Physics-Guided Recurrent Graph Model for Predicting Flow and Temperature in River Networks

Xiaowei Jia[1], Jacob Zwart[2], Jeffrey Sadler[2], Alison Appling[2], Samantha Oliver[2], Steven Markstrom[2], Jared Willard[3], Shaoming Xu[3], Michael Steinbach[3], Jordan Read[2], and Vipin Kumar[3]

[1] University of Pittsburgh, [2]U.S. Geological Survey, [3] University of Minnesota

[1]xiaowei@pitt.edu,[2]{jzwart,jsadler,aappling,soliver,markstro,jread}@usgs.gov,

[3]{willa099,xu000114,stei0062,kumar001}@umn.edu

Abstract

This paper proposes a physics-guided machine learning approach that combines machine learning models and physics-based models to improve the prediction of water flow and temperature in river networks. We first build a recurrent graph network model to capture the interactions among multiple segments in the river network. Then we transfer knowledge from physics-based models to guide the learning of the machine learning model. We also propose a new loss function that balances the performance over different river segments. We demonstrate the effectiveness of the proposed method in predicting temperature and streamflow in a subset of the Delaware River Basin. In particular, the proposed method has brought a 33%/14% accuracy improvement over the state-of-the-art physics-based model and 24%/14% over traditional machine learning models (e.g., LSTM) in temperature/streamflow prediction using very sparse (0.1%) training data. The proposed method has also been shown to produce better performance when generalized to different seasons or river segments with different streamflow ranges.

1 Introduction

Machine learning (ML) models, which have found immense success in commercial applications, e.g., computer vision and natural language processing, are beginning to play an important role in advancing scientific discovery [3, 9, 22]. However, scientific problems often involve non-stationary relationships among physical variables which can change over space and time. In the absence of adequate information about the physical mechanisms of real-world processes, traditional ML approaches are prone to false discoveries due to the difficulty to capture these complex relationships solely from data. Moreover, the data available for many scientific problems is far smaller than what is needed to effectively train advanced ML models.

The focus of this paper is on modeling physical systems that have multiple interacting processes. In particular, we consider the application of predicting flow and temperature in river networks for both observed and unobserved river segments. In this problem, segments in the river network can show different flow and thermodynamic patterns driven by differences in catchment characteristics (e.g. slope, soil characteristics) and meteorological drivers (e.g. temperature, precipitation). These segments also interact with each other through the water advected from upstream to downstream segments. Moreover, there are often only a handful of river segments in a network that are monitored; thus there is limited data to train ML models. Accurate prediction of streamflow and water temperature aids in decision making for resource managers. For example, drinking water reservoir operators in the Delaware River Basin need to supply safe drinking water to New York City while also maintaining sufficient streamflow and cool water temperatures in the river network downstream of the reservoirs [16]. Accurate predictions helps managers optimize when and how much water to release downstream to maintain the flow and temperature regimes.

In scientific domains, physics-based models are often used to study engineering and environmental systems. Even though these models are based on known physical laws that govern relationships between input and output variables, most physics-based models are necessarily approximations of reality due to incomplete knowledge of certain processes or omission of processes to maintain computational efficiency. In particular, existing physics-based approaches for predicting river networks simulate the target variables (e.g., streamflow and temperature) based on general physical relationships such as energy and mass conservation. However, the model predictions still rely on qualitative parameterizations (approximations) based on soil and surficial geologic classification along with topography, land cover and climate input. Hence, such models can only

provide sub-optimal predictive performance. Furthermore, calibration of these physics-based models is often extremely time intensive due to interactions among parameters within segments and across segments.

Intuitively, we can model each river segment independently by an individual ML model such as a Recurrent Neural Network (RNN). However, most of the segments in a real river network are poorly observed or completely unobserved, which makes it impossible to build a purely data driven model for each segment separately. Besides, individual models may ignore the rich spatial and temporal contextual information, e.g., how the streamflows from upstream segments change the water temperature in downstream segments.

In this paper, we propose a global model, Physics Guided Recurrent Graph Networks (PGRGrN), which is applied to all the river segments. The architecture of PGRGrN includes graph convolutional component and recurrent component to explicitly capture the spatial interactions among different river segments as well as their temporal dynamics. Design of such an architecture for this application faces two challenges. First, existing graph convolutional networks (GCN)-based models extract abstract latent variables to propagate over the network but they do not explicitly embed physical information about interactions among different nodes. Such latent variables can become less informative when they are learned from sparser and less representative observation data, which can make the GCN model not generalizable. To address this challenge, we propose to transfer the knowledge from physics-based models to guide the learning of latent variables in the proposed ML model. We implement this by enforcing a physical interpretation to latent variables using the intermediate variables simulated by the physics-based model.

The second challenge is about the variability of target variables in a complex system. For example, streamflow can vary by several orders of magnitude across segments in a river network. When we train a global ML model, the training process using traditional loss functions (e.g., mean squared loss) defined over the entire network can be dominated by river segments that contribute most to the overall error (e.g., segments with high streamflow). However, it is also important to accurately predict river segments with lower streamflow, as accurate prediction for these segments provides important information regarding the habitat for aquatic life and aquatic biogeochemical cycling. To address this challenge, we design a new loss function to ensure that the global ML model can simultaneously capture the local patterns of all the different segments. The local patterns of each segment can be extracted using an individual ML model trained only for this segment using simulation data (which is plentiful). Then during the training of the global ML model, we use a distance-based contrastive loss function to enforce its consistency with the extracted local patterns.

