Physics-Guided Machine Learning from Simulation Data: An Application in Modeling Lake and River Systems

Xiaowei Jia[1], Yiqun Xie[2], Sheng Li[3], Shengyu Chen[1], Jacob Zwart[4], Jeffrey Sadler[4], Alison Appling[4], Samantha Oliver[4], Jordan Read[4]
[1]University of Pittsburgh, [2]University of Maryland, [3]University of Georgia, [4]U.S. Geological Survey
[1]{xiaowei,shc160}@pitt.edu, [2]xie@umd.edu, [3]sheng.li@uga.edu, [4]{jzwart,jsadler,aappling,soliver,jread}@usgs.gov

***Abstract*—This paper proposes a new physics-guided machine learning approach that incorporates the scientific knowledge in physics-based models into machine learning models. Physics-based models are widely used to study dynamical systems in a variety of scientific and engineering problems. Although they are built based on general physical laws that govern the relations from input to output variables, these models often produce biased simulations due to inaccurate parameterizations or approximations used to represent the true physics. In this paper, we aim to build a new data-driven framework to monitor dynamical systems by extracting general scientific knowledge embodied in simulation data generated by the physics-based model. To handle the bias in simulation data caused by imperfect parameterization, we propose to extract general physical relations jointly from multiple sets of simulations generated by a physics-based model under different physical parameters. In particular, we develop a spatio-temporal network architecture that uses its gating variables to capture the variation of physical parameters. We initialize this model using a pre-training strategy that helps discover common physical patterns shared by different sets of simulation data. Then we fine-tune it using limited observation data via a contrastive learning process. By leveraging the complementary strength of machine learning and domain knowledge, our method has been shown to produce accurate predictions, use less training samples and generalize to out-of-sample scenarios. We further show that the method can provide insights about the variation of physical parameters over space and time in two domain applications: predicting temperature in streams and predicting temperature in lakes.**

I. Introduction

Physics-based models have been widely used to study scientific and engineering systems in domains such as hydrology [1], climate science [2], and material science [3]. Even though physics-based models are based on known physical laws that govern relations between input and output variables, most physics-based models are necessarily approximations of reality due to incomplete knowledge of certain processes or excessive complexity in modeling these processes. For example, existing physics-based approaches for predicting river networks simulate target variables (e.g., streamflow and temperature) based on general physical relations such as energy and mass conservation. However, the model predictions still rely on parameterizations of land surface and subsurface processes based on soil and surficial geologic classification along with topography, land cover, and climate input. Hence, such models have limits of prediction performance even after parameter calibration due to constraints from the model structure and simplified representation (e.g., by assuming physical parameters are static in space and/or time). Furthermore, calibration of physics-based models often requires extensive expert knowledge of the system and can be extremely time intensive due to the complex (sometimes chaotic) dynamics in the system and uncertainty in observations, initial conditions, and model error. The limitations of physics-based models cut across disciplinary boundaries and are well known in the scientific community.

Machine learning (ML) models, given their tremendous success in several commercial applications (e.g., computer vision, and natural language processing), are increasingly being considered as promising alternatives to physics-based models by the scientific community. Early results in isolated and relatively simple scenarios have been promising, and the expectations are rising for this paradigm to accelerate scientific discovery and help address some of the biggest challenges that are facing humanity such as food and water security. However, direct application of "black-box" ML models has had limited success in some scientific domains, given that the data available for many scientific problems are far smaller than what are needed to effectively train advanced ML models. Moreover, in the absence of adequate information about the physical mechanisms of real-world processes, ML approaches are prone to false discoveries of patterns that cannot generalize to out-of-sample scenarios.

In recent years, there has been a great interest in developing new approaches that integrate scientific knowledge into ML models (e.g., see a recent survey [4]). From the earliest residual modeling approaches, where an ML model is trained to predict the discrepancy between observations and simulations made by a physics-based model [5]–[7], researchers have now shifted their focus to new methods that leverage knowledge of physics to guide the learning process of ML models. This includes new loss functions to preserve consistency with established physical laws [8]–[13], new model initialization methods by transferring physics [8], [11], [14], [15], and new model architectures by encoding specific physical relationships [16]–[20]. In particular, previous work has shown that ML models can learn more generalizable patterns from limited observation data by transferring knowledge from simulations produced by physics-based models [8], [11], [21].

However, there are two major challenges faced by these methods when applied to real-world problems. First, these methods can require access to a physics-based model that well simulates the target system, which is often not feasible given the high cost of calibration and/or parameterization and the prediction errors that persist even after these procedures. The parameters of a physics-based model modulate the translation of input drivers to predictions of target variables. For example, given the same meteorological drivers for a lake system, the physics-based model can simulate different water temperature profiles by varying the parameter of water clarity, which controls how much light can penetrate into the water column and warm deeper waters. When transferring physics knowledge to an ML model, existing methods are likely to be affected by the inherent bias due to uncertainties remaining after parameterization thereby limiting the model's potential to extract general physical knowledge from the physics-based model. Second, existing methods commonly use physical simulations in a separate training stage [8] or for feature augmentation [10] without fully exploring the relationships between simulations and true observations. Learning such relationships has the potential to identify simulation biases and variations of physical parameters over space and time.

In this paper, we propose a new framework, **SIM**ulation-guided **L**ea**R**ning (SIMLR), which extracts the general physical knowledge jointly from multiple sets of physical simulations with imperfect parameterizations. We also explore the relationship between observation data and simulation data and identify parameter settings that produce the most accurate predictions over different locations and time periods. In particular, we first build a spatial-temporal network (STN) architecture to represent the spatial and temporal relationships in the dynamical system. Given that most physical parameters determine specific conditions that control how the system states react to external changes, we represent such conditioning factors using a set of gating variables in the ML architecture. The gating variables are used to filter the information from the current time step, previous time steps, and the spatial neighborhood. The filtered information is combined to update the state of the ML model. Then we propose a new pre-training strategy that leverages general physical patterns from different sets of simulation data to inform the initialization of the STN model. The idea is that this initialized model can be easily adjusted to fit each set of simulation data by slightly altering gating variables. After the initialization, we further refine the model using true observations via a contrastive learning process. The contrastive learning process aims to explore the similarity of relations between observations and different sets of simulation data and further transfer the knowledge from specific simulations that are closer to the observed reality.

We evaluate the performance in two societally relevant applications, modeling water temperature in a lake system and water temperature in river networks. Although predicting the same variable, these two applications have distinct spatiotemporal drivers of water temperature and focal parameters for physics-based calibration. We demonstrate the effectiveness of model initialization using general physical knowledge and show that our method can achieve good predictive performance even with very sparse observation data. We also analyze the similarity relationships learned from the contrastive loss and provide scientific interpretability. Our method has shown promise in discovering variations of physical parameters across space and time while traditional physics-based model can often take fixed parameter values. Moreover, we show that our method under the guidance of general physical relationships can better generalize to different scenarios. We have released our code and the river dataset in a temporary Google Drive link[1].

II. Related Work

Recently, we have seen an increasing interest of integrating physics into ML models for improving the predictive performance and generalizability in addressing scientific problems. This is commonly conducted in several ways, including developing new model architectures [22], [23], and applying additional loss functions [8], [11]. However, scientific dynamical system can be driven by complex physical processes that are difficult to explicitly include in the loss function or model structure. To overcome this limitation, this paper is focused on transferring rich physical knowledge from physics-based models to ML models using the simulation data.

