mathbf{d}}, \tilde{\mathbf{q}}) = f_2((\mathbf{h}, \hat{\mathbf{q}})) = f_2((\mathbf{h}, f_1((\mathbf{D}, \mathbf{Q}), \Theta)))$. Note that all computations within this component do not involve any learnable parameters, but are only based on the hydraulics of the WDS.

A deeper intuition of this component can be found in the ArXiv version. One important property of the physics-informed algorithm is that if the flows $\hat{\mathbf{q}}$ estimated by the GCN are correct, they remain unchanged by the algorithm:

Theorem 0.3. *If in the setting theorem 0.2, V_r corresponds to the reservoirs with known heads $(h_v^*)_{v\in V_r}$ and $\hat{\mathbf{q}}$ corresponds to the true flows, then $\tilde{\mathbf{q}} = \hat{\mathbf{q}} + \zeta$ holds.*

Therefore, enforcing the equality of $\tilde{\mathbf{q}}$ and $\hat{\mathbf{q}}$ by choosing a suitable loss function will guide us to a physically correct solution for flows, as we discuss in the following section.

[3] We can of course make $\hat{\mathbf{q}}$ the output of the latter function by again adding a second and third factor to each of the tuples and triples, respectively, and using the identity on $\mathbb{R}^{N_e}$.

Overall Model and Training

We obtain an overall model $f(\cdot, \Theta) = f_2 \circ f_1(\cdot, \Theta)$ leveraging GCNs and the water hydraulics. Overall dynamics and training incorporate two further novel design principles:

Since observational *training* data are not easily available, we do not train f in the typical supervised learning sense. Available data are the true demands $\mathbf{d}^* \in \mathbb{R}^{N_n}$ at consumer nodes and the true heads $(h_v^*)_{v \in V_r}$ at the reservoir nodes $V_r \subset V$. This information is used together with error terms which confirm that physical constraints of the hydraulics are fulfilled, as we will detail below.

The *model dynamics* has to ensure that information can spread through all nodes of the WDS; therefore, before updating the model's parameters Θ, for $K \in \mathbb{N}$, it is applied K-times to each sample of a training batch. This iterative scheme is defined as follows: As a first step, for each sample of a batch S_b, we need to initialize the node features, i.e., the demands $\mathbf{D}$, and the edge features, i.e., the flows $\mathbf{Q}$, which are the required inputs to f_1, and the heads $\mathbf{h}$, as the required input to f_2. Thus, for $k = 0$, for the pressure heads $\mathbf{h}^{(0)} = (h_v^{(0)})_{v \in V}$, we define

$$h_v^{(0)} := \begin{cases} h_v^* & \text{if } v \in V_r \\ 0 & \text{if } v \in V \setminus V_r \end{cases}$$

for all $v \in V$. We use the true demand as the input demands $\mathbf{D}^{(0)} = (\mathbf{d_1}^{(0)}, \mathbf{d_2}^{(0)})$ and initialize the input flows $\mathbf{Q}^{(0)} = (\mathbf{q_1}^{(0)}, \mathbf{q_2}^{(0)})$ based on the water hydraulics from eq. (4) (analogously to eq. (8)):

$$d_{v1}^{(0)}, d_{v2}^{(0)} := \begin{cases} 0 & \text{if } v \in V_r \\ d_v^* & \text{if } v \in V \setminus V_r \end{cases},$$

$$q_{e1}^{(0)}, q_{e2}^{(0)} := \operatorname{sgn}(h_v^{(0)} - h_u^{(0)}) \cdot (r_e^{-1} |h_v^{(0)} - h_u^{(0)}|)^{1/x}$$

for all $v \in V$ and all $e = e_{vu} \in E$. Afterwards, we use the outcomes of f_1 and $f = f_1 \circ f_2$ of the k-th iteration in the $k+1$-th iteration: For $k = 0, ..., K-1$, we define