We implement our proposed method in a real-world dataset collected over 36 years from the Delaware River Basin located in the Northeast US and demonstrate our method's superior predictive performance over existing methods. Moreover, we show that the proposed method produces much better predictive performance using sparse observations and also has better generalizability.

2 Related Work

Physics can be incorporated into ML models in several ways [25], including developing new model architectures [1], applying additional loss functions [5, 18], and modeling prediction residuals [11]. New ML architectures have been designed to enforce known physical relationships among multiple internal processes that jointly convert inputs to outputs [1, 14], thus reducing the space for searching parameters. In the context of modeling river networks, Moshe et al. [13] propose HydroNets, which combines the information from its upstream segments for predicting streamflow. It also learns local patterns for each basin by introducing basin-specific model layers. This method focuses on predicting well-observed basins but remains limited in generalizing to different scenarios or learning with less data. Another interesting direction is to build a hybrid architecture of physics-based and machine learning models [24], which is beyond the scope of this paper.

Recently, the Graph Convolutional Networks (GCN) model has shown great promise in modeling node interactions in a graph [4, 8] and also produced improved prediction accuracy in several scientific problems [15, 28]. Besides, the idea of combining GCN with Long-Short Term Memory (LSTM) has been explored in several applications to extract spatio-temporal features [2, 20, 27]. However, the information propagated amongst nodes in GCN is essentially an abstract representation learned by end-to-end training. Such abstract representations are not meant to enforce consistency with known physical relationships among different processes. Given limited training data collected from certain regions and time periods, as in most scientific applications, the learning of these abstract representations can be highly biased towards these regions and time periods, which makes the model less generalizable.

Simulation data have been used to assist in training ML models. Since many ML models require an initial choice of model parameters before training, researchers have explored different ways to physically inform a model starting state. Poor initialization can cause

models to anchor in local minima, which is especially true for deep neural networks. One way to harness physics-based modeling knowledge is to use the physics-based model's simulated data to pre-train the ML model, which also alleviates data paucity issues [7]. In addition, Read et al. [18] show that such models are able to generalize better to unseen scenarios than pure physics-based models.

3 Problem Definition and Preliminaries

3.1 Problem definition Our objective is to model the dynamics of temperature and streamflow 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 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 anywhere upstream of the segment j. The matrix $\mathbf{A}$ represents the adjacency level between each pair of segments, i.e., $\mathbf{A}_{ij} = 0$ means there is no edge from the segment i to the segment j and a higher value of $\mathbf{A}_{ij}$ indicates that the segment i is closer to the segment j in terms of the river distance. More details of how we generate the adjacency matrix are discussed in Section 5.1.

For each river segment i, we have access to its input features at multiple time steps $\mathbf{X}_i = \{\mathbf{x}_i^1, \mathbf{x}_i^2, ..., \mathbf{x}_i^T\}$. The input features $\mathbf{x}_i^t$ are a D-dimensional vector, which include meteorological drivers, geometric parameters of the segments, etc. (detailed in Section 5.1). We also have a set of observed target variables $\mathbf{Y} = \{y_i^t\}$ but they are only available for certain time steps $t \in \{1, ..., T\}$ and certain segments $i \in \{1, ..., N\}$.

3.2 Physics-based Streamflow and Temperature Model The Precipitation-Runoff Modeling System (PRMS) [12] and the coupled Stream Network Temperature Model (SNTemp) [21] is a physics-based model that simulates daily streamflow and water temperature for river networks, and other variables. PRMS is a one-dimensional, distributed-parameter modeling system that translates spatially-explicit meteorological information into water information including evaporation, transpiration, runoff, infiltration, groundwater flow, and streamflow. PRMS has been used to simulate catchment hydrologic variables at regional [10] to national scales [19] in support of resource management decisions. The SNTemp module for PRMS simulates mean daily stream water temperature for each river segment by solving an energy mass balance model which accounts for the effect of inflows (upstream, groundwater, surface runoff), outflows, and surface heating and cooling on heat transfer in each stream segment. Calibration of PRMS-SNTemp is extremely time-consuming because it involves a large number of parameters (84 parameters) and the parameters interact with each other both within segments and across segments.

3.3 Recurrent Neural Networks (RNN) and Long-Short Term Memory (LSTM) The RNN model defines transition relationships for the extracted hidden representation through a recurrent cell structure. In this work, we adopt the LSTM cell which has proven to be effective in 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. Here we omit the subscript i as we do not target a specific river segment. Specifically, the LSTM first computes the candidate cell state $\bar{\mathbf{c}}^t$ and a set of gating variables, as follows:

$$\begin{aligned} \bar{\mathbf{c}}^t &= \tanh(\mathbf{W}_c^h \mathbf{h}^{t-1} + \mathbf{W}_c^x \mathbf{x}^t + \mathbf{b}_c), \\ \mathbf{f}^t &= \sigma(\mathbf{W}_f^h \mathbf{h}^{t-1} + \mathbf{W}_f^x \mathbf{x}^t + \mathbf{b}_f), \\ \mathbf{g}^t &= \sigma(\mathbf{W}_g^h \mathbf{h}^{t-1} + \mathbf{W}_g^x \mathbf{x}^t + \mathbf{b}_g), \\ \mathbf{o}^t &= \sigma(\mathbf{W}_o^h \mathbf{h}^{t-1} + \mathbf{W}_o^x \mathbf{x}^t + \mathbf{b}_o). \end{aligned} \tag{3.1}$$

