Meta-Transfer Learning: An application to Streamflow modeling in River-streams

Rahul Ghosh
University of Minnesota
ghosh128@umn.edu

Bangyan Li
University of Pittsburgh
bal116@pitt.edu

Kshitij Tayal
University of Minnesota
tayal@umn.edu

Vipin Kumar
University of Minnesota
kumar001@umn.edu

Xiaowei Jia
University of Pittsburgh
xiaowei@pitt.edu

***Abstract*—Prediction of response to input drivers by unmonitored entities has been recognized as one of the most important problems in many scientific problems. This problem is challenging due to the non-stationary processes that underlie the dynamics of data observations over space and time. Hence, directly transferring models from well-observed data entities to unmonitored target entity often lead to sub-optimal performance due to the shift in data distribution. This paper proposes a new meta-transfer learning framework that automatically estimates the similarity amongst entities to transfer knowledge from well-observed entities to unmonitored entities. A sequence autoencoder embeds temporal behaviors of time series data and simulations generated by traditional physics-based models. This embedding model is trained in a meta-transfer learning framework under the guidance of source-to-source transferring experiences. We tested this method in streamflow prediction for multiple river segments in the Delaware River Basin, an ecologically diverse region along the eastern coast of the United States. The experimental results demonstrate the superiority of the proposed method in predicting streamflow for unmonitored stream segments compared to a diverse set of baselines. Our method also creates meaningful similarity estimates amongst segments to guide the transfer learning process.**

***Index Terms*—Meta-transfer learning; Streamflow prediction; Metric learning; Representation learning**

I. Introduction

The last decades have witnessed the immense success of machine learning (ML) in commercial applications, e.g., computer vision and natural language processing. Given the capability of ML models in automatically extracting complex relationships from data, there is an expectation for using ML models in addressing essential problems in scientific applications such as hydrology [9], [17], biology, and climate science [7]. The modeling of physical variables in these applications is challenging for traditional physics-based models (PBM) due to the incomplete knowledge or excessive complexity in modeling underlying relationships amongst physical variables [10], [19], [22]. ML models are capable of directly extracting the statistical relationships from data. However, in the absence of adequate information about the physical mechanisms of real-world processes, they are prone to false discoveries. For example, the streamflow in a river stream is governed by complex physical processes that change over space and time. Given the input meteorological drivers (e.g., air temperature, precipitation, wind speed), different streams can exhibit very different water dynamics due to the variation in inherent characteristics of each river stream (e.g., soil properties, land covers, and stream geometry) [4], [24]. As a result, a single global model trained using data from all the entities can have sub-optimal performance for many entities due to the data variability issue.

One intuitive solution to address this issue is to separately train individual models for different entities (e.g., different streams). However, the data available for many scientific problems is far smaller than what is needed to train advanced ML models effectively. Collecting labeled data is often expensive in scientific applications due to the substantial manual labor and material cost required to deploy sensors or other measuring instruments. For example, collecting streamflow data requires deploying sensors within a stream, incurring personnel and equipment costs. This results in disparity in observations across sites, where most of the observations come from a few monitored river streams, and a large number of river streams have no in-situ monitoring data.

The prediction in unmonitored sites has been recognized as one of the most critical problems in hydrology [12]. In this work, we plan to transfer the information in a small population of streams to make predictions in the much larger population of unmonitored sites. Intuitively, we consider leveraging the similarity amongst streams [6] in the following aspects. First, river streams with similar underlying physical characteristics often show similar streamflow responses to the meteorological drivers. For example, physical characteristics like soil and groundwater properties mediate the relationship between the input rainfall and the responses of surface discharge and baseflows. Second, the streams under similar weather conditions (e.g., rainfall, wind speed, solar radiation) over time can have similar temporal streamflow behaviors if their physical characteristics are similar.

Transferring knowledge from one domain to another requires addressing two key research issues: how to transfer and what to transfer [25]. Transfer learning using deep neural networks has shown success in several applications such as image classification [20], sentiment classification [5], and ecological applications such as lake temperature modeling [33]. The issues of "how to transfer" and "what to transfer" can be posed as a problem to be solved by meta-learning [11] or learning from previous learning experiences [37], which is an active area of machine learning research. An explicitly defined meta-level objective measuring the transfer performance of source models to target domains is defined in such methods.

In contrast to traditional transfer learning, meta-learning deals with a broader range of meta representations or meta parameters [30] than solely transferring source model parameters. One way for meta knowledge transfer is to learn the similarity between different domains, which could help identify similar reference domains and transfer the knowledge to the target domain to predict the outcome of interest. The critical part of domain similarity learning is to learn a meaningful and precise metric that can be used to measure the similarity between a pair of river streams.

This paper proposes a novel meta-transfer learning approach that learns a metric space to measure the similarity between domains by leveraging the past transfer experience. The framework is developed in the context of modeling streamflow in river networks, but the framework can be generally applied to many complex physical systems with interacting processes. The architecture is based on recurrent neural networks (RNN) and uses contrastive losses guided by the ordered transfer performance, which implicitly captures the similarity among river streams. In particular, the proposed framework consists of a meta model and several source models. The source models are RNN-based architectures built for each site to extract the temporal information from the time-varying data, such as meteorological data and simulated streamflow from PBM, and stream geometries like depth and elevation and use such information to predict streamflow at each time step. On the other hand, the meta model's goal is to learn when and how to transfer these source models from a multitude of experiences to the target rivers. The meta-model is a bidirectional RNN-based architecture that embeds yearly data for a river stream to a latent space where the similarity across river streams can be measured. To reflect the closeness of river streams based on the similarity of streamflow behaviors, the latent space is learned using a contrastive loss guided by the order of transfer performance from source to source river streams. Once trained, the meta-model can be used to compute the similarity between a new target stream and existing source streams using their respective embeddings in the latent space. The closest source models are retrieved using the computed similarity through several methods like top-K ensemble and clustering, and an ensemble model is created. We evaluate our proposed framework for predicting streamflow in a real-world dataset collected over 36 years from the Delaware River Basin in the Northeastern United States. Our method produces superior prediction performance compared to the global model and other baselines. We also show that the learned similarity closely follows the transferred predictive performance. Code is available at the link [1]

Our contributions can be summarized as follows:

- We introduce a new meta-transfer learning framework applicable in scenarios where observation data is scarce.
- We leverage knowledge from a physics-based model to guide a meta-model for extracting latent variables, which helps measure the similarity among river streams based on underlying physical processes and weather patterns.
- We propose a new contrastive loss function that is used to train the meta-model by leveraging past transferring experiences as guidance.
- We evaluate the framework's utility in the context of an ecologically and societally relevant problem of monitoring river networks.