The most common approach for using simulation data is residual modeling, where an ML model is trained to make corrections to physical model outputs. Most of the work on residual modeling going back several decades has used plain regression models [5], [6], although some recent works [7] have used Long-Short Term Memory (LSTM). Recently, Karpatne et al. introduced a hybrid ML and physics model in which the output of a physics model is fed into an ML model as additional input [10].

More recently, simulation data have been used for pre-training ML models with the aim of improving the initialization of ML models. Intuitively, if physical or other contextual knowledge can be used to help inform the initialization of the weights, model training can be accelerated or improved while also requiring less training samples [8]. One way to inform the initialization to assist in model training and escaping local minima is to use an ML technique known as *transfer learning*. In transfer learning, a model can be *pre-trained* on physics-based model's simulated data prior to being fine-tuned with limited training data to fit the desired task. The pre-trained model serves as an informed initial state that ideally is closer to the desired parameters for the desired task than random initialization. For example, Jia et al. used this strategy in the context of modeling lake temperature dynamics [8]. They pre-trained their Physics-Guided Recurrent Neural Network (PGRNN) models for lake temperature modeling on simulated data generated from a physics-based model and fine tuned the network with small observed data. They showed that pre-training, even using data from a physical model with an

[1] https://drive.google.com/open?id=12l9RhiaGZqwZEp3URFY8GrQ4VAMpRtvy

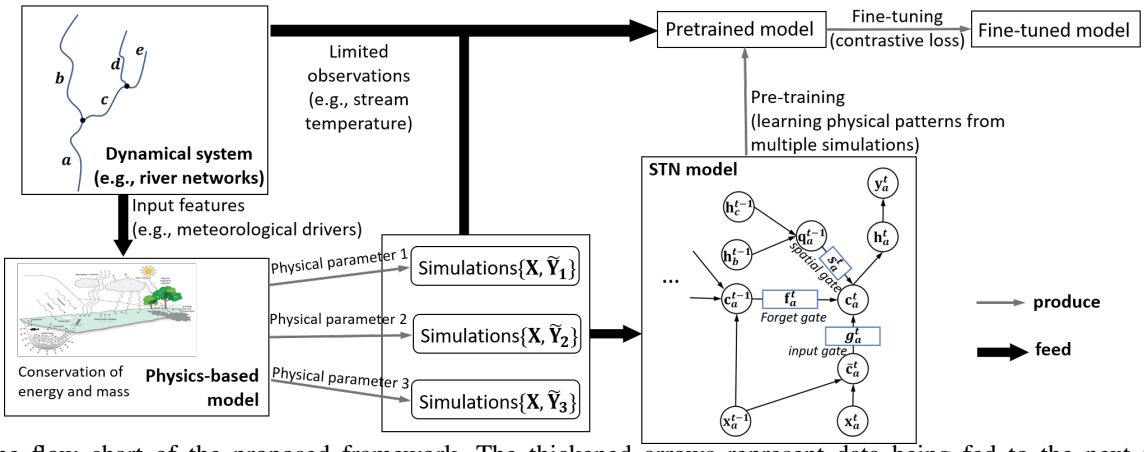


Fig. 1: The flow chart of the proposed framework. The thickened arrows represent data being fed to the next model. For example, we feed limited observations to the obtained pre-trained model and fine-tune it to the final model.

incorrect set of parameters, can still reduce the training data needed for a quality model. In addition, Read et al. [11] demonstrated that such models are able to generalize better to unseen scenarios than pure physics-based models. Such pre-training methods have also been explored in computational biophysics [15], chemistry [24], and climate science [25].

However, the benefits of pre-training are limited by the quality of the physics-based simulations, which in turn is limited by the use of imperfect parameters. In this paper, we will leverage multiple sets of simulation data produced using several default physical parameters and develop a new method to extract general physical relationships from simulation data. Our work is also relevant to existing work on learning from multiple noisy annotators [26]. The difference is that we aim to use physical simulation data only to initialize the ML model and then use true observations to adjust the model and identify the gaps and similarities with different parameter settings. Also, the bias introduced by simulation data are not randomly generated but caused by the deviation of physical parameters used in the governing equations.

III. Problem Definition

Our objective is to predict target variables for each location $i \in \{1, ..., N\}$, and on each date $t \in \{1, ..., T\}$, given input physical variables that drive the dynamics of the physical system. Specifically, we use $\mathbf{x}_i^t$ to represent input features for each location i on a specific date t, and we aim to predict the corresponding target variables $\mathbf{y}_i^t$. In aquatic systems, each location can be a specific depth layer in a lake, or a different segment in a river network. To fully capture the spatial dependencies amongst different locations, we also introduce the neighborhood $\mathcal{N}(i)$ and the adjacency matrix $\mathbf{A}$, where $\mathcal{N}(i)$ represents a set of locations that are spatial neighbors of the location i and $\mathbf{A}_{ij}$ represents the adjacency level between each pair of locations i and j (more details in Section V-A).

In real-world scientific applications, the available observations (i.e., labels) $\mathbf{y}_i^t$ are often sparse due to the substantial manual labor required to collect the observation data. This makes it challenging to directly train an ML model using the limited observation data. To address this issue, we leverage the simulation data generated by the physics-based model to guide the learning of ML model. The physics-based model takes the input features $\{\mathbf{x}_i^t\}$ and simulates target variables based on known physical theory and a set of physical parameters. For example, in the context of modeling lake systems, the physics-based model simulates water temperature based on energy conservation law, and also requires physical parameters such as lake geometry and water clarity, which directly affect the change of temperature in response to external energy fluxes. In this work, we consider learning from ensemble simulation data. In particular, we are provided with K different sets of simulation data generated by using K different parameter values (that are commonly used by domain scientists) for a specific physical parameter (e.g., water clarity). We represent each set of simulated target variables as $\tilde{\mathbf{Y}}_k = \{\tilde{y}_{i,k}^t\}$ for each location i on each date t.

To avoid ambiguity, we use "physical parameters" in this paper to refer to parameters in the physics-based model and otherwise "parameters" refer to parameters in the ML model.

IV. Method

This section presents our proposed SIMLR framework (Fig. 1). We will first describe the architecture of the spatial temporal network (STN) and then discuss how to extract general physical knowledge from simulation data and encode the knowledge using an initialized STN. Finally, we will describe how to refine the STN by exploring the relationships between observation data and simulation data.

A. Spatio-temporal model for scientific systems with different physical parameters

Physics-based models, e.g., General Lake Model [1] and PRMS-SNTemp [27], commonly use parameterized governing equations to represent physical processes that underlie the dynamical system. The physical parameters used in these models have physical definitions and often cannot be easily measured. These physical parameters determine how the model states change in response to external inputs. For example, given the same amount of solar radiation, a lake with higher water clarity

will have a larger increase of water temperature at lower depths compared with a darker lake because more light can penetrate to the lower depths of the water column.