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

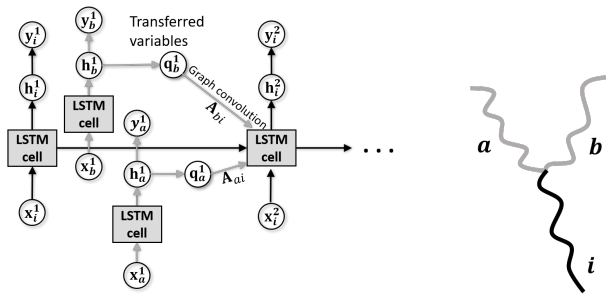
$$\begin{aligned} \mathbf{c}^t &= \mathbf{f}^t \otimes \mathbf{c}^{t-1} + \mathbf{g}^t \otimes \bar{\mathbf{c}}^t, \\ \mathbf{h}^t &= \mathbf{o}^t \otimes \tanh(\mathbf{c}^t), \end{aligned} \tag{3.2}$$

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

4 Method

In this section, we start with introducing details of the model architecture. Then we propose a strategy to help enforce physical relationships by leveraging the physical knowledge embedded in physics-based models. Finally, we introduce a contrastive loss function that attempts to ensures that the model performance on individual river segments is not compromised while optimizing the performance on the entire set of segments.

4.1 Recurrent Graph Network We introduce a global ML model architecture, Recurrent Graph Network (RGrN), which is trained using data collected from all the river segments. Effective modeling of river segments requires the ability to capture their temporal dynamics and the influence received from upstream segments. Hence, we incorporate the information from



(a) (b)

Figure 1: (a) The RGrN architecture for an example river network shown in (b). Here segments a and b are upstream of the segment i. Grey arrows indicate the modeling components for upstream segments.

both previous time steps and neighbors (i.e., upstream segments) when modeling each segment (Fig. 1).

Here we describe the recurrent process of generating the hidden representation $\mathbf{h}^t$ from $\mathbf{h}^{t-1}$, and we repeat this process for the entire sequence from $t = 2$ to T ($\mathbf{h}^1$ learned from an LSTM model). For each river segment i at time $t-1$, the model extracts latent variables which contain relevant information to pass to its downstream segments. We refer to these latent variables as transferred variables. For example, the amount of water advected from each segment and its water temperature can directly impact the change of water temperature for its downstream segments. We generate the transferred variables $\mathbf{q}_i^{t-1}$ from the hidden representation $\mathbf{h}_i^{t-1}$, as follows:

$$\mathbf{q}_i^{t-1} = \tanh(\mathbf{W}_q \mathbf{h}_i^{t-1} + \mathbf{b}_q), \tag{4.3}$$

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

After gathering the transferred variables for all the segments, we develop a new recurrent cell structure for each segment i that integrates the transferred variables from its upstream segments into the computation of the cell state $\mathbf{c}_i^t$. This can be expressed as follows:

$$\mathbf{c}_i^t = \mathbf{f}_i^t \otimes (\mathbf{c}_i^{t-1} + \sum_{(j,i)\in\mathcal{E}} \mathbf{A}_{ji}\mathbf{q}_j^{t-1}) + \mathbf{g}_i^t \otimes \bar{\mathbf{c}}_i^t \tag{4.4}$$

We can observe that the forget gate not only filters the previous information from the segment i itself but also from its neighbors (i.e., upstream segments). Each upstream segment j is weighted by the adjacency level $\mathbf{A}_{ji}$. When a river segment has no upstream segments (i.e., head water), the computation of $\mathbf{c}_i^t$ is the same as with the standard LSTM. In Eq. 4.4, we use $\mathbf{q}_j^{t-1}$ from the previous time step because of the time delay in transferring the influence from upstream to downstream segments (the maximum travel time is approximately one day according to PRMS).

After obtaining the cell state, we can compute the hidden representation $\mathbf{h}_i^t$ by following Eq. 3.2. Finally, we generate the predicted output from the hidden representation as follows:

$$\hat{\mathbf{y}}_i^t = \mathbf{W}_y \mathbf{h}_i^t + \mathbf{b}_y, \tag{4.5}$$

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 using true observations $\mathbf{Y} = \{\mathbf{y}_i^t\}$ on the available time steps and segments, as follows:

$$\mathcal{L}_{\text{RGrN}} = \frac{1}{|\mathbf{Y}|} \sum_{\{(i,t)|\mathbf{y}_i^t \in \mathbf{Y}\}} (\mathbf{y}_i^t - \hat{\mathbf{y}}_i^t)^2. \tag{4.6}$$

4.2 Transferring knowledge from physics-based models Training RGrN directly in an end-to-end fashion can only learn abstract representation for transferred variables while ignoring their physical interpretation. These variables can become less informative when they are learned from sparser and less representative observation data. To this end, we introduce a new strategy to enforce the prior physical relationships amongst different river segments which are encoded by physics-based models. It helps make RGrN model more generalizable and also reduces the amount of observation data required to train a high-quality model. This strategy can be applied to a wide range of scientific problems that are modeled as a set of interacting processes.

