Partial Differential Equation Driven Dynamic Graph Networks for Predicting Stream Water Temperature

Tianshu Bao[1], Xiaowei Jia[2], Jacob Zwart[3], Jeffrey Sadler[3], Alison Appling[3], Samantha Oliver[3], and Taylor T. Johnson[1]
[1]Vanderbilt University, [2]University of Pittsburgh, [3]U.S. Geological Survey
[1]{tianshu.bao, taylor.johnson}@vanderbilt.edu, [2]xiaowei@pitt.edu, [3]{jzwart, jsadler, aappling, soliver}@usgs.gov

***Abstract*—This paper presents a physics-guided machine learning approach that incorporates partial differential equations (PDEs) in a graph neural network model to improve the prediction of water temperature in river networks. The standard graph neural network model often uses pre-defined edge weights based on distance or similarity measures. Such static graph structure can be limited in capturing multiple processes in a physical system that interact and evolve over time. The limitation to represent underlying physical processes can severely affect the performance of the predictive model, especially when we have access to limited training data. To better capture the dynamic interactions among multiple segments in a river network, we built a dynamic graph model, where the graph structure is driven by the PDE that describes underlying physical processes. We further combine the dynamic graph structure and the recurrent layers to model temporal dependencies and improve the prediction. We demonstrate the effectiveness of the proposed method in a subnetwork of the Delaware River Basin. In particular, we show that the proposed method outperforms existing physics-based and machine learning models in temperature prediction using sparse observation data for training. The proposed method has also been shown to produce better performance when generalized to different seasons.**

I. Introduction

Water temperature prediction in river networks is critical for monitoring aquatic ecosystems by providing important information regarding the habitat for aquatic life and aquatic biogeochemical cycling [1], [2]. Effective temperature predictions are also essential for water management decisions. For example, accurate prediction of water temperature can help water managers optimize the water release from reservoirs to maintain the flow and temperature regimes required for quality downstream habitat.

A river network can be considered as a physical system that has multiple interacting processes. In this problem, multiple river segments are connected to each other, and they can show different thermodynamic patterns driven by differences in catchment characteristics (e.g. slope, soil characteristics) and meteorological drivers (e.g. temperature, precipitation). These segments also frequently interact with each other through the water advected from upstream to downstream segments. Rivers are essentially fluid from a physical perspective, with their spatial and temporal patterns described by partial differential equations (PDEs) that govern fluid dynamics. For example, traditional fluid models have used the Navier–Stokes equation [3] for simulating fluid dynamics in many applications including aquatic science, hydraulic modeling, weather and climate modeling, ocean currents, and aerodynamics. When modeling temperature dynamics in river networks, these PDEs capture not only the temporal thermodynamics but also the spatial heat diffusion and convection from connected river segments [4]. Furthermore, these PDEs, along with other known physical relationships, have been used to build more complex physics-based models [5], [6] to simulate multiple interacting processes on different variables in a system. However, these equations and physics-based models have limits in their predictions due to approximations and parameterizations used to represent underlying processes.

Recent advances in machine learning (ML), given their great success in commercial applications, have provided unrealized potential for modeling complex data patterns in scientific problems. The power of these models come from their capacity to extract complex nonlinear patterns from observation data and naturally incorporate spatial and temporal data dependencies. For example, recurrent neural network (RNN) models, which take account of temporal dependencies, have shown extensive applicability in speech recognition and machine translation [7], [8]. Convolutional neural network (CNN)-based approaches have shown tremendous success in learning spatial patterns in many computer vision applications [9]–[11]. Recently, graph neural network models, e.g., graph convolutional networks (GCNs), have shown a great promise for modeling interactions and similarities amongst multiple objects [12]–[15] and also have shown encouraging results for studying river networks [16]–[18].

However, there are several major challenges faced by these existing ML methods when they are directly applied to scientific problems. First, the data available for many scientific problems is far smaller than what is needed for effectively training advanced ML models. In a river network, there are often only a handful of river segments in a network that are monitored due to the high cost associated with data collection. Moreover, despite the promise of existing deep learning techniques, they are not originally designed to exploit the unique characteristics of complex scientific systems. Scientific systems are commonly driven by physical processes and variables that evolve and interact at different spatial and temporal scales. While existing GCN-based methods have shown some success in modeling interactions amongst river segments, they commonly create static graphs based on standard distance metrics (e.g., geographic or stream distances) without fully

exploiting the physical characteristics of river segments and also do not capture dynamic interactions over time. In stream networks, the flow of water from one stream segment takes a certain amount of time to reach to another stream segment and this time depends on multiple factors such as the stream morphology, catchment characteristics, and weather patterns. Additionally, the connection strength between two stream segments depends on the velocity and volume of the water flowing through individual streams, which can also vary over time. Hence, the weight of different upstream segments on a downstream segment can vary across time depending on the variation of these physical variables. Additionally, existing GCN-based models extract abstract hidden variables that are propagated over the network, but these hidden variables may fail to represent true underlying physical relationships that govern the interactions, especially given limited training data.

In this paper, we propose a PDE-guided Dynamic Graph Networks (PDE-DGN) to predict water temperature for all the river segments over a long period. The PDE-DGN captures temporal dependencies with a recurrent layer while also modeling the spatial interactions amongst river segments using dynamic graph structures. Moreover, it incorporates the governing PDE that describes the heat transfer process in the river networks. The PDE represents known physical relationships about dynamic interactions amongst river segments, and thus can be used to guide the design of evolving graph structures. In particular, we use finite difference methods to derive the graph structure from the PDE. The representation of PDEs is also limited in that some physical parameters (e.g., coefficients in PDEs) are unknown and commonly require expensive calibration. In our proposed method, these unknown physical parameters can be estimated together with other neural network parameters efficiently using back propagation.

We implement our proposed method for water temperature prediction using collected data from the Delaware River Basin over 36 years. We demonstrate the superior predictive performance of our proposed method over existing ML methods. Our methods have also been shown to produce better performance when using limited training data that are sparsely distributed over space (i.e., data are only available from certain stream segments) and time (i.e., data are only available from certain seasons). Moreover, the proposed method has better generalizability when tested in data of different distributions. Our contributions can be summarized as follows:

- We introduce a new dynamic recurrent graph network architecture to model a river network with interacting river segments.
- We leverage the knowledge from the underlying PDE to guide the design of evolving graph structure.
- We evaluate the utility in the context of an ecologically and societally relevant problem of monitoring river networks.

II. Related Work

Our proposed method has multiple components, including integrating physics into ML, graph neural network architecture, and machine learning for PDE. These topics have been studied under different contexts.

Recent works have shown the promise of integrating physics into ML models in improving the predictive performance and generalizability in scientific problems. This is commonly conducted in several ways, including physics-guided model architectures [19], [20], physics-guided loss functions [21], [22], and other hybrid approaches [23], [24].

Our proposed method is related to physics-guided model architectures. There are several ways to incorporate known physics into ML models. For example, one can embed known physical principles into neural networks (NN) by ascribing physical meaning for certain neurons in the NN. Muralidlar et al. [25] built a new architecture to insert physics-constrained variables as the intermediate variables in the convolutional neural networks. This method has been tested for predicting drag force on particle suspensions in moving fluids and has achieved improved performance. Another direction is to use ML architecture to encode invariance and symmetries that are inherent of a physical system. In turbulence modeling and fluid dynamics, Ling et al. [26] defines a tensor basis neural network to embed the fundamental principle of rotational invariance into neural networks for improved prediction accuracy.