To represent these relationships and also to facilitate learning from multiple sets of simulation data, we build the STN model architecture. The STN model is essentially an extension of the Long-Short Term Memory (LSTM) structure. It uses a set of gating variables to control the influence from different sources, including the inputs at the current time step, model states from the previous time step, and the effect from spatial neighbors.

Similar to the standard LSTM, the STN preserves a model state $\mathbf{c}_i^t$ for each location i at time t, which serves as a memory and will be updated over time (see Fig. 1). It also outputs a hidden representation $\mathbf{h}_i^t$ at every time step, which encodes the information about the location i and its spatial and temporal context. Now we describe the details of computing model states and hidden representation. First, we generate a candidate state $\bar{\mathbf{c}}_i^t$ by combining $\mathbf{x}_i^t$ and $\mathbf{h}_i^{t-1}$ using a $\tanh(\cdot)$ function, as follows:

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

where $\{\mathbf{W}_c^h, \mathbf{W}_c^x, \mathbf{b}_c\}$ are model parameters.

For each location i, we generate hidden variables $\mathbf{q}_i^{t-1}$ by aggregating the hidden representation from its neighbors based on their adjacency level with the location i, as follows:

$$\mathbf{q}_i^{t-1} = \tanh(\mathbf{W}_q \sum_{j \in \mathcal{N}(i)} \mathbf{A}_{ji} \mathbf{h}_j^{t-1} + \mathbf{b}_q), \tag{2}$$

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

Then we generate three sets of gating variables: forget gating variables $\mathbf{f}_i^t$, input gating variables $\mathbf{g}_i^t$, and spatial gating variables $\mathbf{s}_i^t$. These gating variables are used to filter the information passed from the previous time step, the current time step, and the spatial neighborhood, respectively. Formally, these gating variables are computed using sigmoid function $\sigma(\cdot)$ as follows:

$$\begin{aligned} \mathbf{f}_i^t &= \sigma(\mathbf{W}_f^h \mathbf{h}_i^{t-1} + \mathbf{W}_f^x \mathbf{x}_i^t + \mathbf{b}_f), \\ \mathbf{g}_i^t &= \sigma(\mathbf{W}_g^h \mathbf{h}_i^{t-1} + \mathbf{W}_g^x \mathbf{x}_i^t + \mathbf{b}_g), \\ \mathbf{s}_i^t &= \sigma(\mathbf{W}_s^q \mathbf{q}_i^{t-1} + \mathbf{W}_s^x \mathbf{x}_i^t + \mathbf{b}_s), \end{aligned} \tag{3}$$

where Θ=$\{\mathbf{W}_f^h,\ \mathbf{W}_f^x,\ \mathbf{W}_g^h,\ \mathbf{W}_g^x,\ \mathbf{W}_s^q,\ \mathbf{W}_s^x,\ ,\mathbf{b}_f,\ \mathbf{b}_g,\ \mathbf{b}_s\}$ are model parameters.

Once we obtain the gating variables, we can use them to filter the information from the previous time ($\mathbf{c}_i^{t-1}$), the current time step ($\bar{\mathbf{c}}_i^t$), and the spatial neighborhood ($\mathbf{q}_i^{t-1}$) via element-wise product $\odot$, and combine the filtered information to compute the model state at time t. This can be expressed as follows:

$$\mathbf{c}_i^t = \mathbf{f}_i^t \odot \mathbf{c}_i^{t-1} + \mathbf{g}_i^t \odot \bar{\mathbf{c}}_i^t + \mathbf{s}_i^t \odot \mathbf{q}_i^{t-1}, \tag{4}$$

According to this equation, the change of model states given the inputs over space and time is conditioned on the gating variables $\mathbf{f}_i^t$, $\mathbf{g}_i^t$, and $\mathbf{s}_i^t$. This is analogous to the evolution of a dynamical system, which is conditioned on specific physical parameters. Hence, we can use these gating variables to encode variations in physical parameters. By varying parameters Θ, the STN model can represent the dynamical system using different physical parameters.

After obtaining the model state $\mathbf{c}_i^t$, we generate the output gating variables $\mathbf{o}_i^t$ and use them to filter the model state to compute the hidden representation $\mathbf{h}^t$, as follows:

$$\begin{aligned} \mathbf{o}_i^t &= \sigma(\mathbf{W}_o^h \mathbf{h}_i^{t-1} + \mathbf{W}_o^x \mathbf{x}_i^t + \mathbf{b}_o), \\ \mathbf{h}_i^t &= \mathbf{o}_i^t \odot \tanh(\mathbf{c}_i^t). \end{aligned} \tag{5}$$

Finally, we generate predicted target variables $\hat{\mathbf{y}}_i^t$ using a linear transformation, as follows:

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

The loss function of STN is defined using true observations $\mathbf{Y} = \{\mathbf{y}_i^t\}$ that are available at certain time steps and certain locations, as follows:

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

The model has two sets of parameters. The model parameters Θ=$\{\mathbf{W}_f^h$,$\mathbf{W}_f^x$,$\mathbf{W}_g^h$,$\mathbf{W}_g^x$,$\mathbf{W}_s^q$,$\mathbf{W}_s^x$,$\mathbf{b}_f$,$\mathbf{b}_g,\ \mathbf{b}_s\}$ can capture the difference in physical processes represented using different physical parameters and thus they are specific to each set of simulation data. Later in Section IV-B, we will create different copies of these parameters Θ for different sets of simulation data. We represent other parameters using Φ=$\{\mathbf{W}_c^h$,$\mathbf{W}_c^x$,$\mathbf{W}_q$,$\mathbf{W}_o^h$,$\mathbf{W}_o^x$,$\mathbf{W}_y$,$\mathbf{b}_c$,$\mathbf{b}_q$,$\mathbf{b}_o$,$\mathbf{b}_y\}$. The parameters Φ are shared across different sets of simulation data.

B. Pre-training: Extract knowledge shared across simulations

Although different sets of simulation data are generated using different physical parameters (some are close to reality and some are different), the simulations still share some common patterns of general physical relationships embodied in the physics-based model. At the same time, each set of simulation data also shows patterns specific to its own parameter set. Here we introduce a pre-training strategy for the STN model, which aims to estimate the initial value of Φ_0 and Θ_0 by extracting the general physical relationships. The goal is that the initial value of Θ_0 can later be quickly adjusted to fit different simulation settings while keeping the Φ_0 parameters the same across different simulations.

Our method is inspired by the Model Agnostic Meta Learning (MAML) [28]. The original MAML is designed for learning a model that can be easily adapted to a new task using limited samples. Although the objective of MAML is different from our task, we use the similar method to construct to learning objective so that the pre-trained model can be easily adapted to each set of simulations after slight fine-tuning. In particular, we divide each set of simulation data $\{\mathbf{X}, \tilde{\mathbf{Y}}_k\}$ into a separate training set $\{\mathbf{X}^{\text{tr}}, \tilde{\mathbf{Y}}_k^{\text{tr}}\}$ and validation set $\{\mathbf{X}^{\text{val}}, \tilde{\mathbf{Y}}_k^{\text{val}}\}$. For the k^{th} simulation, we update Θ_0 using its training data

$\{\mathbf{X}^{\text{tr}}, \tilde{\mathbf{Y}}_k^{\text{tr}}\}$ with a learning rate α while not changing the other parameters Φ_0, as follows:

$$\Theta_k = \Theta_0 - \alpha \nabla_\Theta \mathcal{L}_{\text{STN}}(f_{\text{STN}}(\mathbf{X}^{\text{tr}}; \Theta_0, \Phi_0), \tilde{\mathbf{Y}}_k^{\text{tr}}), \tag{8}$$

where $f_{\text{STN}}(\cdot)$ represents the mapping relation from input features to target variables defined by the STN (Eqs. (1)-(6)), Θ_k represents the simulation-specific parameters for gating variables. Here Eq. (8) just shows the adjustment of Θ using a one-step gradient descent. This can be easily extended to multiple update steps, which allows more flexible adjustment of Θ to fit each simulation. In our implementation, we found the update with no more than five steps can already lead to good performance. More discussions on the selection of the number of update steps are in Section V-F.