[1]https://drive.google.com/drive/folders/1wbux6W2ADjM58EmTg6ZkCAoEcuWN9iTN?usp=sharing

II. Related Work

Integrating physics into ML models has improved predictive performance and generalizability in scientific problems. ML models are expected to have sufficient capacity to model such interactions when applied to systems with interacting processes. Moreover, Machine Learning (ML) models (e.g., LSTMs) can provide state-of-the-art performance for many scientific applications [17]. The reason is that ML models can benefit from a large cross-section of diverse training data and thus can transfer knowledge across basins. However, training a global model for all river streams using traditional loss functions for regression problems (such as mean squared loss) tends to be dominated by river segments with more significant errors while degrading the performance of other segments with smaller errors [14]. This transferring local source models to target streams instead of a single global model can be beneficial.

Recently meta-learning has found great success in the few-shot application of meta-learning, where the idea is to perform non-parametric 'learning' at the task level by simply comparing the various tasks. The outer-level optimization corresponds to finding a feature extractor that learns a latent space suitable for comparison. Several advancements have been proposed by including several conditions [3] or designing new metric space [31].

Similarity learning techniques have been used to intelligently select relevant source domains given target domain that help improve the learning performance of ML models [23]. These methods have shown much success in several domains like learning similarity in patients [13] and categories [30]. However, in these approaches, the transfer experience among the source data is not used in learning the metric. Wei et al. [32] proposed a method to automatically determine what and how to transfer by leveraging previous transfer learning experiences. Similar to this strategy, Jared et al. [33] proposed a strategy to train a meta-model that can predict the best source model for a given target domain. However, the meta-model in the proposed strategy uses simple hand-engineered statistics-based features. In contrast, we use a deep-learning-based meta-model that automatically learns the similarity based on the input data.

III. Problem definition

In this work, our objective is to predict streamflow over multiple river segments in a stream network at a daily scale by leveraging temporal contextual information. We consider N river segments in a stream network. For each river segment i, we are provided with input time-series features over multiple daily time-steps represented by $\boldsymbol{X}_i$ as a multivariate time series for T timestamp i.e., $\boldsymbol{X}_i = [\boldsymbol{x}_i^1, \boldsymbol{x}_i^2, \ldots, \boldsymbol{x}_i^T]$ where $\boldsymbol{x}_i^t \in \mathbb{R}^{D_x}$ indicates the dynamic input vector at time $t \in T$. We are also provided with several geometric parameters of the stream segments (such as depth, surface area, shape of lake and others) as static input vector represented as $\boldsymbol{z}_i \in \mathbb{R}^{D_z}$. The observed streamflow response corresponding to $(\boldsymbol{X}_i, \boldsymbol{z}_i)$ for an entity is denoted by $\boldsymbol{Y}_i = [y_i^1, y_i^2, \ldots, y_i^T]$. This observed streamflow is available for certain segments $i \in \{1, ..., N\}$ and on certain dates $t \in 1, ..., T$. More details on the dynamic and static input and output variables can be found in Section V-A.

We consider two sets of river segments, source and target sets. Particularly, we assume that the river segments in the source set have streamflow observations available during the training and test time steps. In contrast, the river segments in the target sets do not have streamflow observations. In streamflow modeling, the goal is to integrate the daily climate drivers $(\boldsymbol{X}_i)$ with the static characteristics $(\boldsymbol{z}_i)$ of a river segment to learn a forward operator $\mathscr{F}$ that predicts the streamflow of water in a river segment at every time step i.e $\mathscr{F} : \boldsymbol{X}_i, \boldsymbol{z}_i \rightarrow \boldsymbol{Y}_i$. The major challenge in building this mapping is to handle the heterogeneity across different sites $i \in \{1, ..., N\}$ to achieve good performance over all the stream segments.

In our proposed method, we also build an individual streamflow model $\mathscr{F}_i$ for each river segment i present in the source set using its data. We assume these models perform well for each source stream segment because we have sufficient training data for all the streams in the source set. For each pair of river segments (i, j), our goal is to learn a similarity/distance metric $\mathscr{D}$ that uses their corresponding input time-series features $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$, stream geometry $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$, and their physical simulations to estimate whether the two river segments are similar (e.g., the two river segments having the same underlying physical process) or not. Accurate metric learning will enable making prediction on a target river segment by transferring a combination of predictions from its most similar source models.

IV. Method

This section provides details of the proposed meta-transfer learning via the metric learning approach. The methods proposed in this paper aim to tackle three sub-tasks: (i) how to represent a river segment using neural networks, (ii) how to estimate the similarity between river segments, and (iii) how to further leverage this similarity for improving streamflow prediction in the target river segments. In Section IV-A, we first introduce the sequence auto-encoder model to represent river segments. Then in Section IV-B, we discuss the metric learning method to estimate the similarity between river segments using the source-to-source transfer performance as guidance. Finally, in Section IV-C, we describe how to leverage the learned similarity to improve the model performance.

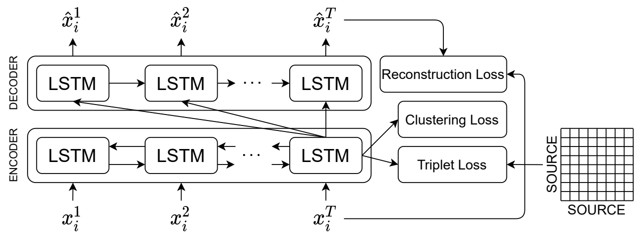


Fig. 1. The sequence autoencoder architecture used for embedding time series data, which uses three loss functions: the reconstruction loss (Eq. 3), the triplet loss (Eq. 5), and the clustering loss (Eq. 11). In our application, the time series data x_i^t represents the concatenation of daily input, static features, and simulated target variables.

A. Sequence Autoencoder

Weather data collected from real-world sensor systems are usually high dimensional and noisy, containing both redundant and irrelevant information. Incorporating these features directly for measuring the similarity between two river segments may hide the discriminative information, resulting in poor performance of metric learning models [16]. Moreover, the data in time series exhibit temporal water dynamics, which reflect the unique characteristics of each river segment and need to be embedded in the representation of each river segment. Domain scientists often use a manual inspection or pre-defined metrics to represent time series data [23], [29]. However, these approaches often require tremendous effort in feature engineering from domain experts. It is also difficult for these approaches to capture long-term temporal data correlations, which are found to be ubiquitous in real time-series datasets and essential for prediction tasks. Therefore, a new mechanism for extracting meaningful and informative representations for time-series data is required for estimating the similarity amongst river segments in a stream network [31].