When applied to systems with interacting processes, e.g., a river network, we need to build ML models that can handle such interactions. The GCN model has proven to be effective in automatically modeling node interactions in a graph. The use of GCN has also shown improved prediction accuracy in several scientific problems [27]–[29]. In a previous work, Jia et al. [17] leveraged physics to guide the extraction of hidden variables that are propagated in the GCN model. This method has been shown to produce better prediction accuracy as well as improved generalizability. Our proposed method is different from this work in that we use underlying physics to directly create new graph structures.

PDEs have been widely used to describe a wide range of phenomena such as fluid dynamics and quantum mechanics that cannot be adequately described by ordinary differential equations (ODEs) [30]. As an example, a typical application for PDEs is the modeling of congestion in highway networks [31], and one popular model for highway control is the Lighthill-Whitham-Richards model [32]. Finite difference methods have been widely used to solve PDEs [33]. In this work, we seek to extend the graph neural network by incorporating the heat transfer PDE. This extension is non-trivial because the discretization of PDEs on a network is inherently more complicated due to the fact that the river segments are not uniformly distributed. An in-depth discussion of these complexities that belong to generalized finite difference methods can be found in [34], [35].

Note that our work is different from existing works on using ML for solving PDEs [36], [37]. These previous methods require access to a perfect PDE (i.e., all the coefficients are known, and the available simulations are always consistent to the PDE). In these studies, the process of numerically solving PDEs was accelerated by using ML as a surrogate model.

In contrast, the proposed method herein aims to improve the prediction of target variables using both observation data and the underlying PDE. Moreover, the PDE contains unknown variables that need to be estimated during the training process. In [38], the authors use neural networks to run continuous time and discrete time models which are obtained through general nonlinear PDEs. This method relies on many training iterations or a very high order Runge-Kutta spatial discretization, and it enforces the physical relationship only through the loss function.

III. Problem definition and preliminary

A. Problem definition

Our objective is to model the dynamics of temperature in a set of connected river segments that together form a river network. The connections amongst these river segments can be represented in a graph structure $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{A}\}$, where $\mathcal{V}$ represents the set of N river segments and $\mathcal{E}$ represents the set of connections amongst river segments. Specifically, we create an edge $(i, j) \in \mathcal{E}$ if the segment i is connected to segment j. Because we consider the dynamic interactions amongst river segments, the adjacency matrices $\mathbf{A} = \{\mathbf{A}^1, \mathbf{A}^2, ..., \mathbf{A}^T\}$ represent the adjacency levels amongst all the segments at each time step from $t = 1$ to T. Here a higher value of $\mathbf{A}^t_{ij}$ indicates that the segment i has a stronger influence on the segment j at time t. $\mathbf{A}^t_{ij} = 0$ means there is no edge from the segment i to the segment j at this time. In this paper, we only consider the evolution of adjacency matrix over time while keeping a static set of river segments (i.e., node set $\mathcal{V}$) and stream connections (i.e., edge set $\mathcal{E}$).

B. Recurrent Neural Networks and Long-Short Term Memory

The RNN model has been widely used to model the temporal patterns in sequential data. The RNN model defines transition relationships for the extracted hidden representation through a recurrent cell structure. In this work, we adopt the Long-Short Term Memory (LSTM) to build the recurrent layer for capturing long-term dependencies. Each LSTM cell has a cell state $\mathbf{c}^t$, which serves as a memory and allows preserving information from the past. Specifically, the LSTM first generates a candidate cell state $\bar{\mathbf{c}}^t$ by combining $\mathbf{x}^t$ and the hidden representation at previous time step $\mathbf{h}^{t-1}$, as follows:

$$\bar{\mathbf{c}}^t = \tanh(\mathbf{W}^h_c \mathbf{h}^{t-1} + \mathbf{W}^x_c \mathbf{x}^t + \mathbf{b}_c). \tag{1}$$

where $\mathbf{W}$ and $\mathbf{b}$ are matrices and vectors, respectively, of learnable model parameters. Then the LSTM generates a forget gate f^t, an input gate g^t, and an output gate o^t via sigmoid function $\sigma(\cdot)$, as follows:

$$\begin{aligned} \mathbf{f}^t &= \sigma(\mathbf{W}^h_f \mathbf{h}^{t-1} + \mathbf{W}^x_f \mathbf{x}^t + \mathbf{b}_f), \\ \mathbf{g}^t &= \sigma(\mathbf{W}^h_g \mathbf{h}^{t-1} + \mathbf{W}^x_g \mathbf{x}^t + \mathbf{b}_g), \\ \mathbf{o}^t &= \sigma(\mathbf{W}^h_o \mathbf{h}^{t-1} + \mathbf{W}^x_o \mathbf{x}^t + \mathbf{b}_o). \end{aligned} \tag{2}$$

The forget gate is used to filter the information inherited from $\mathbf{c}^{t-1}$, and the input gate is used to filter the candidate cell state at t. Then we compute the new cell state as follows:

$$\mathbf{c}^t = \mathbf{f}^t \otimes \mathbf{c}^{t-1} + \mathbf{g}^t \otimes \bar{\mathbf{c}}^t, \tag{3}$$

where $\otimes$ denotes the entry-wise product.

Once obtaining the cell state, we can compute the hidden representation by filtering the cell state using the output gate, as follows:

$$\mathbf{h}^t = \mathbf{o}^t \otimes \tanh(\mathbf{c}^t). \tag{4}$$

According to the above equations, we can observe that the computation of $\mathbf{h}^t$ combines the information at current time step ($\mathbf{x}^t$) and previous time step ($\mathbf{h}^{t-1}$ and $\mathbf{c}^{t-1}$), and thus encodes the temporal patterns learned from data.

IV. Method

In this section, we describe the details of the PDE-DGN method. We start with introducing the dynamic graph model architecture for capturing stream water dynamics and interactions amongst river segments. Then we discuss a new strategy to enforce physical relationships to the dynamic graph structure by leveraging the physical knowledge embedded in known governing PDEs.

A. Dynamic Recurrent Graph Network

Water temperature in rivers has strong spatial and temporal patterns as a result of heat transfer with climate (e.g., solar radiation) and neighboring river segments [4]. The ML model for river networks also needs to capture such spatial and temporal dependencies in order to model the temperature dynamics. In particular, we build a dynamic recurrent graph network, which incorporate the information from both previous time steps and neighbors (i.e., upstream segments) when modeling each river segment (Fig. 1).

The proposed model aims to embed the input data and the spatio-temporal context of each river segments into a hidden representation $\mathbf{h}^t$ at each time step. Our model structure is inspired by the Recurrent Graph Model [17] but we extend it to take account of changes in the graph structure. We now describe the recurrent process of generating the hidden representation $\mathbf{h}^t$ from $\mathbf{h}^{t-1}$.

