Entity-Aware Knowledge-Guided Machine Learning for Scientific Knowledge Discovery

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Rahul Ghosh

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy

Vipin Kumar

December, 2023

Acknowledgements

During my time in graduate school, I was incredibly fortunate to have the support and patience of a number of individuals in my life. As I stand here at this significant moment in my life's journey, I am deeply indebted to all of you who supported and encouraged me. I want to express my deepest gratitude, although no words can fully recognize your efforts in helping me become the person I am today. To everyone who played a role, including those mentioned below, thank you.

To begin, I want to express my heartfelt appreciation to my family. I am deeply grateful to my parents for their unwavering support and encouragement in my journey towards becoming the best version of myself. Your unwavering love, motivation, and endless support, both emotional and practical, have all played a critical role in shaping my character and guiding me through the highs and lows of this demanding academic journey. It is this foundation of hard work and integrity, instilled by your values, that has been instrumental in shaping the person I am today. To my partner, Shreyasi, who has been my rock, and her support every day of my academic career has made it possible for me to get this far. Indeed, words fall short when trying to express my appreciation for your enduring love, unwavering support, and deep understanding that have been more than just emotionally uplifting; they've been my anchor. Your encouragement has given me the strength to face challenges head-on and the inspiration to aim higher. I sincerely appreciate our teamwork relationship and your assistance as my auxiliary brain. I look forward to officially being a full half of the team.

I want to express my sincere gratitude to my advisor, Prof. Vipin Kumar: thank you for your mentorship in all aspects of academia and for setting a great example of respect and support for people's goals and constraints. Despite your accomplishments, your intensity towards research and humility will always be a guiding force in my life.

Thank you for patiently teaching me numerous skills for becoming a successful researcher and providing critical insights into shaping my academic career. Lastly, I am genuinely grateful for your unwavering encouragement throughout the years. I would also like to thank Professor Ju Sun, Professor Yao-yi Chiang, Professor Zhenong Jin, and Professor John Nieber for taking the time to serve on my thesis proposal and final exam committees. Your expertise and helpful insights have shed light on many important research questions I faced and have been crucial in significantly improving the quality of the dissertation. Beyond my committee, I want to acknowledge Professor Christopher Duffy, Professor David Mulla, and Professor Paul Hanson for our collaborations over the years and for opening my mind to different research disciplines and beyond.

It is only possible for me to carry out exciting research projects with the help of talented researchers from different groups, including current and past members of the Kumar lab and Digital Agriculture group. I am grateful to my fellow lab mates, Xiaowei Jia, Praveen Ravirathinam, Arvind Renganathan, Leikun Yin, Ankush Khandelwal, Guruprasad Nayak, Kshitij Tayal, Somya Sharma, and Jared Willard for creating a productive work environment. Lastly, I'd like to express my gratitude to my friends and to all the others who have taken the time over the past five years to keep reaching out to stay in touch. I acknowledge that this work was supported by the funding agencies, NOAA, NSF, and the MNDrive initiative for their financial support of my research.

Dedication

For all the educators in my life. Your belief in me is what made this work possible.

Abstract

Personalized prediction of responses for individual entities caused by external drivers is vital across many disciplines. Recent machine learning (ML) advances have led to new state-of-the-art response prediction models. Models built at a population level often lead to sub-optimal performance in many personalized prediction settings due to heterogeneity in data across entities (tasks). In personalized prediction, the goal is to incorporate inherent characteristics of different entities to improve prediction performance. This thesis focuses on the recent developments in the ML community for such entity-aware modeling approaches. ML algorithms often modulate the network using these entity characteristics when they are readily available. However, these entity characteristics are not readily available in many real-world scenarios, and different ML methods have been proposed to infer these characteristics from the data. This thesis organizes the current literature on entity-aware modeling based on the availability of these characteristics as well as the amount of training data. It proposes novel ML methods for several of those scenarios in the context of environmental modeling. Further it highlights how recent innovations in other disciplines, such as uncertainty quantification, fairness, and knowledge-guided machine learning, can improve entity-aware modeling.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In many real-world environmental and physical sciences applications, there is a need to build robust personalized prediction models for sets of entities (or tasks). For example, the entities can represent a set of hydrological basins [2], and the objective is to model the rainfall-runoff process for several such basins for understanding hydrology cycles, predicting floods and droughts and making operational decisions such as reservoir release. Streamflow also determines the transport of aquatic pollutants across the landscape because water is the primary medium for the movements of particles and dissolved compounds. Similarly, in the healthcare domain, monitoring disease progression among patients (entity) or groups driven by external drivers, demographic or genetic information, and received treatments is essential for understanding the disease dynamics and downstream prediction task [3]. Apart from scientific disciplines, other real-world examples include personalized item prediction based on user behavior in e-Commerce systems [4] or forecasting traffic patterns in different cities or countries [5]. An entity can also be a physical system such as a drone or a spring-mass system where we want to model the trajectory of these systems.

External factors drive these entities, and the entity's response to the external drivers is governed by inherent characteristics specific to each entity. For example, while modeling the rainfall-runoff process of basins/catchments, the response (streamflow) for a given catchment is governed by meteorological drivers (e.g., air temperature, precipitation, wind speed) and complex physical processes (e.g., evapotranspiration, subsurface flow). The nature of these complex and interacting physical processes are best captured by the inherent characteristics of each catchment, e.g., soil properties, land covers, stream geometry, or other geographical, climatological, and geo-morphological characteristics. For example, for the same amount of precipitation (external driver), two catchments (entity) will have very different streamflow (response) values depending on their land-cover distribution (entity characteristics) [6, 7]. Similarly, in a clinical setting, a recorded treatment of medication (driver) for diabetes for a patient (entity) can have remarkably different effects (response) depending on the frequency of self-exercise (entity characteristic) [8]. Additional examples of external drivers include a person's physical activities that affect his/her heart rate. For the same physical activity, two people may have different heart rates depending on the physical fitness of each person.

For entity modeling in scientific disciplines, researchers have built process-based (PB) models to study the response from such environmental and physical systems [9, 10, 11, 12, 13, 14]. For streamflow modeling; hydrological community makes extensive use of PB models, such as SWAT (Soil & Water Assessment Tool) [15], that are built on equations which approximate the underlying bio-geophysical processes of the water cycle. These PB forward models are often described by parameterizing a mathematical model and finding proper system parameters within that model class. For these PB models to provide meaningful results, they need these catchment characteristics/model parameters. The values of these parameters are found by running the forward model for many different combinations to identify the optimal combination that leads to the best fit to observed data.

However, several challenges are encountered while using PB models. First, the PB models are necessarily approximations of reality. They can be impacted by our limited understanding of the underlying processes or our inability to represent known processes at the right level of spatial and temporal resolution (which may not be feasible due to computational constraints). Thus, modeling physical variables in scientific disciplines can be challenging for traditional PB models due to the incomplete knowledge or excessive complexity in underlying relationships amongst physical variables [16, 17, 18]. Second, PB models are often calibrated for each entity individually due to the highly heterogeneous nature of environmental systems (e.g., different entities/catchments can have widely different characteristics). They usually contain many parameters that need to be calibrated for each entity with the help of limited observed data incurring enormous computation costs to search for the optimal set of model parameters. Even in this age of big data, high-quality and plentiful observations are available only for a limited number of entities, and other entities may have limited data or no data at all. Third, the rigid structure of the PB models makes incorporating auxiliary data challenging. In many scientific applications, ancillary information about the system is available beyond the standard input and output variables. For example, streamflow in a river catchment is modeled as a function of weather drivers. However, soil moisture observations from in-situ sensors or earth-observing satellites for vegetation cover and soil moisture can provide valuable information related to underlying processes such as evapotranspiration and base flow. While in principle, such information can be incorporated in PB models via the commonly used data assimilation paradigm in scientific community [19, 20], this is not easy to do due to the sizeable semantic gap with physical quantities represented in the PB models and the multi-modal and multi-resolution remote sensing data.

The last decades have witnessed the immense success of machine learning (ML) in commercial applications where large-scale data are available, e.g., computer vision and natural language processing. Given ML models' capability to extract complex relationships from data automatically, they are increasingly being used in addressing crucial problems in scientific applications such as hydrology [21, 22], biology, and climate science [23]. These ML models learn the mapping from the external drivers to entity response and, thus, essentially emulate the PB models. In contrast to PB models, ML offers excellent potential for dealing with the high degree of heterogeneity that is always present in environmental systems by leveraging a collection of observations from a diverse set of entities to build a powerful global meta-model. The reason is that ML models can benefit from training data from diverse entities and thus can transfer knowledge across entities. Moreover, ML models are data-driven and can directly extract the statistical relationships from data, thus not being impacted by incomplete knowledge of the underlying processes. Finally, ML models can readily incorporate diverse sources of auxiliary data and have the potential to represent complex physical relationships between multiple bio-geo-physical processes that may interact at different spatial and temporal scales. However, ML models can only learn (however complex) patterns in the data used for training and thus fail on unseen data that is outside the range seen in training, whereas

PB models can make predictions for any arbitrary input variables (e.g., heretofore unseen weather patterns that may result from changing climate). Hence, it is critical to incorporate scientific knowledge into the ML framework. Such knowledge-guided machine learning (KGML) models [24, 22, 25] have been shown to improve performance over black-box ML models even with fewer samples and can generalize in unseen scenarios.

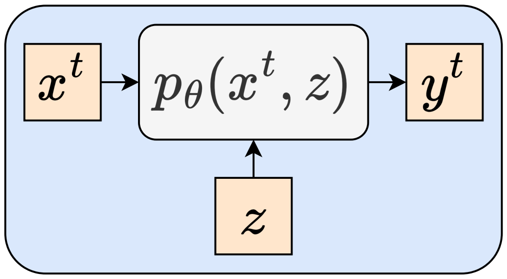


Figure 1.1: *Forward model (p_θ) which uses external drivers ($\boldsymbol{x^t}$) and entity characteristics ($\boldsymbol{z}$) to predict response ($\boldsymbol{y^t}$)*

One of the major challenges in building personalized prediction models is the lack of training for individual entities. Hence, learning individual models can be sub-optimal, as shown by numerous studies in environmental [21] and healthcare [26, 27]. On the other hand, a trivial merging of data from all entities to learn a single global model will also fail to perform well due to diverse response from entities depending on their inherent properties. Hence, KGML methods must consider these entity characteristics to model the driver-response relationship effectively. This strategy of utilizing these entity characteristics to modulate the prediction model is termed as entity-aware modeling (EAM). Figure 1.1 shows the diagrammatic representation of this EAM strategy.

The critical goal of EAM is to build a global model that effectively leverages data from different entities by incorporating entity characteristics to reduce the impact of data scarcity for each entity. These techniques have been applied in several naturally occurring scenarios, as shown in Figure 1.2. When entity-specific characteristics are explicitly available, they are often used directly in ML models for modulation [28, 21, 29, 30]. One advantage of having explicit characteristics (or learned embeddings) for each entity in the training set is that the learned model can be used for out-of-sample entities. However, these characteristics are often unknown or difficult to measure directly in many applications. Thus there is an additional need to build models that do not entirely depend on explicitly available characteristics. Such models either implicitly capture the entity/task characteristics as part of their parameter set [31] or infer entity/task embeddings from the data and use them to modulate the global network for personalization under heterogeneity [32, 33, 34]. Similarly, multi-task learning (MTL) framework [35, 36, 37] can be used but at the expense of increased model complexity because MTL generally requires separate parameters for each task.

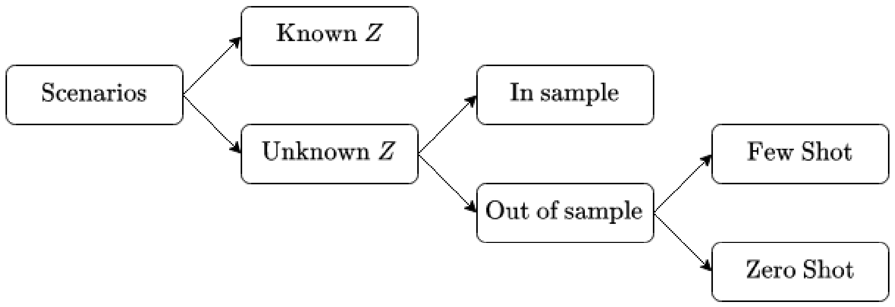
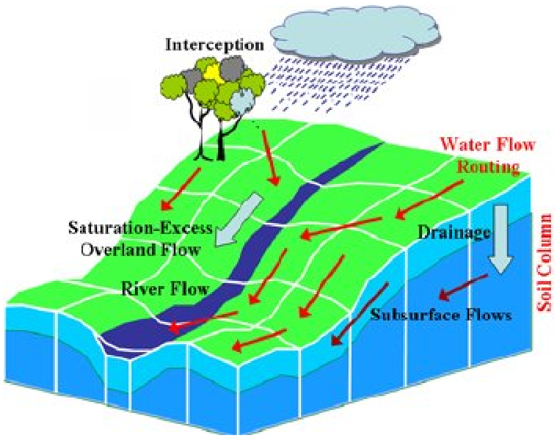
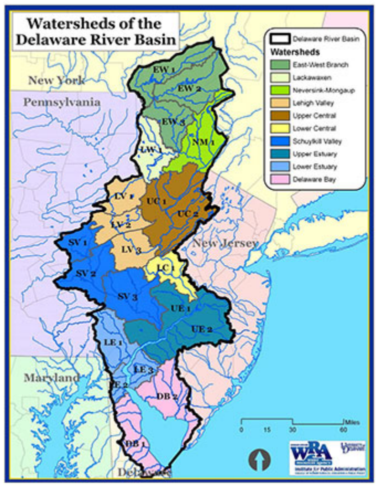


Figure 1.2: *Scenarios*

Figure 1.2 summarizes the scenarios in which the ML models are used in real-world applications. The entity characteristics $\boldsymbol{z}_i$ can be explicitly available in some scenarios. In many scenarios, measurement of entity characteristics may be partially available for some of the characteristics, noisy or uncertain, or completely unavailable. In this situation, the entity characteristics must either implicitly be part of the models or be recovered from the data as latent variables. Further, the trained ML models can be evaluated in two further settings: (1) In-Sample test: the training and testing data are from the same entities but different from each other, and (2) Out-of-Sample test: the training and testing data are from different entities and different periods. In the out-of-sample testing scenario, we further have the few-shot and zero-shot setting depending on whether we have access to few-shot datasets for the testing entities.



(a)



(b)

Figure 1.3: *Diagramatic representation of the motivating use-cases. Fig (a) Rainfall-runoff process in a river catchment [1]; Fig (b) The delaware river basin that consists of an extensive interconnected stream network affected by natural and human interventions.*

1.1 Motivating Use cases

1.1.1 Modeling streamflow:

Most processes dealing with life on earth are associated directly with the ebb and flow of water (the hydrologic cycle) in the atmosphere, on the land surface, within groundwater, in lakes, and oceans. Thus, to predict the functioning of earth's interconnected ecosystems and human systems, it is essential to quantify the storage and flux of water that define these systems. An essential task in hydrologic science is understanding and predicting flows and levels in rivers and streams. Prediction of streamflow in the river catchment (see Figure 1.3a) is used to understand water cycles, predict floods and droughts, and make operational decisions such as reservoir release. Streamflow also determines the transport of aquatic pollutants across the landscape because water is the primary medium for the movements of particles and dissolved compounds. In this regard, there are three main challenges to be addressed by EAM: First, discover efficient model structures that lead to improved forecasting of floods, drought, and water supply.

Second, advance our skill in estimating model parameters and model output and their corresponding uncertainty critical to disaster response where lives and property are at stake. Third, transfer knowledge learned from well-measured watersheds across the continent to improve local streamflow prediction, estimate missing parameters, and reduce uncertainty in ungauged or poorly gauged watersheds.

1.1.2 Monitoring Stream Networks

Healthy aquatic ecosystems are key to the future sustainability of our planet, but unfortunately, water quality continues to degrade due to pressures that range from local demand to global pressure on food, water, and energy networks. Timely monitoring can provide useful information for sound policy and management decisions. One major focus of this proposal is to predict water properties (e.g., temperature, streamflow and nutrients) in stream networks. In this problem, multiple river segments and reservoirs in a stream network can show different flow and thermodynamic patterns driven by differences in catchment characteristics (e.g., slope, soil characteristics) and meteorological drivers (e.g., air temperature, precipitation). These segments and reservoirs also interact with each other through the water flowing from upstream to downstream segments. Moreover, there are often only a handful of river segments in a network that are monitored; thus there is limited data to train ML models. Accurate prediction of streamflow and water temperature aids in decision making for resource managers, establishes relationships between ecological outcomes and streamflow or water temperature, and helps predict other biogeochemical or ecological processes. For example, drinking water reservoir operators in the Delaware River Basin (see Figure 1.3b) need to supply safe drinking water to New York City while also maintaining sufficient streamflow and cool water temperatures in the river segments that are downstream of the reservoirs to maintain the desired habitat for aquatic life and proper aquatic biogeochemical cycling [38]. Accurate prediction of water properties helps managers optimize when and how much water to release downstream to maintain the flow and temperature regimes. Existing physics-based models for modeling river networks, such as PRMS-SNTemp [39], simulate the internal distribution of water properties based on general physical relationships such as energy and mass conservation. However, the model predictions still rely on qualitative parameterizations (approximations) based on soil and surficial geologic classification along with topography, land cover and climate input. Hence, such models can only provide sub-optimal prediction performance. Furthermore, calibration of physics-based models for river networks is often extremely time intensive due to interactions among parameters within segments and across segments and also requires expert knowledge of the system and model to calibrate successfully.

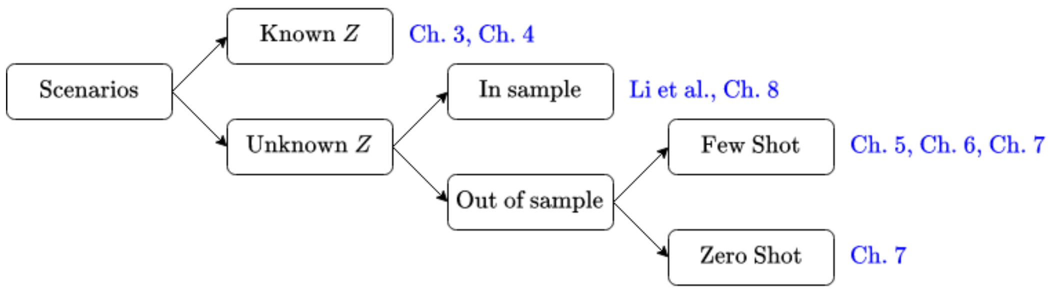


Figure 1.4: *Scenarios and corresponding chapters that contribute to the advancement of methods.*

1.2 Thesis Overview

This thesis proposes methods that contribute to identifying dominant hydrologic processes (causal analysis), estimate parameter uncertainty estimates, increase confidence in streamflow predictions, and facilitates knowledge transfer across watersheds. While the proposed strategies for EAM are in the context of hydrology and modeling stream networks, both crucial environmental applications, these approaches will be applicable to a range of scientific disciplines. Following we provide a brief description of the chapters 1.4.

Chapter 2 provides a survey of the recent developments in the ML community for such entity-aware modeling approaches. ML algorithms often modulate the network using these entity characteristics when they are readily available. However, these entity characteristics are not readily available in many real-world scenarios, and different ML methods have been proposed to infer these characteristics from the data. This chapter organizes the current literature on entity-aware modeling based on the availability of these characteristics as well as the amount of training data.

Chapter 3 introduces Cross-LSTM, a novel LSTM variant explicitly designed to handle cross (multiplicative) interactions. While Deep Neural Networks (DNNs), especially LSTMs, are powerful for scientific applications, they struggle to efficiently model higher-order multiplicative interactions, resulting in increased memory costs and demanding training requirements. This motivates the need for models that can efficiently learn all types of cross features, to model the complex dynamic nature of environmental systems. Cross-LSTM incorporates cross (multiplicative) interaction explicitly rather than modeling them implicitly through non-linear activations.

Chapter 4 proposes a new meta-transfer learning framework to transfer knowledge from well-observed entities to unmonitored entities. 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. Proposed method creates meaningful similarity estimates amongst entities to guide the transfer learning process

Chapter 5 proposes a novel Knowledge-guided Self-Supervised Learning (KGSSL) inverse framework. Existing basin characteristics suffer from noise and uncertainty, among many other things, which adversely impact model performance. To tackle the above challenges, KGSSL extracts system characteristics from driver (input) and response (output) data. This first-of-its-kind framework allows to achieve robust performance in such scenarios while also allowing discovery of characteristics even when they are completely unknown.

Chapter 6 proposes a Entity aware modulation using Representation Learning (EAM-RL) framework that can serve as an alternative to existing state-of-the-art approaches, such as MAML and MMAML, in the absence of entity characteristics. EAM-RL extracts embeddings representing the actual inherent characteristics of these entities and uses that to personalize the predictions for each entity.

Chapter 7 introduces Entity-aware Conditional Variational Inference (EA-CVI), a novel probabilistic inverse modeling approach, to deduce entity characteristics from observed driver-response data. EA-CVI infers probailistic latent representations that has the ability to accurately predict response for diverse entities, particularly in out-of-sample few-shot settings. EA-CVI's latent embeddings can encapsulate diverse entity characteristics within compact, low-dimensional representations. EA-CVI is able to identify dominant modes of variation in responses and offers a physical interpretation of the underlying attributes that shape these responses. EA-CVI is also able to provide a robust estimate of uncertainty, particularly during extreme evenets, which is essential for data-driven decision-making in real-world applications.

Chapter 8 presents a knowledge-guided machine learning (KGML) framework for modeling modes in multi-scale processes for streamflow forecasting in hydrology. Specifically, it proposes a novel hierarchical recurrent neural architecture that factorizes the system dynamics at multiple levels of temporal granularity. Based on inverse modeling, this framework can empirically resolve the system's temporal modes from data (physical model simulations, observed data, or a combination of them from the past) and use them to improve the accuracy of the forecast. The model's hierarchical structure allows it to understand the system's dynamics comprehensively, capturing both rapid fluctuations and long-term trends. Once trained, this framework makes it possible to assimilate observations without requiring loss function optimization (e.g., Kalman Filtering) to improve forecast accuracy.

Chapter 2

Problem Formulation and Literature Review

Personalized prediction of responses for individual entities caused by external drivers is vital across many disciplines. Recent machine learning (ML) advances have led to new state-of-the-art response prediction models. Models built at a population level often lead to sub-optimal performance in many personalized prediction settings due to heterogeneity in data across entities (tasks). In personalized prediction, the goal is to incorporate inherent characteristics of different entities to improve prediction performance. In this chapter, I focus on the recent developments in the ML community for such entity-aware modeling approaches. ML algorithms often modulate the network using these entity characteristics when they are readily available. However, these entity characteristics are not readily available in many real-world scenarios, and different ML methods have been proposed to infer these characteristics from the data. This chapter is organized as follows. Section 2.1 first formulates the problem encountered in predicting the response for a diverse set of entities and discusses the different scenarios in this problem. Section 2.2 then organizes the diverse ML research threads proposed over the years that can be leveraged to tackle this EAM task and discusses the overarching themes between methods and applications.