Our method aims to generate river segment-specific embeddings from time-series data. Specifically, for each river segment i in the source domain, we randomly select a subsequence S_i of length W taken from the time-windows $t_i : t_i + W$. This results in N_s sequences and each element in these sequences are formed by concatenating the input time-series, geometric parameters (through duplication), and simulated streamflow of the river segment s_i^t (i.e., the concatenated features are $[\boldsymbol{x}_i^t; \boldsymbol{z}_i; s_i^t]$). For learning time-series representations, it is crucial to mitigate the inductive biases by choosing the proper objective function so that the learning process adjusts the model towards learning representative features. In this paper, we use a long-short term memory (LSTM)-based encoder-decoder architecture to learn representations from the input time series of a river segment, as shown in Fig. 1. LSTM is particularly suited for our task in which long-term temporal dependencies must be modeled to capture water dynamics. However, LSTMs are designed to run only forward in time, while the similarity estimation requires embedding the overall water behaviors in a sequence by considering the patterns in both forward and backward directions. Hence, we use a bidirectional LSTM-based sequence encoder $q_\phi(\boldsymbol{h}|[\boldsymbol{x}_i^t;\boldsymbol{z}_i;s_i^t]_{t=1:T})$ for the similarity learning model. Specifically, we build two LSTM structures: the forward LSTM and the backward LSTM. The two LSTM structures are the same, except that the time series is reversed for the backward LSTM. Each LSTM uses the following equations to generate the embeddings for a sequence.

$$\begin{aligned}\boldsymbol{i}_t &= \sigma(\boldsymbol{W}_i\left[[\boldsymbol{x}^t;\boldsymbol{z};s^t];\boldsymbol{h}^{t-1}\right]+\boldsymbol{b}_i)\\ \boldsymbol{f}_t &= \sigma(\boldsymbol{W}_f\left[[\boldsymbol{x}^t;\boldsymbol{z};s^t];\boldsymbol{h}^{t-1}\right]+\boldsymbol{b}_f)\\ \boldsymbol{g}_t &= \sigma(\boldsymbol{W}_g\left[[\boldsymbol{x}^t;\boldsymbol{z};s^t];\boldsymbol{h}^{t-1}\right]+\boldsymbol{b}_g)\\ \boldsymbol{o}_t &= \sigma(\boldsymbol{W}_o\left[[\boldsymbol{x}^t;\boldsymbol{z};s^t];\boldsymbol{h^{t-1}}\right]+\boldsymbol{b}_o)\\ \boldsymbol{c}_t &= \boldsymbol{f}_t\odot\boldsymbol{c}_{t-1}+\boldsymbol{i}\odot\boldsymbol{g}_t\\ \boldsymbol{h}_t &= \boldsymbol{o}_t\odot\tanh(\boldsymbol{c}_t)\end{aligned}\tag{1}$$

Each forward and backward LSTM takes a sequence as input and generates corresponding embeddings. These embeddings are essentially the final hidden states of each LSTM. The embeddings for the forward LSTM and backward LSTM are added to get the final embeddings $\boldsymbol{h}=\boldsymbol{h}_{fwd}+\boldsymbol{h}_{bwd}$. This representation $\boldsymbol{h}$ is then fed through the LSTM decoder $p_\theta([\boldsymbol{x}^t;\boldsymbol{z};s^t]_{1:T}|\boldsymbol{h})$ to produce a target sequence, which is the same as the input sequence in the encode-decode architecture. In particular, we use a conditional decoder that iteratively outputs the data at each time $[\boldsymbol{x}^t;\boldsymbol{z};s^t]$ based on the output data from the previous time steps, as follows:

$$\begin{aligned}&p_\theta([\boldsymbol{x}^t;\boldsymbol{z};s^t]_{t=1:T}|\boldsymbol{h})\\ &=p_\theta([\boldsymbol{x}^1;\boldsymbol{z};s^1]|\boldsymbol{h})\prod_{t=2}^{T}p_\theta([\boldsymbol{x}^t;\boldsymbol{z};s^t]|[\boldsymbol{x}^{1:t-1};\boldsymbol{z};s^{1:t-1}],\boldsymbol{h})\end{aligned}\tag{2}$$

A traditional way to train this sequence-to-sequence autoencoder is teacher forcing [34], where ground truth data is used as input instead of the predicted values. Although teacher forcing simplifies the loss landscape and provides faster convergence, this training procedure weakens the encoder as the decoder has to solve a much simpler task. Since we want the encoder to extract good representations, we train our autoencoder in a closed-loop mode, with the network outputs fed back as input. The autoencoder parameters are trained to maximize the likelihood of the data, which under the Gaussian assumption becomes the reconstruction loss computed as the mean-squared error between the reconstructed and the original sequence,

$$\max_{\theta,\phi}E_{\boldsymbol{x},\boldsymbol{z}\sim data}[-logp_\theta([\boldsymbol{x}^t;\boldsymbol{z};s^t]_{t=1:T}|\boldsymbol{h})]\tag{3}$$

This sequence autoencoder, once trained, can extract fixed-length representation from an arbitrary-length sequence. Using their learned representations, we can then calculate the similarity between two river segments. However, choosing a particular similarity function is a critical design choice. We use the cosine between the two embeddings as the similarity measure as they provide softer constraints. Using euclidean distance can lead to exploding loss values as well as it enforces stronger constraints which have the potential to lead to trivial solutions. The cosine similarity between the two latent vectors is calculated as,

$$sim(\boldsymbol{h}_i,\boldsymbol{h}_j)=\frac{\boldsymbol{h}_i\cdot\boldsymbol{h}_j}{\|\boldsymbol{h}_i\|\|\boldsymbol{h}_j\|}\tag{4}$$

However, this calculated similarity is not optimized for effective model transfer. This is primarily because the similarity between the features does not guarantee that the model trained on the source river segment gives the best result on the target river segment. In the following sections, we describe our meta-transfer learning approach, which automatically determines which source models to transfer based on previous transfer learning experiences [32].

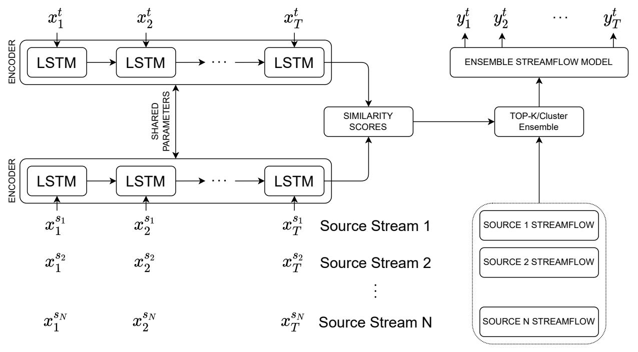
B. Metric Learning for Learning to Transfer

Assume we have $|S|$ river segments in the source domain. We first create individual RNN models (with LSTM structure) $\mathcal{M}_1$, $\mathcal{M}_2$, ..., $\mathcal{M}_{|S|}$ separately for each source segment using its data. Ideally, these individual models can perform well for their corresponding source segment, given sufficient data for each source segment. Note that our metric learning method is agnostic of a specific source model so that it can be used for other predictive models for other applications.