Here we use the river temperature modeling as an example to illustrate the proposed strategy. For each river segment, its temperature change is driven by energy exchanges caused by solar radiation, rainfall, evaporation, conductive and convective heat transfer, and net heat advected into river segments (e.g., groundwater flow, upstream flow, downstream flow). These energy fluxes can be summarized into three categories:

$$\Delta\text{Temperature} \propto F_{in} - F_{out} + F_{up}, \tag{4.7}$$

where F_{in} denotes the incoming energy fluxes from solar radiation, rainfall and other natural sources, F_{out} denotes the outgoing energy fluxes including long-wave emission, evaporation, conductive and convective heat transfer, and F_{up} denotes the net heat advected from upstream segments. The term F_{up} can be estimated by a set of intermediate physical variables from upstream segments, which include upstream flow, upstream water temperature, humidity, and other physical characteristics. These intermediate physical variables can be simulated by PRMS-SNTemp internally.

Our goal is to ensure that the transferred variables $\mathbf{q}_i^t$ at each segment contain sufficient information to represent these intermediate physical variables so that downstream segments can gather all the information needed for capturing F_{up}. For each segment i at time t, we first simulate the set of intermediate variables $\mathbf{s}_i^t$

by running PRMS-SNTemp. Then we use $\mathbf{s}_i^t$ to add supervision on the transferred variables $\mathbf{q}_i^t$ such that we can extract $\mathbf{s}_i^t$ from $\mathbf{q}_i^t$. More formally, we define a loss function on transferred variables as follows:

$$\mathcal{L}_{\text{trans}} = \frac{1}{NT} \sum_i \sum_t ||\mathbf{s}_i^t - (\mathbf{W}_s \mathbf{q}_i^t + \mathbf{b}_s)||^2, \tag{4.8}$$

where $\mathbf{W}_s$ and $\mathbf{b}_s$ are model parameters. We call this model as Physics-Guided Recurrent Graph Networks (PGRGrN) because we leverage the physical relationships between different river segments.

Since the computation of loss $\mathcal{L}_{\text{trans}}$ does not require observation data, we can use it to pre-train RGrN to enforce physical relationships. The pre-training not only explicitly enforces the physical relationships among river segments, but also enables full usage of physical intermediates obtained from physics-based models to enhance the representation learned for $\mathbf{q}_i^t$ and its previous layer $\mathbf{h}_i^t$. In particular, the intermediate physical variables used in this work include streamflow, stream temperature, relative humidity, cloud cover, groundwater and shallow subsurface flow, and surface runoff.

By using intermediate physical variables in the pre-training phase, the ML model is guided but not constrained by the physics-based model output, which allows for more flexibility in the fine-tuning phase to automatically learn information in $\mathbf{q}_i^t$ that is poorly known or not yet discovered while also remaining helpful for modeling the interactions among river segments.

At the same time, we can run PRMS-SNTemp to simulate the final target variables, which are represented as $\tilde{\mathbf{y}}_i^t$. Although the simulated data are not accurate reflection of the observation data, we can generate adequate simulations on every day and for every segment. The simulation data also follow many general physical relationships used to build the physics-based model. Given that observation data is often scarce, we can use simulated target variables to initialize the model via pre-training. Hence, we define another pre-training loss on target variables as follows:

$$\mathcal{L}_{\text{tar}} = \frac{1}{NT} \sum_i \sum_t (\tilde{\mathbf{y}}_i^t - \hat{\mathbf{y}}_i^t)^2, \tag{4.9}$$

Combining Eqs. 4.8 and 4.9, we get the final pre-training loss as follows:

$$\mathcal{L}_{\text{pre-train}} = \mathcal{L}_{\text{tar}} + \lambda \mathcal{L}_{\text{trans}}, \tag{4.10}$$

where λ is a hyper-parameter to balance two losses. We call this model (pre-trained using simulation data and then fine-tuned with true observations) as $\text{PGRGrN}^{\text{ptr}}$.

4.3 Segment-wise contrastive loss The relationship between input features and target variables can be very complex in environmental systems, e.g., slight changes in segment slope and catchment size can drastically alter the streamflow. Traditional loss functions for regression problems, such as mean squared loss, tend to be dominated by river segments with larger errors while degrading the performance on other segments with smaller errors. This issue can be further exacerbated given limited observation data on most river segments, especially low-flow segments. Although improving the segments with smaller errors does not contribute much to reducing the overall error, accurately predicting streamflow at these segments provides important information regarding habitat for aquatic life and biogeochemical cycling.

Our goal is to ensure that the global ML model trained on all the river segments should also be consistent with the local patterns extracted from each river segment. In particular, we train N individual LSTM models, $\mathcal{M}_1$ to $\mathcal{M}_N$, for each segment using the simulation data. Each individual model $\mathcal{M}_i$ is trained to predict simulated target variables (i.e., $\tilde{\mathbf{y}}_i$) for a specific segment i. Even though there is a gap between simulation data and true observation data, these individual models have a better chance at capturing the local temporal patterns of each river segment.