$$\begin{aligned} \hat{\mathbf{q}}^{(k)} &:= f_1\left(\left((\mathbf{d_1}^{(k)}, \mathbf{d_2}^{(k)}), (\mathbf{q_1}^{(k)}, \mathbf{q_2}^{(k)})\right), \Theta\right) \\ (\tilde{\mathbf{h}}^{(k)}, \tilde{\mathbf{d}}^{(k)}, \tilde{\mathbf{q}}^{(k)}) &:= f_2(\mathbf{h}^{(0)}, \hat{\mathbf{q}}^{(k)}) \\ \mathbf{d_1}^{(k+1)} &:= \mathbf{d}^{(0)} \text{ (true demand)} \\ \mathbf{d_2}^{(k+1)} &:= \hat{\mathbf{d}}^{(k)} \text{ (demand from GCN layers)} \\ \mathbf{q_1}^{(k+1)} &:= \hat{\mathbf{q}}^{(k)} \text{ (flow from GCN layers)} \\ \mathbf{q_2}^{(k+1)} &:= \tilde{\mathbf{q}}^{(k)} \text{ (flow from physics-informed algo),} \end{aligned}$$

where we compute the demands $\hat{\mathbf{d}}^{(k)}$ by flows $\hat{\mathbf{q}}^{(k)}$ analogously to eq. (8).[4] More precisely, in each iteration, we update the second node feature by using the updated demands from the output of f_1 while keeping the first node feature fixed to allow better information propagation between iterations. Additionally, we update the edge features by using the updated flows computed by f_1 and f_2.

[4]We do not re-use the demand $\tilde{\mathbf{d}}^{(k)}$ from the physics-informed algorithm as an input node feature, since including it did not improve performance.

After making use of the model $f(\cdot, \Theta)$ K times, we update its parameters by minimizing the following *loss function*[5] through back-propagation:

$$\mathcal{L} = \mathcal{L}(\mathbf{d}^*, \hat{\mathbf{d}}^{(K)}) + \rho \mathcal{L}(\mathbf{d}^*, \tilde{\mathbf{d}}^{(K)}) + \delta \mathcal{L}(\hat{\mathbf{q}}^{(K)}, \tilde{\mathbf{q}}^{(K)}),$$

where ρ and δ are hyperparameters and

- $\mathcal{L}(\mathbf{d}^*, \hat{\mathbf{d}}^{(K)})$ regresses the demands $\hat{\mathbf{d}}^{(K)}$ computed by the GCN layers f_1 against the true demands $\mathbf{d}^*$,
- $\mathcal{L}(\mathbf{d}^*, \tilde{\mathbf{d}}^{(K)})$ regresses the demands $\tilde{\mathbf{d}}^{(K)}$ computed by the whole model $f = f_2 \circ f_1$ against the true demands $\mathbf{d}^*$,
- $\mathcal{L}(\hat{\mathbf{q}}^{(K)}, \tilde{\mathbf{q}}^{(K)})$ regresses the flows $\hat{\mathbf{q}}^{(K)}$ computed by the GCN layers f_1 against the flows $\tilde{\mathbf{q}}^{(K)}$ computed by the whole model $f = f_2 \circ f_1$ (cf. theorem 0.3),
- and $\mathcal{L}$ denotes the L1 loss, i.e., the mean absolute error, over all nodes V and all samples S_b in a batch.

It is important that ρ and δ are considerably smaller than one, because the first term allows the GCN layers to estimate an initial set of flows and the remaining terms guide the model towards a valid solution. Our methodology allows the GCN component f_1 to learn locally, which is augmented by the global physics-informed algorithm f_2. Multiple iterations of this methodology allows the local GCN to learn the global task and solve it accurately.

Experiments

We evaluate our methodology on a number of real-world WDS. To the best of our knowledge, there is no comparable ML approach for the task of state, i.e., head and flow, estimation. Hence, we compare our results with the (computationally demanding) "ground truth" of the hydraulic simulator EPANET (Rossman et al. 2020). EPANET can run two types of simulations, demand driven (DD) and pressure dependent demand (PDD). In DD simulation, demands of all consumers are ensured, while in PDD, all demands may not be met depending on the drop in pressure. Our current methodology emulates DD simulation, although it can be modified to emulate PDD simulation.

Datasets

A WDS can consist of different types of junctions (reservoirs, consumers, tanks) and links (pipes, valves, pumps). A simple WDS normally has a single reservoir, many consumers and pipes. Adding a second reservoir increases the complexity of the problem. If there are valves, the WDS needs to be broken down into parts to estimate the states (heads, flows). Tanks and pumps add a temporal dimension to the dataset. For our experiments, we use datasets based on WDS with a single or multiple reservoirs and where all links are pipes.