In the meta-transfer learning framework, we aim to use the model transfer between each pair of source segments to mimic the transfer process from source to target segments. Since we can access actual observations in source segments, we can measure the performance for source-to-source transfer (e.g., using R^2 value). In the following, we will describe how to record the performance metrics for source-to-source transfer and use them to guide the training of the similarity learning model.

We generate the transferring performance matrix by recording the prediction accuracy when each source model $\mathcal{M}_i$ is applied to the remaining $|S|-1$ river segments to predict daily streamflow. Note that the diagonal will have the best result since the model is being trained and evaluated on the same river segment. We use the $|S|(|S|-1)$ transfer learning experiences in our metric learning framework to guide the learning of embeddings that mimics these experiences. Specifically, for each source segment i, we divide the remaining $|S|-1$ segments into a positive list and a negative list based on a performance metric threshold (e.g., a threshold on R^2 values) using the testing performance of $\mathcal{M}_i$ on each of the remaining river segments. Repeating this process for all the source segments results in S such positive and negative lists for each river segment in the source set. We create $|S|$ triplets for each river segment (anchor) by randomly selecting a river segment from the positive and negative list. Finally, we define the triplet loss that forces the embedding of the anchor river segment $\boldsymbol{h}_i$ to be closer to its positive river segment $\boldsymbol{h}_{p_i}$ and farther from its negative river segment $\boldsymbol{h}_{n_i}$.

$$\mathcal{L}_{Triplet}=max(0,D(\boldsymbol{h}_i,\boldsymbol{h}_{p_i})-D(\boldsymbol{h}_i,\boldsymbol{h}_{n_i})+\alpha)\tag{5}$$



x_1^t x_2^t x_T^t
y_1^t y_2^t ⋯ y_T^t
ENCODER
LSTM
SHARED PARAMETERS
SIMILARITY SCORES
TOP-K/Cluster Ensemble
ENSEMBLE STREAMFLOW MODEL
$x_1^{s_1}$ $x_2^{s_1}$ $x_T^{s_1}$ Source Stream 1
$x_1^{s_2}$ $x_2^{s_2}$ $x_T^{s_2}$ Source Stream 2
$x_1^{s_N}$ $x_2^{s_N}$ $x_T^{s_N}$ Source Stream N
SOURCE 1 STREAMFLOW
SOURCE 2 STREAMFLOW
SOURCE N STREAMFLOW

Fig. 2. The model transfer process using the top-K or cluster ensemble of source models based on the estimated similarity amongst river segments.

Our proposed triplet loss explicitly allows the relationships between river segments based on their transferring performance to be preserved during representation learning. Combining the triplet loss (Eq. 5) and the standard supervised reconstruction loss (Eq. 3), we get the final training loss as follows:

$$\mathcal{L} = \mathcal{L}_{Rec} + \lambda \mathcal{L}_{Triplet} \tag{6}$$

where λ is a hyper-parameter.

C. Ensemble source models

Individual source models trained for each stream segment embed the streamflow behaviors in response to input data. Such behaviors can vary drastically across different segments as some high-flow segments (with higher average streamflow) often exhibit a more significant streamflow variance over low-flow segments. Applying the source model from the most similar source segment is an intuitive solution for a target segment in the target domain. However, the estimated similarity for segments in the target domain may not be entirely accurate, and transferring a sub-optimal model from a diverse set of stream segments could degrade the performance for prediction. Ensemble methods have been shown to obtain better predictive performance than the performance obtained from any single constituent model [28]. Further, in scenarios where significant model diversity exists, ensembles tend to yield better results [18]. Hence, we propose to transfer multiple source models to make predictions for each target river segment, as shown in Fig. 2. However, identifying the groups is essential in creating such ensemble models. In the following sections, we describe three methods for creating ensemble models.

1) Top-K Ensemble: Once the metric learning model is trained, we can use the similarity measure described in Eq. 4 to select the top-k source models for a given target model. The final prediction for the target river segment is the average of the predictions at each step from the individual k source models.

$$\begin{aligned} \boldsymbol{Y}_{\boldsymbol{t_i}} &= \frac{1}{|K_{t_i}|} \sum_{k \in K_{t_i}} \mathscr{F}_k(\boldsymbol{X}_{\boldsymbol{t_i}}, \boldsymbol{z}_{\boldsymbol{t_i}}), \\ \text{where,} \quad K_{t_i} &= \underset{S' \subset S, |S'| = K}{\operatorname{argmax}} \sum_{S_j \in S'} sim(h_{T_i}, h_{S_j}) \end{aligned} \tag{7}$$

However, K is a hyperparameter that needs to be selected manually and can lead to bad predictive performance if k is too large. Thus automatic creation of groups is essential.

2) Cluster Ensemble: We next provide another strategy to automatically select source models for a given target stream without worrying about K. Specifically, we use the source river-stream embeddings to define a clustering structure using K-Means clustering. The trained K-means clustering assigns the target streams to one of the source clusters, and the ensemble model is created by the softmax weight of the source-target distance, as shown,

$$\begin{aligned} \boldsymbol{Y}_{\boldsymbol{t_i}} &= \sum_{k \in K_{t_i}} \alpha_k \mathscr{F}_k(\boldsymbol{X}_{\boldsymbol{t_i}}, \boldsymbol{z}_{\boldsymbol{t_i}}), \\ \text{where,} \quad & \alpha_k \frac{sim(h_{T_i}, h_{S_k})}{\sum_{S_j \in S} sim(h_{T_i}, h_{S_j})} \end{aligned} \tag{8}$$

3) Cluster Ensemble with clustering loss: Although we partition the source river streams into clusters in the previous method, the meta-model is not optimized. The representation learning methods can learn similar representations between low-flow and high-flow streams, which can cause potential confusion amongst various categories of river streams. This challenges the representation learning model to learn a latent space that can correctly cluster all the modes in river streams. Intuitively, suppose we can detect these modes by optimizing a clustering objective. In that case, it will allow the meta-model to learn representations that create a clustering structure of different modes of water bodies. In particular, we adapt DEC [36] as the clustering objective, where the pre-trained autoencoder and the K-Means cluster centroids from the previous method provide an excellent initialization point. The encoder parameters and the centroids are refined by learning from the high-confidence assignments using an Expectation-Maximisation (EM) style algorithm inspired by the previous work [36]. In the E step, the cluster assignment and the target assignment are computed while keeping the encoder parameters and cluster centroids fixed. Specifically, we use a soft assignment based on the similarity of the embedded data point with the cluster centroid, measured using the Student's t-distribution [19]. Specifically, the soft-assignment of data i to cluster j is computed as follows:

$$q_{ij} = \frac{(1 + \|\boldsymbol{h}(\boldsymbol{X}_i; \theta_h) - M_j\|^2/\alpha)^{\frac{\alpha+1}{2}}}{\sum_{j'=1}^{K} (1 + \|\boldsymbol{h}(\boldsymbol{X}_i; \theta_h) - M_{j'}\|^2/\alpha)^{\frac{\alpha+1}{2}}} \tag{9}$$

where $\boldsymbol{h}(\boldsymbol{X}_i; \theta_h)$ is the embedded data point, α is the degree of freedom which is set as 1 in our experiments, and q_{ij} is the probability of assigning the i'th data point to the j'th cluster. To strengthen prediction and to promote learning from data points that are assigned with high confidence, the target assignment is computed as:

$$p_{ij} = \frac{q_{ij}^2 / \sum_i q_{ij}}{\sum_{j'=1}^{K} (q_{ij'}^2 / \sum_i q_{ij'})} \tag{10}$$

Once the cluster assignment and the target assignment are computed, in the M step, we estimate the encoder parameters and the cluster centroids using gradient descent while keeping the cluster and the target assignment fixed. The objective is defined as the KL divergence loss between the soft assignments and the target assignment as follows:

$$\min KL(P\|Q) = \min \frac{1}{N_t} \sum_{i=1}^{N_t} \sum_{j=1}^{K} p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{11}$$

The encoder parameters, decoder parameters and the cluster centroids are refined according to the objective:

$$\mathcal{L} = \mathcal{L}_{Rec} + \lambda_1 \mathcal{L}_{Triplet} + \lambda_2 \mathcal{L}_{clus} \tag{12}$$

where λ_1 and λ_2 are hyper-parameters to control the weights of the triplet loss and the clustering loss, respectively. Once trained, the model can produce the clustering structure during the representation learning process. The prediction for a target river stream is then performed as defined in Eq 8. Here the clusters are provided by the meta-model trained using the clustering objective.

V. Experiments and Results

A. Dataset

All the data used in this work are available through U.S. Geological Survey's National Water Information System [2] and the Water Quality Portal [26]. It is the most extensive standardized water quality data set for inland and coastal water bodies [26]. The methods are evaluated to predict streamflow in the Delaware River Basin, an ecologically diverse region and a watershed along the east coast of the United States that provides drinking water to over 15 million people [35]. Observations at a specific latitude and longitude were matched to river segments that vary in length from 48 to 23,120 meters. The river segments were defined by the national geospatial fabric used for the National Hydrologic Model as described by Regan et al. [27]. The river segments are split up to have roughly a one-day water travel time. We match observations to river segments by snapping observations to the nearest river segment within a tolerance of 250 meters. Observations farther than 5,000 m along the river channel to the outlet of a segment were omitted from our dataset.

We use input features at daily scale from Oct 01, 1980, to Sep 30, 2019 (13,149 dates). The input features include fifteen time-varying features and four time-invariant geometric features of each segment (e.g., elevation, length, slope and width). The time-varying features include meteorological features such as daily average precipitation, daily average air temperature, date of the year, solar radiation, shade fraction, potential evapotranspiration as well as simulated streamflow from PB models. Air temperature and precipitation values were derived from the Daymet gridded meteorological dataset [1]. Other input features (e.g., shade fraction, solar radiation, potential evapotranspiration) are difficult to measure frequently, and we use values produced by the PRMS-SNTemp model [21] as its internal variables.

We study two subsets of the Delaware River Basin. In subset S_1, we include all the river segments with more than 1000 streamflow observations resulting in 63 river segments. Whereas, in subset S_2, we include all the river segments with more than 100 streamflow observations resulting in 128 river segments. From these two subsets, we create different experimental settings. We first sort the river streams in each subset according to their mean streamflow. Dataset I is created from sorted S_1 by selecting the alternate streams into the train and test set. This creates an even distribution of river streams in both sets. Similarly, Dataset II is created from the same sorted S_1, but this time we select the train and test in the ratio of 1:2 to show the effect of reduction in source models. Dataset III is created in the same manner as Dataset I, however, from the subset S_2.

B. Baselines

We compare model performance to multiple baselines, as described below:

- PRMS: The Precipitation-Runoff Modeling System (PRMS) [21] is a physics-based model that simulates daily streamflow 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.
- Global CT-LSTM: We train a global model by concatenating all the input features and feeding them into an LSTM to predict the streamflow.
- Global EA-LSTM: In this approach, we train an entity-aware lstm model by feeding the geometric properties of the river segment in the input gate of the lstm [17]. This model provides interpretability as it modulates the LSTM cell based on the physical properties of the river segment.
- PGTL: We use the river segments' geometric properties to transfer the source model to the target. Specifically, for each
- MAML: We use the model agnostic meta-learning (MAML) [8] approach for fast adaption of the ML model for the target river segments. Since for the target river segments, we do not have the observed streamflow, we use the simulated streamflow of the target river segments and five inner optimization steps to finetune the meta-model.
- PGMTL: We compare the performance of our model to a recently proposed approach that applies meta-transfer learning to machine learning models using regression trees [33]. We use the same four sets of meta-features, i.e., lake attributes, PB0 Simulation statistic, General

TABLE I
R2 VALUES FOR STREAMFLOW MODELING ON THE THREE DATASETS. HERE OUR METHOD AND ITS VARIATIONS ARE COMPARED WITH GLOBAL MODELS, PGTL, MAML, AND THE PGMTL APPROACH. BEST SOURCE IS THE UPPER BOUND OF PERFORMANCE IF WE SPECIFICALLY SELECT THE BEST PERFORMING SOURCE MODEL FOR EACH TARGET RIVER-STREAM

Method	Dataset I	Dataset II	Dataset III
PRMS	-1.93	-1.295	-2.884
Global CT-Lstm	0.412	0.367	0.235
Global EA-Lstm	0.414	0.378	0.219
MAML	0.284	0.425	0.273
AEMTL	0.354	0.302	0.364
PGMTL	0.386	0.421	0.046
Our$_{Topk}$	0.504	0.45	0.452
Our$_{kMeans}$	0.516	0.483	0.378
Our$_{Cluster}$	0.543	0.461	0.401
Best Source	0.594	0.56	0.541

observation statistics, and meteorological statistics, as described by the authors.

In our experiments, we train all global and individual source models for a maximum of 200 epochs. The model is optimized with the ADAM optimizer [15] with the initial learning rate of $5e^{-4}$. All the hidden and gating variables in the RNNs have 20 dimensions. The train, validation, and test set are kept consistent for all models to remove bias between different model runs.