When we apply the global PGRGrN model to a specific segment i, the patterns predicted by the PGRGrN model should be similar to the local patterns learned by the individual model $\mathcal{M}_i$. Specifically, we compute the hidden representation $\mathbf{h}_i^t$ from PGRGrN at each time t, which encodes dynamic patterns that directly output target variables. We also run the individual model $\mathcal{M}_i$ to compute its hidden representation $\tilde{\mathbf{h}}_i^t$, which encodes the local temporal patterns for this segment. The hidden representation $\mathbf{h}_i^t$ from PGRGrN should be close to the corresponding local hidden representation $\tilde{\mathbf{h}}_i^t$ and different from the $\tilde{\mathbf{h}}_j^t$ of other segments, i.e., $j \neq i$. More formally, we define a contrastive loss as follows:

$$\mathcal{L}_{\text{ctr}} = -\frac{1}{NT} \sum_i \sum_t \log \frac{\exp(\mathbf{h}_i^{t\,T} \mathbf{W}_{ctr} \tilde{\mathbf{h}}_i^t)}{\sum_j \exp(\mathbf{h}_i^{t\,T} \mathbf{W}_{ctr} \tilde{\mathbf{h}}_j^t)}, \tag{4.11}$$

where $\mathbf{W}_{ctr}$ is model parameters for computing the similarity of hidden representation. To ensure that the hidden representation produced by PGRGrN ($\mathbf{h}$) and by individual models ($\tilde{\mathbf{h}}$) are comparable, we use shared parameters $\{\mathbf{W}_y,\ \mathbf{b}_y\}$ in the last layer (Eq. 4.5) for individual models and the global PGRGrN model and they are fitted when training individual models.

By combining the contrastive loss and the loss of RGrN (Eq. 4.6), we get the fine-tuning loss as follows:

$$\mathcal{L}_{\text{finetune}} = \mathcal{L}_{\text{RGrN}} + \gamma \mathcal{L}_{\text{ctr}}, \tag{4.12}$$

where γ is a hyper-parameter to balance the supervised loss of PGRGrN and the contrastive loss.

The proposed contrastive loss provides an alternative way in which different segments are comparable with each other. Traditional loss functions do not perform well for every segment because they are defined on target variables which may vary drastically across different segments. Instead, the contrastive loss matches temporal patterns encoded in the space of hidden representation, which alleviates this issue.

5 Experimental Results

We evaluate the proposed method for predicting stream temperature and streamflow 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 [26]. We first describe our dataset and baselines. Then we discuss the results about the predictive performance using sparse data, the effectiveness of pre-training, the spatial distribution of errors, and model generalization.

5.1 Dataset and baselines The dataset is pulled from U.S. Geological Survey's National Water Information System [23] and the Water Quality Portal [17], the largest standardized water quality data set for inland and coastal waterbodies [17]. The river segments were defined by the national geospatial fabric used for the National Hydrologic Model [19], and the river segments are split up to have roughly a one day water travel time. We study a subset of the Delaware River Basin with 42 river segments that feed into the mainstream Delaware River at Wilmington, DE. 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 average precipitation, daily average air temperature, date 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 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. Streamflow observations were available for 18 segments. The number of streamflow observations available for each segment ranges from 4,877 to 13,149 with a total of 206,920 observations across all dates and segments.

We generate the adjacency matrix $\mathbf{A}$ based on the river distance between each pair of river segment outlets, represented as $\text{dist}(i, j)$. We standardize the stream distance and then compute the adjacency level as $\mathbf{A}_{ij} = 1/(1 + \exp(\text{dist}(i, j)))$ for each edge $(i, j) \in \mathcal{E}$.

We compare model performance to multiple baselines, including the physics-based PRMS-SNTemp model, artificial neural networks (ANN), RNN with the LSTM cell, and the state-of-the-art PGRNN method [5] which uses simulation data to pre-train an LSTM model and then fine-tunes it with true observation data (represented as RNN^{ptr}). Since a region-specific calibration PRMS-SNTemp is extremely time-consuming, a version with default values of parameters is widely used in the hydrologic domain [19], and is also adopted for comparison in this paper. We evaluate three variants of the proposed method, RGrN (trained to minimize $\mathcal{L}_{\text{RGrN}}$), $\text{PGRGrN}^{\text{ptr}}$ (pre-training using the strategy in Section 4.2 and fine-tuning to minimize $\mathcal{L}_{\text{RGrN}}$, Eq. 4.6), and $\text{PGRGrN}^{\text{ptr,ctr}}$ (pre-training using the strategy in Section 4.2 and fine-tuning to minimize $\mathcal{L}_{\text{finetune}}$, Eq. 4.12). All the ML models are trained and applied to all the river segments (i.e., all models are global). 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 31, 2016). The hidden representation in these ML models is in 20 dimension (same for $\mathbf{q}_i^t$).

5.2 Overall predictive performance We report the testing performance of different methods for temperature prediction and streamflow prediction in Table 1 and Table 2, respectively. We also test the capacity of each model to learn using less training data by randomly selecting 0.1% and 2% labeled data from first 24 years for training the model. For RNN^{ptr}, $\text{PGRGrN}^{\text{ptr}}$ and $\text{PGRGrN}^{\text{ptr,ctr}}$, we assume the access to simulation data on every single date from Oct 01 1980 to Sep 20 2016 because they can be generated by simply running PRMS-SNTemp model on input drivers. We repeat each experiment five times with random model initialization and random selection of sparser data (0.1%, 2%) and report the mean and standard deviation of the root mean square error (RMSE).