First, for each river segment i, the model needs to aggregate the influence from its neighboring segments j such that $(i, j) \in \mathcal{E}$. Specifically, assuming we have gathered the embeddings $\mathbf{h}^{t-1}_j$ from all the upstream segments at previous time step, we first transform these embeddings through a function $f(\cdot)$, which extracts the information that is most relevant to the downstream segment i. For example, the amount of water advected from this segment and its water temperature can directly affect the change of water temperature for its downstream segments. This function can be implemented using fully connected layers. Then we develop a new recurrent cell structure for each segment i by extending the standard LSTM structure (Eq. 3) that integrates both the historical information from the same river segment (i.e., the previous state $\mathbf{c}^{t-1}_i$) and the spatial context from its upstreams (i.e., $f(\mathbf{h}^{t-1}_j)$ for $(j, i) \in \mathcal{E}$). This can be expressed as follows:

$$\mathbf{c}^t_i = \mathbf{f}^t_i \otimes (\mathbf{c}^{t-1}_i + \sum_{(i,j)\in\mathcal{E}} \mathbf{A}^t_{ij} f(\mathbf{h}^{t-1}_j)) + \mathbf{g}^t_i \otimes \bar{\mathbf{c}}^t_i \tag{5}$$

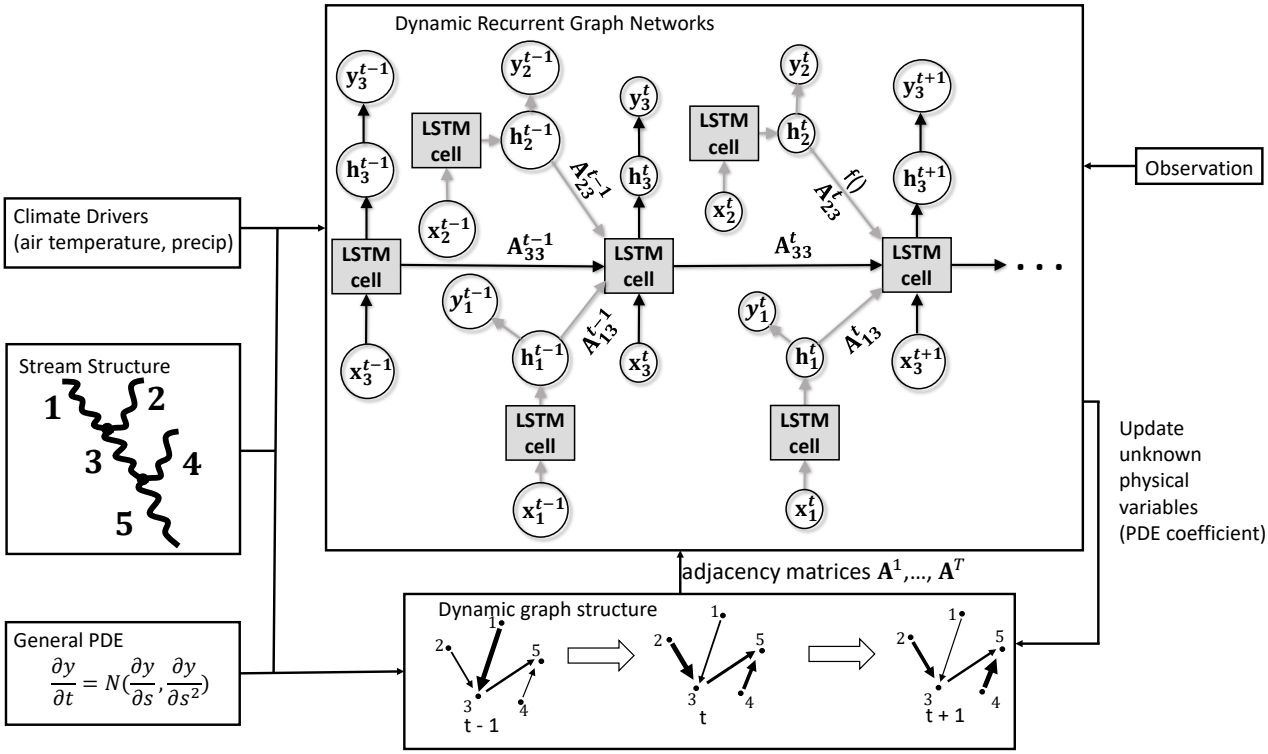


Fig. 1. The overall flow of the proposed method. The dynamic recurrent graphs take the input of climate drivers and dynamic graph structures derived from the stream network and the underlying PDE. The thickness of edges in dynamic graph structures represents different edge weights. The training of the model updates both network parameters and coefficients in the PDE to refine graph structures.

The forget gate not only filters the previous information from the segment i itself but also from its neighbors (i.e., upstream segments). The information from each upstream segment j is weighted by the adjacency level $\mathbf{A}_{ij}^t$ between i and j at time t. The matrix $\mathbf{A}^t$ varies over time due to the change of influence amongst river segments, which is described in Section IV-B. When a river segment has no upstream segments (i.e., headwater stream segment), the computation of $\mathbf{c}_i^t$ is the same as with the standard LSTM. Also, we use the $f(\mathbf{h}_j^{t-1})$ from the previous time step because of the time delay in transferring the influence from upstream to downstream segments.

After obtaining the cell state, we can compute the hidden representation $\mathbf{h}_i^t$ by following Eq. 4. Finally, we generate the predicted output from the hidden representation as follows:

$$\hat{\mathbf{y}}_i^t = \mathbf{W}_y \mathbf{h}_i^t + \mathbf{b}_y, \tag{6}$$

where $\mathbf{W}_y$ and $\mathbf{b}_y$ are model parameters.

After applying this recurrent process to all the time steps, we define a loss, $\mathcal{L}_{\text{DGN}}$, using true observations $\mathbf{Y} = \{\mathbf{y}_i^t\}$ that are available at certain time steps and certain segments, as follows:

$$\mathcal{L}_{\text{DGN}} = \frac{1}{|\mathbf{Y}|} \sum_{\{(i,t)|\mathbf{y}_i^t \in \mathbf{Y}\}} (\mathbf{y}_i^t - \hat{\mathbf{y}}_i^t)^2. \tag{7}$$

B. PDE-Driven Dynamic Graph Structure

Water temperature changes in rivers as a result of the heat transfer process. The heat transfer process has been widely studied and also used to build process-based models to simulate water temperature change along stream networks and through time [4]. Here we introduce a new strategy that leverages such underlying physical process to estimate the graph structure used in our proposed method. By enforcing such general physical relationships, the model stands a better chance at learning generalizable patterns even with small amounts of training data.

In particular, we consider the temperature change resulted from the net gain of energy fluxes by following the heat transfer formula described in [4]. This formula is expressed as follows:

$$\frac{\partial y}{\partial t} = -U\frac{\partial y}{\partial s} + D\frac{\partial^2 y}{\partial s^2} + \frac{H_{total}}{\rho \cdot c_p \cdot d}, \tag{8}$$

where $y = y(s,t)$ is the water temperature (°C) at time t and location s, U is mean channel velocity (m s^{-1}), D is a longitudinal dispersion coefficient (m s^{-2}), ρ is the density of water (1000 kg m^{-3}), c_p is the specific heat of water (41.8×10^3 J kg-1 °C^{-1}), and d is the mean channel depth (m). Here $\frac{\partial y}{\partial t}$ denotes the water temperature change over time (s) while $\frac{\partial y}{\partial s}$ denotes the water temperature change over stream distance (m). H_{total} represents the total energy available for

transfer to or from the river channel. Eq. 8 describes the dynamics of river temperature from a temporal perspective; its rearrangement in the form $\frac{\partial y}{\partial s}$ also permits the calculation of river temperature in a spatial framework [39]. When using Eq. 8, we assume that the channel is well mixed and does not contain notable lateral temperature gradients. In order to derive the spatial relationships from Eq. 8, we use the finite difference method. Finite difference methods (FDMs) are a group of approaches used for approximating real values or variable derivatives of a function on predefined mesh points by solving algebraic equations containing finite differences and values from nearby points. The error between the discrete solutions and the exact solutions is measured through Taylor expansions.