C. Prediction performance

In Table I, we report the performance of each method for streamflow prediction. For all the methods, we assume that the simulation data are available on every single date from Oct 01, 1980, to Sept 20, 2016. This is because they can be generated by running the PRMS process-based model on input drivers. We report the means R2 across the test basins for three datasets, D1, D2, and D3. We can observe that the proposed method outperforms baselines by a considerable margin for all three datasets. All versions of our proposed method perform better than the global models because they utilize the past source-source transfer experience, which is critical for an accurate estimation of source-target transfer performance.

We first observe that the global models do not perform well, as shown by their low R2 values. This is because ML models optimize the overall performance while low-flow stream segments (mostly headwaters) are a minority in the entire river network and contribute less to the loss function. We also observe that the proposed method performs better than MAML, which is fine-tuned using the simulated data. This can be explained by the poor performance of the output from the PRMS method on the target streams. The MAML method in the fine-tuning step utilizes the simulated observation to generate the individual models. Although AEMTL uses the meta-model, it is not trained using the transfer experience matrix between the source set. On the other hand, PGMTL uses the transfer matrix but uses few hand-engineered features in a simple Gradient-boosted tree-based meta-model. However, all the variants of our method use the time-varying feature values and the source-to-source transfer matrix to learn a latent space and determine the most relevant sources to assign to a target river stream. This is reflected in its performance gain over other baseline methods. Moreover, in some cases (Dataset I and Dataset II), our method's cluster variants perform better than our method's topK version due to the reasons associated with selecting only one single source model for a target river stream. Fig. 3 shows the performance of our method and several baselines on all the streams in Dataset I. Moreover, we show the streamflow prediction on one of the target streams for all the test time steps in Fig. 4. Note how the global CT-Lstm model trained on all source river streams over-estimates the streamflow for the reasons discussed above.

D. Similarity Learning

Here we aim to evaluate the performance of the models in learning similarities between river streams. We particularly compare the performance of our meta-model in learning similarity between river streams by utilizing the $tripletloss$ to the plain recurrent auto-encoder variant. We evaluate the models using two strategies. First, we visually compare the several learned similarity matrices to the ground truth. Further, we quantitatively evaluate the learned similarity using commonly used metrics in recommendation systems.

1) Visualizing Predicted Similarity: In Fig. 5 we visually compare the learned similarity matrices with the ground truth in the train and test set for all the datasets. Each matrix has the source river streams used to train the individual source models on the y-axis and the target river streams on the x-axis. In both axes, we order the river streams in the increasing order of their mean streamflow. In the case of the train set, the target river streams are the same as the source river streams, and this denotes the transferring performance where each source model is applied to every other source river stream. To avoid temporal correlation, we use the data during the test years in this analysis. The ground truth column shows a matrix containing the prediction accuracy of the models in terms of R^2 values, whereas the other two columns show the learned similarity calculated by taking the cosine similarity of the embeddings (Eq 4). In each matrix, brighter color denotes higher similarity, whereas a darker color shows that the pair of river streams are not similar. We first observe that the similarity matrix obtained from our method matches more closely to the ground truth than the $AEMTL$. Moreover, we observe a block structure in the ground truth matrix for both the train and test set. This shows that the low streamflow source models usually do not perform well on the high streamflow target streams and vice-versa. $AEMTL$ cannot capture this pattern without explicitly modeling this information in the form of triplet loss. However, this block pattern is also observed in our method for both datasets. This shows that the experience-guided triplet loss can learn from this pattern in the training set and apply this learned transferring knowledge in the test set.

2) Evaluating similarity via nearest neighbor retrieval: In addition to visually inspecting the learned similarity matrix, we also give quantitative metrics to evaluate them. The task of learning to transfer appropriate source models for each

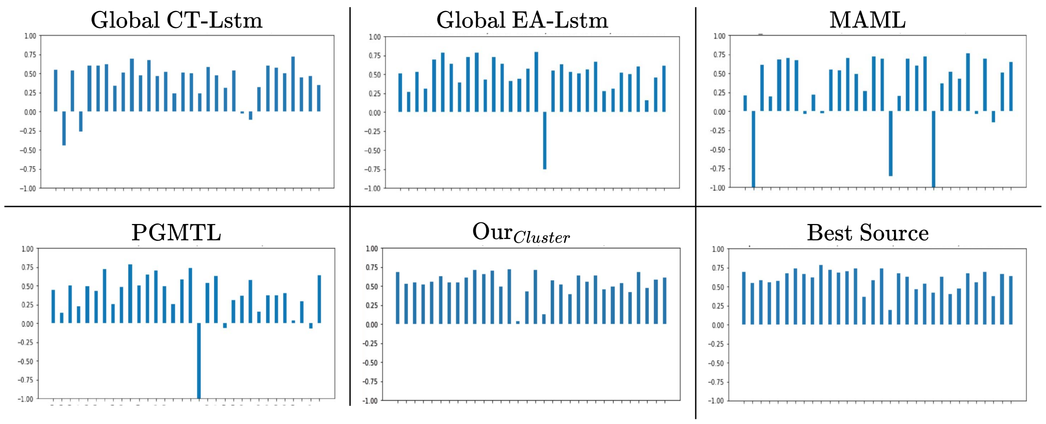


Fig. 3. Streamflow performance on each target river stream by all the models. Y-axis shows the R^2 value, whereas the river streams are on the x-axis.

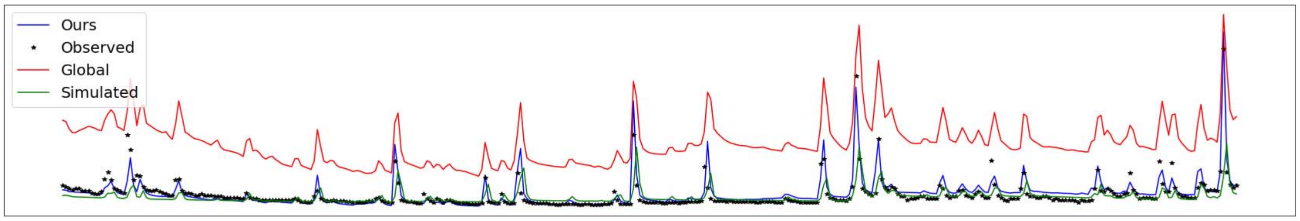


Fig. 4. Streamflow predictions made by the proposed method, the global CT-LSTM model, and the physics-based PRMS model.

TABLE II
EVALUATION OF THE SIMILARITY LEVELS ESTIMATED BY AEMTL, PGMTL, AND THE PROPOSED METHOD, USING RETRIEVAL METRICS.