Hanoi is one of the most popular WDS being used in the domain of WDS. Different scenarios are available with varying demand patterns, diameters, length and roughness coefficients (Vrachimis et al. 2018). L-TOWN is a well known

[5]As true demands at the reservoirs are unknown, with a slight abuse of notation, here the demand vectors exclude all $v \in V_r$.

MRAE (%)	Hanoi	Fossolo	Pescara	L-Town Area-C	Zhi Jiang
All samples					
EPANET	0.001 ± 0.0005	0.115 ± 0.022	0.157 ± 0.080	0.362 ± 0.349	0.013 ± 0.002
GCN Model	0.179 ± 0.200	0.324 ± 0.050	2.135 ± 1.614	1.004 ± 0.622	0.415 ± 0.067
Excluding 5% outliers					
EPANET	0.001 ± 0.0004	0.113 ± 0.019	0.149 ± 0.075	0.310 ± 0.272	0.013 ± 0.002
GCN Model	0.147 ± 0.058	0.316 ± 0.034	1.907 ± 1.123	0.891 ± 0.321	0.404 ± 0.048

Table 1: MRAE on estimated vs true demands across nodes and 20,160 samples.

MRAE (%)	Hanoi	Fossolo	Pescara	L-Town Area-C	Zhi Jiang
All samples					
Flows	0.295 ± 1.936	3.972 ± 1.586	4.663 ± 8.400	0.336 ± 0.335	0.859 ± 0.444
Heads	0.003 ± 0.003	0.002 ± 0.001	0.010 ± 0.004	0.005 ± 0.004	0.015 ± 0.013
Excluding 5% outliers					
Flows	0.144 ± 0.150	3.757 ± 1.290	3.240 ± 2.080	0.278 ± 0.115	0.794 ± 0.345
Heads	0.002 ± 0.001	0.002 ± 0.001	0.009 ± 0.003	0.004 ± 0.003	0.013 ± 0.009

Table 2: MRAE for GCN Model vs EPANET on flows and heads across nodes and 20,160 samples.

Time in seconds	Hanoi	Fossolo	Pescara	L-Town Area-C	Zhi Jiang
EPANET	22.87	719.86	1318.86	1380.40	274.80
GCN Model	5.98	7.24	10.44	11.12	15.58

Table 3: Evaluation times (seconds) to simulate/emulate 20,160 samples

large WDS with one available demand pattern (Vrachimis et al. 2020). We use only Area-C of L-TOWN, where we treat it as a separate WDS by attaching a reservoir of head $200m$ and changing the demand multiplier to 5. Moreover, Fossolo, Pescara and Zhi Jiang are real world international WDS datasets (Dandy 2016a; Hall 2021; Dandy 2016b) with no demand patterns available.

We are particularly interested in emulations which can address different WDS configurations (regarding link attributes such as diameter) and varying demands within a single model. To eliminate the need for re-training we thus train for multiple scenarios for each WDS. For Hanoi, such scenarios are already available. For L-Town Area-C, there is one demand pattern available. For the other three WDS, we generate demands by sampling from a normal distribution $\mathcal{N}(1,\, 0.1)$. Except for Hanoi, for each scenario, we add variation to the demands by adding noise sampled from $\mathcal{N}(0,\, 0.1)$. Moreover, we vary the diameters by adding noise sampled from $\mathcal{N}(0,\, 1/30)$. Additionally, we use different seeds for each scenario. Note that varying the diameter gives enough variation for the model to generalize to varying values of r_e, as for $e \in E$, instead of the magnitudes l_e, d_e, and c_e, the model gets r_e as input (cf. eq. (5)).

Training Setup

We implement all models in Pytorch and train them using the ADAM optimizer. We do not use bias in any of the layers. Hyperparameters are identical for all WDS (see ArXiv version). Since our model uses a physics-informed algorithm, the error does not increase with further iterations after convergence. Therefore, when choosing the number of GCN layers I and iterations K, we only have to ensure that $I \cdot K$ is sufficiently larger than the diameter of the WDS. We use 20 scenarios for training with two days of data each. The sampling rate is every 30 minutes (48 samples per day) and 60:20:20 train:validation:test splits are used. We do not normalize the data in order to preserve the hydraulic relationships between demands, flows and heads. During training, we vary the number of iterations K randomly within a range for every epoch to prevent over-fitting. We use learning rate scheduler and gradient clipping for smoother training. EPANET simulations are carried out using the WNTR python library (Klise, Murray, and Haxton 2018).