Once we gather the Θ_k that are specific to each set of simulation data, we define the pre-training loss using the k^{th} simulation's validation set, as follows:

$$\mathcal{L}_{\text{pre}} = \sum_k \mathcal{L}_{\text{STN}}(f_{\text{STN}}(\mathbf{X}^{\text{val}}; \Theta_k, \Phi_0), \tilde{\mathbf{Y}}_k^{\text{val}})/K. \tag{9}$$

During the pre-training process, we minimize the loss $\mathcal{L}_{\text{pre}}$ with respect to the initial parameters Φ_0 and Θ_0. These estimated parameters are used to initialize the STN model, which will then be fine-tuned using true observations. We also collect the obtained intermediate parameters Θ_k values for k=1 to K, which encode the information specific to each set of simulation data. These simulation-specific parameters will also be used for model fine-tuning.

C. Fine-tuning: Contrastive Learning

After initializing the parameters Φ and Θ using the values Φ_0 and Θ_0, we refine these parameters using the available observed target variables. We will also further leverage the knowledge extracted from simulation data by exploring the relationships between the observation data and different sets of simulation data. Because each set of simulation data can be considered as an ideal version of real data under certain physical parameter settings, for each observed sample, it is possible to find its matched counterpart in the set of simulation data. Here we will introduce a new loss function for fine-tuning that captures this relationship.

In particular, we first generate hidden representation for each location i at time t, as follows:

$$\mathbf{h}_i^t = g_{\text{STN}}(\mathbf{x}_i^{1:t}; \Phi, \Theta), \tag{10}$$

where the function $g_{\text{STN}}(\cdot)$ represents the function defined by the STN model to extract hidden representation $\mathbf{h}_i^t$ from input data by following Eqs. (1)-(5).

Using the collected Θ_k for k^{th} simulation setting, we also generate the corresponding hidden representation $\tilde{\mathbf{h}}_{i,k}^t$. It is noteworthy that these hidden representations are generated using gating variables that are specific to each set of simulation data. This process can be expressed as follows:

$$\tilde{\mathbf{h}}_{i,k}^t = g_{\text{STN}}(\mathbf{x}_i^{1:t}; \Phi, \Theta_k). \tag{11}$$

Here the obtained $\tilde{\mathbf{h}}_{i,k}^t$ encodes the spatial and temporal patterns under the specific parameter settings used to generate k^{th} set of simulation data.

After gathering these hidden representations, we define a similarity mapping $\mathbf{h}_i^t \rightarrow \tilde{\mathbf{h}}_{i,k}^t$ for each k=1 to K using the inner product of these two vectors. Once we obtain the similarity values for all sets of simulation data (i.e., k=1 to K), we normalize the obtained similarity values and convert them into a distribution $\mathcal{Q}(\mathbf{h}_i^t \rightarrow \tilde{\mathbf{h}}_{i,k}^t)$ via a softmax function. More formally, this can be expressed as follows:

$$\mathcal{Q}(\mathbf{h}_i^t \rightarrow \tilde{\mathbf{h}}_{i,k}^t) = \frac{\exp(\mathbf{h}_i^t \cdot \tilde{\mathbf{h}}_{i,k}^t)}{\sum_k' \exp(\mathbf{h}_i^t \cdot \tilde{\mathbf{h}}_{i,k'}^t)} \tag{12}$$

We aim to ensure that the patterns extracted from observation data are similar to certain sets of simulation data that use more accurate physical parameters, but are different from other simulation settings. Specifically, we define a contrastive loss based on the entropy of the similarity probability, as follows:

$$\mathcal{L}_{\text{ctr}} = -\sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{k=1}^{K} \mathcal{Q}(\mathbf{h}_i^t \rightarrow \tilde{\mathbf{h}}_{i,k}^t) \log \mathcal{Q}(\mathbf{h}_i^t \rightarrow \tilde{\mathbf{h}}_{i,k}^t)/NT. \tag{13}$$

As a side benefit, this method also enables the discovery of more accurate physical parameters for each location at each time step. Compared to standard physics-based model which commonly assumes static parameters in space and/or time, the proposed method has a better chance at capturing the variability of underlying physical processes.

Combining the contrastive loss and the standard supervised loss (Eq. (7)) using the observation data, we get the final fine-tuning loss, as follows:

$$\mathcal{L}_{\text{ft}} = \mathcal{L}_{\text{STN}} + \lambda \mathcal{L}_{\text{ctr}}, \tag{14}$$

where λ is a hyper-parameter.

V. Experiments

A. Datasets

We apply our SIMLR model to two different environmental modeling challenges, predicting depth-specific water temperatures in a lake and predicting water temperatures for segments in a stream network. Both problems require accurately accounting for variations across space (e.g. lake depth, stream reaches) and time (e.g. daily weather patterns, seasonal climate). The problems differ substantially in the nature of the spatial relationships: lakes exchange heat mostly at the water surface, and various processes operating at different timescales act to distribute heat through the lake vertically. In contrast, because streams are well-mixed, the entire stream warms or cools primarily due to energy exchange with the atmosphere and other water sources (e.g., groundwater). In-stream heat almost exclusively flows downhill along the river network and at the same pace as the water. The focal parameters are different in each problem and have very different effects: lake geometry affects the degree, timing, and duration of thermal stratification and affects the response of the lake to wind

events, and lake water clarity controls the depths at which incoming solar energy is absorbed, while the groundwater residence time in stream reaches controls the temperature of incoming groundwater.

D1: Predicting water temperature in Lake Mendota. This dataset was collected from Lake Mendota in Wisconsin, USA [11]. This lake system is reasonably large ($\sim$40 km^2 in area) and the lake has a maximum depth of 25 meters. It also exhibits large changes in water temperatures in response to seasonal and sub-seasonal weather patterns and thermally stratifies during the summertime. Observations of lake temperature were collected from North Temperate Lakes Long-Term Ecological Research Program [11].

The input features that describe prevailing meteorological conditions are available on a continuous daily basis from April 02, 1980, to December 30, 2014 (12,690 dates). We used a set of seven input features, including short-wave and long-wave radiation, air temperature, relative humidity, wind speed, frozen and snowing indicators. These were acquired and/or computed from the North American Land Data Assimilation System. Temperature observations vary in their distribution across depths and time, i.e., there are certain days when observations are available only on a few depths or no observations are available.