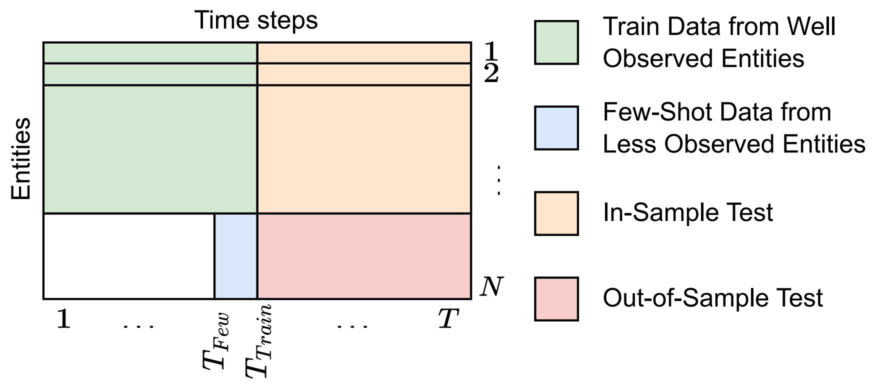

Time steps
Entities
1
2
N
1
T
T_{Few}
T_{Train}
Train Data from Well Observed Entities
Few-Shot Data from Less Observed Entities
In-Sample Test
Out-of-Sample Test

Figure 2.1: *Problem Setting*

2.1 Problem Formulation

This thesis focuses on the setting where there are a set of N entities/tasks, as shown in Figure 2.1. There exists a variability in the amount of training data available for entities - abundant, sparse or none. There may be a subset of well-observed entities, such that for each entity i in this set, we have access to a training dataset $\mathcal{D}_i = \{(x_i^1, y_i^1), (x_i^2, y_i^2), \ldots, (x_i^{T_i^{Train}}, y_i^{T_i^{Train}})\}$, with (drivers, response) pairs. This is denoted by the green region in Figure 2.1. From the remaining entities, there may be a subset of less-observed entities where, for each entity j in the remaining set, we have access to a few-shot dataset $\mathcal{D}_j = \{(x_j^1, y_j^2), \ldots, (x_j^{T_{Few}}, y_j^{T_{Few}})\}$. This is denoted by the blue region in Figure 2.1. The rest of the entities are completely unobserved.

The objective is to learn the mapping function from input variables x_i^t to target variables y_i^t. In conventional supervised machine learning, we train a predictive model $\hat{y}_i^t = p_{\theta_i}(x_i^t)$, parameterized by θ_i, by finding the parameters that minimize the empirical risk on the training data:

$$\theta_i^* = \arg\min_{\theta_i} \mathcal{L}(\mathcal{D}_i; \theta_i) \tag{2.1}$$

Given sufficient training data for each entity, we can train individual ML models that capture these inherent biases in each entity within the learned parameter set θ_i^*. However, the data from all the entities are combined due to the lack of training data to learn a robust model for each entity. Learning a global model by the trivial merging of data from different entities can lead to sub-optimal results due to the heterogeneity across different sites. Because these entities are differentiated by their inherent characteristics $\boldsymbol{z}_i$, the functions are of the form $\hat{y}_i^t = p_\theta(x_i^t, \boldsymbol{z}_i)$, where θ denotes the function class shared by the target systems and $\boldsymbol{z}_i$ denotes entity-specific inherent characteristics as shown in Figure 1.1.

Figure 1.2 summarizes the scenarios in which the ML models are used in real-world applications. The entity characteristics $\boldsymbol{z}_i$ can be explicitly available in some scenarios. In many scenarios, measurement of entity characteristics may be partially available for some of the characteristics, noisy or uncertain, or completely unavailable. In this situation, the entity characteristics must either implicitly be part of the models or be recovered from the data as latent variables. Further, the trained ML models can be evaluated in two further settings: (1) In-Sample test: the training and testing data are from the same entities but different from each other (shown by yellow region in Figure 2.1), and (2) Out-of-Sample test: the training and testing data are from different entities and different periods (shown by red region in Figure 2.1). In the out-of-sample testing scenario, we further have the few-shot and zero-shot setting depending on whether we have access to few-shot datasets for the testing entities. Next, we describe the most relevant literature for these scenarios.

2.2 Literature Review

This section aims to organize the diverse ML research threads proposed over the years that can be leveraged to tackle this EAM task. Here I discuss the overarching themes between methods and applications. Furthermore, I enumerate the gaps and opportunities for advancing research in each direction.

2.2.1 Known $\boldsymbol{z}$

In the scenario where entity-specific characteristics are known, the most standard approach is to modulate the network using them [28, 30]. Several studies have used these source characteristics by concatenating them along with the drivers and then passing them into the ML models [21]. Other approaches have used entity-specific characteristics/features to modulate the architecture or transform the input features. In the context of store placement prediction in multiple cities, [29] use the city-specific parameters in an attention network to modulate and adapt the base feature extractor. Further, [40] perform feature-wise affine transformation based modulation using entity characteristics based conditioning. Similarly, [41] explicitly model the interaction of the input drivers and entity characteristics through a deep and cross network. One advantage of having explicit characteristics for each entity is that the learned model can be used for out-of-sample entities. Thus we do not further divide this scenario based on whether the ML models are being applied in the in-sample or out-of-sample setting, which is trivial. However, all the methods discussed in the later settings can be easily adapted for this scenario. A challenge commonly faced in this scenario is handling the model's bias towards certain types of entities caused due to the fact that the training set may be imbalanced in the types and occurrences of entities, as discussed in Section ??.

2.2.2 Unknown z & In-sample

When the characteristics are unavailable, EAM methods that learn to leverage entity relationship implicitly is required. **Multi-task Learning** (MTL) is the most common approach used for several personalized prediction applications in this setup, such as mood prediction [35], app-usage prediction [36], and human mobility prediction [37]. The different tasks/entities (used interchangeably here) share a common network in multi-task learning followed by task-specific weights to achieve personalization. However, there are two main challenges with MTL for personalized prediction. First, the number of entity-specific weights increases rapidly with the number of entities. Furthermore, the shared network should have sufficient capacity to handle a large set of entities [42]. Hierarchical Dirichlet processes have been used to combine similar tasks at the expense of the increased model and computational complexity. Recently, [43] proposed a deep MTL framework that aims to learn entity similarity from the data to reduce the impact of limited training data while training entity-specific parameters. Another approach in this scenario is to train global models by assigning one-hot/random vector to each entity [44]. Here the entity-specific parameters do not depend on the number of entity but on the dimensionality of random characteristics, thus reducing the high model complexity of the MTL framework. More complex methods, such as meta-learning (discussed later), can also be applied in this scenario.

2.2.3 Unknown z, out-of-sample & few-shot

When few samples of observation are available for the out-of-sample entities, few-shot learning [45] methods can be used. MTL methods are not the right approach as a model trained using traditional learning schemes is not easily adaptable to a different set of entities [31]. The solution is to use the few-shot data to either adapt the models to the new entities through gradient-based optimization or infer the entity characteristics and use them to modulate the prediction model.

Meta Learning: Recently meta-learning has gained much attention in few-shot learning applications by leveraging the shared structure between existing training tasks, leading to better generalization and adaptation [46]. In particular, Model Agnostic Meta Learning (MAML) [31] aims to learn a global meta model that can be easily adapted to create personalized models for each entity. This is commonly done by formulating the training scheme as a bi-level optimization problem:

$$\begin{aligned} &\theta^* = \arg\min_{\theta} \sum_{i \in \mathcal{P}(i)} \mathcal{L}(\mathcal{D}_i^{val}; \theta_i^*) \\ \text{s.t.} \quad &\theta_i^* = \arg\min_{\theta} \mathcal{L}(\mathcal{D}_i^{train}; \theta) \end{aligned} \tag{2.2}$$

During meta-training, the individual models θ_i are finetuned for each entities using their meta-training samples $\mathcal{D}_i^{train}$. These individual models are used to calculate the loss on meta-test samples $\mathcal{D}_i^{val}$, which serves as the training error for the meta model θ^*. This meta-model can adapt to each entity using one or a small number of gradient steps to find the task-adapted parameter of the prediction model.

Meta-learning has gained much attention in recent years for several EAM tasks. [47] obtain a high-performance personalized model using meta learning and few-shot entity data. [26] bridge the modeling of infrequent patients (entites) and rare diseases (tasks) by designing a meta learning approach based on hierarchical patient subtyping mechanism. [48] show the benefit of meta-learning over individual models in forecasting a diverse set of air pollution. [49] developed a MAML framework for multiple clinical risks prediction in healthcare application. Other applications include using the prior consumption data from multiple source cities to predict optimal store placement in a new city [29].

Adapting the whole parameter set may put extensive burden on the optimization procedure, possibly biasing the solution of the inner-level optimization. Recently, variations of MAML have been proposed that adapt only the high-level layers instead of the whole meta-model [50, 51]. This strategy can be adapted for the EAM modeling. The key idea is to freeze the prediction model in the inner loop and assume only entity characteristics as a trainable vector. This strategy has been used in engineering [28], finance [52], and vision domains [53]. Recently, [54] used an invertible neural network to infer lake attributes using a few observations. Similarly, [55] proposed to jointly identify and predict systems using similar bi-level optimization of MAML:

$$\begin{aligned} \theta^* &= \arg\min_{\theta} \sum_{i \in \mathcal{P}(i)} \mathcal{L}(\mathcal{D}_i^{val}; \theta, z_i^*) \\ \text{s.t.} \qquad z_i^* &= \arg\min_{z} \mathcal{L}(\mathcal{D}_i^{train}; \theta, z) \end{aligned} \tag{2.3}$$

Here, the total parameters are separated into prediction model θ shared by the target entities and entity-specific characteristics z. Additionally, many MAML-based methods assume that all train and test entities are drawn from the same distribution. Thus, a single meta-initialization could be challenging to adapt due to the data distribution in different entities being different and multimodal [32]. Furthermore, the training process is computationally expensive and sensitive to hyperparameter choices [56].

Conditional Meta Learning: If the entity distribution is multi-modal with disjoint and far apart modes (e.g. patients/groups from different countries), a set of separate meta-learners could better master the full distribution. Several strategies have been proposed to learn meta-learners that acquire mode-specific prior parameters by formulating the bi-level optimization problem as,

$$\begin{aligned} \theta^* &= \arg\min_{\theta} \sum_{i \in \mathcal{P}(i)} \mathcal{L}(\mathcal{D}_i^{val}; \theta_i^*) \\ \text{s.t.} \quad \theta_i^* &= \arg\min_{\theta} \mathcal{L}(\mathcal{D}_i^{train}; \theta) \quad \textit{and} \quad \theta = \mathcal{T}(\boldsymbol{z}) \end{aligned} \tag{2.4}$$

Here, the meta-model is conditioned on additional side information $\mathcal{T}(\boldsymbol{z})$ that contains descriptive features associated to the entity/task. Several works have been proposed that advocate this conditional perspective. They have been called several names such as heterogeneous meta learning [57], conditional meta learning [58] or multi-modal meta learning [32].

Associating each entity with one of the meta initializations require additional entity characteristics, which is often unavailable or could be ambiguous when the modes are not disjoint. Under this setting, the most common strategy is to learn another network that converts training data from seen entities into entity-specific embeddings that modulate the shared prediction network [32]. The prediction and embedding networks can be trained either jointly or alternately. [33] learn embedding metric space that characterizes disease (entity) relationships for disease prediction and shows promising results for solving the data scarcity problem in healthcare decision support. [59] learn a mixture of hierarchical Bayesian models by incorporating entity-specific parameters as latent variables. This allows the meta-learner to perform entity-specific parameter selection instead of consolidating inductive biases into a single meta-model. Further, [5] showed that utilizing the traffic data from data-rich cities improves the prediction in cities with only a short period of data using a conditioned MAML (CMAML) based approach. Similarly, [60] proposed a graph-based conditional meta-learning approach for predicting water quality and quantity variables in a diverse set of basins. One challenge of CMAML that needs to be addressed is quantifying the diversity needed to merit these methods. Further, most of CMAML methods utilize a metric that measures the similarity between entities. Thus we need to investigate novel metrics that better capture this similarity.

Neural Process: The neural process (NP) family has been used in EAM in a variety of fields, including robotics, computer vision, and natural language processing [61]. The Neural Process (NP) family of methods started with Conditional Neural Processes (CNPs) [62], which combine the benefits of Deep neural networks and Bayesian methods, such as Gaussian Processes (GPs), to exploit prior knowledge and quickly infer the shape of a new function. The defining characteristic of the NP framework is that it conditions the prediction function on the observations via an inferred entity embedding. The resulting model can be boiled down to three core components, as shown below,

$$\begin{aligned} h_c &= q_\phi(x_c, y_c) && \text{encoder} \\ z &= h_1 \oplus \cdots \oplus h_n && \text{aggregator} \\ y_c &= p_\theta(x_c, z) && \text{conditional decoder} \end{aligned} \tag{2.5}$$

Here, the encoder produces a representation from each (input, output) pair, that are aggregated to form an embedding. The conditional decoder outputs the target predictions using the embedding and inputs. Further advancements have been proposed, such as introduction of latent variables [63] instead of deterministic embeddings, using bootstrapping for multiple latent variables [64], or introducing attention-based versions [65].

The NP framework has found applications in a wide range of domains, given its flexibility, modeling capacity, and computational efficiency. NPs have been used in designing recommender systems [34] for personalized prediction of an item for each user (entity). In neuroscience, NPs have been used to predict the responses of neurons (entities) in the visual cortex to natural stimuli [66] and neural spike sorting [67]. [68] propose a Multi-fidelity Hierarchical Neural Process (MF-HNP) that can leverage the cheap data from low-fidelity simulators for epidemiology tasks across individuals from multiple age groups and climate modeling for diverse sites. [69] propose an extension to the NP framework for multi-task classification settings that can quickly adapt to a new task without costly retraining. [22] use a self-supervised contrastive loss to infer the entity embeddings for personalized streamflow prediction.

Despite the success of the NP framework, there exist open challenges and limitations that need to be addressed. Deciding the aggregator function is still an open direction of research [70]. Further in many scientific process there are multiple processes within an entity which can lead to multiple contexts, as discussed in Sec ??. Thus a potential direction is to impose a manifold structure on the latent distribution or a hierarchy among the latent distributions from the diverse contexts for the same entity.

2.2.4 Unknown z, out-of-sample & zero-shot

In many scenarios, a good model is expected for these out-of-sample entities, despite collecting high-quality data for all possible entities (e.g., abnormalities/diseases in healthcare) being challenging. While meta-learning is the common approach for few-shot learning scenarios, it cannot be used when we have no data available for out-of-sample entities (zero-shot setting). Since there is no data/knowledge about the entity characteristics, the characteristics should only be inferred from the training entities' drivers and responses. There could be multiple choices for the latent characteristics that can yield the same data distribution. However, only one (or a small subset) contributes a robust model. Additionally, the global model could be biased toward the in-sample entities.

Disentangled Representation Learning: Disentangled representation learning is proposed to partition the hidden representation h into independent factors of variations, which are aligned with data generative factors [71]. For example, in the image classification task, a disentangled representation might encode the shape and color of an object separately. Based on generative models like variational autoencoders (VAEs) and generative adversarial networks (GANs) structure, disentanglement is encouraged by new regularizations and training techniques [72, 73]. Better disentanglement could be achieved if there is more information about the latent generative factors, like hierarchical priors or a group of entities sharing a common factor [74, 75]. To build an entity-aware model using disentangled representations, one option is to separate the entity-dependent representations from the representations which are shared by all the entities, i.e., $h_i = [h_{shared}, z_i]$. Recent progress in disentangled representation learning provides opportunities for this approach. For example, using identification label, [76] introduced identity shuffle GAN (IS-GAN) to disentangle identity-related (e.g., clothing) and unrelated features (e.g., human pose) from person images. [77] introduced a disentangled sequential auto-encoder. The latent representation is learned to separate time-independent (e.g., static entity characteristics) and time-dependent features (e.g., states of the entity).

State Space Model: State space model (SSM) is a model designed for sequential data, which assumes the observational data is generated from latent state variables through *emission model*. The transitions between latent states are modeled by *transition model*. Given observations $[x^t, y^t]$ and latent states h^t, the vanilla state space model [78] can be formulated as

$$\begin{aligned} h^t &= g(h^{t-1}) + \varepsilon_z && \textit{(transition model)} \\ [x^t, y^t] &= f(h^t) + \varepsilon_x && \textit{(emission model)} \end{aligned} \tag{2.6}$$

According to the equation above, the hidden Markov model (HMM) can be seen as a special state space model, where the latent state is discrete and the transition only depends only on the previous latent state. Recently, researchers added neural structures to the conventional state space model for better approximation and to learn nonlinear latent states. The deep SSMs are usually solved by variational learning algorithm, which includes a inference network to approximate the intractable posterior of latent states and a generative model to approximate the transition and emission model. [79] proposed an inference algorithm to learn continuous latent states of deep Markov models (DMMs), where the emission distributions are modeled by deep neural networks, and the transition distributions are estimated by an RNN-based inference network. [3] put attention mechanism on latent states to investigate the dependence between the current and all past states, which generalize the transition model in Eq.2.6. The authors also proposed an inference algorithm for a discrete latent state. [80] further combined state space model with transformer architectures, which uses attention mechanism instead of RNNs to model latent state dynamics. Given its flexibility and interpretability, the state space model is widely used for time series modeling and forecasting in different domains like computer vision [81], and healthcare [82]. Recent progress in the state space model provides great promise for EAM. The overall idea is to use prior knowledge about entities to learn better latent state representations and dynamics. For example, [27] introduced a model that allows the transition of latent states depends on the characteristics of entity (e.g., genetics, demographics). The learned latent representations thus implicitly captures the entity-related knowledge (e.g., clinical phenotypes, pharmacodynamic) from the observations. In computer vision domain, [81] used a Kalman variational auto-encoder (KVAE) as inference network. The KVAE is designed to separate the object's representation from latent state describing its dynamics in a unsupervised manner, which overlaps with disentangled representation.

Causal Representation Learning: This group of methods focus on the discovery of latent causal variables and the robust prediction in the downstream task [83]. Most disentangled representation learning methods are insufficient to learn causal representations since they try to disentangle independent factors from observations. However, causal factors are usually dependent on each other, which forms an underlying causal structure. To fill this gap, a line of recent work focuses on recovering the causal representation from the disentangled factors. [84] proposed to add a causal layer in the VAE-based model to transform independent factors into causal representation. [85] introduced a weakly supervised disentanglement method when the dependency among hidden generative factors is only caused by confounders (common parents). [86] used a trainable structural causal model as the prior distribution to enforce causal disentanglement, instead of an independent one. In causal representation learning, the *Causality*

Assumption [87] states that the environment e does not change the relationship between covariates X and target variables Y.The environment $e \in \mathcal{E}$ is a special case of an entity, which is also referred to as experimental setting, sub-population, or perturbation. For example, basins from different locations, and different patient populations can be seen as different environments. Leveraging such invariance across entities could yield a robust model. Given the assumption that environment e only change the distribution of covariates X, recent progress show empirically and theoretically that causal representations enable out-of-distribution generalization [88].

Entity-aware models built on causal representation can resist the distributional shifts induced by interventions, and selection bias. However, this approach may fail when entities have divergences other than distributional shifts. Further, domain knowledge on latent causal variables/mechanisms could be critical for the causal identification [89]. Successful adoption of the causal representation learning methods require addressing the challenge of the identifiability of causal variables [71]. The detailed discussion of the problem of identifiability can be found in section ??.

Chapter 3

Deep and Cross LSTM for Modeling Complex Dynamic Systems

3.1 Introduction

As water is kept in basins for later redistribution globally in most of the world, streamflow prediction is an issue that is highly societally relevant. We need reasonable estimates of the time patterns of streamflow in each basin to estimate the release schedules necessary to meet societal contracts and environmental laws. A wide variety of scientific models can be considered a mapping function between drivers x_t (e.g., weather drivers, climate forcings) and response y_t (e.g., streamflow in a river basin, global average temperature). However, each entity's inherent characteristics (e.g., slope, land cover) best capture the evolution of these complex physical processes, i.e., for a given entity (river-basin/catchment), drivers (meteorological data, e.g., air temperature, precipitation, wind speed), and complex physical processes specific to each entity ([2, 6]) govern its response(streamflow). For example, two river basins will have different streamflow response values for the same amount of precipitation depending on their land-cover type. Thus, the data-driven methods must consider these ancillary basin characteristics to distinguish basins and effectively model these relationships. Deep neural networks are beginning to provide remarkable performance in various environmental time series applications such as rainfall-runoff modeling in a hydrologic basin that predicts streamflow $\boldsymbol{y_t}$, given drivers $\boldsymbol{X_t}$, and entity characteristic $\boldsymbol{x^s}$, due to their ability to model arbitrarily complex functions ([21]). In most of these scientific models, the cross feature interaction of static entity characteristics with dynamic characteristics governs the evolution of y_t.

Traditionally LSTMs are extensively used for environmental modeling where static and time-series variables are supplied as input (here, static characteristics are repeated at each time-step). Recently, EA-LSTM ([21]), and CT-LSTM have emerged as the state-of-the-art hydrological model that allow processing the time-series meteorological drivers conditioned on static characteristics. In general, LSTMs are a big step in what we can accomplish with RNNs for a prediction task in a multivariate time series. However, the original RNN models were not designed to exploit cross-feature interactions between static and dynamic features. DNNs are known to capture complicated interactions through deep but thin layers due to their many nonlinear activated neurons, giving DNNs their high functional capacity. Intuitively, ReLU/Nonlinear activations can approximate multiplicative interactions (crosses). However, as shown in ([**?**, **?**, 41]) practically, DNNs are inefficient to even approximately model second or third-order cross feature, as they implicitly generate the interactions.

This brings the question, is there another big step? Exploring potentially rare cross features is essential in predicting environmental systems better due to their complex dynamic nature. This motivates the need for models that can more easily express and efficiently learn all types of cross features. The traditional way of solving this issue is to further increase the model capacity through wider or deeper networks([**?**, **?**, 41]), but doing so is often at the cost of significantly more challenging training, higher complexity, and often a need for a large amount of data as the generalizable capacity of DNN are proportionally balanced with the amount of supervision in the training data([90]). Another way of modeling feature crosses is to explicitly model and introduce multiplicative operations($x_1 \times x_2$), but they are inefficient in DNN ([41, 91]).

We propose a novel variant of LSTM called Cross-LSTM that borrows from and builds further on the idea in ([41]) to let every step of an RNN pick the cross(multiplicative) information more efficiently. Unlike a traditional LSTM where we directly multiply inputs with hidden weights and multiply hidden vectors with hidden weights, we create an explicit feature cross at every gate, rather than just using nonlinear activations to model those interactions implicitly. Cross-LSTM is more expressive yet remains cost-efficient at modeling complex multiplicative interactions in dynamic systems. We demonstrate the usefulness of Cross-LSTM in streamflow modeling using CAMELS (Catchment Attributes and MEteorology for Large-sample Studies), a widely used hydrology benchmark data set. In a comprehensive empirical study, we observed that the Cross-LSTM approach outperforms the state-of-the-art methods like CT-LSTM and EA-LSTM by 16.20% and 23.64% in streamflow prediction for out-of-sample testing on the popular benchmark CAMELS. The improved Cross-LSTM is more expressive yet remains cost-efficient at modeling complex multiplicative interactions in dynamic systems.