Results and Analysis

We evaluate all models on entirely unseen scenarios. Each of these 30 scenarios consist of 14 days of data (20,160 samples). The evaluations are done on NVIDIA GeForce RTX 4090. We compare the estimated values of demands against the true demands by computing the mean relative absolute error [6], given by

$$\text{MRAE}(S) = \frac{1}{N_s \cdot N_s} \sum_{\iota=1}^{N_s} \sum_{v \in V} \frac{|d^*_{\iota v} - \hat{d}_{\iota v}|}{|d^*_{\iota v}|}$$

over all samples $S = \{(\mathbf{d}^*_\iota, \hat{\mathbf{d}}_\iota) \mid \iota = 1, ..., N_s\}$ (cf. table 1). Since there are no true values available for heads and flows, we compare our results with those from the EPANET simulator using an analogous formula (cf. table 2). We also compute the conformity C of the results to eq. (4) by

$$C := \sum_{\iota=1}^{N_s} \sum_{v \in V; u \in \mathcal{N}(v)} \text{sgn}(\tilde{h}_{\iota v} - \tilde{h}_{\iota u}) - \text{sgn}(\tilde{q}_{e_{\iota v u}}),$$

[6] We do not use the conventional mean absolute error (MAE) since our data is not normalized, i.e. heads generally have big values $\gg 0$, while flows and demands are much smaller (in decimals). Hence, MAE for flows and demands would be very low but unjust.

leading to zero error for all experiments (hence not reported in tables). We achieve very good accuracy on head estimation and demand reconstruction. Most importantly, we report evaluation times, which are reduced by orders of magnitude for larger WDS as compared to EPANET (cf. table 3). Moreover, these do not increase drastically with the the number of samples (cf. fig. 2). We added noise sampled from $\mathcal{N}(0,\ 0.1)$ to the demands to create different scenarios. Great evaluation results on 30 unseen scenarios show that the model can generalize to approximately 30 percent change in demands. We also varied diameters by adding noise sampled from $\mathcal{N}(0, 1/30)$, meaning that the model can also generalize to up to 10 percent change in diameters. These results clearly support our model as a viable alternative to the hydraulic simulator for WDS planning and expansion.

Limitations

We empirically evaluated the model by adding noise to the diameters up to standard deviation $\sigma = 0.1$ and observed that the model has less than 5% error up to $\sigma = 0.08$, well beyond $\sigma = 0.033$ used for training (cf. fig. 3). We also investigated the comparatively higher standard deviation of errors in case of flows. We discovered that in some outlier samples, our model estimates some small flows erroneously. Such a sample will exhibit very high error for some flows but that will not affect the demand and head estimation by the same magnitude. Hence, we include a set of results in both tables 1 and 2 excluding 5 percent outliers showing significantly lower standard deviations.

Social Impact

Sustainable urban development is not possible without efficient and timely WDS planning and expansion. Scientists at water institutes face the challenge of quick decision making on a daily basis as well as long-term planning of WDS refurbishing and extension in the light of deep uncertainties. Since hardly any new city starts from scratch, most of those decisions are about modifications to or expansion of an existing WDS. Given the structure and parameters (reservoirs, demands, pipe lengths, diameters, roughness etc.) of a WDS, even changing or adding a few new pipes can require a multitude of simulations, which is extremely costly using current hydraulic simulation. Therefore, a faster DL alternative is needed. We expect that our proposed model can be directly applied to such tasks and thus can contribute to sustainable development of critical infrastructure in cities.

Conclusion and Future Work

We present a physics-informed DL solution to the task of state estimation in WDS. The task is of utmost importance for planning and expansion of WDS. The hydraulic solver EPANET is the current go to solution for this task for researchers, scientists and engineers in the field of WDS. However, the solver suffers from long computation times since it scales non-linearly with the size of WDS. Moreover, even the slightest changes in the configuration of WDS requires a complete re-run. We utilize GCN layers and a unique physics-informed algorithm to build a DL alternative that is vastly faster than the hydraulic simulator. To the best of our knowledge, this is the first DL approach that does not use any additional information to solve the task. We use a limited number of GCN layers iteratively thus keeping the model size small. Unlike conventional ML tasks, we infer two features (heads, flows) from a given feature (demand) and achieve great accuracy using hydraulic principles.