We use the observed data from April 02, 1980, to October 31, 1991, and the data from June 01, 2003, to December 30, 2014, as training data (in total 8,037 observations). Then we applied the trained model to predict the temperature at different depths for the period from November 01, 1991, to May 31, 2003 (in total 5,121 observations).

D2: Predicting water temperature in Delaware River Basin. The dataset is pulled from U.S. Geological Survey's National Water Information System [29] and the Water Quality Portal [30]. The river segments were defined by the network used for the National Hydrologic Model [31], 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, Delaware.

We use input features at the daily scale from October 01, 1980, to September 30, 2016 (13,149 dates). The input features have 10 dimensions which include precipitation, air temperature, day of year, solar radiation, shade fraction, potential evapotranspiration, and the geometric features of each segment (e.g., elevation, length, slope, and width). Air temperature, precipitation, and solar radiation values were derived from the gridMET dataset [32]. Other input features (e.g., shade fraction, potential evapotranspiration) are difficult to measure frequently, and we use values produced by the PRMS-SNTemp model as its internal variables. 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 observed segment ranged from 1 to 9,810 with a total of 51,103 observations across all dates and segments. We use the observed data from October 01, 1980, to September 30, 1992, and the data from October 01, 2004, to September 30, 2016, as training data (in total 34,985 observations). Then we applied the trained model to predict the temperature for the period from October 01, 1992, to September 30, 2004 (in total 16,118 observations).

Simulation data: In D1, we use the physics-based General Lake Model (GLM) [1] to generate different simulation data by varying the lake geometry and water clarity. Specifically, we used "cone," "barrel," and "martini" shapes to define the depth-area parameters in the GLM to generate three sets of simulation data. Then we fix the geometry as "cone" and use three different clarity levels "normal" (Kw=0.45), "dark" (Kw=1.20), and "clear" (Kw=0.25). Water clarity affects the penetration of solar radiation into the deeper water. In D2, we use PRMS-SNTemp [27] to generate different simulations by setting average groundwater residence time (τ) as 10 days, 45 days, and 100 days. A shorter τ means the groundwater quickly moves through the groundwater aquifer and enters the stream at a temperature more similar to recent air temperatures, whereas a longer τ means groundwater temperatures are more seasonally stable. These physical parameter values represent a range of values observed across lake and stream systems, but are not tailored to our target systems.

Goals: In D1, we aim to predict water temperature in each depth of the water column. In D2, we aim to predict water temperature in each river segment. Here we assume the river temperature is the same across depth because rivers tend to be well-mixed and shallower.

Implementation details: We implement STN using Tensorflow and GTX 2080 GPU. All the hidden variables and gating variables have 20 dimensions. We use five update steps for obtaining simulation-specific parameters Θ_k during each epoch of pre-training. The model is pre-trained for 150 epochs with learning rate 0.001 before being fine-tuned for 100 epochs with learning 0.0005. The hyper-parameter λ is set as 0.5.

We generate the adjacency matrix $\mathbf{A}$ based on the inverse relation of the distance between each pair of locations i and j. We represent the distance as $\text{dist}(i, j)$. In D1, we use the distance between different layers across depth. In D2, this represents the stream distance between the endpoints of each pair of river segments and we only consider i as a neighbor of j when i is anywhere upstream from j (so j is affected by water flow from i). We standardize the distance and then compute the adjacency level as $\mathbf{A}_{ij} = 1/(1 + \exp(\text{dist}(i, j)))$ for each pair of locations (i, j).

B. Evaluation of predictive performance

We compare our method against multiple baselines, i.e., the physics-based model (GLM in D1 and PRMS-SNTemp in D2); Recurrent Neural Network (RNN) with the LSTM cell; and the state-of-the-art methods, PGRNN [8] and PGRGrN [33], which have shown success in modeling lake temperature and river temperature, respectively. We also compare to the STN model that is pre-trained using a specific set of simulation data (as proxy labels) and then fine-tuned with observed labels. Such a comparison aims to show the advantage of our proposed pre-training strategy in extracting general physical

TABLE I: Performance (as measured by root mean squared error (RMSE) in degrees Celsius) using simulations with different parameters in modeling lake water temperature (D1) and river water temperature (D2). The first three rows for each dataset represent the simulation data produced by the physics-based model using different parameters. The superscript $p(k)$ means that the model is first pre-trained using simulations generated using parameter k. Here % columns are percent observations used during fine-tuning phase. The +/- values represents the range of values across replicates with random starting weights, and NA's for the +/- values are for models that did not have multiple model runs. The bolded values are the best performing models in each dataset and data sparsity level.

Dataset	Method	0%	0.2%	2%	20%	100%
D1 - geometry	$GLM^{(cone)}$	2.664(±NA)	-	-	-	-
	$GLM^{(barrel)}$	3.791(±NA)	-	-	-	-
	$GLM^{(martini)}$	5.919(±NA)	-	-	-	-
	RNN	-	4.615(±0.173)	2.311(±0.240)	1.531(±0.083)	1.489(±0.091)
	STN	-	3.349(±3.805)	1.848(±1.997)	1.387(±1.737)	1.393(±0.070)
	$PGRNN^{p(cone)}$	2.469(±0.168)	**2.056**(±0.184)	1.595(±0.097)	1.452(±0.113)	1.374(±0.074)
	$STN^{p(cone)}$	**2.289**(±0.175)	2.181 (±0.173)	1.591(±0.107)	1.408(±0.089)	1.368(±0.075)
	$STN^{p(barrel)}$	2.996(±0.102)	2.808(±0.187)	1.642(±0.102)	1.339(±0.084)	1.312(±0.075)
	$STN^{p(martini)}$	5.386(±0.124)	2.955(±0.074)	1.821(±0.071)	1.419(±0.110)	1.402(±0.081)
	STN^{SIMLR}	2.914(±0.116)	2.103(±0.076)	1.634(±0.144)	1.411(±0.086)	1.373(±0.045)
	$STN^{SIMLR\text{-}ctr}$	2.914(±0.116)	2.431(±0.196)	**1.535**(±0.132)	**1.366**(±0.060)	**1.248**(±0.061)
D1 - clarity	$GLM^{(normal)}$	2.664(±NA)	-	-	-	-
	$GLM^{(dark)}$	3.053(±NA)	-	-	-	-
	$GLM^{(clear)}$	1.723(±NA)	-	-	-	-
	RNN	-	4.615(±0.173)	2.311(±0.240)	1.531(±0.083)	1.489(±0.091)
	STN	-	3.349(±3.805)	1.848(±1.997)	1.387(±1.737)	1.393(±0.070)
	$PGRNN^{p(clear)}$	2.518(±0.135)	2.050(±0.120)	1.648(±0.128)	1.399(±0.088)	1.371(±0.076)
	$STN^{p(normal)}$	2.289(±0.175)	2.179 (±0.206)	1.594(±0.100)	1.416(±0.096)	1.377(±0.074)
	$STN^{p(dark)}$	2.582(±0.164)	2.084(±0.195)	1.634(±0.099)	1.421(±0.047)	1.326(±0.031)
	$STN^{p(clear)}$	**2.214**(±0.133)	1.847(±0.205)	1.645(±0.116)	1.408(±0.105)	1.308(±0.056)
	STN^{SIMLR}	2.425(±0.044)	1.817(±0.049)	1.601(±0.035)	1.415(±0.037)	1.372(±0.034)
	$STN^{SIMLR\text{-}ctr}$	2.425(±0.044)	**1.806**(±0.036)	**1.503**(±0.029)	**1.360**(±0.19)	**1.263**(±0.031)
D2 - τ	$PRMS\text{-}SNTemp^{(\tau 10)}$	**2.618**(±NA)	-	-	-	-
	$PRMS\text{-}SNTemp^{(\tau 45)}$	3.558(±NA)	-	-	-	-
	$PRMS\text{-}SNTemp^{(\tau 100)}$	5.840(±NA)	-	-	-	-
	RNN	-	2.867(±0.147)	1.732(±0.083)	1.479(±0.023)	1.445(±0.027)
	STN	-	2.356(±0.135)	1.858(±0.105)	1.427(±0.025)	1.397(±0.030)
	$PGRGrN^{p(\tau 10)}$	2.852(±0.103)	2.362(±0.098)	1.628(±0.063)	1.417(±0.032)	1.396(±0.033)
	$STN^{p(\tau 10)}$	2.738(±0.094)	2.259(±0.123)	1.697(±0.096)	1.405(±0.023)	1.403(±0.022)
	$STN^{p(\tau 45)}$	3.632(±0.084)	2.409(±0.124)	1.874(±0.079)	1.499(±0.050)	1.473(±0.029)
	$STN^{p(\tau 100)}$	5.596(±0.079)	2.480(±0.089)	1.871(±0.092)	1.487(±0.046)	1.457(±0.027)
	STN^{SIMLR}	3.235(±0.045)	**2.009**(±0.130)	1.636(±0.066)	1.416(±0.021)	1.403(±0.014)
	$STN^{SIMLR\text{-}ctr}$	3.235(±0.045)	2.103(±0.079)	**1.618**(±0.058)	**1.392**(±0.018)	**1.362**(±0.021)