We can observe that the proposed method outperforms baselines by a considerable margin (Tables 1 and 2). The improvement from ANN to RNN shows that the recurrent component is helpful for capturing temporal patterns. RGrN performs better than RNN because it utilizes upstream-downstream dependencies which are critical for an accurate accounting of temperature and streamflow.

We also observe that $\text{PGRGrN}^{\text{ptr}}$ has much better performance than RGrN using just 0.1% or 2% data. This is because we leverage the physical knowledge to learn representative latent variables without risking overfitting small amount of observations.

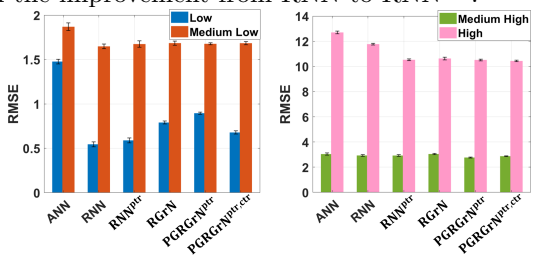
Table 1: Prediction RMSE (± standard deviation) for temperature modeling using 0.1%, 2% and 100% training labels. Rows in grey color represent methods using simulation data.

Method	0.1%	2%	100%
PRMS-SNTemp	3.661	3.661	3.661
ANN	3.706±0.114	2.159±0.059	1.575±0.035
RNN	3.234±0.057	1.908±0.048	1.546±0.045
RNN^{ptr}	2.818±0.059	1.810±0.057	1.444±0.039
RGrN	2.849±0.049	1.906±0.063	1.408±0.068
PGRGrN^{ptr}	2.556±0.045	1.715±0.041	1.406±0.035
PGRGrN^{ptr,ctr}	2.464±0.105	1.636±0.056	1.402±0.034

Table 2: Prediction RMSE for streamflow modeling using 0.1%, 2% and 100% training labels.

Method	0.1%	2%	100%
PRMS-SNTemp	6.834	6.834	6.834
ANN	7.116±0.120	5.777±0.063	4.801±0.055
RNN	6.885±0.068	5.718±0.114	4.406±0.064
RNN^{ptr}	6.367±0.067	5.529±0.053	4.104±0.049
RGrN	6.299±0.053	5.473±0.064	4.139±0.067
PGRGrN^{ptr}	5.824±0.075	4.708±0.032	4.106±0.046
PGRGrN^{ptr,ctr}	5.895±0.069	4.679±0.082	4.076±0.059

PGRGrN^{ptr,ctr} further improves the performance by enforcing local patterns of each segment. The standard pre-training method is also helpful for the RNN model given the improvement from RNN to RNN^{ptr}.



(a) (b)

Figure 2: Distribution of prediction errors in (a) low and medium-low segments, and (b) medium-high and high-flow segments.

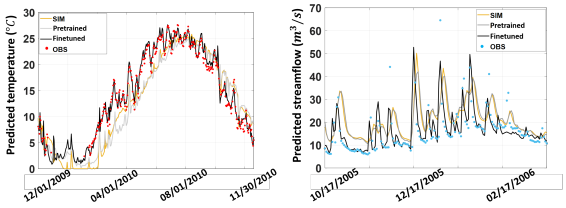
To understand how the performance varies across different types of river segments, we show the streamflow prediction errors for four types of segments, low ($<0.5m^3/s$), medium low (0.5-$2m^3/s$), medium high (2-$5m^3/s$) and high ($>5m^3/s$) in Fig. 2. RGrN and PGRGrN^{ptr} generally perform better than other methods in medium-high and high-flow segments but perform much worse in low-flow segments. This is because the neighboring river segments tend to have similar embeddings after graph convolution and thus the training of RGrN pays even less attention to low-flow segments given the fact that there are only a few low-flow segments in the river network. As shown in Fig. 2 (a), this issue is partly addressed by using the contrastive loss. An alternative solution to this issue is to intelligently select the most suitable model for different types of river segments, which we suggest as future work.

Table 3: RMSE of temperature prediction on individual segments after removing training observation data.

Segment	Method	With Obs	Without Obs
Seg A	RNN	2.297±0.082	3.328±0.132
	RGrN	2.135±0.060	2.749±0.079
	PGRGrN^{ptr,ctr}	2.084±0.053	2.501±0.037
Seg B	RNN	1.116±0.064	1.384±0.065
	RGrN	0.981±0.037	1.214±0.032
	PGRGrN^{ptr,ctr}	1.047±0.024	1.205±0.016
Seg C	RNN	1.082±0.083	1.804±0.041
	RGrN	1.013±0.033	1.796±0.077
	PGRGrN^{ptr,ctr}	0.989±0.026	1.589±0.040
Seg D	RNN	0.955±0.053	1.805±0.064
	RGrN	0.902±0.026	1.597±0.024
	PGRGrN^{ptr,ctr}	0.996±0.025	1.297±0.017
Seg E	RNN	1.067±0.045	1.646±0.075
	RGrN	0.977±0.031	1.357±0.033
	PGRGrN^{ptr,ctr}	1.013±0.025	1.345±0.041

5.3 Assessing performance on unobserved segments Here we aim to test the performance of temperature prediction for the segments which have no observation data (Tables 3). Such segments commonly exist in a real-world basin system. Seg A to Seg E are five river segments which have sufficient observation data for both stream temperature and streamflow. 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 predictive performance of RNN, RGrN, and PGRGrN^{ptr,ctr} only on this segment during the testing period (Oct 01, 2004 to Sep 31, 2016).