Our objective is to construct matrix $\mathbf{A}^t$ from the PDE (Eq. 8). Real-world river systems are complex. Multiple river segments in a network may interact with each other and these interactions are non-uniform in space and time. Moreover, river segments are usually non-uniformly distributed in space which makes the equal distance FDMs fail. Hence, there is a need to develop a new numerical scheme to handle irregular distributed points. In the following, we will discuss how we use the PDE to contruct the graph structure in two parts: (1) How to numerically approximate the solution of the PDE using irregularly distributed points. (2) How to use use the PDE under special conditions (e.g., intersections and boundaries).

1) PDE over irregular points: A feasible way towards the first problem is to use a step variational FDM which is often referred to as generalized finite difference methods (GFDMs). GFDMs are meshless methods and have been used in a wide variety of applications [34]. For the ease of presenting PDEs and GFDMs, we explicitly represent the set of time steps as $\{t_1, t_2, ..., t_T\}$, where the time interval between consecutive time steps is Δt. We also assume the spatial locations of N segments as $\{s_1, s_2, ...s_N\}$. During our presentation, we consider s_{i-1} and s_{i+1} to be the closest upstream and downstream segments to s_i, respectively. Using GFDM, the first order temporal derivative can be approximated as follows:

$$y_t(s_i, t_n) = \frac{\partial y(s_i, t_n)}{\partial t} \approx \frac{y_i(s_i, t_{n+1}) - y_i(s_i, t_n)}{\Delta t}. \quad (9)$$

Consider one river segment with the left distance Δs_1 to the left observation point and right distance Δs_2 to the right observation point (see Fig. 2). The first order spatial derivative are calculated as follows:

$$y_s(s_{i-1/2}, t_n) = \frac{\partial y(s_{i-1/2}, t_n)}{\partial s} \approx \frac{y(s_i, t_n) - y(s_{i-1}, t_n)}{\Delta s_1}, \quad (10)$$

$$y_s(s_{i+1/2}, t_n) = \frac{\partial y(s_{i+1/2}, t_n)}{\partial s} \approx \frac{y(s_{i+1}, t_n) - y(s_i, t_n)}{\Delta s_2}, \quad (11)$$

Here, $s_{i+1/2}$ represents the middle points of s_i and s_{i+1}. $s_{i-1/2}$ represents the middle points of s_i and s_{i-1}. We use these two approximated first order spatial derivatives to approximate a second order spatial derivative which can be estimated as follows:

$$y_{ss}(s_i, t_n) = \frac{\partial y_s(s_i, t_n)}{\partial s} \approx \frac{y_s(s_{i+1/2}, t_n) - y_s(s_{i-1/2}, t_n)}{(\Delta s_1 + \Delta s_2)/2}. \quad (12)$$

The truncation error for Eqs. 9, 10, 11 and 12 are $O(\Delta t)$, $O(\Delta s_1)$, $O(\Delta s_2)$ and $O(((\Delta s_1 + \Delta s_2)/2)^2)$, respectively.

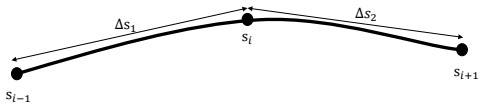


Fig. 2. The river segment diagram for estimating first-order and second-order spatial derivatives (Eqs. 10, 11 and 12).

Now we can use Eqs. 9-12 to approximate the derivatives in Eq. 8. By substituting the approximated derivatives into Eq. 8, we can rewrite the original PDE as follows:

$$\begin{aligned}&\frac{y(s_i, t_{n+1}) - y(s_i, t_n)}{\Delta t}\\ =&-U\frac{y(s_{i+1}, t_n) - y(s_i, t_n)}{\Delta s_2} + D\frac{y_s(s_{i+1/2}, t_n) - y_s(s_{i-1/2}, t_n)}{(\Delta s_1 + \Delta s_2)/2}\\ &+ \frac{H_{total}}{\rho \cdot c_p \cdot d}.\end{aligned} \quad (13)$$

Here we use Eq. 11 to approximate $y_s(s_i, t_n)$. Then by rearranging the terms in Eq. 13, we get the following representation for $y(s_i, t_{n+1})$, as follows:

$$\begin{aligned}&y(s_i, t_{n+1})\\ =&(\frac{2\Delta t D}{\Delta s_1(\Delta s_1 + \Delta s_2)})y(s_{i-1}, t_n)\\ &+ (\frac{U\Delta t}{\Delta s_2} - \frac{2\Delta t D}{\Delta s_1(\Delta s_1 + \Delta s_2)} - \frac{2\Delta t D}{\Delta s_2(\Delta s_1 + \Delta s_2)})y(s_i, t_n)\\ &+ (\frac{2\Delta t D}{\Delta s_2(\Delta s_1 + \Delta s_2)} - \frac{U\Delta t}{\Delta s_2})y(s_{i+1}, t_n) + \frac{H_{total}}{\rho \cdot c_p \cdot d}\Delta t.\end{aligned} \quad (14)$$

Because we have multiple rivers in a river graph, we use a vector $\boldsymbol{Y}(t_n) = \{y(s_1, t_n), y(s_2, t_n), ..., y(s_N, t_n)\}$ to represent the temperatures on all the river segments at a specific time step t_n. Note that in Eq. 13, $i-1$ and $i+1$ represent two segments that are connected to the segment i, i.e., $(i-1, i) \cup (i, i+1) \subset \mathcal{E}$, and they are the closest upstream and downstream segments to the segment i, respectively. Then we can derive the update formula at each time step by converting Eq. 14 into the following form:

$$\boldsymbol{Y}(t_{n+1}) = \mathbf{A}^{t_{n+1}}\boldsymbol{Y}(t_n) + \text{Constant}, \quad (15)$$

where each row of $\mathbf{A}^{t_{n+1}}$ can be determined by Eq. 14. We ignore the constant term when we build the graph structure.

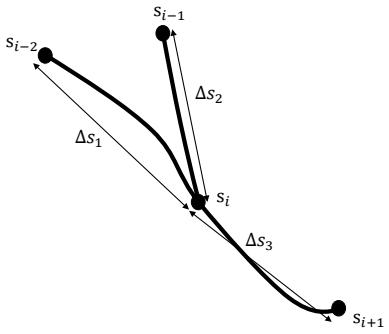


Fig. 3. River intersection diagram. The center point has two upstream points and one downstream point. In this scenario, we consider $Dy_{ss} = D_1 y_{ss,1} + D_2 y_{ss,2}$.

For other variables, the channel velocity U is calculated as the quotient of streamflow$_i^t/ca_i$, where ca_i represents the cross-sectional area of each stream segment i, and we use the streamflow values that are simulated by a physics-based model PRMS [40]. Because we do not have the measured value of ca_i and D, we estimate these values from observations of water temperature.