patterns jointly from multiple sets of simulations. Additionally, we implement two versions of our proposed method STN^{SIMLR} and $STN^{SIMLR\text{-}ctr}$. They have the same pre-training process but STN^{SIMLR} only uses the supervised loss (Eq. (7)) in fine-tuning while $STN^{SIMLR\text{-}ctr}$ uses the contrastive loss (Eq. (14)).

In Table I, we report the performance of different methods in predicting water temperature in lake systems using different parameters of lake geometry and clarity, as well as the performance in predicting water temperature in river networks using different parameters for average residence time in groundwater flow (i.e., τ). For methods PGRNN and PGRGrN, because they can only learn from one set of simulation data, we show the performance of these methods using the simulation data that produce the best performance ("cone" for lake geometry, "clear" for lake clarity, and "τ=10 days" in river modeling).

We can observe that our method outperforms other methods by a considerable margin in both applications. The improvement from RNN to STN shows the effectiveness of incorporating spatial dependencies in modeling thermodynamic patterns. The physics-based models (i.e., GLM and PRMS-SNTemp) perform poorly because of their inherent model bias due to approximations and imperfect parameterizations. Nevertheless, the proposed pre-training method (i.e., STN^{SIMLR}) can still extract useful physical knowledge from the imperfect simulation data and thus performs better than STN, especially when we are using less training data. The performance is further improved after we use the contrastive loss in fine-tuning (i.e., $STN^{SIMLR\text{-}ctr}$). This is because $STN^{SIMLR\text{-}ctr}$ can better learn from specific sets of simulation data that are closer to the reality.

As we reduce the amount of training data, all of the methods produce larger prediction error. However, we can clearly see that the models that are pre-trained using simulation data (i.e., methods in the second and third blocks of each dataset) have much lower error than non-pre-trained models when we use fewer training samples. These models can learn a better initialized state from a large amount of simulation data (available at every day and every location) and thus require less observation data for fine-tuning. Also, our proposed method generally performs better than existing methods with the pre-

training process (i.e., PGRNN and PGRGrN) given limited observation data (e.g., 0% and 0.2%). The pre-training strategy used in STN jointly learns from multiple sets of simulation data and updates different components of the model (e.g., gating variables and other layers) in a deliberate way to reflect the difference in physical parameters. Hence, compared to PGRNN and PGRGrN, STN has a better chance at capturing general physical knowledge while reducing the effect of imperfect physical settings.

We can also observe that models pre-trained using different sets of simulation data can have very different performance. Specifically, the martini shape is very different from the true shape of Lake Mendota so the model pre-trained with the martini simulations has relatively poor performance. Similarly, the river temperature model $\text{STN}^{p(\tau 100)}$ has worse performance because the residence time for shallow groundwater is thought to be generally less than 100 days for many segments in the Delaware River Basin especially during higher stream flows (Martin Briggs, U.S. Geological Survey, written commun., Feb. 8, 2021). However, these pre-trained models ($\text{STN}^{p(\text{martini})}$ and $\text{STN}^{p(\tau 100)}$) can get much better performance when refined using even a small amount of data (e.g., 2%), and predictions can still be much better than the STN model without pre-training. The methods PGRNN and PGRGrN show similar results since they are also pre-trained using simulations.

Moreover, we can observe that some pre-trained models (e.g., $\text{STN}^{p(\text{cone})}$, $\text{STN}^{p(\text{clear})}$, and $\text{STN}^{p(\tau 10)}$) have better performance than our proposed $\text{STN}^{\text{SIMLR}}$ before fine-tuning (i.e., with 0% data for fine-tuning). However, in practice we may not know the most suitable parameters when training the model. We can see that our method can still get comparable performance even without access to such information. Moreover, our method ($\text{STN}^{\text{SIMLR-ctr}}$) after fine-tuning has better performance than all the baselines by using the contrastive loss to explore the relationship between observations and different sets of simulation data.

C. Pre-training

Here we discuss the effect of pre-training in different scenarios. In Fig. 2 (rows 1-2), we show the predictions made by the pre-trained model using our method (SIMLR) and using each set of simulation data over different depths of Lake Mendota. Pre-trained models are always biased because they are trained from simulations with imperfect parameterization. We can observe that SIMLR predictions are generally in the middle of predictions made by other pre-trained models and also follow the similar temporal patterns. This is because SIMLR extracts general patterns that are shared by all these different sets of simulation data. Besides, SIMLR is able to achieve reasonable accuracy compared with observations even without the awareness of the best parameter setting.

In Fig. 2 (rows 3-4), we show the predictions made by the pre-trained model using our proposed method and the fine-tuned model using 2% data. Although the pre-trained model has bias compared to true observations, it is able to capture many general physical relationships (e.g., seasonal patterns, temperature variation across depths), and thus it can be easily refined to match observations even using just 2% data.

D. Similarity mapping

A goal of this work is to better understand the relation between observations and simulations through the similarity learned from the fine-tuning process (i.e., Eq. (12)). In Fig. 3 (a) we show the similarity with different clarity values in different lake depths (averaged over time). In Fig. 3 (b), we show the similarity with different geometries (cone, barrel, and martini) in different months (averaged over depth). We can see that the model is closer to clear simulations in depths 5-12m but closer to dark simulations in lower depths. As none of the physics-based models provided accurate predictions for all depths, SIMLR revealed unaccounted-for or poorly parameterized processes that could be addressed given these insights, such as introducing a clarity parameter that varies with depth (as is common in the natural environment) or modifying vertical mixing parameters that could alter bottom water warming rates. We can also see that the model is closer to cone simulations in the summer and closer to barrel simulations in the fall and winter, which indicates the fall cooling period and under ice temperatures were better simulated when cooling was slowed by using the barrel lake shape.