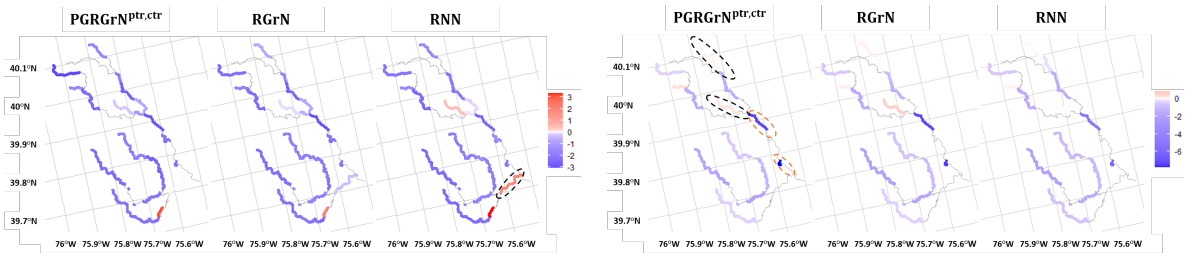
We can observe larger errors produced by all the three models after we remove training data for a segment. This is expected because segment-specific observation data has not been used for refinement of the model. However, we observe that the drop in performance of PGRGrN^{ptr,ctr} is consistent and often significantly smaller than that of the RNN model.



(a) (b)

Figure 3: The predictions of pre-trained and fine-tuned models for (a) temperature and (b) streamflow.

5.4 Effectiveness of pre-training In Fig. 3, we randomly select two example segments to show how predictions change from the pre-trained model to the fine-tuned model using 2% training data. Here the fine-tuned model is the same with the PGRGrN^{ptr} model



(a) (b)

Figure 4: Prediction errors across different segments for (a) temperature prediction and (b) streamflow prediction. Here we show the error as the RMSE of each method minus the RMSE of PRMS-SNTemp simulation data to get a better contrast. Darker blue indicates better performance of the ML model over the physics-based model. In (a), we do not discuss the red colored segment on the bottom (predicted poorly by all the three methods) and head water (75.7°W, 39.95°N) predicted poorly by RNN because they just have one test observation.

used in the previous results but is only fine-tuned using 2% observations and the pre-trained model does not use observations for fine-tuning. We can observe that the pre-trained model matches the simulated temperatures very well and thus can capture general temperature or streamflow patterns even without using observation data. There is still a gap from the pre-trained model to the true observations since it emulates PRMS-SNTemp, which has inherent bias due to an incomplete representation of physics. Nevertheless, after learning general patterns from simulation data, the model can be fine-tuned to match true observations using much less training data. This can be verified as we show that the model fine-tuned with just 2% data can match true observations much better.

5.5 Spatial distribution of errors Fig. 4 shows the distribution of prediction errors across different segments. In Fig. 4 (a), we can observe that RGrN and PGRGrN$^{\text{ptr,ctr}}$ produce smaller temperature error than RNN in many segments. We find one segment (in dashed circle) where RNN performs much worse than the proposed methods. This is the only segment in the data set for which we have no training data but sufficient testing data. The reason why RGrN can produce better predictions for this unlabeled segment is that it leverages the dependencies with other segments to learn the temperature patterns even without training data from this specific segment.

For streamflow prediction (Fig. 4 (b)), it can be seen that RGrN and PGRGrN$^{\text{ptr,ctr}}$ have lower RMSE in several high-flow segments (e.g., the segments in red dashed circles). However, RGrN performs worse than RNN on headwater segments (in black dashed circles) and these segments have lower streamflow. Compared with RGrN, it can be seen that PGRGrN$^{\text{ptr,ctr}}$ alleviates this issue and produces smaller errors in these low-flow segments by using the contrastive loss.

Table 4: Temperature RMSE in summers from 2005 to 2016. Each model is trained using observation data from colder seasons (column 1) or all the observations data (column 2) from Oct 1980 to Sep 2004.

Method	Train on cold seasons	Train on all
ANN	2.138±0.093	1.794±0.032
RNN	2.104±0.080	1.789±0.034
RNN$^{\text{ptr}}$	1.893±0.085	1.555±0.021
RGrN	1.939±0.062	1.539±0.024
PGRGrN$^{\text{ptr}}$	1.853±0.034	1.530±0.014
PGRGrN$^{\text{ptr,ctr}}$	1.744±0.053	1.416±0.019

5.6 Generalization test Here we test model generazability for temperature modeling. In particular, we test the model performance on predicting temperatures in a season that was not included in training data. 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 4. We also include the testing performance of the model trained using all the observations from first 24 years as a baseline in the second column of Table. 4. Note that PGRGrN$^{\text{ptr}}$ and PGRGrN$^{\text{ptr,ctr}}$ performs much better than other methods because the incorporation of physical knowledge and minimizing the contrastive loss force the model to learn generalizable patterns for each segment. It can be seen that ANN and RNN have large errors because they do not fully capture spatial and temporal processes that drive target physical phenomena. We include the generalization test for streamflow using training data from segments of different streamflow ranges and include the results in the full version of the paper [6].