2) Dealing with special conditions: Now we discuss the estimation of spatial derivatives under two special conditions; intersections and boundaries. First, we consider the intersections in river networks where the target river segment can have multiple upstream segments and one downstream segment (Note that in real-world river networks, it is very rare for naturally flowing river segments to have more than one downstream segment). We show one such example in Fig. 3. The calculation of the first-order derivative y_s requires data points from both the target segment and the neighboring segments (Eqs. 10 and 11). Additional complexity arises if we want to estimate the first-order gradients using multiple upstream segments. Hence, we use the data point from the single downstream segment to estimate the first-order derivative by following Eq. 11.

The estimation of second-order derivative y_{ss} (i.e., heat diffusion effect) requires data points from both sides of the target segment. To adapt Eq. 12 to handle multiple upstream segments, a straightforward method is to only focus on the closest upstream segment and neglect all the other points, termed as 2-points approximation. Although this method results in a simplified solution, it may degrade the performance due to the ignorance of the influence from other upstream segments. This issue can be further exacerbated if no observations of water temperature are available for the closest upstream.

Another method is to aggregate the diffusion effect from all the upstream segments. Because more upstream segments lead to more heat exchange, we sum the second-order derivative y_{ss} over all the upstream segments.

In particular, in the heat transfer PDE (Eq. 8), we convert $D\,y_{ss}$ into $\sum_{i=1}^{k} D_i\, y_{ss,i}$ where k is the total number of upstream segments at the current river segment, and $y_{ss,i}$ represents the second-order derivative estimated using the upstream segment i and the downstream segment following Eq. 12. As a result, each row of $\mathbf{A}^t$ contains k values from upstream segments, one value from the downstream segment and one value from itself.

Second, we consider segments that are located at the boundary (i.e., when they have no upstream or downstream segments). It is challenging for estimating spatial derivatives (especially second-order derivatives) for these segments due to the missing neighbors on one side. To this end, we assume the values outside the boundary are always identical to the temperatures on the boundary, i.e., we have the Neumann boundary condition $y_s(s_b, t) = 0$ where s_b denotes a boundary point. Here we show the computation of the second-order derivative for headwater segments (i.e., segments with no upstream segments) as follows:

$$y_{ss}(s_b, t_n) \approx \frac{y_s(s_{b+1/2}, t_n)}{(\Delta s_1 + \Delta s_2)/2}, \tag{16}$$

where Δs_1 is a virtual distance measure and is pre-defined in our implementation.

Algorithm 1 Calculation of dynamic graph structures at a specific time step t.

for i = 1 : number of river segments **do**
  U = streamflow$_i^t/ca_i$
  for k = 1 : number of upstreams of i **do**
    $coef_{up,k} = \frac{2\Delta t crossarea_i}{\Delta s_{1,k}(\Delta s_{1,k}+\Delta s_2)}$
    $coef_{i,k} = \frac{U\Delta t}{\Delta s_2} - \frac{2\Delta t D_L}{\Delta s_{1,k}(\Delta s_{1,k}+\Delta s_2)} - \frac{2\Delta t D_L}{\Delta s_2(\Delta s_{1,k}+\Delta s_2)}$
    $coef_{dn,k} = \frac{2\Delta t D_L}{\Delta s_2(\Delta s_{i,k}+\Delta s_2)} - \frac{U\Delta t}{\Delta s_2}$
  end for
  $coef_i = \sum_k coef_{i,k}$
  $coef_{dn} = \sum_k coef_{dn,k}$
  for k = 1 : number of upstreams of i **do**
    $A^t_{i,up(k)} = coef_{up,k}$, where $up(k)$ is the index of k^{th} upstream segment
  end for
  $A^t_{i,i} = coef_i$
  $A^t_{i,dn} = coef_{dn}$
end for

We show the detailed process of computing the dynamic graph structures in Algorithm 1 and then summarize our proposed method in Algorithm 2. In each iteration, we first estimate the graph structure using the PDE and the current value of ca_i and $\{D_i\}$. Then we update ca_i and $\{D_i\}$ as well as other model parameters using available training observations.

V. EXPERIMENTAL RESULTS

We evaluate the proposed method for predicting stream temperature using real-world data collected from the Delaware River Basin, which is an ecologically diverse region and a societally important watershed along the east coast of the United States as it provides drinking water to over 15 million people [41]. We first describe our dataset and baselines. Then

Algorithm 2 The flow of the proposed PDE-DGN model.

initialize the network model and the physical parameters $\{D_i\}$ and $\{ca_i\}$
for $epoch$ = 1 : number of training iterations **do**
  for t = 1 : number of time steps **do**
    Estimate adjacency matrix A^t using the current values of $\{D_i\}$ and $\{ca_i\}$ following Algorithm 1
    Make predictions using the recurrent graph network following Eqs. 5-6
    Add the accumulated errors to the loss function (Eq. 7)
  end for
  update model parameters (i.e., networks weights) and physical parameters (i.e., $\{D_i\}$ and ca_i)
end for

we discuss the results about the predictive performance using sparse data, the spatial distribution of errors, and model generalization. All experiments are conducted using Tensorflow on a computer with the following configuration: Intel Core i7-8750H CPU @2.20GHz $\times$ 6 Processor, 16 GiB Memory, 64-bit Win10 OS.

A. Dataset and baselines

The dataset is pulled from U.S. Geological Survey's National Water Information System [42] and the Water Quality Portal [43], the largest standardized water quality dataset for inland and coastal waterbodies [43]. Observations at a specific latitude and longitude were matched to river segments that vary in length from 48 to 23,120 meters. The river segments were defined by the national geospatial fabric used for the National Hydrologic Model as described by Regan et al. [44], and the river segments are split up to have roughly a one day water travel time. We match observations to river segments by snapping observations to the nearest stream segment within a tolerance of 250 m.

We study a subset of the Delaware River Basin (Christina River Watershed) with 42 river segments that feed into the mainstem Delaware River at Wilmington, Delaware. We use input features at the daily scale from Oct 01, 1980, to Sep 30, 2016 (13,149 dates). The input features have 10 dimensions which include daily total precipitation, daily mean air temperature, day of the year, solar radiation, shade fraction, potential evapotranspiration and the geometric features of each segment (e.g., elevation, length, slope and width). Water temperature observations were available for 32 segments but the temperature was observed only on certain dates. The number of temperature observations available for each segment ranges from 1 to 9,810 with a total of 51,103 observations across all dates and segments. We compare model performance to multiple baselines, which are described as follows:

- Artificial neural networks (ANN): We train an ANN model using data collected from all the segments on all the dates. The model is applied to predict water temperature on each date separately.
- Recurrent neural networks (RNN): We train an RNN model with the LSTM cell for modeling temperature dynamics across consecutive dates. The RNN model takes the daily input drivers but the loss is only defined on those dates with observations.
- HydroNets [16]: This method also takes into account both the temporal dependencies and spatial river structures using customized model architecture. It has poor performance on our dataset because its river-specific model parameters cannot be effectively trained for segments without observations.
- Recurrent Graph Neural Networks (RGrN) [17]: This baseline combines the graph convolutional networks and LSTM, and has shown promising results in predicting water temperature in streams.
- 2-points DGN: This is a variant of the proposed method, which only utilizes one closest upstream segment when estimating the second-order derivatives (as discussed in Section IV-B).