In the future, we plan to extend the proposed methodology to adapt to more complex WDS. For instance, a WDS with valves adds a break to the continuity of the hydraulic principles. A pump adds a Markov property to the flows and will need further adjustments to our model. The current DD simulations can be modified to run PDD simulations by adding certain constraints. On a graph level, the impact of changes to WDS structure on the performance of our methodology needs to be explored. Finally, a model able to generalize across different WDS will be the ideal solution.

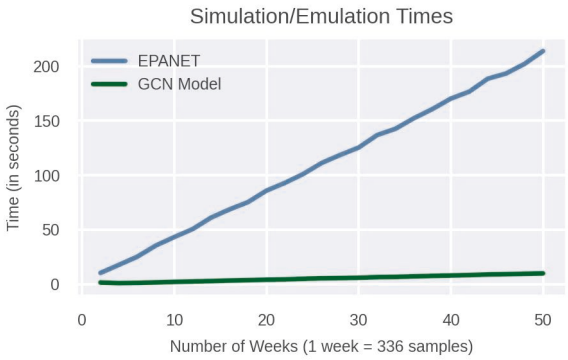

Figure 2: Comparison of simulation/emulation times on L-TOWN Area-C for EPANET and GCN Model.

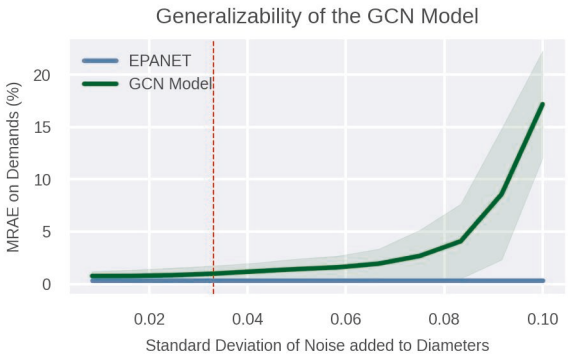

Figure 3: MRAE on Demands (L-TOWN Area-C) with increasing σ of noise added to diameters. Red line indicates the maximum σ used for training.

Acknowledgements

We gratefully acknowledge funding from the European Research Council (ERC) under the ERC Synergy Grant Water-Futures (Grant agreement No. 951424) and the research training group "Dataninja" (Trustworthy AI for Seamless Problem Solving: Next Generation Intelligence Joins Robust Data Analysis) funded by the German federal state of North Rhine-Westphalia.

References

Ashraf, I.; Hermes, L.; Artelt, A.; and Hammer, B. 2023. Spatial Graph Convolution Neural Networks for Water Distribution Systems. In Crémilleux, B.; Hess, S.; and Nijssen, S., eds., *Advances in Intelligent Data Analysis XXI*, 29–41. Cham: Springer Nature Switzerland. ISBN 978-3-031-30047-9.

Bruna, J.; Zaremba, W.; Szlam, A. D.; and LeCun, Y. 2014. Spectral Networks and Locally Connected Networks on Graphs. *CoRR*.

Dandy, G. 2016a. "03 Fossolo" International Systems. 3. https://uknowledge.uky.edu/wdst_international/3. Accessed: 2024-01-25.

Dandy, G. 2016b. "06 Zhi Jiang" International Systems. 6. https://uknowledge.uky.edu/wdst_international/6. Accessed: 2024-01-25.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *NIPS*, 29: 3844–3852.

Dick, K.; Russell, L.; Dosso, Y. S.; Kwamena, F.; and Green, J. R. 2019. Deep Learning for Critical Infrastructure Resilience. *JIS*, 25(2): 05019003.

Eichenberger, C.; and et al. 2022. Traffic4cast at NeurIPS 2021 - Temporal and Spatial Few-Shot Transfer Learning in Gridded Geo-Spatial Processes. In *Proceedings of the NeurIPS 2021 Competitions and Demonstrations Track*, volume 176, 97–112. PMLR.

Fan, X.; Zhang, X.; and Yu, X. . B. 2021. Machine learning model and strategy for fast and accurate detection of leaks in water supply network. *Journal of Infrastructure Preservation and Resilience*, 2(1): 10.

Gao, H.; Wang, Z.; and Ji, S. 2018. Large-scale learnable graph convolutional networks. In *SIGKDD*, 1416–1424.