3.2 Related Work

Most of the current state-of-the-art deep learning approaches for predicting streamflow [21] are modeled using a standard LSTM. However, in hydrology, just like in many other complex environmental systems, higher-order crosses exist between input features due to their complex dynamic nature. However, recent studies have shown that nonlinear activations are inefficient in capturing explicit feature crosses (multiplicative relations) even with a deeper and wider network, for even second-order feature crosses [41, **?**, **?**, 91]. This means, to effectively and efficiently model complex dynamic systems, we need an efficient method that explicitly incorporates them. Many recent works [41, **?**, 92] have tried to tackle this challenge, most of them have the key idea of leveraging the implicit high-order crosses implicitly learned from DNNs, with explicit and bounded-degree feature crosses which are known to be effective in linear models. Another work [91] empirically showed that it becomes increasingly difficult for a neural network to fit the dot product function as the number of samples needed scales polynomially with the increasing dimensions and reduced error. This motivates us to design a model that incorporates cross(multiplicative) interaction more efficiently rather than modeling them implicitly through nonlinear activations. The procedure of our work builds upon the work in [41] which proposed a new model DCN-V2 to model explicit crosses in an expressive yet simple manner. However, our problem domain of learning crosses in complex time-varying physical processes is fundamentally different, and is more suited for RNNs, hence the need for modifying LSTM architecture rather than just using the model formulated in [41] which dealt with static feature space.

3.3 Methods

In this work, we study the driver-response relation in dynamical systems. Specifically, we assume a dataset consisting of N entities (an entity can be a lake, river-basin or streams in a river-network). For each entity i, the daily drivers are 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 input vector at time $t \in T$ with D_x dimension. $\boldsymbol{x_i^s} \in \mathbb{R}^{D_s}$ denotes the static characteristic vector of an entity with D_s dimensions. The daily response corresponding to $(\boldsymbol{X_i}, \boldsymbol{x_i^s})$ for an entity is denoted by $\boldsymbol{Y_i} = [y_i^1, y_i^2, \ldots, y_i^T]$. Our final objective is to predict target variables for each location $i \in \{1, ..., N\}$, and on each date t $\in \{1, ..., T\}$.

3.3.1 Data Description

We use the CAMELS (Catchments attributes and meteorology for large sample studies) dataset, extensively used for investigating hydrology processes, particularly streamflow prediction [93]. CAMELS provides a dataset encompassing 671 watersheds/basins across the contiguous US. Each basin i is supplied with observed streamflow discharge (y_i^t) and multi-variate meteorological driver data (X_i^t) from ground observations and remote sensing products at a daily scale. Meteorological inputs are daily precipitation, minimum air temperature, maximum air temperature, average short-wave radiation, and vapor pressure. Daily meteorological weather inputs and discharge data cover reasonably long records spanning from 1980 to 2014. In addition, each basin i is characterized from climatology, geomorphology, and geology perspectives by 27 basin characteristics. All of these 27 basin characteristics are relatively static over time; hence they are static in the CAMELS dataset. All of these 27 basin characteristics and the weather drivers make it possible to leverage recent developments in machine learning, particularly deep learning, in the hydrology community to advance continental hydrology modeling [21]. In particular, for the CAMELS dataset, Kratzert et al. [21] showed that a global scale LSTM model (that uses known static characteristics as input in addition to weather drivers) can outperform state-of-the-art physics-based hydrological models that are individually calibrated for each basin.

3.3.2 Architecture

Complex environmental models commonly have higher-order feature crosses between their dynamic drivers and static basin characteristics. The feature crosses associated with these models often cannot be easily learned by standard RNNs like LSTM, which uses a nonlinear activation function in its gates. These feature crosses often determine how the model states change in response to external inputs. For example, two river basins will have very different streamflow response values for the same amount of precipitation depending on their land-cover type.

To represent these crosses, we propose a Cross-LSTM model architecture. Essentially, Cross-LSTM builds upon a standard LSTM. It also uses a set of gating variables to control the influence from different sources/drivers, including the inputs at the current time step, model states from the previous time step, and explicit feature crosses.

Similar to the standard LSTM model, Cross-LSTM preserves a model state C_t at time t, which serves as a memory and is updated over time. Like an LSTM, Cross-LSTM also outputs a hidden representation h_t at every time step, which encodes the temporal information about the basin. The core of idea of Cross-LSTM lies in the use of cross layers. Unlike a traditional LSTM where we directly multiply inputs with hidden weights, and multiply hidden vectors with hidden weights and hope for nonlinear activation to learn crosses, we create an explicit feature cross at every gate, Which means instead of something like $x_t^d U^i + b_x^i$, we instead model that tensor multiplication as $x^s \odot (x_t^d U^i + b_x^i) + x_t^d + x^s$ to explicitly incorporate feature crosses with static basin characteristics like in [41].

Let x_t^d be the dynamic features at every timestep, and x^s the static basin features.

$$\text{Let } x_t = [x_t^d, x^s] \tag{3.1}$$

x_d of dimension q and x_s of z dimension are both first encoded to the same dimension through a fully connected neural network before they are fed into the model to enable point wise multiplication $\odot$.

$$x^s = x^s \times A^T + B \tag{3.2}$$

$$x_t^d = x_t^d \times C^T + D \tag{3.3}$$

where, $\{A \in R^{d\times z}$, $B \in R^{d}$, $C \in R^{d\times q}$, $D \in R^{d}\}$ are the learnable model parameters.

Just like in an LSTM, we first generate a candidate state $\tilde{C}_t$ by combining x_t and h_{t-1} using a $\tanh(\cdot)$ function in the following manner.

$$\tilde{C}_t = \tanh(x^s \odot (x_t^d U^g + b_x^g) + x_t^d + x^s + x^s \odot (h_{t-1} W^g + b_h^g) + h_{t-1}) \tag{3.4}$$

where, $\{U^g \in R^{dxd}$, $b_x^g \in R^d$, $W^g \in R^{dxd}$, $b_h^g \in R^d\}$ are the learnable model parameters.

Next, we generate two sets of gating variables: forget gating variables f_t, input gating variables g_t. These gating variables, which now incorporate explicit crosses, are used as filters for the information passing from the previous time step to the current time step. They are further computed using a $\sigma(\cdot)$ function in the following manner,

$$i_t = \sigma(x^s \odot (x_t^d U^i + b_x^i) + x_t^d + x^s + x^s \odot (h_{t-1} W^i + b_h^i) + h_{t-1}) \tag{3.5}$$

$$f_t = \sigma(x^s \odot (x_t^d U^f + b_x^f) + x_t^d + x^s + x^s \odot (h_{t-1} W^f + b_h^f) + h_{t-1}) \tag{3.6}$$

where, $\{U^f \in R^{dxd}$, $b_x^f \in R^d$, $W^f \in R^{dxd}$, $b_h^f \in R^d$, $U^i \in R^{dxd}$, $b_x^i \in R^d$, $W^i \in R^{dxd}$, $b_h^i \in R^d\}$ are the learnable model parameters.

Once the gating variables are obtained, we use them to filter the information from the previous time step and current time step by using $\odot$ in the following manner.

$$C_t = \sigma(f_t * C_{t-1} + i_t * \tilde{C}_t) \tag{3.7}$$

i.e the current state is conditioned upon forget gate and input gate. A Cross-LSTM model can represent and learn complex dynamic systems by varying these learnable parameters.
Once model state C_t is obtained, we generate the output gating variables o_t and use it to filter the model state to compute the hidden representation h_t in the following manner.

$$o_t = \sigma(x^s \odot (x_t^d U^o + b_x^o) + x_t^d + x^s + x^s \odot (h_{t-1} W^o + b_h^o) + h_{t-1}) \tag{3.8}$$

$$h_t = \tanh(C_t) * o_t \tag{3.9}$$

where, $\{U^o \in R^{dxd}$, $b_x^o \in R^d$, $W^o \in R^{dxd}$, $b_h^o \in R^d\}$ are the learnable model parameters.

Once the hidden representation is obtained we generate the predicted target variable by a feed-forward layer $\mathscr{I}$ on hidden representation h_t to estimate $\hat{\boldsymbol{y}} = \mathscr{I}(\boldsymbol{h})$, and then we define MSE loss using observed target variable $\mathrm{Y} = \{y^t\}$ as is defined in the following to optimize our learnable parameters.

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}(y_i^t - \hat{y}_i^t)^2 \tag{3.10}$$

3.4 Experiment and Results

3.4.1 Baselines

We compare our proposed approaches with CT-LSTM (An LSTM with static basin characteristics concatenated to the meteorological inputs at each time step), and EA-LSTM [21] with respect to Nash–Sutcliffe model efficiency coefficient (NSE) score and RMSE for each forward model run. CT-LR and Cross-LR are just a two-layer fully connected network version of CT-LSTM and Cross-LSTM, where the first uses feature concatenation and the latter uses feature crosses. Both the traditional LSTMs and Cross-LSTMs use a hidden dimension of 32. All the implementations were identical in all the models except for the feature interaction component for a fair comparison.

3.4.2 Synthetic Dataset

It is important to study in a clean setting with known ground-truth models to understand in which cases the traditional state-of-the-art models like CT-LSTM would become inefficient compared to Cross-LSTM. To test our Hypothesis that plain LSTM has difficulties capturing cross-feature interaction, we create a synthetic dataset by the following methodology. We randomly pick four basins from CAMELS, which gives us four sets of dynamic and four sets of static characteristics. Let us call them Static A, Static B, Static C, and Static D and Dynamic A, Dynamic B, Dynamic C, and Dynamic D. To generate a simpler input, we further sub-select the first three dynamic features in the dynamic set and the first six static features in the static feature set. Next, by picking different combinations of static and dynamic features, we generate our final input parameter set **S** which consists of 16 different parameter ensembles.

Next we generate our synthetic output for each of our parameter sets in the ensemble by the following function

$$f(\mathrm{x}) = \sum_{(i,j)\in S} w_{ij} x_i^s x_j^d \quad \mathrm{x}^{\mathrm{s}} \in \mathbb{R}^3, \mathrm{x}^{\mathrm{d}} \in \mathbb{R}^6, |S| = 16 \tag{3.11}$$

where weights are randomly assigned from uniform U(0,10), and x^d and x^s are samples from parameter ensemble.

Our proposed method is evaluated in the context of predicting the synthetic output Eq 3.11. The 16 available parameter sets are split into two groups, training basins, and testing basins. The training period starts on October 1st, 1999, and ends on September 30th, 2008. The testing period ranges from October 1st, 1989 to September 30th, 1999. This selection of starting and ending years follows the definition of the water year in the hydrology community. We train our model on training basins during training years, i.e., in the training quadrant, as shown in Fig. 3.1. We create input sequences of length 365 using a stride of half the sequence length, i.e., 183. This results in 19 windows each for the training and the testing period. All the hidden variables and gating variables are of 10 dimensions. Table 3.1a and Table 3.1b reports the mean of per basin RMSE, per sample RMSE, # parameters(no. of parameters) for all the models in the in-sample and out-of-sample test sets out of five runs for our synthetic dataset.

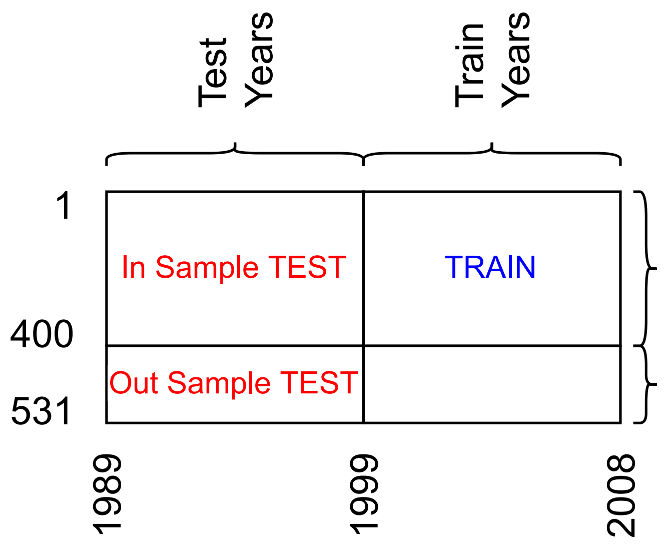


Figure 3.1: *Experimental setting followed in this Chapter for training and testing of the ML models for both synthetic and CAMELS dataset.*

Table 3.1: *RMSE values for different models when applied to Synthetic Dataset*

(a) *Performance in In-Sample Test*

MODELS	# Parameters	Per Sample RMSE	Per Node RMSE
CT_LR	1409	0.444	0.238
Cross_LR	1441	0.008	0.006
CT-LSTM	8801	2.364	0.859
EA-LSTM	7777	1.381	0.839
Cross-LSTM	9089	1.324	0.824

(b) *Performance in Out-Sample Test*

MODELS	# Parameters	Per Sample RMSE	Per Node RMSE
CT_LR	1409	1.927	1.044
Cross_LR	1441	0.022	0.014
CT-LSTM	8801	1.678	1.031
EA-LSTM	7777	1.436	0.968
Cross-LSTM	9089	1.435	0.967

For the In-sample test, our model Cross-LSTM outperforms the baseline EA-LSTM by 1.79% and CT-LSTM by 3.98% when comparing the mean node RMSE. For the Out of sample test, our model outperforms the baseline CT-LSTM by 6.23% and has equivalent performance to EA-LSTM when comparing the mean node RMSE. Our model outperforms both the baselines EA-LSTM and CT-LSTM. We can observe this more clearly when considering per sample RMSE for comparing performance. This shows that nonlinear activations are inefficient in capturing explicit feature crosses (multiplicative relations) even with a deeper network for even second-order feature crosses.

3.4.3 CAMELS Dataset

As suggested by [21], in this study, we selected 531 out of the total 671 basins by removing watersheds whose boundaries are likely mis-delineated and therefore cause unwanted spatial heterogeneity issues. Accounting for data quality controls from multiple watersheds, only a segment of those records is used for training and testing. The training and testing periods are the same as the ones used in the synthetic dataset. We create input sequences of length 365 using a stride of half the sequence length, i.e., 183. This results in 19 windows each for both training and the testing period. Moreover, we divide the basins into the training and testing sets. The training set has 400 basins, and the remaining basins are put into the testing set. We train our model on training basins during training years and predict on testing basins during testing years. Figure 3.1 shows the quadrant for visualizing testing, training basins and testing, and training years. All the hidden variables and gating variables are of 32 dimension.

Table 3.2: *RMSE and NSE Values for different models when applied to CAMELS.*

(a) *Performance in In-Sample Test*

MODELS	# Parameters	Per Sample		Per Node	
		RMSE	NSE	RMSE	NSE
CT_LR	2145	2.972	0.456	2.505	0.131
Cross_LR	2177	2.878	0.467	2.417	0.113
CT-LSTM	9537	1.955	0.766	1.668	0.507
EA-LSTM	8513	1.962	0.765	1.685	0.484
Cross-LSTM	9825	1.903	0.771	1.604	0.513

(b) *Performance in Out-Sample Test*

MODELS	# Parameters	Per Sample		Per Node	
		RMSE	NSE	RMSE	NSE
CT_LR	2145	3.038	0.410	2.589	-0.182
Cross_LR	2177	2.875	0.444	2.463	-0.290
CT-LSTM	9537	2.191	0.700	1.848	0.216
EA-LSTM	8513	2.145	0.719	1.815	0.203
Cross-LSTM	9825	2.080	0.726	1.779	0.251

Table 3.2a and Table 3.2b reports the no of parameters, mean of per basin RMSE, per sample RMSE, per sample, and per basin Nash–Sutcliffe model efficiency coefficient (NSE) score for all the models in the in-sample and out-of-sample test sets out of five runs for camels dataset. For the In-sample test, our Model outperforms the baseline EA-LSTM by 5.05% and CT-LSTM by 3.99% in terms of mean node RMSE. Furthermore, for mean per node NSE, it outperforms EA-LSTM and CT-LSTM by 5.99% and 1.2%, respectively. For Out of sample test, our Model outperforms the baseline EA-LSTM by 2.02% and CT-LSTM by 3.73% in terms of mean node RMSE. Furthermore, for mean per node NSE, it outperforms EA-LSTM and CT-LSTM by 23.64% and 16.20%, respectively. In Figure 3.2 we plot the actual observed streamflow along with the predicted values for both the baseline(CT-LSTM), and the cross-LSTM model for year 1991-1992, for an out-of-sample basin. This shows that nonlinear activations are inefficient in capturing explicit feature crosses (multiplicative relations), which are essential for predicting complex dynamic models like environmental systems.

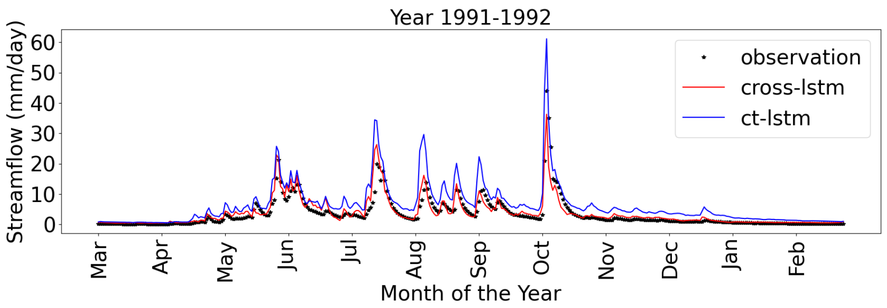


Figure 3.2: *Stream flow prediction for the cross lstm model, and baseline ct-lstm model when it is tested on a out-of-sample basin for year 1991-1992.*

3.5 Conclusion

This work shows that it is possible to build an RNN model that can learn cross(multiplicative) information more efficiently than a baseline LSTM model by creating explicit feature crosses at every gate. We have shown the applicability of this methodology in the context of a synthetic dataset, and streamflow prediction in hydrology using CAMELS dataset. Cross-LSTM outperforms state-of-the-art methods like CT-LSTM and EA-LSTM on synthetic datasets and the popular benchmark CAMELS. The improved Cross-LSTM is more expressive yet remains cost-efficient at modeling complex multiplicative interactions in dynamic systems. This methodology can be used in many scientific applications as exploring potentially rare cross features is essential in predicting environmental systems better due to their complex dynamic nature. In future work, the proposed Cross-LSTM can be further enhanced to incorporate much more higher-order crosses by explicitly adding more cross layers in the RNN cell.

Chapter 4

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

4.1 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 [21, 22], biology, and climate science [23]. 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 [16, 17, 18]. 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) [2, 6]. 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 [94]. 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 [95] 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 [96]. Transfer learning using deep neural networks has shown success in several applications such as image classification [97], sentiment classification [98], and ecological applications such as lake temperature modeling [99]. The issues of "how to transfer" and "what to transfer" can be posed as a problem to be solved by meta-learning [46] or learning from previous learning experiences [100], 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 [101] 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 Chapter 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.

4.2 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 [21]. 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 [11]. 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

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

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 [102] or designing new metric space [103].

Similarity learning techniques have been used to intelligently select relevant source domains given target domain that help improve the learning performance of ML models [104]. These methods have shown much success in several domains like learning similarity in patients [105] and categories [101]. However, in these approaches, the transfer experience among the source data is not used in learning the metric. Wei et al. [106] proposed a method to automatically determine what and how to transfer by leveraging previous transfer learning experiences. Similar to this strategy, Jared et al. [99] 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.

4.3 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 4.5.1.

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.

4.4 Method

This section provides details of the proposed meta-transfer learning via the metric learning approach. The methods proposed in this Chapter 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 4.4.1, we first introduce the sequence auto-encoder model to represent river segments. Then in Section 4.4.2, 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 4.4.3, we describe how to leverage the learned similarity to improve the model performance.

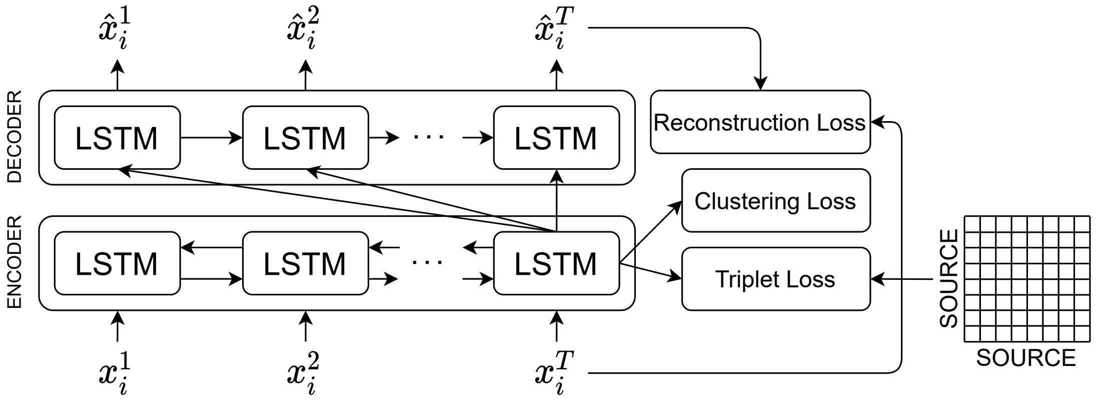


Figure 4.1: *The sequence autoencoder architecture used for embedding time series data, which uses three loss functions: the reconstruction loss (Eq. 4.3), the triplet loss (Eq. 4.5), and the clustering loss (Eq. 4.11). In our application, the time series data x_i^t represents the concatenation of daily input, static features, and simulated target variables.*

4.4.1 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 [107]. 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 [104, 108]. 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 [103].

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 Chapter, 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. 4.1. 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.

$$\begin{aligned}
\boldsymbol{i}_t &= \sigma(\boldsymbol{W}_i \left[[\boldsymbol{x}^t; \boldsymbol{z}; s^t]; \boldsymbol{h}^{t-1}\right] + \boldsymbol{b}_i) \\
\boldsymbol{f}_t &= \sigma(\boldsymbol{W}_f \left[[\boldsymbol{x}^t; \boldsymbol{z}; s^t]; \boldsymbol{h}^{t-1}\right] + \boldsymbol{b}_f) \\
\boldsymbol{g}_t &= \sigma(\boldsymbol{W}_g \left[[\boldsymbol{x}^t; \boldsymbol{z}; s^t]; \boldsymbol{h}^{t-1}\right] + \boldsymbol{b}_g) \\
\boldsymbol{o}_t &= \sigma(\boldsymbol{W}_o \left[[\boldsymbol{x}^t; \boldsymbol{z}; s^t]; \boldsymbol{h}^{\boldsymbol{t-1}}\right] + \boldsymbol{b}_o) \\
\boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i} \odot \boldsymbol{g}_t \\
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t)
\end{aligned} \tag{4.1}$$

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:

$$\begin{aligned}&p_\theta([\boldsymbol{x}^t;\boldsymbol{z};s^t]_{t=1:T}|\boldsymbol{h})\\&=p_\theta([\boldsymbol{x}^1;\boldsymbol{z};s^1]|\boldsymbol{h})\prod_{t=2}^{T}p_\theta([\boldsymbol{x}^t;\boldsymbol{z};s^t]|[\boldsymbol{x}^{1:t-1};\boldsymbol{z};s^{1:t-1}],\boldsymbol{h})\end{aligned} \tag{4.2}$$

A traditional way to train this sequence-to-sequence autoencoder is teacher forcing [109], 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,

$$\max_{\theta,\phi} E_{\boldsymbol{x},\boldsymbol{z}\sim data}[-log p_\theta([\boldsymbol{x}^t;\boldsymbol{z};s^t]_{t=1:T}|\boldsymbol{h})] \tag{4.3}$$

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,

$$sim(\boldsymbol{h}_i,\boldsymbol{h}_j)=\frac{\boldsymbol{h}_i\cdot\boldsymbol{h}_j}{\|\boldsymbol{h}_i\|\|\boldsymbol{h}_j\|} \tag{4.4}$$

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 [106].