All the models are trained and applied to all the river segments. In the following experiments, we train each ML model using data from the first 24 years (Oct 01, 1980, to Sep 30, 2004) and then test in the next 12 years (Oct 01, 2004, to Sep 30, 2016). The hidden representation in these ML models is in 20 dimensions. We set the learning rate to be 0.0005 and update the model for 100 epochs for modeling water temperature.

B. Predictive performance using sparse data

We report the testing performance of different methods for temperature prediction and streamflow prediction in Table I. We also test the capacity of each model to learn using less training data by randomly selecting 5% and 10% labeled data from first 24 years for training the model. We repeat each experiment five times with random model initialization and random selection of sparser data (5%, 10%) and report the mean and standard deviation of the root mean square error (RMSE).

We observe that all the methods have larger RMSE values when we reduce the amount of training data. Our proposed method PDE-DGN outperforms baselines using different amounts of training data. In particular, RGrN and HydroNets perform worse than PDE-DGN because these models use a static graph based on the river network structure and thus they are limited in fully capturing dynamic interactions amongst streams. Although the standard graph convolutional structure can also extract different hidden representation for different segments (given their individual input features) and propagate the extracted information to the neighbors, such extraction process is conducted using a set of parameters shared over all the segments and over time. Hence, they cannot model how segment-specific physical variables (e.g., cross-sectional areas) and time-varying variables (e.g., streamflow) affect the segment interactions. Moreover, by leveraging the knowledge encoded by the PDE, the proposed method stands a higher chance at extracting more generalizable patterns. Our method

TABLE I

AVERAGE RMSE (± STANDARD DEVIATION) FROM FIVE RUNS FOR TEMPERATURE PREDICTION USING 5%, 10%, AND 100% TRAINING LABELS. HERE OUR METHOD IS COMPARED WITH ARTIFICIAL NEURAL NETWORKS (ANN), RECURRENT NEURAL NETWORKS (RNN), HYDRONETS, RGRN, AND PDE-DGN. BOLDED VALUES INDICATE THE BEST PERFORMING MODEL FOR EACH OF THE PERCENT TRAINING LABELS USED.

Method	5%	10%	100%
ANN	3.706±0.114	2.159±0.059	1.529±0.017
RNN	1.841±0.107	1.731±0.119	1.484±0.051
HydroNets	1.768±0.120	1.666±0.021	1.474±0.016
RGrN	1.744±0.073	1.654±0.077	1.443±0.017
2 points DGN	1.756±0.088	1.633±0.021	1.459±0.046
PDE-DGN	**1.740**±0.081	**1.574**±0.063	**1.428**±0.024

also outperforms the 2-points DGN method, which confirms the effectiveness of incorporating multiple upstream segments in estimating the second-order derivatives for representing the heat diffusion process.

Also, the improvement from RNN to RGrN shows that the incorporation of upstream-downstream dependencies in river networks is helpful to improve the accuracy for predictions. Such improvement is especially obvious as we use less training data. In terms of speed, the overall running time for a complete PDE-DGN model is around 45 minutes which is slightly higher than RGrN method since the only extra overhead during training stage is updating the adjacent matrix.

C. Assessing performance on unobserved segments

One important task for modeling river networks is to make predictions on unobserved river segments, which commonly exist in real-world basins. In this test, we evaluate different models for predicting river segments with no observation data. We report the results in Table II. Here Seg A to Seg E are five river segments that have sufficient observation data for stream temperatures. Each row shows the results for an individual experiment where we intentionally remove the temperature observations for a specific segment during the training period (Oct 01, 1980, to Sep 30, 2004). Then we report the prediction performance of RNN, RGrN, and PDE-DGN only on this segment during the testing period (Oct 01, 2004, to Sep 30, 2016) before and after we remove the training data.

We can observe larger errors produced by all these models after we remove training data for a segment. This is expected because different segments may exhibit different patterns and observations when the target segment is not used for training the model. However, we observe that the drop in performance of PDE-DGN is consistently smaller than that of the RNN model and the RGrN model. This confirms that the incorporation of the governing PDE helps the ML model to learn more generalizable patterns. We can also observe that RGrN generally performs better than the RNN model, which demonstrates the effectiveness of the graph structure in propagating relevant information to unobserved segments.

In Fig. 4, we show the predictions made by different models on segments A-E after we intentionally hide the training data

TABLE II

RMSE OF TEMPERATURE PREDICTION ON INDIVIDUAL SEGMENTS AFTER REMOVING TRAINING OBSERVATION DATA. HERE WE COMPARE THE PERFORMANCE OF RNN, RGRN, AND PDE-DGN MODELS. BOLDED VALUES INDICATE THE BEST PERFORMING MODEL FOR EACH SEGMENT AND TRAINING SCENARIO.

Segment	Method	With Obs	Without Obs
Seg A	RNN	2.297±0.082	4.104±0.921
	RGrN	2.176±0.070	3.724±0.637
	PDE-DGN	**2.151**±0.020	**3.127**±0.366
Seg B	RNN	1.116±0.064	1.440±0.068
	RGrN	1.014±0.016	1.387±0.068
	PDE-DGN	**0.994**±0.062	**1.289**±0.055
Seg C	RNN	1.082±0.083	2.302±0.124
	RGrN	1.007±0.032	2.021±0.217
	PDE-DGN	**0.992**±0.027	**1.936**±0.219
Seg D	RNN	0.955±0.053	2.527±0.161
	RGrN	0.943±0.020	2.278±0.391
	PDE-DGN	**0.917**±0.063	**1.971**±0.122
Seg E	RNN	1.067±0.045	1.461±0.097
	RGrN	**0.979**±0.018	1.277±0.063
	PDE-DGN	0.980±0.051	**1.243**±0.046

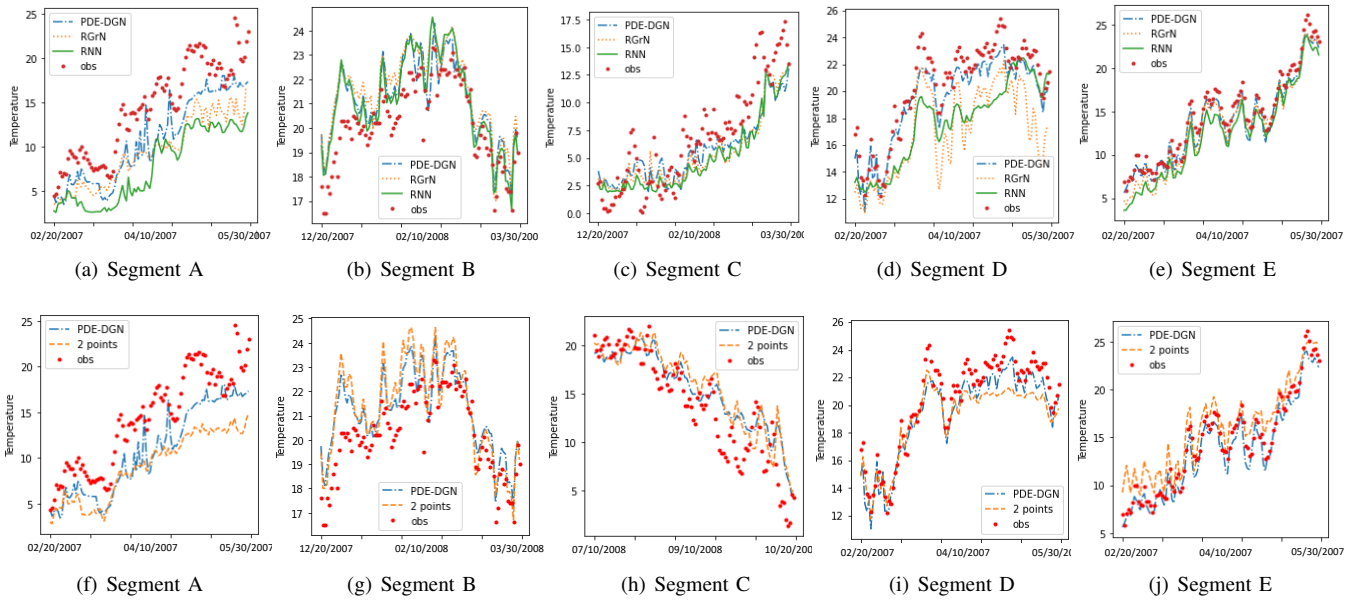
from each of these segments. PDE-DGN better matches true observations compared with other methods when the model does not have access to the training data from these segments. RNN generally predicts a smoother trajectory but does not capture temperature changes very well.