For the river modeling, SIMLR detects that most segments are more similar to simulation with groundwater residence time τ=10 days during March-May. A lower τ value indicates the groundwater temperature is more similar to recent air temperatures. Although PRMS-SNTemp encodes a constant value of τ throughout the year, seasonality in groundwater residence times has been confirmed in nearby watersheds, with shallow groundwater (having lower τ) contributing more in the spring than in other seasons [34]. The SIMLR approach transforms a constant physics-based model parameter into a flexibly time-varying parameter that is more consistent with observed temperatures and known processes.

E. Generalization test

We expect our method has better generalizability to different scenarios given its ability to extract and transfer general physical relationships. Generalizability is important for scientific problems because most observation data may be collected from certain periods or locations for which it is easier to deploy sensors. As a strong test of generalizability for modeling lake temperature, we train the model using observations from colder seasons in the training period and then test in the summer time of the testing period. Although real-world temperate data collection procedures more often provide data in summer than in winter, training only on cold seasons is more challenging because Lake Mendota has highly dynamic patterns in summer and also a unique stratification across different layers due to the temperature difference between the surface and the lake bottom.

In Table II, we report the performance in the summer seasons of the test period. We also include the testing performance

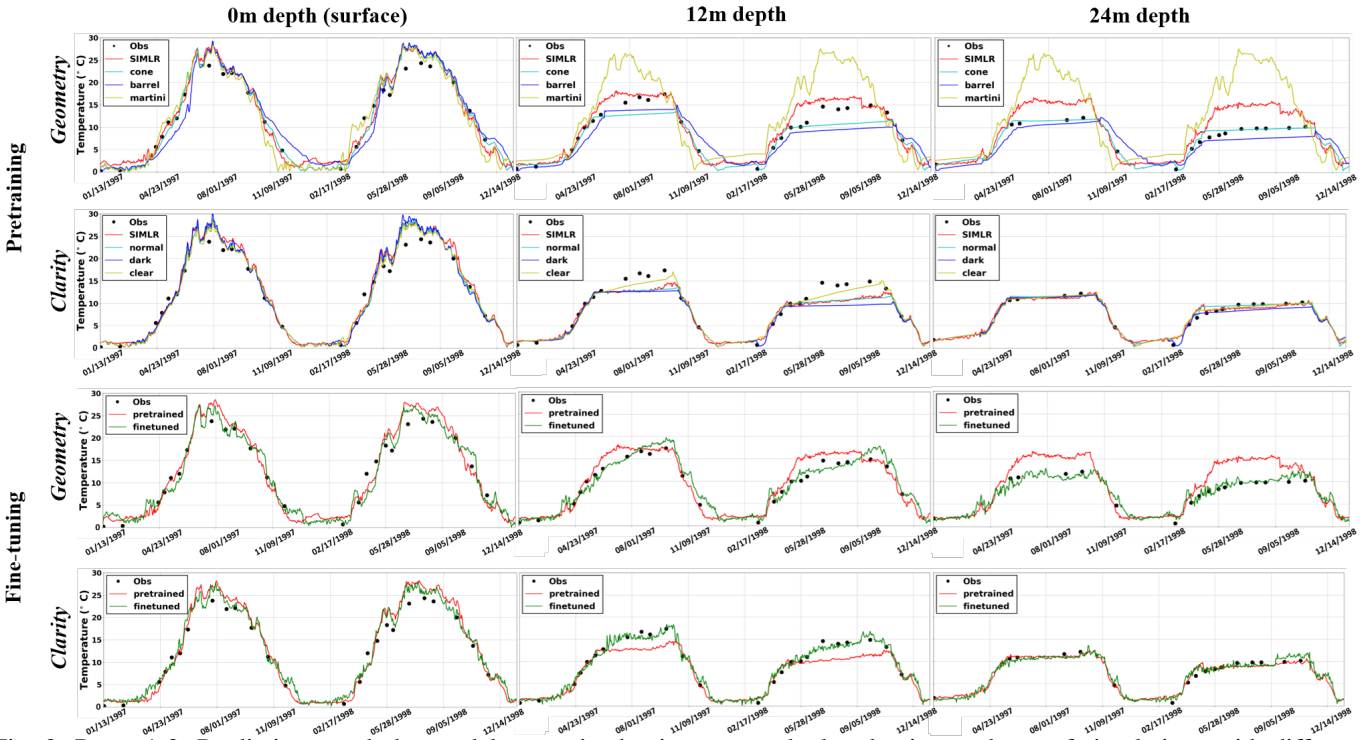
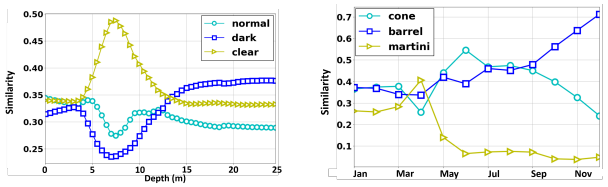


Fig. 2: Rows 1-2: Predictions made by models pretrained using our method and using each set of simulations with different parameters for geometry (first row) and clarity (second row). Rows 3-4: Predictions made by both pretrained and fine-tuned (using 2% observations) models using simulations with different parameters for geometry (third row) and clarity (fourth row). All the predictions are shown at 0m depth, 12m depth, and 24m depth (Columns 1-3).



(a) (b)

Fig. 3: (a) The similarity to different clarity settings in different depths of the Lake Mendota. (b) The similarity to different geometry settings in different months.

TABLE II: Temperature root mean squared error (RMSE) in the summer seasons of the testing period using models trained from spring, fall, and winter in the training period (the first column) and trained using all the data in the training period (the second column).

Method	Train on cold seasons	Train on all the data
$\text{GLM}^{\text{clear}}$	2.037(±NA)	2.037(±NA)
RNN	2.587(±0.245)	1.500(±0.035)
STN	2.180(±0.092)	1.389(±0.045)
$\text{STN}^{\text{SIMLR}}$	1.724(±0.061)	1.402(±0.037)
$\text{STN}^{\text{SIMLR-ctr}}$	1.685(±0.066)	1.325(±0.034)

of the model trained using all observations from the training period as a baseline in the second column of Table II. We can see that all the methods have larger errors when they are trained only on colder seasons. However, our proposed method still yields better performance than other methods. This is because our method learns the general physical relationships that hold in different scenarios.

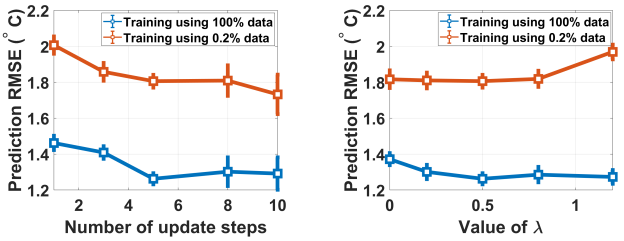
F. Sensitivity test

Here we test the performance using different settings of hyper-parameters. First, we test different number of update steps (iteration of Eq. (8)) and report the predictive performance of lake temperature modeling using simulations with clarity settings. Specifically, we show the performance using 100% data and 0.2% observation data in Fig. 4 (a). We can see that the model has similar performance when we set the number of update steps to be greater or equal to five. Although more update steps can slightly improve the performance, it will bring additional computational cost to the training process.