4.4.2 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 $\boldsymbol{h}_{n_i}$.

$$\mathcal{L}_{Triplet} = max(0, D(\boldsymbol{h}_i, \boldsymbol{h}_{p_i}) - D(\boldsymbol{h}_i, \boldsymbol{h}_{n_i}) + \alpha) \tag{4.5}$$

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. 4.5) and the standard supervised reconstruction loss (Eq. 4.3), we get the final training loss as follows:

$$\mathcal{L} = \mathcal{L}_{Rec} + \lambda \mathcal{L}_{Triplet} \tag{4.6}$$

where λ is a hyper-parameter.

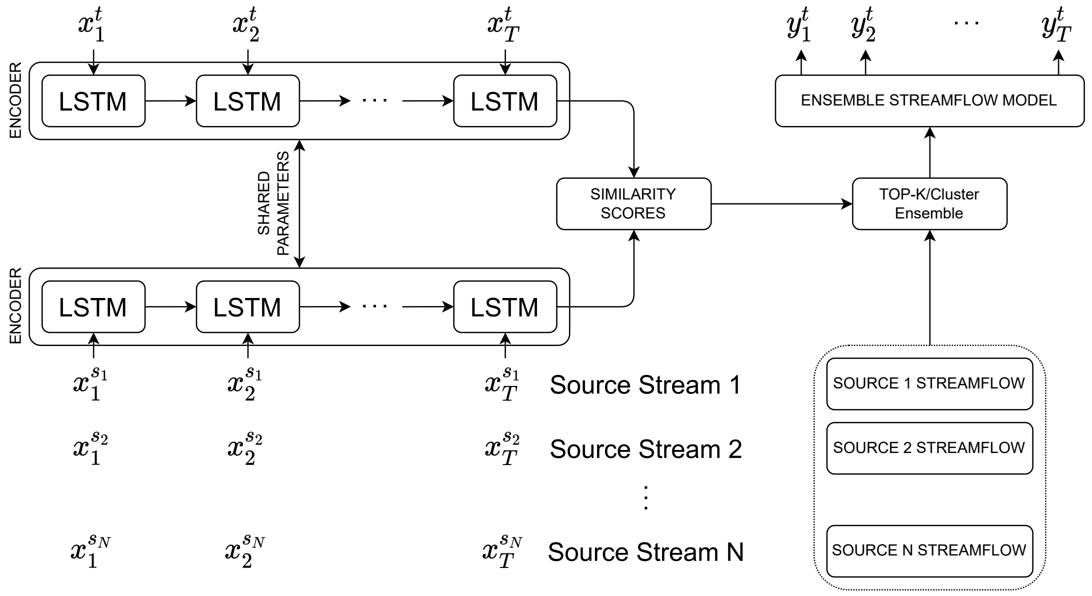


Figure 4.2: *The model transfer process using the top-K or cluster ensemble of source models based on the estimated similarity amongst river segments.*

4.4.3 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 [110]. Further, in scenarios where significant model diversity exists, ensembles tend to yield better results [111]. Hence, we propose to transfer multiple source models to make predictions for each target river segment, as shown in Fig. 4.2. However, identifying the groups is essential in creating such ensemble models. In the following sections, we describe three methods for creating ensemble models.

Top-K Ensemble

Once the metric learning model is trained, we can use the similarity measure described in Eq. 4.4 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.

$$\begin{aligned} \boldsymbol{Y}_{\boldsymbol{t_i}} &= \frac{1}{|K_{t_i}|} \sum_{k \in K_{t_i}} \mathscr{F}_k(\boldsymbol{X}_{\boldsymbol{t_i}}, \boldsymbol{z}_{\boldsymbol{t_i}}), \\ \text{where,} \quad K_{t_i} &= \underset{S' \subset S, |S'|=K}{\operatorname{argmax}} \sum_{S_j \in S'} sim(h_{T_i}, h_{S_j}) \end{aligned} \tag{4.7}$$

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.

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,

$$\begin{aligned} \boldsymbol{Y}_{\boldsymbol{t_i}} &= \sum_{k \in K_{t_i}} \alpha_k \mathscr{F}_k(\boldsymbol{X}_{\boldsymbol{t_i}}, \boldsymbol{z}_{\boldsymbol{t_i}}), \\ \text{where,} \quad \alpha_k & \frac{sim(h_{T_i}, h_{S_k})}{\sum_{S_j \in S} sim(h_{T_i}, h_{S_j})} \end{aligned} \tag{4.8}$$

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 [112] 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 [112]. 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:

$$q_{ij} = \frac{(1 + \|\boldsymbol{h}(\boldsymbol{X}_i;\theta_h) - M_j\|^2/\alpha)^{\frac{\alpha+1}{2}}}{\sum_{j'=1}^{K}(1 + \|\boldsymbol{h}(\boldsymbol{X}_i;\theta_h) - M_{j'}\|^2/\alpha)^{\frac{\alpha+1}{2}}} \tag{4.9}$$

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:

$$p_{ij} = \frac{q_{ij}^2/\sum_i q_{ij}}{\sum_{j'=1}^{K}(q_{ij'}^2/\sum_i q_{ij'})} \tag{4.10}$$

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:

$$\min KL(P\|Q) = \min \frac{1}{N_t} \sum_{i=1}^{N_t} \sum_{j=1}^{K} p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{4.11}$$

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

$$\mathcal{L} = \mathcal{L}_{Rec} + \lambda_1 \mathcal{L}_{Triplet} + \lambda_2 \mathcal{L}_{clus} \tag{4.12}$$

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 4.8. Here the clusters are provided by the meta-model trained using the clustering objective.

4.5 Experiments and Results

4.5.1 Dataset

All the data used in this work are available through U.S. Geological Survey's National Water Information System [113] and the Water Quality Portal [114]. It is the most extensive standardized water quality data set for inland and coastal water bodies [114]. 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 [115]. 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. [116]. 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 [117]. 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 [118] 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.

4.5.2 Baselines

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

- PRMS: The Precipitation-Runoff Modeling System (PRMS) [118] 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 [21].

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) [31] 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 [99]. We use the same four sets of meta-features, i.e., lake attributes, PB0 Simulation statistic, General 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 [119] 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.

4.5.3 Prediction performance

Table 4.1: *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

In Table 4.1, 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.

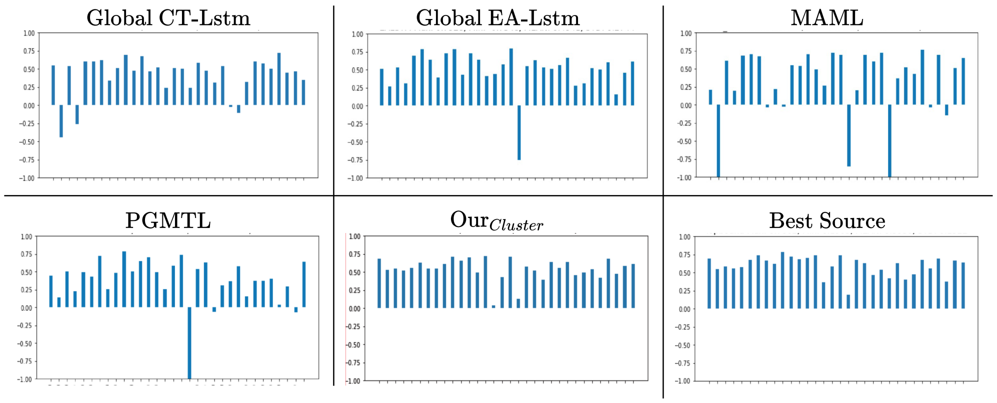


Figure 4.3: *Streamflow performance on each target river stream by all the models. Y-axis shows the R^2 value, whereas the river streams are on the x-axis.*

Fig. 4.3 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. 4.4. Note how the global CT-Lstm model trained on all source river streams over-estimates the streamflow for the reasons discussed above.

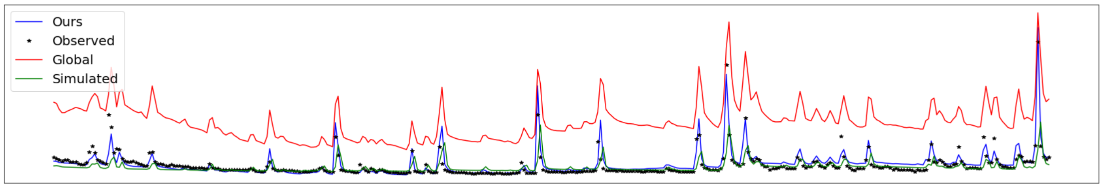


Figure 4.4: *Streamflow predictions made by the proposed method, the global CT-LSTM model, and the physics-based PRMS model.*

4.5.4 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.

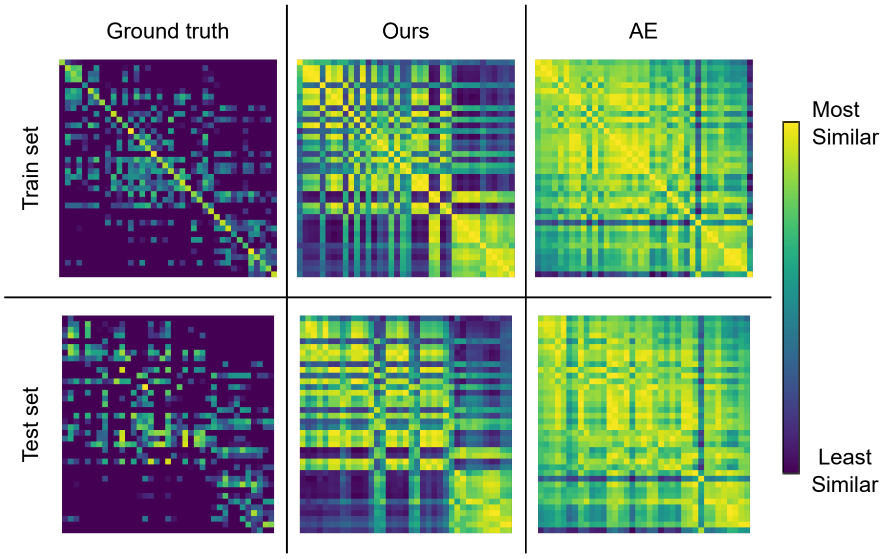


Figure 4.5: *The similarity matrices between source-to-source segments (1st row) and between source-to-target segments (2nd row). Each entry (i,j) in the ground truth matrix (1st column) represents the R2 value obtained by applying the source model of the segment i to the data of segment j. The matrices 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).*

Visualizing Predicted Similarity

In Fig. 4.5 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 4.4). 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.

Table 4.2: *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

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 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,

$$Prec@k = \frac{\text{Top } k \text{ recommendations} \cap \text{Top } k \text{ ground truth}}{k} \tag{4.13}$$

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 (4.8), 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,

$$AP@k = \frac{1}{k}\sum_{i=1}^{k}(Prec@i * relevant@i) \tag{4.14}$$

  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,

$$MRR = \frac{1}{|T|}\sum_{i=1}|T|\frac{1}{rank_i} \tag{4.15}$$

Table 4.2 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 4.1) compared to other baselines. We attribute this characteristic to learning from past transfer experience in the training set.

4.5.5 Sensitivity Tests

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

Sensitivity to Top-k values

Fig. 4.6 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.

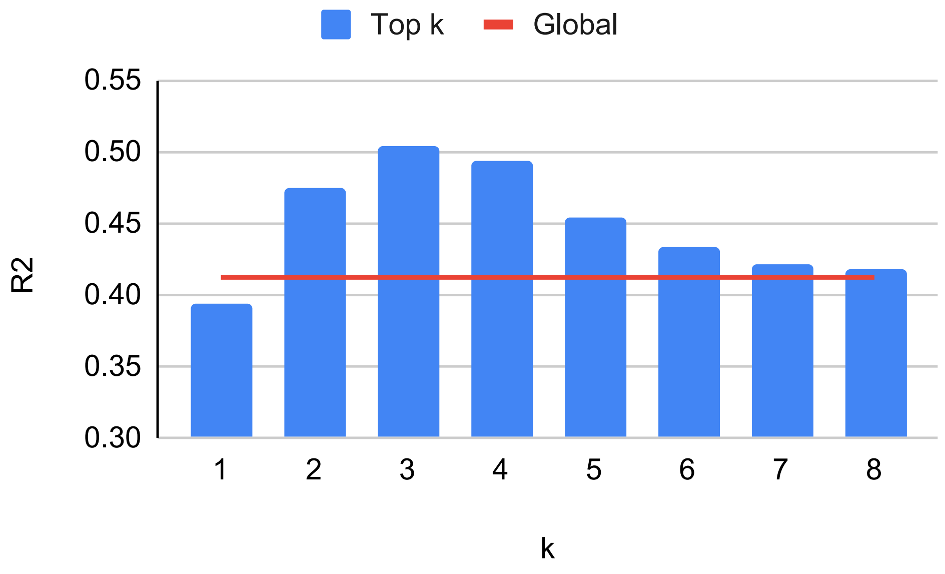


Figure 4.6: *Predictive performance (in terms of R^2 values) using different K values in the top-K ensemble transfer method.*

Sensitivity to Cluster numbers

In Fig. 4.7, 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.

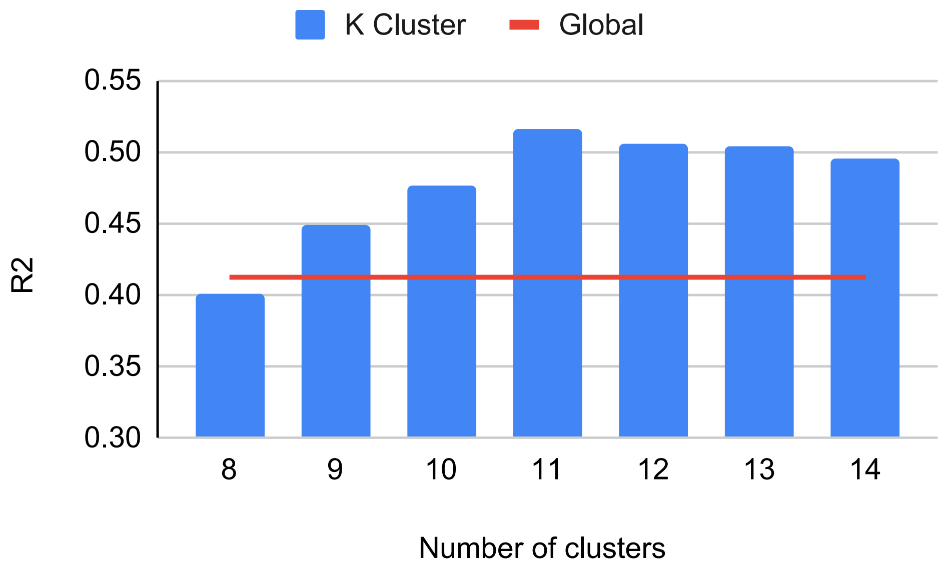


Figure 4.7: *Predictive performance (in terms of R^2 values) using different numbers of clusters in the cluster ensemble transfer method.*

4.6 Conclusion

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

Chapter 5

Robust Inverse Framework using Knowledge-guided Self-Supervised Learning: An application to Hydrology

5.1 Introduction

Machine learning (ML) is increasingly being used to solve challenging tasks in scientific applications such as hydrology, fresh-water ecology, and crop yield monitoring. Consider the case of hydrology, where streamflow prediction is used for understanding hydrology cycles, water supply management, flood mapping, and making operational decisions such as reservoir release. For a given entity (basin/catchment, we use either term interchangeably), the response (streamflow) is governed by drivers (meteorological data e.g., air temperature, precipitation, wind speed) and complex physical processes specific to each entity [2]. These complex physical processes are best captured by the inherent characteristics of each entity (e.g., slope, land-cover). For example, for the same amount of precipitation, two basins will have very different streamflow response values depending on their land-cover type. The streamflow modeling is just one example of a wide variety of scientific models that can be considered as a mapping function between drivers $\boldsymbol{x_t}$ (e.g. weather drivers, climate forcings), and response y_t (e.g., streamflow in river basin, global average temperature), governed by entity characteristics. Such problems are often solved using a mechanistic forward model f that predicts the response y_t, given drivers $\boldsymbol{x_t}$ and entity characteristic $\boldsymbol{z}$. Figure 5.1a shows the diagrammatic representation of this forward model. More recently, Machine Learning (ML) models (e.g. LSTMs) have been shown to provide state of the art performance for forward modelling in many scientific applications [21, 120]. The reason is that ML models are able to benefit from training data from a diverse range of entities and thus can transfer knowledge across entities. There is also much interest in the development of ML algorithms guided by scientific knowledge [24]. Such knowledge-guided machine learning (KGML) models have been shown to provide improved performance over black-box ML models even with fewer sample, and are able to generalize in unseen scenarios [25, 121, 122, 123].

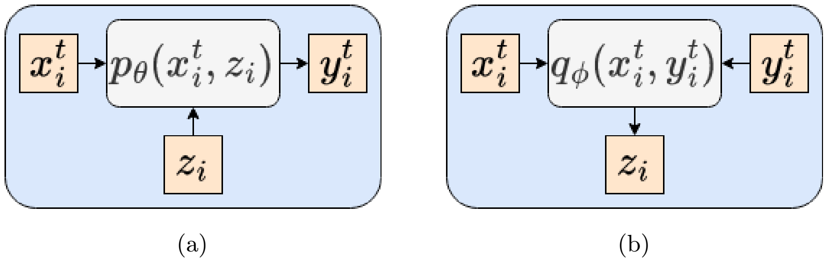


Figure 5.1: *(a) Forward model which uses meteorological drivers ($\boldsymbol{x_i^t}$) and entity characteristics ($\boldsymbol{z_i}$) to predict response ($\boldsymbol{y_i^t}$) (b) The inverse model which approximates entity characteristics ($\boldsymbol{z_i}$) by inverting the forward process.*

In streamflow modeling (as well as many other scientific applications), entity characteristics are often surrogate variables of the true basin characteristics [124] and thus can lead to several challenges. First, there often exists high uncertainty in hydrological measurement, which in turn causes corruption in basin characteristics. Uncertainty can also arise due to temporal change, spatial heterogeneity, sufficiency of the characteristic itself to explain the rainfall/runoff process, measurement error, missing data, and correlation among characteristics that collectively contribute to streamflow. Second, the full set of basin characteristics may not be measured across all the river basins, resulting in the incompleteness of basin characteristics. Missing characteristics hinder the building of a global model that can leverage data across multiple basins and constrains the transferability of models built from one region to another. Finally, some basin characteristics may be essential in modeling the rainfall-runoff response relation but may be completely unknown, not well understood, or not present in the available set of basin characteristics. Thus, the ability to infer these time-invariant basin characteristics from the time-varying meteorological and streamflow data is essential for model prediction and hydrological process understanding. However, traditional methods used by the physical science community for inferring these characteristics are compute intensive, as they require a large number of forward model runs (especially if z has a large dimension).

This Chapter presents an inverse modeling methodology that can be used to identify or reconstruct static characteristics of an environmental system given its input and output over time. Figure 5.1b shows the diagrammatic representation of this inverse problem. Inverse problems [125] appears in many fields of engineering when the goal is to recover "hidden" characteristics of a system from "observed" data. In recent years, deep learning techniques have shown remarkable success for solving inverse problems in various fields such as compressed sensing, medical imaging [126], and many more (see [125] for a recent overview). In general, the inverse problem is ill-posed, i.e., one may not be able to uniquely recover the input field given noisy and incomplete observations [125].

This Chapter proposes to compute $\boldsymbol{z}$ given $\boldsymbol{x^t}$ and $\boldsymbol{y^t}$ efficiently. Deep learning methods traditionally solve inverse problems by minimizing a cost function [127] that consists of a data-fit term, which measures how well the reconstruction matches the observations and a regularizer. These methods, largely based on convolution operator tend to work for inverse problems such as image denoising, super-resolution, and compressed sensing, but for capturing time varying physical processes such as ours, the traditional method fails. It presents a novel inverse framework leveraging knowledge from the hydrological domain in a self-supervised learning framework to implicitly extract complex correlations embedded in the input data. This methodology is termed as knowledge-guided self supervised learning (KGSSL). KGSSL enables the extraction of time-invariant characteristics autonomously in the form of embeddings, by forcing them to be similar for the same basin but different years and dissimilar with other basins. In cases where certain basin characteristics are known, this framework further adds a pseudo-inverse loss on top of the learned embeddings to guide the learning using the known basin characteristics.

KGSSL's usefulness is demonstrated in the context of stream flow modeling using

CAMELS (Catchment Attributes and MEteorology for Large-sample Studies) [7] which is a widely used hydrology benchmark data set. Specifically, this chapter shows that the methodology can effectively impute basin characteristics (if they are missing) or reduce their uncertainty (if they are uncertain). It additionally demonstrates KGSSL's usefulness in situations where static characteristics $\boldsymbol{z}$ are not known for any basins. Here, it shows that the similarity between basins using learned embeddings closely follow the similarity based on the actual characteristics. Further, it shows that the learned embeddings can act as an effective replacement for static characteristics in a forward model. Our main contributions are listed below:

- This Chapter demonstrates the power of leveraging domain knowledge in a self-supervised framework for solving an inverse problem.
- Extensive evaluation in the context of a widely used hydrology benchmark show that KGSSL outperforms baseline by 16 % in predicting missing characteristics. In addition, the framework achieves robust performance even when the characteristics are corrupted or missing.
- In the context of forward modelling, KGSSL inferred characteristics provide a 35% improvement in performance over a standard baseline when the static characteristic are unknown.

5.2 Related Works

Inverse problems [125] always exist together with their forward problem. The goal of the inverse problem is to recover "hidden" information (which we cannot observe directly or is very expensive to observe) from readily available "observed" data. Unfortunately, the inverse is often both intractable and ill-posed, since crucial information is lost in the forward process. However, the inverse process is required to inform us about physical parameters of the system (e.g., mass, temperature, physical dimensions, or structure), sources of influence, reconstruction of the coefficients in the equations that we cannot observe otherwise. Inverse problems are studied for many environmental science branches, i.e., hydrogeology [128], geophysics [129], oceanography [130], meteorology [131], remote sensing [132], etc. For example, an inverse problem arises when we reconstruct

Earth's interior by modeling the physical propagation of seismic waves [133]. Similarly, in reservoir engineering [134], given various measurements of geophysical fields, an inverse problem arises to determine the subsurface properties, such as the permeability field. Most of the recent deep learning approaches [135] model forward/inverse mapping within a single network. However, in hydrology, the initial physical parameters are not known reliably [136] for the basins/catchment due to temporal and spatial heterogeneity. This leads to a noisy forward operator, which makes existing inverse approaches ineffective. This motivates us to design a robust inverse framework impervious to corrupted basin characteristics.