Metric	Prec@5	MAP@5	MRR
AEMTL	0.157	0.101	0.353
PGMTL	0.407	0.317	0.595
Ours	0.519	0.410	0.737

target river stream can be viewed as a recommender system problem. Specifically, we recommend personalized source models unique to each target river stream. We compare the models based on metrics defined as follows

- **Prec@k** : Precision@k is a fraction of top k recommended items relevant to the user. It evaluates recommender systems' decision-making capacity, i.e., the system recommends correct source models in the set. We calculate this metric for all models by setting k as 5, as shown below,

$$Prec@k = \frac{\text{Top } k \text{ recommendations} \cap \text{Top } k \text{ ground truth}}{k} \tag{13}$$

  We report the average of the $Prec@k$ values for all the target river-streams.
- **AP@k** : AveragePrecision@k evaluates a recommender system based on the ranked ordering of relevant items. It rewards the model for placing the correct recommendations on top of the list. Since we use weighted averaging of prediction (8), having correct source models on the top of the list will allow the method to put more weight on its prediction. We calculate average precision for each target river stream as shown below,

$$AP@k = \frac{1}{k}\sum_{i=1}^{k}(Prec@i * relevant@i) \tag{14}$$

  where, $relevant@i$ is equal to 1 if i^{th} recommendation is in Top k ground truth, otherwise 0. We report the mean of $AP@k$ values for all the target river-streams.
- **RR** : Reciprocal rank is the "multiplicative inverse" of the rank of the first correct source model. We calculate the RR for all target streams and report the mean of the values as shown,

$$MRR = \frac{1}{|T|}\sum_{i=1}|T|\frac{1}{rank_i} \tag{15}$$

Table II compares the models on Dataset I using the metrics defined above. We observe that our model outperforms the autoencoder baseline and the recently published PGMTL baseline in both the classification-based (Prec@k) and rank-based (AP@k and RR) metrics. This shows that our model can correctly recommend relevant source models as well as recommend them at the top of the list. This explains our model's high predictive performance (Table I) compared to other baselines. We attribute this characteristic to learning from past transfer experience in the training set.

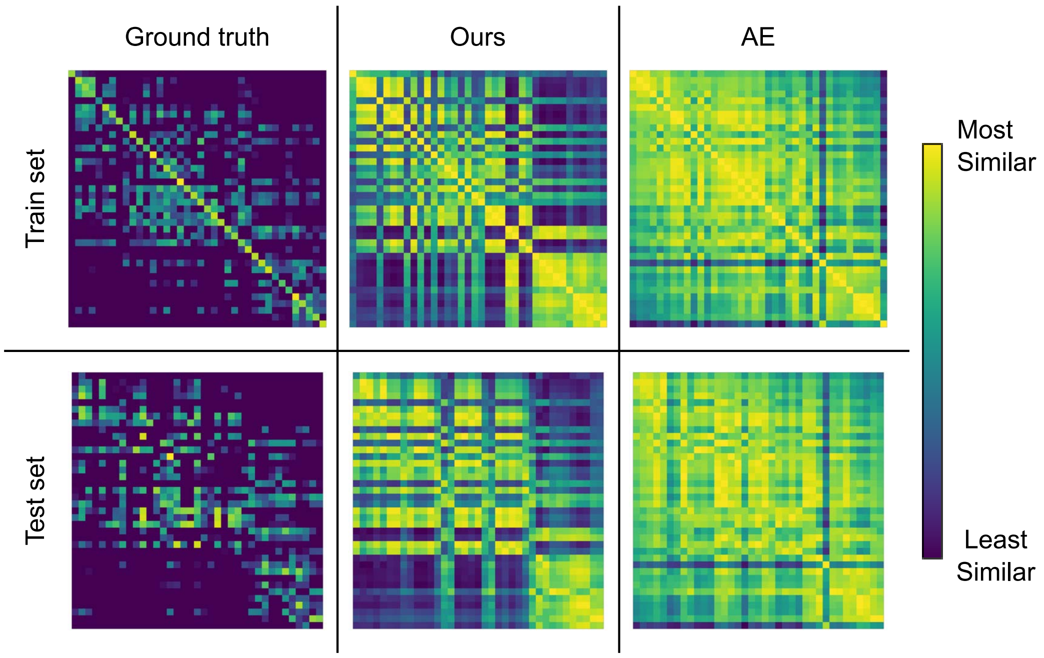


Fig. 5. The similarity matrices between source-to-source segments (1st row) and between source-to-target segments (2nd row). Each entry (i,j) in the ground truth matrix (1st column) represents the R2 value obtained by applying the source model of the segment i to the data of segment j. The matrices for our method and the AE method show the estimated cosine similarity between each pair of segments using the obtained embeddings. Here the yellow color indicates a higher R2 score (in the 1st column) or a higher similarity level (in the second and third columns).

E. Sensitivity Tests

Here we test the sensitivity of the model to different hyper-parameter settings. In particular, we compare the performance of the model with various top-k values and cluster numbers.

1) Sensitivity to Top-k values: Fig. 6 shows the variation of the predictive performance using different K values in the top-K ensemble transfer method. It can be seen that the performance using a small K value ($K = 1$) or very large K values ($K > 7$) can result in worse performance compared to the global model. With $K = 1$, we are only transferring the most similar source model, and the performance can be affected by the errors in estimating the similarities for segments in the target set. When we set a very large K value, we are averaging the predictions from a large number of models. It is likely that we mistakenly include some models from those segments that are less similar to the target segment, degrading the predictive performance.

2) Sensitivity to Cluster numbers: In Fig. 7, we show the performance variation with respect to different numbers of clusters. The performance is generally better than the global LSTM model except when we have a small number of clusters, e.g., when the number of clusters is smaller than 9. This is because the model needs to aggregate the prediction from many source models, and some of their corresponding source segments can be less similar to the target segment.

VI. Conclusion

This paper proposes a new meta-transfer learning framework for predicting target variables in unmonitored stream

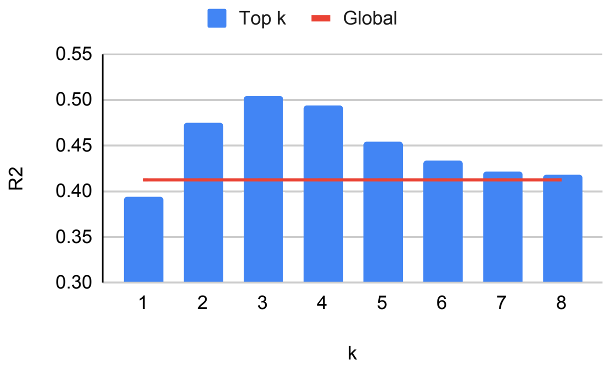


Fig. 6. Predictive performance (in terms of R^2 values) using different K values in the top-K ensemble transfer method.