We also study the effect of hyper-parameter λ (the weight for the contrastive loss) on the performance (see Fig. 4 (b)). As we increase the value from 0, the prediction error decreases gradually. Such decrease is especially obvious when we use 100% data because the contrastive loss can better explore the relationship between observation and simulation data. In particular, if we set a very high value for λ, then the prediction error becomes larger when we use limited data (i.e., 0.2% data). This is because the small data may better fit certain simulations that are not close to reality and thus mislead the training process.

VI. Conclusion

In this paper, we propose a new method for modeling spatial and temporal patterns in dynamical systems while also accommodating uncertainties in physics-based model parameters. We



(a) (b)

Fig. 4: The prediction root mean squared error (RMSE) using (a) different number of update steps and (b) the value of λ.

extract the general physical relationships over space and time to inform the initialization of the ML model. Then we fine-tune the model by exploring the similarity between available observation data and simulation data. We have demonstrated the superiority of the proposed method, which is better equipped to learn from limited observation data, provide insights about the value of physical parameters, and better generalize to unseen scenarios. The proposed method can be widely applied to other dynamical systems simulated by physics-based models with uncertain parameters. For example, physics-based models for hydrologic and climate systems commonly also have bias which stems from the uncertainty in selecting physical parameters.

While our method has shown the improved predictive performance by considering the variations on certain physical parameters, we could certainly explore a larger number of variations in future studies. Moreover, one could explore ways to learn from variations of multiple physical parameters at the same time. We anticipate the knowledge discovered by our method can advance the design of both physics-based models and machine learning models.

VII. Acknowledgement

X.J. and S.C. is support by the USGS Award G21AC10207 and Pitt Momentum Award. Y.X. is supported in part by NSF awards 2105133and 2126474, Google's AI for Social Good Impact Scholars program, and the DRI award at the University of Maryland. 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.

References

[1] M. R. Hipsey *et al.*, "A general lake model (glm 3.0) for linking with high-frequency sensor data from the global lake ecological observatory network (gleon)," 2019.

[2] P. Cox *et al.*, "The impact of new land surface physics on the gcm simulation of climate and climate sensitivity," *Climate Dynamics*, 1999.

[3] M. R. Tonks *et al.*, "Mechanistic materials modeling for nuclear fuel performance," *Annals of nuclear energy*, 2017.

[4] J. Willard, X. Jia, S. Xu, M. Steinbach, and V. Kumar, "Integrating physics-based modeling with machine learning: A survey," *arXiv preprint arXiv:2003.04919*, 2020.

[5] U. Forssell and P. Lindskog, "Combining semi-physical and neural network modeling: An example of its usefulness," *IFAC*, 1997.

[6] T. Xu *et al.*, "Data-driven methods to improve baseflow prediction of a regional groundwater model," *Computers & Geosciences*, 2015.

[7] Z. Y. Wan *et al.*, "Data-assisted reduced-order modeling of extreme events in complex dynamical systems," *PloS one*, 2018.

[8] X. Jia, J. Willard, A. Karpatne, J. S. Read, J. A. Zwart, M. Steinbach, and V. Kumar, "Physics-guided machine learning for scientific discovery: An application in simulating lake temperature profiles," *ACM/IMS Transactions on Data Science*, 2021.

[9] F. Fioretto *et al.*, "Predicting ac optimal power flows: Combining deep learning and lagrangian dual methods," in *AAAI*, 2020.

[10] A. Karpatne, W. Watkins, J. Read, and V. Kumar, "Physics-guided neural networks (pgnn): An application in lake temperature modeling," *arXiv preprint arXiv:1710.11431*, 2017.

[11] 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," *WRR*, 2019.

[12] R. Stewart and S. Ermon, "Label-free supervision of neural networks with physics and domain knowledge." in *AAAI*, 2017.

[13] P. C. Hanson, A. B. Stillman, X. Jia, A. Karpatne, H. A. Dugan, C. C. Carey, J. Stachelek, N. K. Ward, Y. Zhang, J. S. Read *et al.*, "Predicting lake surface water phosphorus dynamics using process-guided machine learning," *Ecological Modelling*, 2020.

[14] D. M. Hurtado *et al.*, "Deep transfer learning in the assessment of the quality of protein models," *arXiv preprint arXiv:1804.06281*, 2018.

[15] M. M. Sultan, H. K. Wayment-Steele, and V. S. Pande, "Transferable neural networks for enhanced sampling of protein dynamics," *Journal of chemical theory and computation*, 2018.

[16] A. Daw, R. Thomas, C. Carey, J. Read, A. Appling, and A. Karpatne, "Physics-guided architecture (pga) of neural networks for quantifying uncertainty in lake temperature modeling," *arXiv:1911.02682*, 2019.

[17] N. Muralidhar *et al.*, "Phynet: Physics guided neural networks for particle drag force prediction in assembly," in *SDM*, 2020.

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

[19] L. Zhang *et al.*, "End-to-end symmetry preserving inter-atomic potential energy model for finite and extended systems," *NeurIPS*, 2018.

[20] K. T. Schütt *et al.*, "Schnet: A continuous-filter convolutional neural network for modeling quantum interactions," *NeurIPS*, 2017.

[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 *SDM*, 2019.

[22] B. Anderson, T. S. Hy, and R. Kondor, "Cormorant: Covariant molecular neural networks," in *NeurIPS*, 2019.

[23] N. Muralidhar *et al.*, "Incorporating prior domain knowledge into deep neural networks," in *IEEE Big Data*. IEEE, 2018.

[24] J. Lu, K. Yao, and F. Gao, "Process similarity and developing new process models through migration," *AIChE journal*, 2009.

[25] Y.-G. Ham, J.-H. Kim, and J.-J. Luo, "Deep learning for multi-year enso forecasts," *Nature*, 2019.

[26] Y. Li, J. Gao, C. Meng, Q. Li, L. Su, B. Zhao, W. Fan, and J. Han, "A survey on truth discovery," *ACM Sigkdd Explorations Newsletter*, 2016.

[27] S. L. Markstrom *et al.*, "Prms-iv, the precipitation-runoff modeling system, version 4," *USGS Techniques and Methods*, no. 6-B7, 2015.

[28] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," *arXiv preprint:1703.03400*, 2017.

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

[30] 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.

[31] 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.

[32] "gridmet - climatology lab," http://www.climatologylab.org/gridmet.html.

[33] X. Jia, J. Zwart, J. Sadler, A. Appling, S. Oliver, S. Markstrom, J. Willard, S. Xu, M. Steinbach, J. Read *et al.*, "Physics-guided recurrent graph model for predicting flow and temperature in river networks," in *SDM*. SIAM, 2021.

[34] D. A. Burns, P. S. Murdoch, G. B. Lawrence, and R. L. Michel, "Effect of groundwater springs on no3- concentrations during summer in catskill mountain streams," *WRR*, 1998.