Due to abundant unlabeled data in computer vision, recently, researchers have started investigating self-supervised methods [137] for model training. In self-supervised learning, the models are trained using pretext tasks instead of an actual task. For example, image colorization [138], image inpainting [139], solving image-jigsaw [140], predicting rotations [141], etc. For a comprehensive understanding of self-supervised representation learning, we would like to redirect the reader to a survey by Jing et al. [137]. Our work utilizes a self-supervised loss called InfoNCE loss [142] to counter the uncertainty in the basin characteristics by implicitly extracting complex correlations embedded in the meteorological drivers. The use of InfoNCE loss in our work is closely related to that of [143], which trained the teacher-student network using contrastive loss to recover the true feature of a corrupted image. However, our problem domain (learning relationship between time-varying complex physical processes) is fundamentally different from vision-related inverse problems. In addition, our method differs in the following aspects. First, we employ the self-supervised learning method in the time-series domain, whereas most of the applications are in the vision domain. Second, we design novel pretext tasks in hydrology using domain knowledge. Finally, we focus extensively on robustness in our work and showcase our methodology's use on both supervised and unsupervised learning.

5.3 Method

In this work, we study the driver-response relation in dynamical systems. Specifically, we assume a dataset consisting of N entities (an entity can be a lake, basin or streams in a river-network). For each entity i, the daily drivers are 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 input vector at time $t \in T$ with D_x dimension. $\boldsymbol{z_i} \in \mathbb{R}^{D_z}$ denotes the static characteristic vector of an entity with D_z dimensions. The transient response corresponding to $(\boldsymbol{X_i}, \boldsymbol{z_i})$ for an entity is denoted by $\boldsymbol{Y_i} = [y_i^1, y_i^2, \ldots, y_i^T]$.

Our proposed method KGSSL, infers time-invariant entity characteristics $(\boldsymbol{z_i})$ given the time-varying driver $(\boldsymbol{X_i})$ and response $(\boldsymbol{Y_i})$ data. KGSSL has several components. First, a *Sequence Encoder* is used to extract a fixed length representation from the driver-response time-series. Second, a *reconstruction loss* $(\mathcal{L}_{Rec})$ that forces the fixed length representation to capture the information stored in driver-response time-series by penalizing bad driver-response time-series reconstructions. Third, a *Knowledge-guided Contrastive Loss* $(\mathcal{L}_{Cont})$ that implicitly extract complex correlations embedded in the driver-response time-series and enforces the physical knowledge that the entity characteristics are time-invariant. Finally, a *PseudoInverse loss* $(\mathcal{L}_{Inv})$ that encourages robust reconstruction of entity characteristics from the fixed length representation using a feed-forward network. Thus, the final loss function for training KGSSL is

$$\mathcal{L} = \lambda_1 \mathcal{L}_{Rec} + \lambda_2 \mathcal{L}_{Cont} + \lambda_3 \mathcal{L}_{Inv} \tag{5.1}$$

where λ_1, λ_2, λ_3 are hyper-parameters to control the weights of three loss terms. $\mathcal{L}_{Inv}$ is added only when the entity characteristics are known and available for training. We can also train KGSSL using only the self-supervised loss functions, $\mathcal{L}_{Rec}$ and $\mathcal{L}_{Cont}$. In the following subsections, we discuss each of these components in detail and provide intuition behind such design choices.

KGSSL generates time invariant and entity specific embeddings from driver-response time-series data. Specifically, for each entity i in a given set of N entities, we randomly select two sequences of length W. Let S_{a_i} and S_{p_i} be the two sequences taken from the time-windows $t_{a_i} : t_{a_i} + W$ and $t_{p_i} : t_{p_i} + W$, respectively. This results in $2N$ sequences and each element in these sequences are formed by concatenating the drivers-response time-series of the entity $([\boldsymbol{x_i^t}; y_i^t])$.

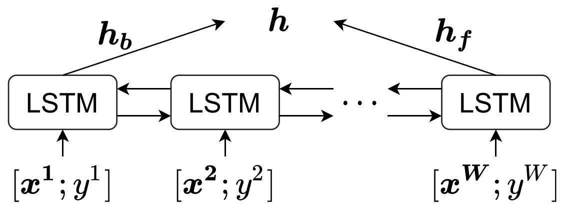


Figure 5.2: *Bidirectional LSTM based Sequence Encoder*

5.3.1 Sequence Encoder

We use a sequence encoder to encode the temporal information and the interaction between the driver and response in these sequences. LSTM is particularly suited for our task where long range temporal dependencies between driver and response exist as they are designed to avoid exploding and vanishing gradient problems. However, LSTMs are designed to run forward in time and cannot provide explainability on the current time-steps given the future data. To capture this information we use a Bidirectional LSTM based sequence encoder $\mathscr{E}$ (Figure 5.2). 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 set of equations to generate the embeddings for a sequence,

$$\begin{aligned}
\boldsymbol{i_t} &= \sigma(\boldsymbol{W_i}\left[[\boldsymbol{x^t}; y^t]; \boldsymbol{h^{t-1}}\right] + \boldsymbol{b_i}) \\
\boldsymbol{f_t} &= \sigma(\boldsymbol{W_f}\left[[\boldsymbol{x^t}; y^t]; \boldsymbol{h^{t-1}}\right] + \boldsymbol{b_f}) \\
\boldsymbol{g_t} &= \sigma(\boldsymbol{W_g}\left[[\boldsymbol{x^t}; y^t]; \boldsymbol{h^{t-1}}\right] + \boldsymbol{b_g}) \\
\boldsymbol{o_t} &= \sigma(\boldsymbol{W_o}\left[[\boldsymbol{x^t}; y^t]; \boldsymbol{h^{t-1}}\right] + \boldsymbol{b_o}) \\
\boldsymbol{c_t} &= \boldsymbol{f_t} \odot \boldsymbol{c_{t-1}} + \boldsymbol{i} \odot \boldsymbol{g_t} \\
\boldsymbol{h_t} &= \boldsymbol{o_t} \odot \tanh(\boldsymbol{c_t})
\end{aligned} \tag{5.2}$$

Each of the forward and backward LSTM takes in a sequence S as input and generates corresponding embeddings $\boldsymbol{h_f}$ and $\boldsymbol{h_b}$ ($\boldsymbol{h} = \mathscr{E}(S)$). These embeddings are essentially the final hidden states of each LSTM. The embeddings for the forward LSTM ($\boldsymbol{h_f}$) and backward LSTM ($\boldsymbol{h_b}$) are added to get the final embeddings $\boldsymbol{h}$ as shown in Figure 5.2.

These embeddings capture the temporal information as well as the driver-response interaction by modeling the change in streamflow due to the weather drivers in both forward and backward directions.

5.3.2 Reconstruction Loss

To preserve the key information from driver-response data, we use a standard LSTM based decoder $\mathscr{D}$ that reconstructs the sequence back from the embedding ($\hat{S} = \mathscr{D}(\boldsymbol{h})$). The LSTM decoder uses its own output at the previous time-step as the input for the current time-step and thus can be regarded as a sequence generator using the embedding $\boldsymbol{h}$ as a prior. The reconstruction error is computed as the mean-squared error between the reconstructed and the original sequence, as shown below,

$$\mathcal{L}_{Rec} = \frac{1}{2N} \sum_{e \in \{a,p\}} \sum_{i=1}^{N} MSE(\hat{S}_{e_i}, S_{e_i}) \tag{5.3}$$

Here $\mathcal{L}_{Rec}$ acts as a regularizer in representation learning, by extracting meaningful information from the time-varying input data. However, since we are interested in extracting the time-invariant information from the time-series data, solely relying on $\mathcal{L}_{Rec}$ leads to sub-optimal performance. $\mathcal{L}_{Rec}$ promotes preservation of information about the time-series in the embeddings which will later be used by the decoder to reconstruct back the input time-series.

5.3.3 Knowledge-guided Contrastive Loss

Each entity's response to a given driver is governed by complex physical processes captured by its inherent physical characteristics that remain constant through time. Moreover, different entities have different responses to the same driver due to the differences in their inherent characteristics. We use this physical knowledge of entities to define a self-supervised contrastive loss [142, 144]. Specifically, the sequences S_{a_i} and S_{p_i} of an entity form a positive pair, and for each positive pair, we treat the other 2(N-1) sequences within a batch as negative examples. Thus, the contrastive loss forces the embeddings $\boldsymbol{h_{a_i}}$ and $\boldsymbol{h_{p_i}}$ resulting from the sequences S_{a_i} and S_{p_i} of the same entity to be similar and different from the embeddings of other basins. For a given positive pair, the loss is calculated as,

$$\begin{aligned} l(a_i, p_i) = & \frac{\exp\left(sim(\boldsymbol{h_{a_i}}, \boldsymbol{h_{p_i}})/\tau\right)}{\sum_{e\in\{a,p\}} \sum_{j=1}^{N} \exp\left(sim(\boldsymbol{h_{a_i}}, \boldsymbol{h_{e_j}})/\tau\right)} \\ + & \frac{\exp\left(sim(\boldsymbol{h_{p_i}}, \boldsymbol{h_{a_i}})/\tau\right)}{\sum_{e\in\{a,p\}} \sum_{j=1}^{N} \exp\left(sim(\boldsymbol{h_{p_i}}, \boldsymbol{h_{e_j}})/\tau\right)} \end{aligned} \tag{5.4}$$

where, $sim(\boldsymbol{h_{a_i}}, \boldsymbol{h_{p_i}}) = \frac{\boldsymbol{h_{a_i}}^T \boldsymbol{h_{p_i}}}{\|\boldsymbol{h_{a_i}}\| \|\boldsymbol{h_{p_i}}\|}$. Thus, the total contrastive loss for 2N such positive pairs is given as,

$$\mathcal{L}_{Cont} = \frac{1}{2N} \sum_{i=1}^{N} l(a_i, p_i) \tag{5.5}$$

Both $\mathcal{L}_{Cont}$ and $\mathcal{L}_{Rec}$ do not require any supervised information and thus can work with a large number of entities for which we only know the driver-response time-series. Moreover, later in the results, we show that using only one of these losses leads to sub-optimal performance and, thus we use a combination of these two losses.

5.3.4 PseudoInverse Loss

Reconstruction Loss and Knowledge-guided Contrastive Loss is used to extrapolate entity characteristics from the time-varying driver ($\boldsymbol{X_i}$) and response ($\boldsymbol{Y_i}$). However, if some entity characteristics are known (albeit noisy/uncertain), the above loss functions fail to account for them during the model training. To improve our inverse framework, we propose using PseudoInverse loss that utilizes incomplete/uncertain missing entity characteristics as a source of supervision. Specifically, we add a feed-forward layer $\mathscr{I}$ on sequence encoder output h to estimate $\hat{\boldsymbol{z}} = \mathscr{I}(\boldsymbol{h})$ and then we define regression loss with the available set of entity characteristics (z) as shown in Figure 5.3 . Pseudoinverse loss is defined as follows:

$$\mathcal{L}_{Inv} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{z} \sum_{j=1}^{z} (z_i^j - \hat{z}_i^j)^2 \tag{5.6}$$

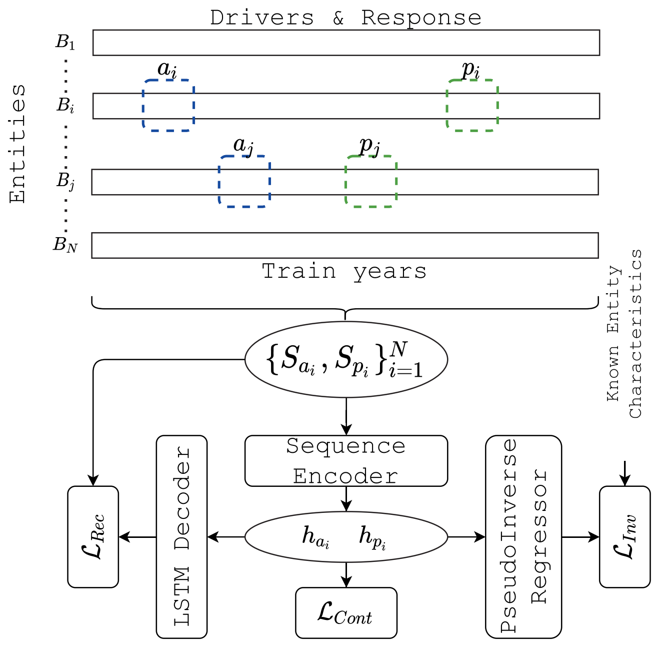


Figure 5.3: *Proposed inverse model generates embeddings for a basin from the LSTM Encoder (Figure 5.2) and is trained in a self-supervised manner. Strong supervision* ($\mathcal{L}_{Inv}$) *is added when ground-truth characteristics are available for a limited number of entities.*

5.3.5 Reconstructing static characteristics given temporal data

Our KGSSL framework can be used to generate entity-specific embeddings as well as static characteristics. Specifically, given input-drivers $X_i = [\boldsymbol{x_i^1}, \boldsymbol{x_i^2}, \ldots, \boldsymbol{x_i^T}]$ where $\boldsymbol{x_i^t} \in \mathbb{R}^{D_x}$ and output-response $Y_i = [y_i^1, y_i^2, \ldots, y_i^T]$ time-series of length T for an entity, we break the combined time-series into T/W sequences of length W. Each of these sequences S_i^j are fed to the encoder $\mathscr{E}$ to generate an embedding $\boldsymbol{h}_i^j$, which are further fed into the inverse regressor $\mathscr{I}$ to predict the static characteristics $\hat{\boldsymbol{z}}_i^j$. By taking the element-wise mean of the embeddings, we get the final embeddings of the entity. Similarly, we get the final estimate of the static characteristics along with their uncertainties by taking the element-wise mean and standard deviation of the sequence specific predictions, as shown,

$$\boldsymbol{h}_i = \frac{W}{T}\sum_{j=1}^{T/W}\boldsymbol{h}_i^j \quad \hat{\boldsymbol{z}}_i = \frac{W}{T}\sum_{j=1}^{T/W}\hat{\boldsymbol{z}}_i^j \quad \boldsymbol{unc}_i = \sqrt{\frac{W}{T}\sum_{j=1}^{T/W}(\hat{\boldsymbol{z}}_i^j - \hat{\boldsymbol{z}}_i)^2} \tag{5.7}$$

As more and more years of data are made available for an entity, the embeddings and the predictions of the static characteristics become more certain and informative.

5.4 Experimental Results

5.4.1 Datasets and Implementation details

We evaluate KGSSL using the CAMELS (Catchment Attributes and MEteorology for Large-sample Studies) dataset, which is extensively used for investigating hydrology processes, in particular, streamflow prediction [93]. CAMELS compiles meteorological forcing data (e.g. precipitation, air temperature), streamflow observation, calibrated physical model simulation, and catchment characteristics(see Appendix A.1 for a complete list), all of which makes it possible to leverage recent developments in machine learning, in particular deep learning, in the hydrology community to advance continental hydrology modeling [21, 145]. In particular, using the CAMELS dataset, Kratzert et al. [21] showed that a global scale LSTM model (that uses known static characteristics as input in addition to weather drivers) can outperform state-of-the-art physics based hydrological model that are individually caliberated for each basins.

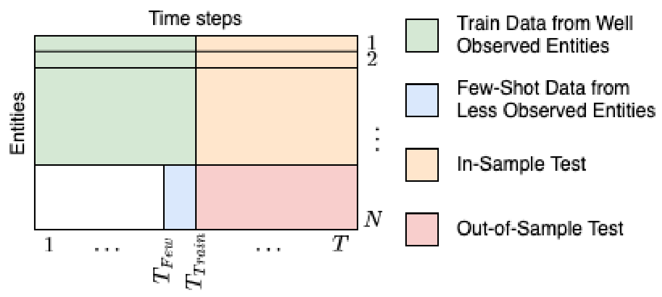


Figure 5.4: *Experimental setting followed in this Chapter for training and testing of the ML models.*

Following the set up used by Kratzert et al. [21], our study uses data for 531 basins from CAMELS for the periods (1989-2009). Of these, (2001-2008) is used for model building, and the rest is used for testing. Fig. 5.4 show the experimental setting followed in this Chapter. Like Kratzert et al. our study uses 27 basin characteristics organized by physically meaningful groups: climatology, soils/geologic conditions and geomorphology/land-cover. These three groups of characteristics can generally be assumed to represent physical characteristics that contribute more or less to the rainfall/runoff process in any given catchment. We create input sequences of length 365 using a stride of half the sequence length, i.e., 183. This results in 13 windows for the data used for model training and 19 for the testing period. All LSTMs used in our architecture have one hidden layer with 64 units. The feed-forward network to reconstruct characteristics has one hidden layer followed by activation to introduce non-linear combinations of the embeddings. The hyperparameter λ_1, λ_2, and λ_3 are set at 1, 1, and 1 respectively. The value of λ_1, λ_2 and λ_3 are selected to balance the supervised and unsupervised components of the loss function. Higher values for λ_3 lead to lower training loss but at the expense of loss of robustness to noise in the static characteristics(see Appendix A.2 for more details about the hyperparameter search). To reduce the randomness typically expected with network initialization, we train five models with different initialization of deep learning model weights. The predictions were then further combined into an ensemble by averaging prediction from these five models.

In sections 5.4.2, 5.4.3, 5.4.4 we evaluate the ability of KGSSL to estimate the entity characteristics in test basins under various conditions, including when characteristics in the training set are corrupted or missing. Section 5.4.5 considers the case where the catchment characteristics are not available during training. In addition, section 5.4.6 shows the ability of KGSSL to improve the forward modeling task.

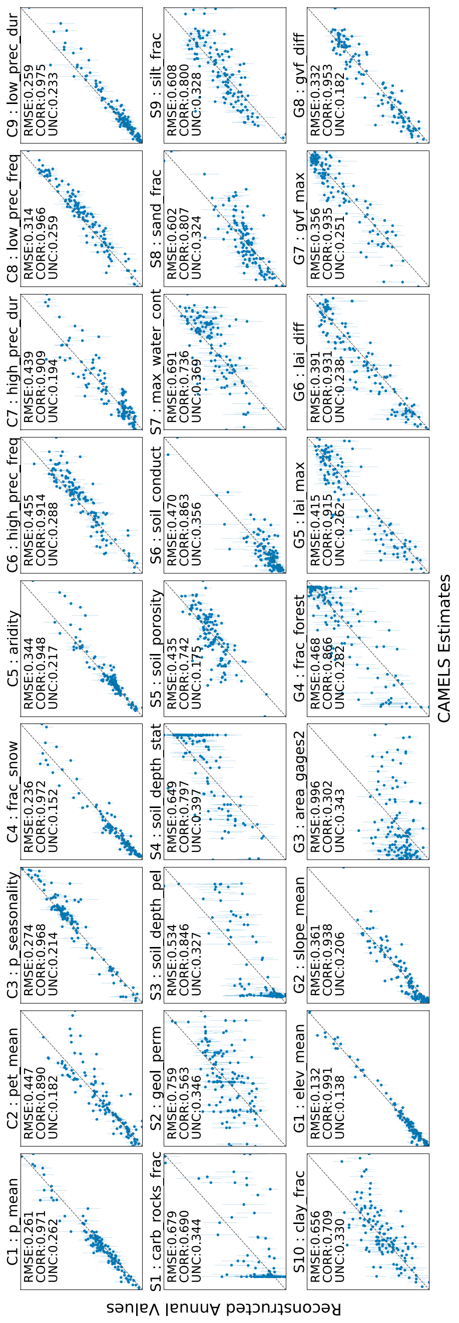


Figure 5.5: *Scatter plot of the CAMELS Estimates for static characteristics (x-axis) for the testing basins vs. reconstructed annual characteristics (y-axis). The error bar across the points show the variation of the reconstructed characteristics annually across test years.*

Method	RMSE	CORR
LSTM	0.540	0.795
KGSSL($L_{Rec+Inv}$)	0.493	**0.831**
KGSSL($L_{Cont+Inv}$)	0.514	0.815
KGSSL($L_{Rec+Cont+Inv}$)	**0.465**	0.824

Table 5.1: *Average root mean square error (RMSE) and correlation (CORR) for* 131 *test basins during testing period.*

5.4.2 Estimating the entity characteristics

We train our model using 400 train basins and reconstruct the static characteristics of the remaining 131 test basins. Table 5.1 reports average root mean square error (RMSE) and correlation (CORR) for 131 test basins during testing period. The entities have different scale values. Since the RMSE value is not scale-invariant, we also report a correlation metric, which is scale-independent. Moreover, the RMSE value measures prediction error, whereas correlation captures the trend.

We make the following high-level observations from our results: a) KGSSL, which uses both supervised and unsupervised loss functions to infer entity characteristics, has superior performance (**16**% better RMSE) as compared to LSTM, which was trained using mean square error loss. b)Each of the self-supervised losses, i.e., L_1 and L_2 individually, leads to sub-optimal performance; thus, combining these two losses with L_3 helps capture the complex physical process accurately. Fig. 5.5 illustrates the ability of KGSSL to reconstruct the 27 basin characteristics in the CAMELS dataset. Note that the reconstructed values are annual averages for each year in the testing period 1989-1999, while vertical lines show the uncertainty (UNC) in the prediction as described in Eq. 5.7. Each individual scatter plot showcases RMSE, Correlation (CORR) and uncertainty (UNC). Note that in general KGSSL performs well (correlation$>$0.8) in 20 out of 27 cases with correlation $>$ 0.9 for 14 of them, and of the remaining 7 cases only one has a correlation $<$0.5.

The results (Table 5.1 and Figure 5.5) exhibit that KGSSL is able to predict the characteristics with acceptable accuracy (corr$>$0.8) for most of the characteristics. In general, the average RMSE and average correlation of the predicted values are 0.465

and 0.824, respectively. However, the characteristics reconstruction performance varies among the individual characteristics. The ones with higher reconstruction RMSEs are usually accompanied with higher standard deviation. The features with satisfactory reconstruction performance (lower RMSEs and high correlation) are also more temporally consistent (lower standard deviation). As discussed in the next paragraph, inconsistencies in reconstruction performance among the individual characteristics can be reasoned based on domain knowledge and reflect uncertainties present in the original CAMELS data set. This interpretation of the modeling results is arguably an important scientific discovery of our proposed KGSSL framework.