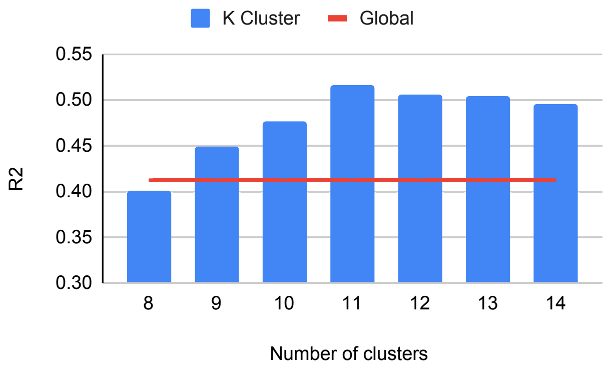


Fig. 7. Predictive performance (in terms of R^2 values) using different numbers of clusters in the cluster ensemble transfer method.

segments. It uses a sequence autoencoder to create embeddings for all the segments by combining input time series data and simulated data generated by the physics-based model. The representation learning model is trained in the meta-transfer learning framework by modeling the similarity amongst stream segments from source to source transferring experiences. We tested this method in the Delaware River Basin, an ecologically diverse region along the eastern coast of the United States. The experimental results reveal that our method can achieve superior predictive performance for unmonitored stream segments compared to a diverse set of baselines. Moreover, our method is shown to create meaningful similarity estimates amongst segments to guide the transfer learning process. Although our method is evaluated in the context of streamflow prediction, it can be generally applied to a wide range of applications that involve multiple heterogeneous entities, and some entities have limited annotations. For example, monitoring greenhouse emissions needs to be conducted over large regions, but the data are often collected from flux towers at specific locations. Similarly, patients in different demographic groups may have different amounts of annotated data in clinics, which poses a significant challenge for automated early disease detection for all the patients.

VII. Acknowledgements

This work was supported by the National Science Foundation (NSF) Awards 1934721 and 2147195, USGS Awards G200011317, G21AC10207 and G21AC10564, and NASA program grant 80NSSC22K1164. Access to computing facilities was provided by the Minnesota Supercomputing Institute and the University of Pittsburgh Center for Research Computing.

References

[1] gridmet - climatology lab. http://www.climatologylab.org/gridmet.html. Accessed: 2021-10-01.

[2] Usgs water data for the nation: U.s. geological survey national water information system database. http://dx.doi.org/10.5066/F7P55KJN. Accessed: 2021-10-01.

[3] Antreas Antoniou and Amos J Storkey. Learning to learn by self-critique. *Advances in Neural Information Processing Systems*, 2019.

[4] Gopal Bhatt et al. A tightly coupled gis and distributed hydrologic modeling framework. *Environmental modelling & software*, 2014.

[5] John Blitzer et al. Domain adaptation with structural correspondence learning. *Proceedings of the 2006 conference on empirical methods in natural language processing*, 2006.

[6] Martin Erlandsson et al. Thirty-five years of synchrony in the organic matter concentrations of swedish rivers explained by variation in flow and sulphate. *Global Change Biology*, 2008.

[7] James H Faghmous and Vipin Kumar. A big data guide to understanding climate change: The case for theory-guided data science. *Big data*, 2014.

[8] Chelsea Finn et al. Model-agnostic meta-learning for fast adaptation of deep networks. *International conference on machine learning*, 2017.

[9] Rahul Ghosh et al. Robust inverse framework using knowledge-guided self-supervised learning: An application to hydrology. *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022.

[10] Hoshin V Gupta and Grey S Nearing. Debates—the future of hydrological sciences: A (common) path forward? using models and data to learn: A systems theoretic perspective on the future of hydrological science. *Water Resources Research*, 2014.

[11] Timothy Hospedales et al. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*, 2020.

[12] Markus Hrachowitz et al. A decade of predictions in ungauged basins (pub)—a review. *Hydrological sciences journal*, 2013.

[13] Mengdi Huai et al. Uncorrelated patient similarity learning. *Proceedings of the 2018 SIAM International Conference on Data Mining*, 2018.

[14] Xiaowei Jia et al. Physics-guided recurrent graph model for predicting flow and temperature in river networks. *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, 2021.

[15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[16] Deguang Kong et al. Exclusive feature learning on arbitrary structures via $\ell_{1,2}$ norm. *Advances in neural information processing systems*, 2014.

[17] Frederik Kratzert et al. Towards learning universal, regional, and local hydrological behaviors via machine learning applied to large-sample datasets. *Hydrology and Earth System Sciences*, 2019.

[18] Ludmila I Kuncheva et al. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine learning*, 2003.

[19] Upmanu Lall. Debates—the future of hydrological sciences: A (common) path forward? one water. one world. many climes. many souls. *Water Resources Research*, 2014.

[20] Mingsheng Long et al. Learning transferable features with deep adaptation networks. *International conference on machine learning*, 2015.

[21] Steven L Markstrom et al. Prms-iv, the precipitation-runoff modeling system, version 4. *US Geological Survey Techniques and Methods*, 2015.

[22] Jeffrey J McDonnell and Keith Beven. Debates—the future of hydrological sciences: A (common) path forward? a call to action aimed at understanding velocities, celerities and residence time distributions of the headwater hydrograph. *Water Resources Research*, 2014.

[23] Garrett W Meigs et al. A landsat time series approach to characterize bark beetle and defoliator impacts on tree mortality and surface fuels in conifer forests. *Remote Sensing of Environment*, 2011.

[24] Brent D Newman et al. Ecohydrology of water-limited environments: A scientific vision. *Water resources research*, 2006.

[25] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 2009.

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

[27] R Steven Regan et al. Description of the national hydrologic model for use with the precipitation-runoff modeling system (prms). Technical report, US Geological Survey, 2018.

[28] Lior Rokach. Ensemble-based classifiers. *Artificial intelligence review*, 2010.

[29] FB Schwing et al. The northern oscillation index (noi): a new climate index for the northeast pacific. *Progress in oceanography*, 2002.

[30] Jake Snell et al. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 2017.

[31] Flood Sung et al. Learning to compare: Relation network for few-shot learning. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.

[32] Ying Wei et al. Transfer learning via learning to transfer: Supplementary material. *35th International Conference on Machine Learning, ICML 2018*, 2018.

[33] Jared D Willard et al. Predicting water temperature dynamics of unmonitored lakes with meta-transfer learning. *Water Resources Research*, 2021.

[34] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1989.

[35] Tanja N Williamson et al. Summary of hydrologic modeling for the delaware river basin using the water availability tool for environmental resources (water). 2015.

[36] Junyuan Xie et al. Unsupervised deep embedding for clustering analysis. *International conference on machine learning*, 2016.

[37] Wei Ying et al. Transfer learning via learning to transfer. *International Conference on Machine Learning*, 2018.