The 2-points DGN method does not work as well as the complete PDE-DGN because the DGN model only consider one closest upstream. However, if a river segment has a single upstream and downstream segment, the two approaches may perform similarly (e.g., Segment B shown in Fig. 4 (g)). In contrast, the segment E has two upstream segments, and that is why PDE-DGN performs better than 2-points DGN. The segment C also has two upstream segments but the PDE-DGN has similar performance with the 2-points DGN. This is because its two upstreams observation points are far away from the current segment thus making very little contribution (i.e., having much lower weights in the adjacency matrix) to the segment C.

D. Generalization test

It is known that traditional ML models are limited in generalizing to a new scenario that is very different from training data. We hypothesize that the proposed PDE-DGN has better generalizability because it follows general physical relationships that govern underlying processes. Here we test model generazability for stream temperature prediction across different seasons. In particular, we train each model using data only from colder seasons (spring, fall and winter) in the first 24 years and then test in summers in the next 12 years, as shown in Table III (first column). We also show a baseline in which each model is trained using all the data from the first 24 years and then tested in summers in the last 12 years (Table III second column).

We can observe that PDE-DGN performs better than other methods because of its awareness of the underlying physical



(a) Segment A (b) Segment B (c) Segment C (d) Segment D (e) Segment E

(f) Segment A (g) Segment B (h) Segment C (i) Segment D (j) Segment E

Fig. 4. Predictions made by (a-e) RNN, RGrN, PDE-DGN and (f-j) 2-points DGN and the complete PDE-DGN in Segments A-E after we intentionally hide the training data for each segment.

relationships. Because the adjacency matrices are created based on the governing PDE, the model has a higher chance to learn physically consistent patterns. Compared to the static graph model of the RGrN, the dynamic graph model of the PDE-DGN is advantageous because the connection strength between stream segments can change substantially in one season compared to another. The HydroNets model performs poorly because it is more likely to overfit the training data due to the higher model complexity.

TABLE III
TEMPERATURE RMSE IN SUMMERS FROM 2005 TO 2016. EACH MODEL IS TRAINED USING OBSERVATION DATA FROM SPRING, FALL, AND WINTER SEASONS (COLUMN 1) OR THE OBSERVATIONS DATA FROM ALL THE SEASONS (COLUMN 2) FROM OCT 1980 TO SEP 2004. HERE OUR METHOD IS COMPARED AGAINST ANN, RNN, HYDRONETS, RGRN, 2-POINTS DGN, AND PDE-DGN). BOLDED VALUES INDICATE THE BEST PERFORMING MODEL FOR EACH TRAINING SCENARIO.

Method	Train on cold seasons	Train on all the data
ANN	2.138±0.093	1.529±0.017
RNN	2.104±0.080	1.484±0.051
HydroNets	2.792±0.018	1.474±0.016
RGrN	2.085±0.046	1.443±0.017
2 points DGN	2.106±0.064	1.459±0.046
PDE-DGN	**2.055**±0.051	**1.428**±0.024

VI. CONCLUSION

In this paper, we propose a novel method called PDE-DGN for modeling interacting segments and predicting temperatures in river networks. We leverage the prior physical knowledge about segment-to-segment interactions embedded in PDE-based models to enhance the learning of latent representation in the proposed ML model. Moreover, we approximate the physical meaning by applying FDMs on the river segments guided by PDEs. Although there were marginal improvements in model performance when trained on all the data, our proposed method demonstrated superiority when handling data-sparse conditions and in generalizing to unseen scenarios. The proposed method also estimates physically meaningful parameters (e.g., PDE coefficients) that could inform other modeling or resource management activities and increase trust in deep learning models. In addition to modeling variables in river networks, the proposed method can be adjusted to model other complex systems which involve dynamic interacting processes. For example, this method could be potentially used for modeling particle interactions in quantum mechanics and gene coexpression in biological research.

VII. ACKNOWLEDGEMENTS

This work was support by the USGS Award G21AC10207, the National Science Foundation (NSF) under awards 1910017, 1918450, and 2028001, and Pitt Momentum Award. This research was supported in part by the University of Pittsburgh Center for Research Computing through the resources provided. Any use of trade, firm, or product names is for descriptive purposes only and does not imply endorsement by the U.S. Government. All model drivers used in this analysis can be found at [45].

REFERENCES

[1] J. R. Brett, "Energetic responses of salmon to temperature. a study of some thermal relations in the physiology and freshwater ecology of sockeye salmon (oncorhynchus nerkd)," *American zoologist*, vol. 11, no. 1, pp. 99–113, 1971.

[2] J. J. Magnuson, L. B. Crowder, and P. A. Medvick, "Temperature as an ecological resource," *American Zoologist*, vol. 19, no. 1, pp. 331–343, 1979.

[3] C. K. Batchelor and G. Batchelor, *An introduction to fluid dynamics*. Cambridge university press, 2000.

[4] S. J. Dugdale, D. M. Hannah, and I. A. Malcolm, "River temperature modelling: A review of process-based approaches and future directions," *Earth-Science Reviews*, vol. 175, pp. 97–113, 2017.

[5] F. Theurer, K. Voos, and W. Miller, "Instream water temperature model. instream flow information paper 16. us fish wildl serv," *Div. Biol. Serv., Tech. Rep. FWS OBS*, vol. 84, no. 15, pp. 11–42, 1984.

[6] S. L. Markstrom, R. S. Regan, L. E. Hay, R. J. Viger, R. M. Webb, R. A. Payn, and J. H. LaFontaine, "Prms-iv, the precipitation-runoff modeling system, version 4," *US Geological Survey Techniques and Methods*, no. 6-B7, 2015.

[7] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[8] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 2013, pp. 6645–6649.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[11] H. Gao, Z. Wang, L. Cai, and S. Ji, "Channelnets: Compact and efficient convolutional neural networks via channel-wise convolutions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[12] N. Ma, S. Mazumder, H. Wang, and B. Liu, "Entity-aware dependency-based deep graph attention network for comparative preference classification," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 5782–5788.

[13] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 974–983.