KGSSL inferred all nine climate characteristics quite accurately. This result is consistent with the fact that the climate characteristics published in the CAMELS data set are derived directly from the meteorological forcing data $\boldsymbol{X_i}$. We reason that the *elev_ mean* was also quite accurately inferred (0.132 RMSE and 0.138 standard deviation) because the mean elevation is related to climate patterns, and this reasoning also holds true for the catchment slope characteristic, *slope_ mean*, and vegetation characteristics (*gvf_ diff, gvf_ max, lai_ max, and lai_ diff, frac_forest*), as these should all be correlated to meteorological characteristics. The remaining seven characteristics are uncertain by nature because of involved spatial and temporal heterogeneities. Some of them also possess uncertainties in the original data source from which they are derived. Most of these remaining characteristics are soil-related (e.g., *carbonate_ rocks_frac*, *geol_ permeability*, *soil_ depth_ pelletier*) and are derived as spatial averages from the catchments. Such derivation overly simplifies catchment spatial heterogeneity, in particular for large catchments. Therefore, this simplification might explain the large variance recognized in those characteristics. Furthermore, as mentioned in [93], the spatial gridded data where those soil characteristics are derived are uncertain and erroneous in certain geographic regions. It also only characterizes top layer soils and ignores deep soil information. Consequently, soil related characteristics are poorly constructed ones. In addition, *area_ gages2* is the contributing area where surface runoff is generated, and this is spatially and temporally highly non-uniform due to the spatial variability of soil properties, spatial variability of antecedent conditions, and non-uniformity of incident rainfall. Thus, our reconstruction performance on *area_ gages2* is also unsatisfactory.

Group	Indexes	Sec 5.4.2	Sec 5.4.3			Sec 5.4.4	
		Original	90% $\mathcal{N}(0, \sigma_i)$	50% $\mathcal{N}(0, 2\sigma_i)$	90% $\mathcal{N}(0, 2\sigma_i)$	50% Missing	90% Missing
Climate	$C1 - C9$	0.935	0.906	0.890	0.854	0.933	0.870
Soil-Geology	$S1 - S10$	0.711	0.658	0.585	0.546	0.665	0.550
Geomorphology-land cover	$G1 - G8$	0.841	0.812	0.783	0.751	0.825	0.783
Mean		0.824	0.786	0.745	0.709	0.802	0.725

Table 5.2: *The correlation of reconstructed characteristics for test basins in reference to true characteristics for different levels of noise and missing values in train basins.*

5.4.3 Robustness to Corruption in available characteristics

As highlighted in Sec.5.1, we expect uncertainty in the characteristics, , and furthermore the nature of this uncertainty may be due to temporal and/or spatial variability, lack of representativeness, measurement error and/or missing data. The inverse model learns generalizable patterns and hence can potentially denoise the corrupted characteristics. To emulate this uncertainty in measurement we randomly corrupt 50% and 90% of the characteristics. Three experimental setting are thus created. First, to 50% of the characteristics, a Gaussian noise with 0 mean and 2 standard-deviation is added while the remaining characteristics are left unchanged. Second, to 90% of the characteristics a Gaussian noise with 0 mean and 1 standard-deviation is added. Finally, to the same 90% of the characteristics a Gaussian noise with 2 standard-deviation is added. Those scenarios are created to capture two perspectives: a small number of characteristics can have a high level of noise, and a large number of characteristics are corrupted with a relatively low level noise. We train separate models on the training data using the corrupted values of these three settings, and the basin characteristics were predicted using the data from the test years for all the basins and compared to the original values.

We compare the performance of KGSSL trained using the corrupted data with the KGSSL trained using original catchment characteristics. Table 5.2 shows the performance of various methods in terms of correlation of the predicted characteristics with the original characteristics. The impact of noise on characteristics varies among groups, which can be explained by their dependence on weather data that characteristics are learned. Climate characteristics are the least sensitive ones because their original characteristics are derived from weather data. Noisy original characteristics will not downgrade the reconstruction performance because characteristics can be learned from weather data anyway. Though not directly related to weather data, geomorphology-land cover characteristics exhibit geomorphology, land cover patterns that are implicitly characterized from weather data because of involved plant growing mechanisms and terrestrial processes. Thus, their reconstruction performance is much less impacted by noise in the training set. The worst responses in soil-geology characteristics are likely because they characterize subsurface processes whose interactions with weather data are relatively negligible. Such limited usable information in weather data for soil-geology characteristics constrains the capability of our model to learn. Figure 5.6 shows the RMSE computed for corrupted (in blue) and reconstructed characteristics (in green) with respect to true characteristics averaged across all the 400 train basins for these two models. We can observe that KGSSL significantly reduces measurement error in characteristics by an average RMSE of 1.369.

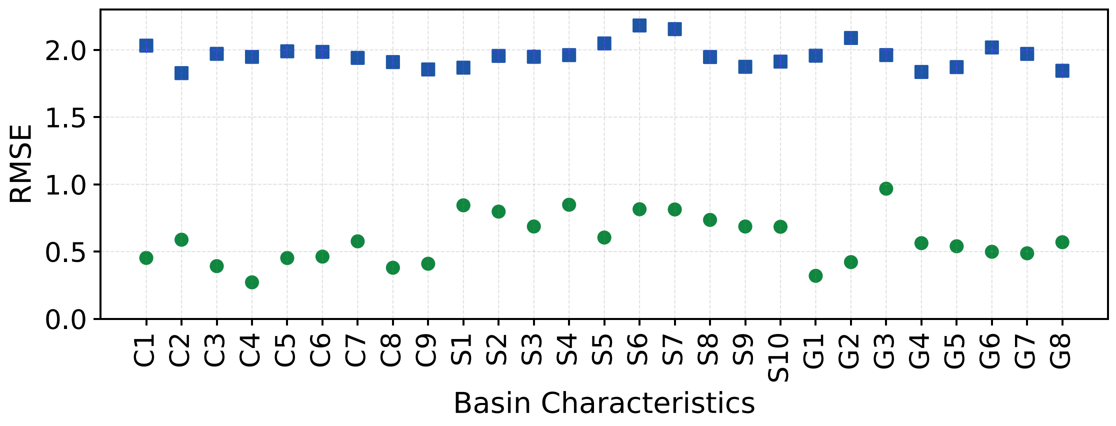
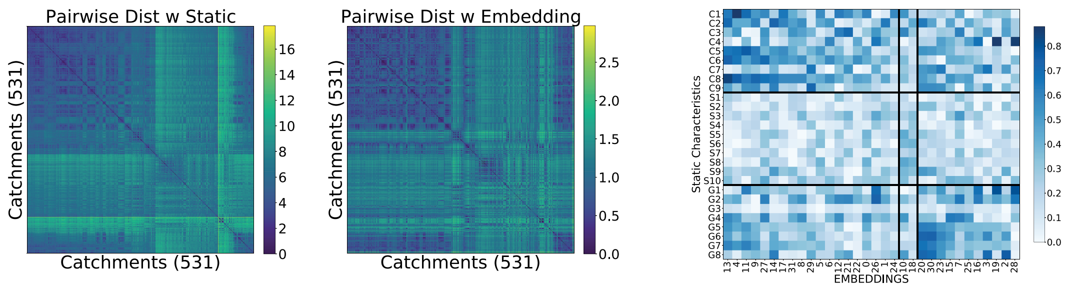


Figure 5.6: *Comparison of the RMSE of the corrupted values (in blue) generated by adding Gaussian noise ($\mathcal{N}(0, 2\sigma_i)$) to 50% of basins and the reconstructed values (in green).*

5.4.4 Robustness to Missing characteristics

Representing physical processes, catchment characteristics often serve as a unique catchment signature. However, owing to the availability of various data sources, characteristics that represent one region are likely not available in another region. It therefore creates a common and important application scenario where a complete set of catchment characteristics across catchments are not assured. This limitation is more pronounced for cross-continental catchments whose characteristics are overlapping rather than exactly matching with each other. For instance, over half of catchment characteristics (e.g., main stream length, bulk density) in CAMELS-CN (a version of CAMELS for China) [146] are not included in the characteristics set of the catchments in the CAMELS dataset being used in this Chapter (which only contains basins from USA) [93]. The same scenario is also present in CAMELS version for Great Britain [147], Chili [148], and Brazil [149]. In addition, insufficient understanding of catchment processes will also lead to a select set of characteristics that miss the opportunity to capture certain hydrological processes beyond current hydrological understanding. To address this issue, the KGSSL can potentially estimate catchment characteristics when they are missing for some catchments. To emulate such a scenario of missing catchment characteristics, we use a similar set up as in Sec. 5.4.3. Instead of adding Gaussian noise, we treated 50% and 90% of the characteristics to be missing. We trained separate models for each of these settings on train years and train basins, where $\mathcal{L}_{Inv}$ was calculated and used for training the model only when characteristics were available. The catchment characteristics were predicted using the data from the test years for all the basins and compared to the original catchment characteristics.

The predicted catchment characteristics using data from the test years are compared to the original catchment characteristics. For the setting with 50% missing data, the average RMSE and average correlation of the predicted values are 0.540 and 0.802 respectively, whereas for 90% missing data, the average RMSE and average correlation of the predicted values are 0.646 and 0.725 respectively. This result suggests that KGSSL can potentially be used to impute the missing characteristics. Further, Table 5.2 shows the robustness of our method, where we predict the characteristics for the 131 test catchments using the data from the test years using both the models and compare the prediction performance to the model trained using the clean data (Section 5.4.2).



(a) (b) (c)

Figure 5.7: *Represent pairwise distance matrices for 531 catchments. Fig (a) Entry (i,j) is the pairwise distance between characteristic vector of catchments i and catchment j; Fig (b) Entry (i,j) is the pairwise distance between embedding vectors generated using KGSSL for catchment i and catchment j (c) Correlation of each dimension of the learned embeddings with each physical characteristic.*

5.4.5 Discovering characteristics in the absence of ground truth(known characteristics)

Here we investigate the ability of KGSSL to identity time invariant characteristics that may be missing from available characteristics. We train the inverse model without using any knowledge of available characteristics that we can use as a constraint. $\mathcal{L}_{Rec}$ and $\mathcal{L}_{Cont}$ are used for training the model using the data from train years for all 531 basins. Further, using the data from the test years the embeddings for each basin are computed. To empirically demonstrate the characteristics captured by the learned embeddings, we calculate the pairwise-euclidean distance between two basins using their 27d physical characteristics (Figure 5.7a) and compare them with the distances computed using learned embeddings (Figure 5.7b). We generate (Figure 5.7a) by reordering the rows in the distance matrix computed using 27d physical characteristics such that basins with the least distances between themselves are placed close to each other to form a band-like structure. The exact order of basins used to generate (Figure 5.7a) is then further applied to the distance matrix computed using learned embeddings to generate (Figure 5.7b). From the figure we observe similar patterns in both the distance matrices which shows that KGSSL generates embeddings that contains meaningful similarity structure between basins. Further, we calculate the correlation between the learned embeddings with each of the physical characteristics for 531 basins. Figure 5.7c provides a measure of the relative contribution of the 3 groups (C1-9 Climate, S1-10 Soils/ Geology, G1-8 Geomorphology/Landcover) for explaining the rainfall-runoff process. The vertical axis represents the 27 Static Characteristics, and the horizontal axis is the embeddings ranked from highest average correlation across characteristics (left) to lowest average correlation (right). Note that S1 and G3 have a weak correlation across all embeddings, Collectively the Climate characteristics show the strongest correlation, followed by geomorphology/landcover. The soil and geology group represent the weakest correlation. This might be expected since soil , and geologic properties have high spatial variability as discussed earlier.

5.4.6 Forward Modeling based evaluation

In previous sections, we observed that KGSSL is able to recover characteristics under missing/uncertain scenarios. In this section, we take one step further and plug our retrieved values in state-of-the-art hydrological models to evaluate the gains achieved in streamflow prediction performance by these retrieved values compared to the missing/uncertain values. LSTMs are extensively used for environmental modeling where both static and time-series variables are supplied as input (here, static characteristics are repeated at each time-step). However, the original RNN models were not designed to exploit static data. Recently, EA-LSTM [21] has emerged as one of the state-of-the-art ML-based forward models used in hydrology that processes the time-series meteorological drivers conditioned on static characteristics. Henceforth, we compare the streamflow prediction performance of the EA-LSTM model in two settings: KGSSL inferred features and original basin characteristics. We report Nash–Sutcliffe model efficiency coefficient (NSE) score for each forward model run.

Method	**Mean NSE**
Baseline(uses known characteristics)	0.704
Baseline (100% missing)	0.491
KGSSL (10% missing)	0.697
KGSSL (50% missing)	0.700
KGSSL (90% missing)	0.664
KGSSL (100% missing)	0.669

Table 5.3: *Model performance for different percentages of missing values.*

Forward modeling with missing entity characteristics

By design, KGSSL is trained using both supervised and unsupervised loss and has generalizations capability to infer the missing basin characteristics that can eventually enhance the streamflow prediction when basin characteristics are missing/not available. We train all models on all 531 basins during the train years and test the performance during the test years. Table 5.3 (first two rows) report performance of state-of-the-art EA-LSTM (baseline) trained with all and no static characteristics [21]. The baseline model where all characteristics are present performs **43**% better (mean NSE) than the baseline model when some or all basin characteristics are missing. This shows the importance of the basin characteristics in modulating the driver-response network.

To evaluate how much inferred basin features help in the forward model, we randomly treat 10%, 50%, 90%, and 100% of the characters to be missing. We impute missing characters using our KGSSL pipeline and run it through the forward model. Table 5.3 (last 4 rows) report NSE performance when our model was used to fill in the static characteristics for different percentages of missing values. We observe that the forward model trained with reconstructed characteristics from KGSSL with *10%* and *50%* missing values perform similar to the baseline trained with all characteristics. Further with *90%* missing characteristics, the forward model observes only 5% drop when compared with baseline. In addition, for 100% missing characteristics, we use a total unsupervised setting in our KGSSL framework, i.e., generate embeddings instead of characteristics. The KGSSL model with no supervision perform **35**% better to the baseline with 100% missing characteristics. Even more impressive is the fact that KGSSL with no supervision (last line) is only slightly worse than the baseline that uses known characteristics. We attribute this success to our framework's knowledge-guided component, which implicitly extracts complex correlations embedded in the input data.

Method	**Mean NSE**
Baseline(actual characteristics)	0.560
Baseline ($0.5\sigma_i$ noise)	0.474
Baseline ($1\sigma_i$ noise)	0.245
KGSSL (1 year)	0.460
KGSSL (2 year)	0.535
KGSSL (3 year)	0.554
KGSSL (9 year)	0.582

Table 5.4: *Forward model performance with corrupted characteristics and using KGSSL embeddings. In KGSSL (n-year), the n refers to the number of years of data utilized to learn the embedding.*

Forward modeling with corrupted entity characteristics

As shown by Kratzert et al. [21], uncertainty or corruption in basin-characteristics can have detrimental effect on the forward modeling. To demonstrate this, we trained the baseline model on the 400 train basins in the train years and test the performance on the 131 test basins in the test years. Table 5.4 (first row) shows the baseline performance of the forward model. To model uncertainty in the static characteristics, we add Gaussian noise ($\mathcal{N}(0, 0.5\sigma_i)$,$\mathcal{N}(0, 1\sigma_i)$) to test characteristics and measure the performance (Table 5.4 - 2^{nd}, 3^{rd} row) of the baseline model. As expected, the forward model is susceptible to noise in the basin characteristic, and the performance drops significantly with slight noise. Specifically, the mean NSE drops by 50% with a 1 standard deviation noise.

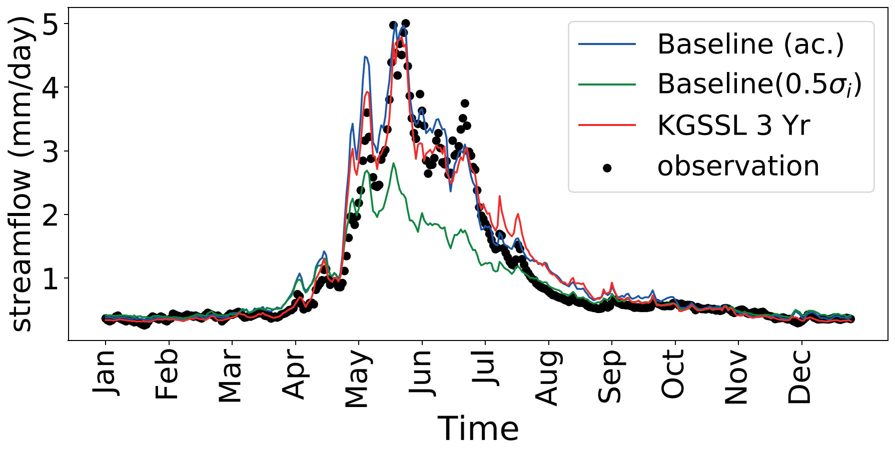


Figure 5.8: *Basin at year 1992 (Best seen in color)*

If the basin characteristics are corrupted, we can utilize representation obtained from the KGSSL trained in self-supervised manner using n-years of observations (note that this approach does not need any information about characteristics but it does need a small amount of data to create the embeddings). Table 5.4 (second set of rows), showcases the power of this methodology. As expected, the performance improves as we use more data to generate embeddings. Note that with only 2 years of data, the EALSTM model using *KGSSL(2 year)* outperforms the EALSTM using *Corrupted* $\mathcal{N}(0, 0.5\sigma_i)$ characteristics. Moreover, *KGSSL(9 year)* generated with 9 years of train data outperform the model with *actual characteristics*. In Figure 5.8 we plot the actual observed streamflow (black dot) as well as the predicted streamflow using the various settings of the forward model. We observe that baseline imputation with corruption performs poorly and is nowhere close to the actual values. We also note that baseline prediction (blue line) closely matches our KGSSL predicted values using only 3 years of data (red line). We attribute this good result to our novel pretext task that is able to handle time invariant physical processes.

5.5 Discussion and Future Work

In this work, we build a novel inverse framework KGSSL, and demonstrate the power of leveraging domain knowledge between entities in the context of streamflow. We performed extensive experiments on the hydrological benchmark dataset and show that KGSSL outperforms baseline significantly by a margin of 16-35 % under various situations. KGSSL is a first-of-its-kind knowledge-guided framework that implicitly extracts system characteristics given its driver and response data. This Chapter addresses an important problem in the hydrologic domain, which is societally relevant. We note that the proposed method is general and can add value in other applications such as computer vision (self-driving car), where additional features are used to capture variations in light, weather, and object poses. Note that KGSSL learns representations without focusing on optimizing the response variable (i.e., streamflow prediction in our hydrology application). KGSSL can be further extended by combining both the forward and inverse model in a unified framework that first uses the inverse model to generate a representation and then uses the learned representation to modulate the forward model. Such an extension can leverage the recent work from task-aware modulation in machine-learning [32, 53], and will be considered in future work.

Chapter 6

Few Shot Entity Aware Modulation: An Application in Hydrology

6.1 Introduction

In practical, real-world applications, constructing robust, individualized prediction models with limited training data for each task or entity is often necessary. Machine learning (ML) techniques that share information between entities/tasks/sources have the potential to address data scarcity. However, source heterogeneity (variation in data distribution and generating process) can lead to sub-optimal personalized predictions if the data is trivially merged. Such heterogeneities are prevalent in many real-world scenarios, such as differences in user behavior in e-commerce systems [4], and in streamflow dynamics, in different catchments, [150, 21].

For example, in streamflow modeling, the response (streamflow) is influenced by meteorological drivers (e.g., air temperature, precipitation, wind speed) and specific complex physical processes unique to each entity [2]. These complex physical processes are best captured by the entity's inherent characteristics (e.g., slope, land cover). As a result, two entities can have different streamflow responses to the same amount of precipitation based on their land cover. The streamflow modeling is just one example of a wide variety of scientific models that can be considered as a mapping function between drivers $\boldsymbol{x_t}$ (e.g., weather drivers, climate forcings) and response y_t (e.g., streamflow in a river basin, global average temperature), governed by entity characteristics. Thus ML models must utilize these entity characteristics to personalize the predicted response to external drivers for each entity. In Deep Learning, this is often referred to as process modulation, where an auxiliary input such as entity characteristics modulates the output [21, 120].

However, these entity characteristics are often surrogate variables of the true characteristics [124] and thus can lead to several challenges. First, there is often uncertainty associated with these inherent characteristics in hydrology, which arise due to temporal change, spatial heterogeneity, the sufficiency of the characteristic to explain the rainfall/runoff process, measurement error, missing data, and correlation among characteristics that collectively contribute to streamflow. Second, the complete set of basin characteristics may only be measured across some of the river basins, resulting in the incompleteness of basin characteristics. Missing characteristics hinder building a global model that can leverage data across multiple basins and constrains the transferability of models built from one region to another. Finally, some basin characteristics may be essential in modeling the rainfall-runoff response relation but may be completely unknown, poorly understood, or not present in the available basin characteristics. These problems often lead to a performance drop, as shown in [22, 44].

Model agnostic meta-learning (MAML) has been a popular paradigm for personalized prediction tasks in the presence of data scarcity [46]. The central idea of MAML is to train a meta-model on a diverse set of tasks in two steps: a)meta-training and b) meta-validation. In meta-training, the model is finetuned for a given task, and then the finetuned model is evaluated during meta-validation. The meta-model is trained using the loss calculated during meta-validation. This training procedure simulates the few-shot setting; thus, the trained meta-model is expected to adapt to new tasks with a few examples. However, because the properties of different tasks differ, a single meta-model impedes efficient adaptation for a varied group of functions. To overcome this issue, task-aware modulation techniques that alter the shared hidden features based on task embeddings of inherent tasks have been proposed [32, 58]. One such popular state-of-the-art approach is Multimodal model-agnostic meta-learning (MMAML) [32], which augments MAML with the capability to identify the mode of tasks sampled from a multimodal task distribution and adapts to that mode through gradient updates. However, adapting the whole parameter set through a few gradient updates can lead to instability issues depending on the architecture and hyperparameter choices. These challenges are further exacerbated by significant training times and a cumbersome hyperparameter selection process.

This Chapter presents a novel framework called Task aware modulation using representation learning (TAM-RL) that can incorporate inherent entity characteristics into ML algorithms to improve personalized predictions. We propose an embedding-based lightweight adaptation strategy that does not suffer from the instability problems of MMAML. In TAM-RL, task-specific embeddings are learned from data, which is further trained along with the forward model in a unified fashion. We evaluate two key strategies to address heterogeneity between entities: task-aware representation learning and task-aware adaptation. This Chapter focuses on predicting in limited gauged basins scenario. The model is trained in well-observed basins, transferred to new sparsely observed basins through inherent features, and adapted using the limited observations available. Predicting streamflow for limited gauged basins is an important problem with societal impact because, for much of the world, accurate streamflow measurements are very sparse and often expensive, especially in developing countries where unavailability or limited availability of historical streamflow data poses a significant limitation in streamflow forecasting. This Chapter shows that entity modulation can be achieved without explicit entity-specific inherent characteristics. Using a real-world continental scale hydrology dataset CARAVAN-GB, we show that Entity Aware modulation can outperform existing approaches such as MAML and MMAML in a few-shot learning setting. We further explore various artificial scenarios using simple synthetic datasets to compare the performance of TAM-RL with other approaches like MAML and MMAML. Our main contributions can be summarized as follows:

- We introduce a new Meta Learning framework for learning ML models for a diverse set of entities for a few-shot setting.
- We perform an extensive evaluation to show TAM-RL's state-of-the-art performance in predicting streamflow in a continental scale hydrology benchmark dataset CARAVAN-GB while being more straightforward and much faster to train.

- We present an empirical evaluation via synthetic data to explore the impact of heterogeneity amongst the entities on the relative performance of MAML, MMAML, and TAM-RL.