6 Conclusion

In this paper, we propose a novel method PGRGrN for modeling interacting segments in river networks. We leverage the prior physical knowledge about segment-to-segment interactions embedded in physics-based models

to enhance the learning of the proposed ML model. Moreover, we improve the loss function to optimize both the overall performance over the river network and the local performance on each individual river segment. We have demonstrated the superiority of the proposed method in handling the scarcity of labeled data and in generalizing to unseen scenarios. The proposed method can also be adjusted to model other complex systems which involve interacting processes. For example, this method could be potentially used for material discovery, biological research and quantum chemistry to capture interactions between different atoms or molecules.

While our method performs much better than existing models, it remains limited in precisely predicting special segments (e.g., segments with extremely low streamflows). To advance understanding, future ML modeling efforts may consider uncertainty of the global ML model and determine whether ML should be used to replace physics-based models in different situations.

7 Acknowledgments

This research was supported by the NSF award 1934721. Any use of trade, firm, or product names is for descriptive purposes only and does not imply endorsement by the U.S. Government.

References

[1] Brandon Anderson et al. Cormorant: Covariant molecular neural networks. In *NeurIPS*, 2019.

[2] Jinyin Chen et al. Gc-lstm: Graph convolution embedded lstm for dynamic link prediction. *arXiv preprint arXiv:1812.04206*, 2018.

[3] D Graham-Rowe et al. Big data: science in the petabyte era. *Nature*, 2008.

[4] Will Hamilton et al. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

[5] Xiaowei Jia et al. Physics guided rnns for modeling dynamical systems: A case study in simulating lake temperature profiles. In *SDM*, 2019.

[6] Xiaowei Jia et al. Physics-guided recurrent graph networks for predicting flow and temperature in river networks. *arXiv preprint arXiv:2009.12575*, 2020.

[7] Xiaowei Jia, Jared Willard, Anuj Karpatne, Jordan Read, Jacob Zwart, Michael Steinbach, and Vipin Kumar. Physics guided rnns for modeling dynamical systems: A case study in simulating lake temperature profiles. In *SDM*, 2019.

[8] Xiaowei Jia, Handong Zhao, Zhe Lin, Ajinkya Kale, and Vipin Kumar. Personalized image retrieval with sparse graph representation learning. In *SIGKDD*, 2020.

[9] TO Jonathan, AM Gerald, B Sandrine, et al. Special online collection: dealing with data. *Science*, 2011.

[10] Jacob H LaFontaine et al. Application of the precipitation-runoff modeling system (prms) in the apalachicola-chattahoochee-flint river basin in the southeastern united states. *USGS*, 2013.

[11] Dehao Liu and Yan Wang. Multi-fidelity physics-constrained neural network and its application in materials modeling. *Journal of Mechanical Design*, 2019.

[12] Steven L Markstrom et al. Prms-iv, the precipitation-runoff modeling system, version 4. *USGS*, 2015.

[13] Zach Moshe et al. Hydronets: Leveraging river structure for hydrologic modeling. 2020.

[14] Junyoung Park and Jinkyoo Park. Physics-induced graph neural network: An application to wind-farm power estimation. *Energy*, 2019.

[15] Yanlin Qi, Qi Li, Hamed Karimian, and Di Liu. A hybrid model for spatiotemporal forecasting of pm2. 5 based on graph convolutional neural network and long short-term memory. *STOTEN*, 2019.

[16] Arun Ravindranath et al. An environmental perspective on the water management policies of the upper delaware river basin. *Water Policy*, 2016.

[17] Emily K Read et al. Water quality data for national-scale aquatic research: The water quality portal. *Water Resources Research*, 2017.

[18] Jordan S Read et al. Process-guided deep learning predictions of lake water temperature. *WRR*, 2019.

[19] R Steven Regan et al. Description of the national hydrologic model for use with the precipitation-runoff modeling system. Technical report, USGS, 2018.

[20] Luana Ruiz et al. Gated graph recurrent neural networks. *arXiv preprint arXiv:2002.01038*, 2020.

[21] Michael J Sanders et al. Documentation of a daily mean stream temperature module—an enhancement to the precipitation-runoff modeling system. Technical report, US Geological Survey, 2017.

[22] Terrence J Sejnowski et al. Putting big data to good use in neuroscience. *Nature neuroscience*, 2014.

[23] US Geological Survey. National water information system data available on the world wide web (usgs water data for the nation). 2016.

[24] Rui Wang, Karthik Kashinath, Mustafa Mustafa, Adrian Albert, and Rose Yu. Towards physics-informed deep learning for turbulent flow prediction. In *SIGKDD*, 2020.

[25] Jared Willard, Xiaowei Jia, Shaoming Xu, Michael Steinbach, and Vipin Kumar. Integrating physics-based modeling with machine learning: A survey. *arXiv preprint arXiv:2003.04919*, 2020.

[26] Tanja N Williamson et al. Summary of hydrologic modeling for the delaware river basin using the water availability tool for environmental resources (water). Technical report, US Geological Survey, 2015.

[27] Bing Yu et al. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875*, 2017.

[28] Di Zhu et al. Understanding place characteristics in geographic contexts through graph convolutional neural networks. *AAG*, 2020.