Hajgató, G.; Gyires-Tóth, B.; and Paál, G. 2021. Reconstructing nodal pressures in water distribution systems with graph neural networks. *arXiv preprint arXiv:2104.13619*.

Hall, A. 2021. "04 Pescara" International Systems. 3. https://uknowledge.uky.edu/wdst_international/4. Accessed: 2024-01-25.

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *NIPS*, 1025–1035.

Hammer, B. 2000. *Learning with recurrent neural networks*, volume 254 of *Leacture Notes in Control and Information Sciences*. Springer.

Henaff, M.; Bruna, J.; and LeCun, Y. 2015. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*.

Kammoun, M.; Kammoun, A.; and Abid, M. 2022. Leak Detection Methods in Water Distribution Networks: A Comparative Survey on Artificial Intelligence Applications. *Journal of Pipeline Systems Engineering and Practice*, 13(3): 04022024.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

Klambauer, G.; Unterthiner, T.; Mayr, A.; and Hochreiter, S. 2017. Self-Normalizing Neural Networks. In *NIPS*, NIPS'17, 972–981. ISBN 9781510860964.

Klise, K.; Murray, R.; and Haxton, T. 2018. An Overview of the Water Network Tool for Resilience (WNTR):(075). In *WDSA/CCWI Joint Conference Proceedings*, volume 1.

Levie, R.; Monti, F.; Bresson, X.; and Bronstein, M. M. 2018. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing*, 67(1): 97–109.

Li, G.; Xiong, C.; Thabet, A.; and Ghanem, B. 2020. Deepergcn: All you need to train deeper gcns. *arXiv preprint arXiv:2006.07739*.

Li, R.; Wang, S.; Zhu, F.; and Huang, J. 2018. Adaptive Graph Convolutional Neural Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32. Association for the Advancement of Artificial Intelligence (AAAI).

Magini, R.; Moretti, M.; Boniforti, M. A.; and Guercio, R. 2023. A Machine-Learning Approach for Monitoring Water Distribution Networks (WDNs). *Sustainability*, 15(4).

Monti, F.; Boscaini, D.; Masci, J.; Rodola, E.; Svoboda, J.; and Bronstein, M. M. 2017. Geometric deep learning on graphs and manifolds using mixture model cnns. In *IEEE conference on computer vision and pattern recognition*, 5115–5124.

Niepert, M.; Ahmed, M.; and Kutzkov, K. 2016. Learning convolutional neural networks for graphs. In *ICML*, 2014–2023. PMLR.

Omitaomu, O. A.; and Niu, H. 2021. Artificial Intelligence Techniques in Smart Grid: A Survey. *Smart Cities*, 4(2): 548–568.

Raissi, M.; Perdikaris, P.; and Karniadakis, G. 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378: 686–707.

Ritchie, H.; and Roser, M. 2018. Urbanization. *Our World in Data*. Https://ourworldindata.org/urbanization.

Rossman, L.; Woo, H.; Tryby, M.; Shang, F.; Janke, R.; and Haxton, T. 2020. EPANET 2.2 User's Manual, Water Infrastructure Division. *CESER*.

Ruff, E.; Russell, R.; Stoeckle, M.; Miotto, P.; and How, J. P. 2023. Surrogate Neural Networks for Efficient Simulation-based Trajectory Planning Optimization. arXiv:2303.17468.

Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G. 2009. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1): 61–80.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. *ICLR*. Accepted as poster.

Vrachimis, S.; Kyriakou, M.; Eliades, D.; and Polycarpou, M. 2018. LeakDB: A benchmark dataset for leakage diagnosis in water distribution networks description of benchmark. In *Vol 1 of Proc., WDSA/CCWI Joint Conf. Kingston, ON, Canada: Queen's Univ*.

Vrachimis, S. G.; Eliades, D. G.; Taormina, R.; Ostfeld, A.; Kapelan, Z.; Liu, S.; Kyriakou, M.; Pavlou, P.; Qiu, M.; and Polycarpou, M. M. 2020. BattLeDIM: Battle of the leakage detection and isolation methods. In *CCWI/WDSA Joint Conf*.

Wu, Y.; Wang, X.; Liu, S.; Yu, X.; and Wu, X. 2023. A weighting strategy to improve water demand forecasting performance based on spatial correlation between multiple sensors. *Sustainable Cities and Society*, 93: 104545.

Xing, L.; and 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).

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations*.