6.2 Related Works

Several works have shown the benefit of building a global model that effectively leverages data from different entities. They usually assume that the task-specific characteristics are explicitly available and incorporate them in the network modulation. These methods have been proposed across multiple disciplines such as precision engineering [28], healthcare [30], and environmental sciences [150]. However, this approach may not always be feasible in real-world situations where such task characteristics are often unknown. In this scenario, generalizing to out-of-sample tasks, i.e., tasks that the model has yet to not encountered during training, becomes challenging. However, when few observation samples are available for the out-of-sample tasks, few-shot learning methods, such as Meta Learning [46], have been proposed.

Meta Learning: Meta-learning methods leverage the shared structure between different training tasks, leading to better generalization and adaptation in few-shot learning applications [46]. Model Agnostic Meta-Learning (MAML) [31] is a popular approach that learns a global meta-model, which can then be easily adapted to create personalized models for each entity using limited training data. One common challenge faced by the MAML has been adapting the whole parameter set of the ML model using the few-shot observations. Several variations of MAML have been proposed that address the issue by only adapting the high-level layers of the meta-model [50, 51]. A common assumption in many MAML-based methods is that all entities, both during training and testing, are drawn from the same distribution. This assumption can pose challenges when dealing with entities whose data distributions are different and multimodal, and thus, it could be challenging to adapt a single meta-initialization [32].

Conditional Meta Learning: If the entity distribution has multiple and distinct modes, multiple meta-learners may be more effective at handling the entire distribution. Forming groups of tasks such that a separate meta-model can be assigned to each group requires additional entity information. Some methods assume the availability of task-specific information beforehand [29], but this is not always possible when the modes are not distinct. In these cases, a common approach is to train another network to convert training data from seen entities into entity-specific embeddings, which modulate a shared prediction network [32]. The prediction and embedding networks can either be trained together or alternately. However, these methods face the same challenge as MAML of adapting the whole parameter set.

Representation Learning: Another line of research has been to encode the tasks into low-dimensional latent embeddings that enable better adapt the behavior of the model [63]. The advantage of this approach is that compared to MAML-based approaches that are computationally expensive and sensitive to hyperparameter choices, they do not need to instantiate and explicitly maintain a unique set of model parameters. In this direction, Ghosh et al. [22] introduced a novel framework called KGSSL that can infer time-invariant entity attributes from driver-response behavior. By using these inferred attributes, they demonstrated performance comparable to that of an LSTM network with known basin characteristics. Our proposed approach is different from KGSSL in the way that the task encoder and the forward model are trained jointly.

6.3 Problem Formulation

In this work, we focus on learning driver-response behavior using ML models for a set of multi-modal entities. An entity can be a physical system such as a lake or river basin, a task, a person, or a domain/distribution. We have access to multiple driver/response pairs of time series sequences for each entity i. The daily drivers are represented by $\boldsymbol{X_i}$ as a multivariate time series 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 input vector at time $t \in T$ with D_x dimension, $\boldsymbol{Y_i} = [y_i^1, y_i^2, \ldots, y_i^T]$ be the corresponding output. Given the set of n entities in the Training set, for each entity i, we can access the set of multivariate time series instances of corresponding inputs and output pairs, D_i^{Train} $= [(\boldsymbol{X_i^1}, y_i^1), (\boldsymbol{X_i^2}, y_i^2), \ldots, (\boldsymbol{X_i^{T_{Train}}}, y_i^{T_{Train}})]$. For the set of entities in the test set, a few shot data of inputs and outputs, $D_j^{Few} = [(\boldsymbol{X_j^1}, y_j^1), (\boldsymbol{X_j^2}, y_j^2), \ldots, (\boldsymbol{X_j^{T_{Few}}}, y_j^{T_{Few}})]$ is provided for each entity j.

The goal is to learn a regression function $\mathcal{F} : X \rightarrow Y$ that maps the input drivers to the output response for each entity. Individual ML models can be trained for each entity with sufficient training data. However, this is not feasible for many entities which lack sufficient training data. The behavior of entities is often governed by their inherent characteristics (z_i). Thus the forward model is represented as $\mathcal{F}_\theta(x_i^t, z_i)$, where θ denotes the function class shared by the target systems and z_i denotes entity-specific inherent characteristics. The major challenge lies in handling the heterogeneity across different entities to achieve good performance over all entities, particularly in scenarios where the measurement of entity characteristics is unavailable. In this Chapter, we propose an entity-aware modeling approach to overcome the challenge of building a global model that can accurately predict the response of a new entity with limited observations, despite the heterogeneity across different entities.

Conforming to the Meta-learning literature, we divide the total data for each entity into a support set $D_i^{support}$ and a query set D_i^{query}. For the entities used in training, both $D_i^{support}$ and D_i^{query} are derived from the training data D_i^{Train}. Whereas, for the entities in the test set, the few shot data D_j^{Few} forms the support set. During training, meta-learning approaches utilize $D_i^{support}$ to capture the temporal correlations, multivariate relationships within the time series, and the inherent task characteristics associated with each entity i and use it for prediction in D_i^{query}. Thus, the goal is to train the model on D_i^{Train} by minimizes the prediction error:

$$\arg\min_\theta ||y_i^t - \mathcal{F}(\boldsymbol{x_i^t}, D_i^{support})||^2 \quad s.t. \quad (\boldsymbol{x_i^t}, y_i^t) \in D_i^{query} \tag{6.1}$$

During inference, the model uses the few-shot data D_j^{Few} to accurately predict the target value for a new, unseen multivariate time series instance for the new entity, as shown,

$$\hat{y}_j^t = \mathcal{F}(\boldsymbol{x_j^t}, D_j^{Few}) \tag{6.2}$$

6.4 Preliminaries

The objective of meta-learning is to rapidly learn task-specific functions that can transform input data into the desired output using D_j^{Few} for various tasks j, even when the amount of data is limited. Tasks are defined by the distribution that generates the data $\mathcal{P}(\mathrm{X})$ and the conditional probability $\mathcal{P}_t(\mathrm{Y} \mid \mathrm{X})$. This study aims to quickly adapt to a novel task from a multimodal task distribution (a mixture of different classes of functional families/modes).

MAML: MAML [31] seeks to find an optimal initial set of parameters, θ, for a meta-learner that can be efficiently adapted to new tasks with just a few gradient steps. This adaptation involves minimizing the task-specific loss on the training data, $\mathcal{D}_i^{\text{support}}$, while ensuring that the adapted parameters generalize well to the validation data, $\mathcal{D}_i^{\text{query}}$. To train the initial parameters, MAML samples mini-batches of tasks from $\mathcal{D}_i^{\text{support}}$, computes the adapted parameters for each task in the batch, evaluates the adapted parameters on the validation data $\mathcal{D}_i^{\text{query}}$, and updates the initial parameters, θ, based on the gradients from the validation losses. The overall training process is formulated as a bi-level optimization problem, as shown in equation 6.3. This allows MAML to find the optimal initial parameters that lead to the best adaptation to new tasks.

$$\begin{aligned} &\theta^* = \arg\min_{\theta} \sum_{i \in \mathcal{P}(i)} \mathcal{L}(\mathcal{D}_i^{query}; \theta_i^*) \\ \text{s.t.} \qquad &\theta_i^* = \arg\min_{\theta} \mathcal{L}(\mathcal{D}_i^{support}; \theta) \end{aligned} \tag{6.3}$$

MMAML: MMAML, as introduced in [32], builds upon the Model-Agnostic Meta-Learning (MAML) algorithm [31]. Its aim is to find an optimal initial set of parameters, θ, for a meta-learner that can be efficiently adapted to new tasks with just a few gradient steps. MMAML achieves its objective by using two complementary neural networks. The first network, consists of two components a task encoder $\mathcal{E}$ and an MLP layer . The task encoder first uses $\mathcal{D}_i^{\text{support}}$ to predict a task embedding $\mathbf{h}$, the task embedding is then passed through the MLP layer to predict task specific parameter τ. These task specific parameter τ are then further used to modulate the prior parameters of the second network, the task network. The modulated task network is then optimized for the target task through few steps of gradient-based optimization. The overall training process is formulated as a bi-level optimization problem, as shown in equation 6.4.

$$\begin{aligned} &\theta^* = \arg\min_{\tau,\theta} \sum_{i \in \mathcal{P}(i)} \mathcal{L}(\mathcal{D}_i^{query}; \theta_i^*) \\ \text{s.t.} \quad &\theta_i^* = \arg\min_{\tau} \mathcal{L}(\mathcal{D}_i^{support}; \theta; \tau) \end{aligned} \tag{6.4}$$

Neural Process: Neural processes [63] bring together the advantages of Deep

Neural Networks and Bayesian techniques like Gaussian Processes (GPs) to use prior knowledge and efficiently predict the form of a new function. The defining aspect of NP is its use of an inferred entity embedding to condition the prediction function on observed data. The model consists of three main components: an encoder that creates a representation from each input-output pair, an aggregator that combines these representations into a single embedding, and a conditional decoder that generates target predictions using the embedding and inputs.

$$\begin{aligned} h_i^t &= q_\phi(\boldsymbol{x_i^t}, y_i^t) \quad s.t. \quad (\boldsymbol{x_i^t}, y_i^t) \in D_i^{support} && \text{encoder} \\ z_i &= h_1 \oplus \cdots \oplus h_n && \text{aggregator} \\ y_i^t &= p_\theta(x_i^t, z) \quad s.t. \quad (\boldsymbol{x_i^t}, y_i^t) \in D_i^{query} && \text{conditional decoder} \end{aligned} \tag{6.5}$$

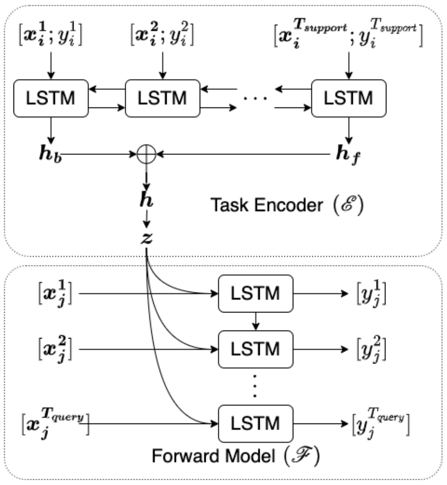

$[\boldsymbol{x_i^1}; y_i^1]$
$[\boldsymbol{x_i^2}; y_i^2]$
$[\boldsymbol{x_i^{T_{support}}}; y_i^{T_{support}}]$
LSTM
LSTM
LSTM
$\boldsymbol{h_b}$
$\boldsymbol{h_f}$
$\boldsymbol{h}$
Task Encoder ($\mathscr{E}$)
$\boldsymbol{z}$
$[\boldsymbol{x_j^1}]$
$[\boldsymbol{x_j^2}]$
$[\boldsymbol{x_j^{T_{query}}}]$
LSTM
LSTM
LSTM
$[y_j^1]$
$[y_j^2]$
$[y_j^{T_{query}}]$
Forward Model ($\mathscr{F}$)

Figure 6.1: *Model overview: The task encoder produces an embedding h , which is used to modulate the forward model to fit the target task.*

6.5 Method

Our proposed task-aware modulation using representation learning (TAM-RL) is designed to extract task embeddings (latent features or underlying characteristics) of entities based on the time-varying driver (X_i) and response (Y_i) data. These features are then utilized to make predictions of an entity's response given the drivers in a unified fashion. The method is structured with two main components: a sequence encoder $\mathcal{E}$

(known as the task encoder) and a decoder $\mathcal{F}$ (forward model), as illustrated in Fig. 6.1. The encoder network is trained to learn a task embedding that represents the entity's characteristics and enhances the forward modulation process. On the other hand, the decoder network is trained to produce the entity's response using the task embeddings and the driver data as inputs. In the following sections, we will discuss the choice of neural network architectures used and the details of the training process.

6.5.1 Task Encoder

We implement a sequence encoder, specifically a Long Short-Term Memory (LSTM) network, to encode the temporal information and the relationship between the driver and response in sequences. LSTMs are well-suited for tasks that involve long-range temporal dependencies between the driver and response and are designed to overcome problems with exploding and vanishing gradients. However, LSTMs can only process sequences in the forward direction and do not provide insight and explainability into the current time steps based on future data. We utilize a Bidirectional LSTM ([151] based sequence encoder to address this. This encoder is made up of two LSTMs, a forward LSTM and a backward LSTM, where the time series is reversed for the backward LSTM. Each of our forward and backward LSTM uses the following equations to generate embeddings for a sequence, which capture the temporal information and the interaction between the driver and response, as shown below.

$$\begin{aligned}
\boldsymbol{i_t} &= \sigma(\boldsymbol{W_i}\left[[\boldsymbol{x^t}; y^t]; \boldsymbol{h^{t-1}}\right] + \boldsymbol{b_i}) \\
\boldsymbol{f_t} &= \sigma(\boldsymbol{W_f}\left[[\boldsymbol{x^t}; y^t]; \boldsymbol{h^{t-1}}\right] + \boldsymbol{b_f}) \\
\boldsymbol{g_t} &= \sigma(\boldsymbol{W_g}\left[[\boldsymbol{x^t}; y^t]; \boldsymbol{h^{t-1}}\right] + \boldsymbol{b_g}) \\
\boldsymbol{o_t} &= \sigma(\boldsymbol{W_o}\left[[\boldsymbol{x^t}; y^t]; \boldsymbol{h^{t-1}}\right] + \boldsymbol{b_o}) \\
\boldsymbol{c_t} &= \boldsymbol{f_t} \odot \boldsymbol{c_{t-1}} + \boldsymbol{i} \odot \boldsymbol{g_t} \\
\boldsymbol{h_t} &= \boldsymbol{o_t} \odot \tanh(\boldsymbol{c_t})
\end{aligned} \tag{6.6}$$

The final embeddings (h) are obtained by adding the last hidden states from the forward (h_f) and backward LSTMs (h_b) as shown in the Task encoder $\mathcal{E}$ in Fig 6.1. These learned embeddings are further passed through a multi-layer perceptron (MLP) layer to obtain the latent entity characteristics.

$$\begin{aligned} \boldsymbol{h} &= \mathbf{BiLSTM}([x^t; y^t]_{1:T}; \phi_h) \\ \boldsymbol{z} &= \mathbf{MLP}(h; \phi_z) \end{aligned} \tag{6.7}$$

where, ϕ_h and ϕ_z are the BiLSTM and MLP parameters, respectively.

Algorithm 1 TAM-RL Training Algorithm

```
Input: $W^{Tr_\mathcal{T}} = [W^{support_\mathcal{T}}, W^{query_\mathcal{T}}]$ generated from $D^{Train}$, $\alpha$: step size hyper-
    parameter; We abbreviate support as s and query as q
 1: Randomly Initialize the pipeline with weights $\theta$ and $\omega$
 2: for epoch = 1 to $N$ do
 3:     while not Done do
 4:         Sample batches of entities $W_k^{Tr_\mathcal{T}} \sim W^{Train_\mathcal{T}}$
 5:         for all k do
 6:             Infer z = $\mathcal{E}(W_k^{s_\mathcal{T}}; \omega)$
 7:             Update $\theta$ and $\omega$ with $\alpha \nabla_{\theta,\omega} \mathcal{L}_{W_k^{Tr_\mathcal{T}}}(\mathcal{F}(x_k^{q_t}; \theta, z); W_k^{q_\mathcal{T}})$
 8:         end for
 9:     end while
10: end for
```

6.5.2 Forward Model (Decoder)

Next, we implement an LSTM-based sequence decoder $\mathcal{F}$ to predict the response given input drivers and the inferred hidden characteristics z_i from the task encoder $\mathcal{E}$, as shown in the Forward model block of Fig 6.1. Specifically, it is a sequence-to-sequence LSTM network such that the response $y^t = \mathcal{F}_\theta(\boldsymbol{z}, \boldsymbol{x^{1:t}}; \theta)$, where $(\boldsymbol{x^t}, y^t) \in D^{query}$ and $z = \mathcal{E}([\boldsymbol{x^t}, y^t]^{1:T}; \phi)$, where $(\boldsymbol{x^t}, y^t) \in D^{support}$. During inference time, when adaptation is used for TAM-RL, the first network, the task encoder $\mathcal{E}$, is used to infer entity-specific parameters that are used to modulate the prior parameters of the second network, the forward model $\mathcal{E}$. The modulated forward model is then optimized for the target task through gradient-based optimization for the required number of inner steps, just like in the case of MMAML [32].

The whole framework is trained end to end using a forward loss, which can be any supervised loss depending on the problem. In our experiments, we train our model with mean squared error (MSE) as shown in equation 6.8.

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{T}\sum_{j=1}^{T}(y_i^j - \hat{y}_i^j)^2 \tag{6.8}$$

Algorithm 2 TAM-RL Inference Algorithm

```
Input: $W_j^{support_{\mathcal{T}}}$ generated from $D_i^{Few}$, $x_j^{query_t}$, $\beta$: step size hyper-parameter,
    num_inner_steps: no. of gradient steps for the inner loop; We abbreviate sup-
    port as s and query as q
Output: Output $y_j^{qt}$ for input driver $x_j^{qt}$ of an entity j
1: load $\theta$ and $\omega$ from trained model
2: Infer z = $\mathcal{E}(W_j^{s\mathcal{T}};\omega)$
3: for epoch = 1 to num_inner_steps do
4:     Update $\theta$ with $\beta\nabla_\theta\mathcal{L}_{W_j^{s\mathcal{T}}}(\mathcal{F}(x_j^{st};\theta,z);W_j^{s\mathcal{T}})$
5: end for
6: return $y_j^{qt} = (\mathcal{F}(x_j^{qt};\theta,z)$
```

6.5.3 Pseudo Code

To handle the difficulty of feeding a long time series into a model, during the training phase, we create $\mathcal{T}$ sliding windows ($W^{Tr_{\mathcal{T}}}$) from the given training data (D^{Train}). These windows are split into support windows ($W^{support_{\mathcal{T}}}$) and query windows ($W^{query_{\mathcal{T}}}$), and used for training the model. The procedure is described in Algorithm 1.

In the inference phase, given a few shots of data (D_j^{few}) for a new entity j, we generate $\mathcal{T}$ support sliding windows ($W_j^{support_{\mathcal{T}}}$). We use these support windows and the input data ($x_j^{query_t}$) to make predictions. The procedure is outlined in Algorithm 2.

6.6 Experimental Results

In this work, we aim to answer two main research questions:

- Can TAM-RL learn from a diverse set of entities and perform well on a new set of entities in a few-shot learning setting for a real-world problem?

- How does it compare to the other state-of-the-art approaches like MAML and MMAML for personalized response prediction in diverse entities?

In this section, we evaluate TAM-RL on real-world and synthetic datasets. Specifically, using these datasets, we shed light on the scenarios where TAM-RL gives similar, better, or worse performance than the baseline approaches.

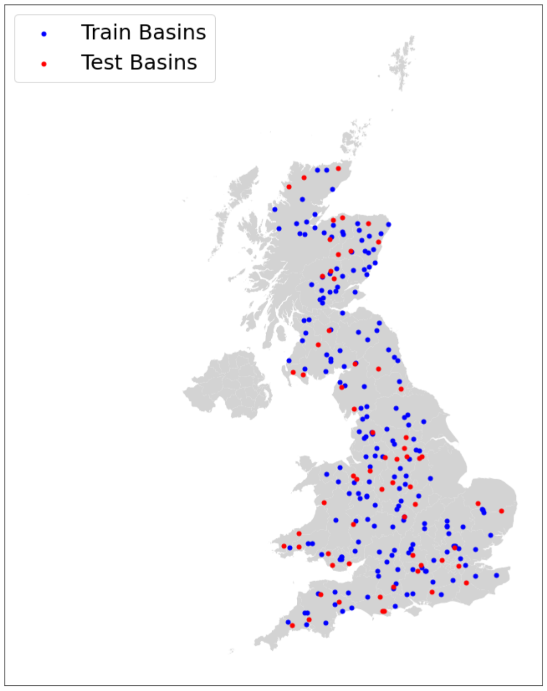


Figure 6.2: *Geographical locations of Train (red) and Test (yellow) basins Caravan-GB*

6.6.1 Datasets

CARAVAN GB: We evaluate the performance of our approach and the baselines on a publicly available real-world hydrology benchmark dataset of streamflow modeling [152]. CARAVAN is a newly published global benchmark comprising 6830 basins aggregated from several existing open datasets worldwide from the US, UK, GB, Australia, Brazil, Chile, etc. This dataset provides meteorological forcing data (e.g., precipitation, potential evaporation, temperature, surface net solar radiation, etc.), streamflow observations, and basin characteristics (complete list in Appendix). We use a subset encompassing 408 basins of the UK present in CARAVAN and will be referred to as the CARAVAN-GB in subsequent discussions.

Synthetic Dataset: We follow the synthetic data generation setup used by Vuorio et al. [32]. Specifically, we create a set of task distributions for regression with different modes. Modes are a collection of functions that includes sinusoidal functions, linear functions, quadratic functions, $\ell 1$ norm functions, and hyperbolic tangent functions. Each task/function from these modes is created by changing as shown:

- Sinusoidal functions: $A \cdot \sin w \cdot x + b + \epsilon$, with $A \in [0.1, 5.0]$, $w \in [0.5, 2.0]$ and $b \in [0, 2\pi]$
- Linear functions: $A \cdot x + b$, with $A \in [-3, 3]$
- Quadratic functions: $A \cdot (x - c)^2 + b$, with $A \in [-0.15, -0.02] \cup [0.02, 0.15]$, $c \in [-3.0, 3.0]$ and $b \in [-3.0, 3.0]$)
- $\ell 1$ norm functions: $A \cdot |x - c| + b$, with $A \in [-0.15, -0.02] \cup [0.02, 0.15]$, $c \in [-3.0, 3.0]$ and $b \in [-3.0, 3.0]$
- Hyperbolic tangent functions: $A \cdot tanh(x - c) + b$, with $A \in [-3.0, 3.0]$, $c \in [-3.0, 3.0]$ and $b \in [-3.0, 3.0]$

We add Gaussian noise ($\mu = 0$ and standard deviation $= 0.3$) to each data point obtained from the individual task.

6.6.2 Baselines & Implementation Details

- $LSTM$: We train a global model by feeding only the input drivers into an LSTM to predict the response without using static characteristics of the entities. To improve the model's performance for each specific basin, we further fine-tune it using the respective few-shot data to create the $LSTM_{Adapt}$ baseline. We use streamflow data from test basins in a few-shot setting and five inner optimization steps to fine-tune the global model during inference.
- $CTLSTM$: In this we feed the static characteristics of the entities along with the input drivers in a concatenated (CT) fashion into an LSTM to predict the

response for respective entities. Similar to $LSTM$, we further fine-tune the base $CTLSTM$ model to create the $CTLSTM_{Adapt}$ baseline. Note that only this baseline approach uses the entity characteristics which we assume to be unknown for the other methods. This baseline has the most information available to it.