[14] D. Li and H. Ji, "Syntax-aware multi-task graph convolutional networks for biomedical relation extraction," in *Proceedings of the Tenth International Workshop on Health Text Mining and Information Analysis (LOUHI 2019)*, 2019, pp. 28–33.

[15] Y. Ye, S. Hou, L. Chen, J. Lei, W. Wan, J. Wang, Q. Xiong, and F. Shao, "Out-of-sample node representation learning for heterogeneous graph in real-time android malware detection." in *IJCAI*, 2019, pp. 4150–4156.

[16] Z. Moshe, A. Metzger, G. Elidan, F. Kratzert, S. Nevo, and R. El-Yaniv, "Hydronets: Leveraging river structure for hydrologic modeling," *arXiv preprint arXiv:2007.00595*, 2020.

[17] X. Jia, J. Zwart, J. Sadler, A. Appling, S. Oliver, S. Markstrom, J. Willard, S. Xu, M. Steinbach, J. Read, and V. Kumar, "Physics-guided recurrent graph model for predicting flow and temperature in river networks," in *SIAM International Conference on Data Mining*. SIAM, 2021.

[18] X. Jia, B. Lin, J. Zwart, J. Sadler, A. Appling, S. Oliver, and J. Read, "Graph-based reinforcement learning for active learning in real time: An application in modeling river networks," in *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*. SIAM, 2021, pp. 621–629.

[19] B. Anderson, T. S. Hy, and R. Kondor, "Cormorant: Covariant molecular neural networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 14 510–14 519.

[20] N. Muralidhar, M. R. Islam, M. Marwah, A. Karpatne, and N. Ramakrishnan, "Incorporating prior domain knowledge into deep neural networks," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 36–45.

[21] X. Jia, J. Willard, A. Karpatne, J. Read, J. Zwart, M. Steinbach, and V. Kumar, "Physics guided rnns for modeling dynamical systems: A case study in simulating lake temperature profiles," in *Proceedings of the 2019 SIAM International Conference on Data Mining*. SIAM, 2019, pp. 558–566.

[22] J. S. Read, X. Jia, J. Willard, A. P. Appling, J. A. Zwart, S. K. Oliver, A. Karpatne, G. J. Hansen, P. C. Hanson, W. Watkins *et al.*, "Process-guided deep learning predictions of lake water temperature," *Water Resources Research*, 2019.

[23] J.-X. Wang, J.-L. Wu, and H. Xiao, "Physics-informed machine learning approach for reconstructing reynolds stress modeling discrepancies based on dns data," *Physical Review Fluids*, 2017.

[24] D. Liu and Y. Wang, "Multi-fidelity physics-constrained neural network and its application in materials modeling," *Journal of Mechanical Design*, vol. 141, no. 12, 2019.

[25] N. Muralidhar, J. Bu, Z. Cao, L. He, N. Ramakrishnan, D. Tafti, and A. Karpatne, "Phynet: Physics guided neural networks for particle drag force prediction in assembly," in *Proceedings of the 2020 SIAM International Conference on Data Mining*. SIAM, 2020, pp. 559–567.

[26] J. Ling, A. Kurzawski, and J. Templeton, "Reynolds averaged turbulence modelling using deep neural networks with embedded invariance," *J. Fluid Mech*, 2016.

[27] Y. Qi, Q. Li, H. Karimian, and D. Liu, "A hybrid model for spatiotemporal forecasting of pm2. 5 based on graph convolutional neural network and long short-term memory," *Science of the Total Environment*, 2019.

[28] T. Xie and J. C. Grossman, "Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties," *Physical review letters*, vol. 120, no. 14, p. 145301, 2018.

[29] D. Zhu, F. Zhang, S. Wang, Y. Wang, X. Cheng, Z. Huang, and Y. Liu, "Understanding place characteristics in geographic contexts through graph convolutional neural networks," *Annals of the American Association of Geographers*, vol. 110, no. 2, pp. 408–420, 2020.

[30] L. C. Evans, "Partial differential equations," 2010.

[31] A. M. Bayen, R. L. Raffard, and C. J. Tomlin, "Network congestion alleviation using adjoint hybrid control: Application to highways," in *International Workshop on Hybrid Systems: Computation and Control*. Springer, 2004, pp. 95–110.

[32] M. J. Lighthill and G. B. Whitham, "On kinematic waves ii. a theory of traffic flow on long crowded roads," *Proc. R. Soc. Lond. A*, vol. 229, no. 1178, pp. 317–345, 1955.

[33] G. D. Smith, G. D. Smith, and G. D. S. Smith, *Numerical solution of partial differential equations: finite difference methods*. Oxford university press, 1985.

[34] B. Heinrich, *Finite difference methods on irregular networks: a generalized approach to second order elliptic problems*. Springer, 1987.

[35] J. Benito, F. Urena, and L. Gavete, "Influence of several factors in the generalized finite difference method," *Applied Mathematical Modelling*, vol. 25, no. 12, pp. 1039–1053, 2001.

[36] L. Arsenault, A. Lopez-Bezanilla, O. von Lilienfeld, and A. Millis, "Machine learning for many-body physics: The case of the anderson impurity model," *Phys. Rev. B*, 2014.

[37] K. Rudd and S. Ferrari, "A constrained integration (cint) approach to solving partial differential equations using artificial neural networks," *Neurocomputing*, 2015.

[38] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations," *arXiv preprint arXiv:1711.10561*, 2017.

[39] G. Garner, I. A. Malcolm, J. P. Sadler, and D. M. Hannah, "What causes cooling water temperature gradients in a forested stream reach?" *Hydrology and Earth System Sciences*, vol. 18, no. 12, p. 5361, 2014.

[40] S. L. Markstrom, R. S. Regan, L. E. Hay, R. J. Viger, R. M. Webb, R. A. Payn, and J. H. LaFontaine, "Prms-iv, the precipitation-runoff modeling system, version 4," *US Geological Survey Techniques and Methods*, no. 6-B7, 2015.

[41] T. N. Williamson, J. G. Lant, P. Claggett, E. A. Nystrom, P. C. Milly, H. L. Nelson, S. A. Hoffman, S. J. Colarullo, and J. M. Fischer, "Summary of hydrologic modeling for the delaware river basin using the water availability tool for environmental resources (water)," US Geological Survey, Tech. Rep., 2015.

[42] U. G. Survey, "National water information system data available on the world wide web (usgs water data for the nation)," 2016.

[43] E. K. Read, L. Carr, L. De Cicco, H. A. Dugan, P. C. Hanson, J. A. Hart, J. Kreft, J. S. Read, and L. A. Winslow, "Water quality data for national-scale aquatic research: The water quality portal," *Water Resources Research*, vol. 53, no. 2, pp. 1735–1745, 2017.

[44] R. S. Regan, S. L. Markstrom, L. E. Hay, R. J. Viger, P. A. Norton, J. M. Driscoll, and J. H. LaFontaine, "Description of the national hydrologic model for use with the precipitation-runoff modeling system (prms)," US Geological Survey, Tech. Rep., 2018.

[45] S. K. Oliver, A. Appling, R. A. Atshan, W. D. Watkins, J. M. Sadler, H. R. Corson-Dosch, J. A. Zwart, and J. S. Read, "Predicting water temperature in the delaware river basin," *U.S. Geological Survey data release*, 2021.