- $KGSSL{-}CTLSTM$: We first train a global model (inverse model) using knowledge-guided self-supervised learning (KGSSL) based inverse framework [22] to first infer the entity attributes in the form of embeddings. Then we train another global CTLSTM model (forward model) using those inferred entity attributes to predict streamflow. During inference, we use the few-shot streamflow observations from the test basins to first infer entity attributes using the inverse model, and use that entity attributes further along with drivers to predict response using the forward model. We use the proposed version of the model, which does not use static characteristics in a few-shot setting.

- $MAML$: We train a global model using MAML [31] as the baseline to compare with existing methods. For our real-world dataset, MAML uses LSTM as the base model. For synthetic datasets, it uses a 4-layer fully connected neural network with ReLU non-linearity for each layer. $MAML_{Adapt}$ refers to the model that is obtained after finetuning the base-model using the few-shot data.

- $MMAML$: We train a global model using $MMAML$ [32] as the baseline to compare with existing methods. For our real-world dataset, $MMAML$ uses LSTM as the task encoder and CTLSTM as the modulation network, which uses input drivers and inherent task/basin attributes as the base model. We use CTLSTM for the base model in a real-world setting as it is the current state-of-the-art model when basin characteristics are available in the case of hydrology. For the synthetic dataset, $MMAML$ uses LSTM as the task encoder and a 4-layer fully connected neural network with ReLU non-linearity for each layer as the modulation network. $MMAML_{Adapt}$ is the finetuned version of $MMAML$.

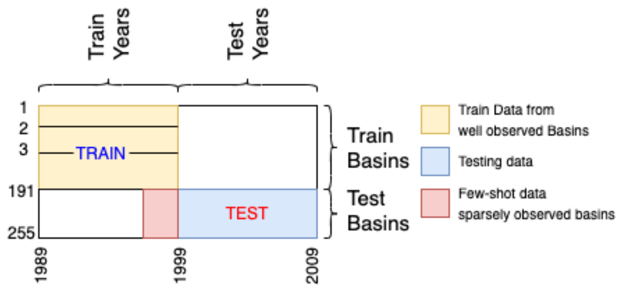


Figure 6.3: *Experimental setting for CARAVAN GB*

Table 6.1: *Mean Ensemble NSE values for streamflow modeling on the benchmark datasets for TAM-RL and the baselines.*

Architecture	**0 Years**	**2 Year**	**5 Years**
$LSTM$	0.18	-	-
$LSTM_{Adapt}$	0.18	0.31	0.41
$MAML$	0.26	-	-
$MAML_{Adapt}$	-	0.48	0.50
$MMAML$	-	0.28	0.28
$MMAML_{Adapt}$	-	0.53	0.56
$KGSSL-CTLSTM$	-	0.53	0.56
$TAM-RL$	-	0.60	0.60
$TAM-RL_{Adapt}$	-	**0.62**	**0.67**
$CTLSTM$	0.50	-	-
$CTLSTM_{Adapt}$	0.50	0.60	0.66

6.6.3 Experiment: CARAVAN GB

Our study utilizes data from October 1st, 1989, to September 30th, 2009, where the model training phase uses data from 1989-1999, while the testing phase covers 1999-2009. Additionally, the period of 1989-1999 is divided into training years (1989-1997) and validation years (1997-1999). Of the available 408 Basins in CARAVAN-GB, we selected 255 basins for our experiment as they had no missing streamflow observation from 1989-1999. Fig 6.3 shows the experimental setting followed in this experiment. We first divide our set of basins into train and test basins. We then use our train basins to build and train our ML models and test basins to evaluate the performance of our models. The selection of train and test basins is made by sorting all the available 255 basins by their mean streamflow. Then, we consecutively select train and test basins using a 3:1 ratio to create a relatively even distribution for train and test sets with 191 and 64 basins, respectively. Fig 6.2 shows the location of the train and test basins where train basins are in the color red, and test basins are in color yellow. When using a test set of basins to evaluate performance, we use limited years of data (two years, five years) from the training period as our few-shot as shown in Fig 6.3 to both fine-tune and infer characteristics if and when required as part of the model.

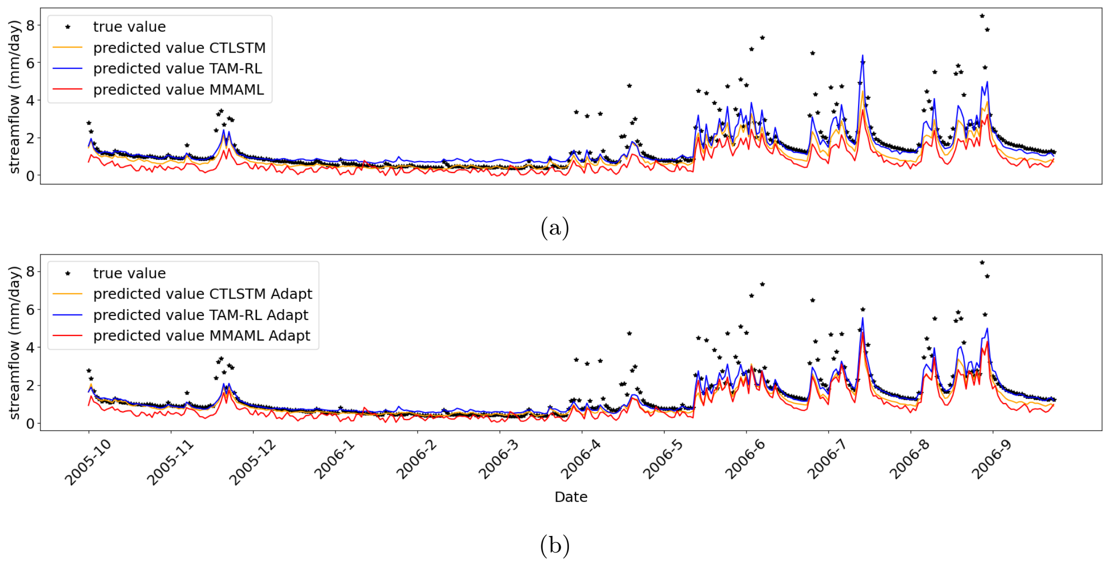


Figure 6.4: *Observed Streamflow and predicting streamflow by different model architectures a) Models w/0 adaptations b)models with adaptations (Best seen in color).*

We use our available time series data to create sliding windows of length 365 days, where each is strided by half the sequence length (183 days). This results in 19 windows for the data, each for model training and testing. The LSTMs in our experiments are sequence-to-sequence networks that use 365-length sequences for input, and output is generated at a stride of 183 days). LSTM used in the base models, forward model in $TAM-RL$, or the baselines have one hidden layer of 256 dimensions. In contrast, the LSTM used as the encoder in $KGSSL-CTLSTM$, $MMAML$, and $TAM-RL$ are bidirectional and use one hidden layer of 32 dimensions. All our model architectures were trained for a maximum of 100 epochs with patience set to 10 on the validation set's performance. $CTLSTM$, $LSTM$, $MAML$, and $TAM - RL$ use a single ADAM [119] optimizer with an initial learning rate of 1×10^{-3}, and $MMAML$ uses two ADAM optimizers for the base model and embedding parameters respectively. Both gradient-based meta-models $MAML$ and $MMAML$ use a fast learning rate of 1×10^{-3} for inner loop gradient updates and five gradient steps during the adaptation in training and inference phases. We train five models with different initialization of deep learning model weights for all architectures to reduce the randomness typically expected with network initialization. The predictions were then combined into an ensemble by averaging predictions from these five models to get a robust performance.

To evaluate the streamflow prediction performance, we use Nash–Sutcliffe efficiency (NSE) as the evaluation metric, commonly used in the hydrological analysis for comparing different models. An NSE score of 1 indicates a perfect time series prediction. An NSE score of zero means model has the same predictive performance as the mean of the time-series in terms of the sum of squared errors, a negative NSE represents a performance worse than a mean of time-series prediction in terms of the sum of squared errors. We train our model using 191 train basins and evaluate our performance in 64 test basins. Table 6.1 reports a mean ensemble NSE of 5 runs for 64 test basins during the testing period for each architecture shown in the "Architecture" column. The Years represent the years of limited available observations used for adaptation in test basins during inference.

As we would expect, Table 6.1 shows that LSTM performs very poorly as its global model was built without using static/inherent characteristics, which have previously been shown to be important in modulating the driver-response network in [150, 22]. The $LSTM_{Adapt}$ which further adapts (finetunes) the global model for each entity individually using the limited observations, performs relatively better but not well enough as it is difficult for a global model to adapt to a new set of tasks/basins efficiently with a limited observation without knowing inherent characteristics when the distribution of task/basins consists of multiple different functional modes. Next, we have $CTLSTM$, which uses given inherent characteristics for streamflow modulation of entities. Even though in this Chapter we assume the characteristics to be unavailable, we report the performance of $CTLSTM$ as a method that is the upper bound in terms of information available to it. Even without using any limited observation from test basins, we can see that it already performs better than $LSTM_{Adapt}$ showing the importance of including inherent task characteristics for modulation in a multimodal distribution setting. $CTLSTM_{Adapt}$ finetunes the $CTLSTM$ model further by using limited observation along with the given inherent characteristics to improve the global model for every test entity further individually. Next, $MAML$ and $MAML_{Adapt}$ show relatively better performance when compared to $LSTM$, and $LSTM_{Adapt}$ which uses the same amount of information, $MAML$ performs better because it can efficiently adapt in few-shot setting when compared to the $LSTM$ model. However, the performance is still relatively poor for $MAML$ because it lacks the notion of inherent characteristics, which leads to its failure as it is challenging for a single meta-model to lead to an effective adaption for all tasks, especially for multimodal distribution tasks/basins.

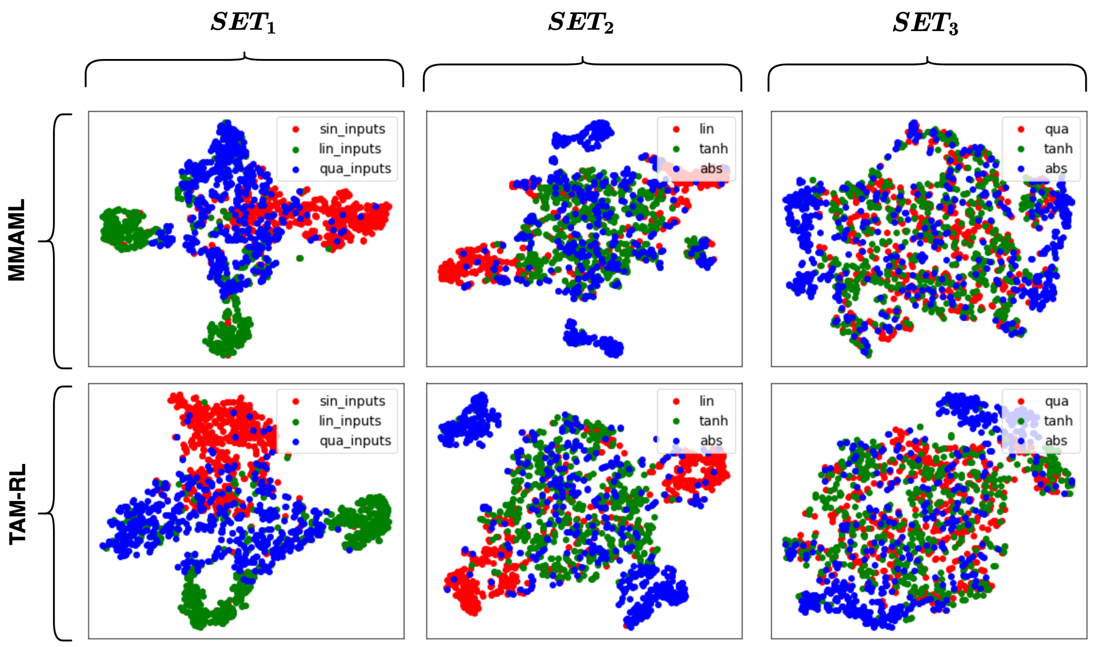


Figure 6.5: *tSNE plots of the task embeddings produced by $MMAML$ (top row) and $TAM-RL$ (bottom row) for the three sets of multimodal tasks (show in columns).*

All three models, $KGSSL-CTLSTM$, $MMAML$, and $TAM-RL$ infer inherent characteristics that they further use for modulation. $KGSSL-CTLSTM$ performs better than the other models, which do not use characteristics, but not better than

$MMAML$ and $TAM - RL$ because KGSSL learns inherent representations as an independent step without focusing on optimizing the response variable (i.e., streamflow prediction in our hydrology application) but $MMAML$ and $TAM-RL$ train their Task Encoder and Forward Model (Decoder) in a unified fashion. Amongst $MMAML$ and $TAM - RL$, even an $TAM - RL$ modulated using two years of limited observations without adaptation performs better than an $MMAML_{Adapt}$, which is modulated and adapted using five years of observation. $TAM - RL_{Adapt}$, which modulates and adapts using limited observation, further improves the performance given by $TAM - RL$. An $TAM-RL_{Adapt}$ with just two years of data performs very close to the performance shown by $CTLSTM - Adapt$ without using given inherent characteristics, and by the time we have five years of data, it already outperforms $CTLSTM - Adapt$, which uses the most amount of information and the other models. $TAM - RL$ uses the driver-response data to first infer the entity characteristics in the form of embeddings, whereas $CTLSTM$ is constrained by the known characteristics, which may be incomplete and thus do not capture the entire driver-response relationship. When the measurements of these inherent entity characteristics are not available, which happens in many naturally occurring scenarios, the next best-performing model after $TAM - RL_{Adapt}$ is $MMAML_{Adapt}$ with $TAM - RL_{Adapt}$ outperforming the later by 17% with two years of data and 19.6% with five years of data, while being 5.5 times faster in terms of training time ($TAM-RL$ takes 20 min, $MMAML$ takes 110 min for training on a v100 machine for CARAVAN-GB for each model run). As we are dealing with a multimodal real-life dataset, we can observe that all the methods without any notion of inherent task/basin characteristics perform poorly. As expected, our results improve as more limited observations are available for modulation and adaptation.

In Figure 6.4, we plot the observed streamflow (black dot) and the predicted streamflows from $TAM - RL$, $CTLSTM$, and $MMAML$ for a randomly selected test basin during the test years. Figure 6.4a shows the prediction from the models without adaptation. We can observe that the $TAM - RL$ predictions match the observed streamflow better than both $CTLSTM$ and $MMAML$. Figure 6.4b shows the prediction from the models on adapting with only two years of data. From the plot, we observe that all models' prediction improves in general. However, $TAM - RL$ gives a prediction closer to the ground truth.

6.6.4 Experiment: Synthetic Dataset with Multiple Modes (sinusoidal, tanh, linear, quadratic, $\ell 1$ norm)

In the previous section, when working with the real-life hydrology dataset for streamflow prediction in CARAVAN-GB, we saw that the $TAM-RL_{Adapt}$ significantly outperforms $MMAML_{Adapt}$. We further study this phenomenon in detail to understand in what situation $TAM-RL_{Adapt}$ outperforms $MMAML_{Adapt}$, and vice-versa. We do this by creating three sets of multimodal task distribution, each consisting of three different families of functions (modes) to mimic an artificial scenario.

Table 6.2: *Mean square error (MSE) on the multimodal 5-shot regression with different combinations of 3 modes for different architectures. Gaussian noise with $\mu = 0$ and $\sigma = 0.3$ is applied to each function.*

Architecture	***SET_1* {Sin, Linear, Quadratic}**			***SET_2* {Linear, Tanh,$\ell 1$ norm}**			***SET_3* {Quadratic, Tanh, $\ell 1$ norm}**		
	Sin	**Linear**	**Quadratic**	**Linear**	**Tanh**	**$\ell 1$ norm**	**Quadratic**	**Tanh**	**$\ell 1$ norm**
$MAML_{Adapt}$	$5.623 \pm$ 2.40e-01	$4.880 \pm$ 4.81e-01	$0.550 \pm$ 4.6e-02	$7.216 \pm$ 3.39e-01	$1.489 \pm$ 5.2e-02	1.63 $\pm 8.5e-02	$0.913 \pm$ 2.5e-02	$1.484 \pm$ 4.3e-02	$1.965 \pm$ 1.11e-01
	Set Mean MSE: $3.684 \pm$ 1.76e-01			Set Mean MSE: $3.446 \pm$ 1.15e-01			Set Mean MSE: $1.454 \pm$ 4.0e-02		
$MMAML_{Adapt}$	$0.815 \pm$ 2.08e-01	$0.440 \pm$ 6.9e-02	$0.547 \pm$ 4.5e-02	$0.664 \pm$ 7.8e-02	$0.697 \pm$ 6.9e-02	$1.895 \pm$ 3.55e-01	$0.581 \pm$ 5.4e-02	$0.977 \pm$ 1.02e-01	$0.713 \pm$ 9.3e-02
	Set Mean MSE: $0.601 \pm$ 7.4e-02			Set Mean MSE: $1.085 \pm$ 1.21e-01			Set Mean MSE: **$0.757 \pm$ 5.0e-02**		
$TAM-RL$	$0.553 \pm$ 1.00e-01	$0.391 \pm$ 4.6e-02	$0.539 \pm$ 4.5e-02	$0.873 \pm$ 9.3e-02	$0.577 \pm$ 3.2e-02	$1.696 \pm$ 1.84e-01	$0.620 \pm$ 4.8e-02	$1.386 \pm$ 1.39e-01	$0.825 \pm$ 9.9e-02
	Set Mean MSE: **$0.494 \pm$ 4.0e-02**			Set Mean MSE: **$1.049 \pm$ 6.9e-01**			Set Mean MSE: $0.944 \pm$ 6.0e-02		

The three different sets of multimodal task distribution are the following:

- SET_1: $\{Sine, Linear, Quadratic\}$
- SET_2: $\{Linear, Tanh, \ell 1 norm\}$
- SET_3: $\{Quadratic, Tanh, \ell 1 norm\}$

We create the above three sets to have relatively different levels of mode similarity in a set. For each set of multimodal task distribution, we select 375,000 tasks for each family of functions (task modes), and each task has five meta-train and five meta-validation examples for training. For batching, we sample them uniformly by selecting 25 tasks from each task mode in a batch, leading to a total of 3*25=75 tasks for each batch and a total of 15000 batches for each set. We train a $MAMl_{Adapt}$, $MMAML_{Adapt}$, and $TAM-RL_{Adapt}$ for each of the above three sets of multimodal distributions. Like in the case of [32], we use a 4-layer fully connected neural network with a hidden dimension of 100 and ReLU non-linearity for each layer as a base model for all three models $MAML$,

$MMAML$, $TAM - RL$. In $MMAML$ and $TAM - RL$, an additional model with a Bidirectional LSTM of hidden size 40 is trained to generate task embedding h, which is then further used to generate z (see Figure 6.1), which is used to modulate each layer of the base model. We use ADAM as the meta-optimizer and use the same hyperparameter settings as the regression experiments in [32]. To evaluate each set, we sample 12,500 new tasks for each family of functions (task mode) in the set, which equals 37,500 new tasks in total for the whole set. We evaluate all models with five gradient steps during adaptation using five meta-train examples for these new tasks. We report the mean squared error (MSE) of five meta-validation examples of these new tasks as the evaluation criterion.

Fig 6.5 show the tSNE plots [153] of the task embeddings produced by $TAM-RL$ and $MMAML$, respectively, from randomly sampled tasks for each set. From the tSNE plot, we observe that as we move from SET_1 to SET_3, the class of functions becomes more and more homogeneous. For example, a quadratic function can resemble a sinusoidal or linear function, but generally, a sinusoidal function is dissimilar from a linear function. SET_3 is very cluttered and homogenous because quadratic, $\ell 1$, and hyperbolic tangent functions are very similar, particularly with the inclusion of Gaussian noise in the output space. Further, we can observe from the tSNE plots that $TAM - RL$ and $MMAML$ task embeddings generate similar tSNE plots and clusters.

The quantitative results for all three sets are shown in Table 6.2. For each architecture, we report the MSE for each mode of function in a set and the mean MSE for the set. We can observe that SET_1 has a reasonably heterogeneous split, as shown in Fig 6.5. For SET_1, the performance of $TAM - RL_{Adapt}$ is 21.6% better than $MMAML_{Adapt}$. As the set becomes more homogenous, like in SET_2, the performance gap becomes smaller, and $TAM - RL_{Adapt}$ is only 3.5% better than $MMAML_{Adapt}$, and they are almost in the same tier. As the set becomes completely homogenous, like in SET_3, the performance gap inverts, and $MMAML_{Adapt}$ is now 24.7% better than $TAM-RL_{Adapt}$. From Table 6.2, we can also observe that MAML has the highest error in all settings and that incorporating task identity through task embeddings matters significantly in multimodal task distributions.

The general trend we can observe from above is that as the set becomes more and more heterogeneous, $TAM - RL_{Adapt}$ outperforms $MMAML_{Adapt}$ and vice versa. One possible explanation is that when the tasks are from a homogeneous set, the whole parameter set can quickly and accurately adapt to each task. Since the whole parameter set is adapted, MMAML shows higher performance. On the other hand, $TAM-RL$ uses the few-shot data to infer the task embeddings. These task embeddings modulate only a part of the forward model to adapt to a new unseen task. This lightweight adaptation technique does not utilize bi-level optimization training to find model parameters that can be finetuned easily. However, in the heterogenous case, adapting the whole parameter set may burden the optimization procedure extensively, possibly biasing the solution of the inner-level optimization. This is where $TAM - RC$ shows better performance when the lightweight adaptation procedure does not bias the solution.

6.7 Conclusion

We build a novel task-aware modulation framework using representation learning in this work. This framework infers the entity characteristics and leverages them for the modulation of responses given driver data. We performed extensive experiments on a hydrological benchmark dataset CARAVAN GB, showing that our framework TAM-RL outperforms baseline models for less-observed entities. We further showed that TAM-RL outperforms MMAML when multimodal task distribution is more heterogeneous. Our proposed method is general and can add value in other applications where global models are to be learned in a setting with a diverse set of entities, where only few-shot of information are available for a new entity. Currently, our framework cannot handle missing driver or response data observations. Our methodology can be further extended to handle missing observations. [22] showed that incorporating Knowledge-guided Self-Supervised Learning intro, their task encoder improved their prediction performance as now the task embeddings had semantic meaning. One potential research direction is incorporating similar knowledge guidance into our methodology.

Chapter 7

Towards Entity-Aware Conditional Variational Inference for Heterogeneous Time-Series Prediction: An application to Hydrology

7.1 Introduction

Across numerous scientific and environmental disciplines, researchers study how engineered and natural systems/entities respond to external factors [154]. In hydrology, e.g., predicting the streamflow (response) of a river basin/catchment (entity) due to external drivers (meteorological data, e.g., air temperature, precipitation, etc) is crucial to understanding hydrology cycles, water management, flood mapping, and making operational decisions. An entity's response to external drivers is influenced by its inherent properties, referred to as entity characteristics. For instance, the streamflow of two river basins can vary significantly in response to the same amount of precipitation due to differences in their land-cover types [7]. Despite increasing data availability, in many of these applications, data seldom exist at appropriate spatiotemporal resolution or coverage for scientific studies or management decisions. Developing models that are able to transfer information from highly observed sytems to sparsely observed or unmonitored systems is of interest in many environmental applications [120] has been a longstanding area of research. Traditionally, this transfer of information has relied on the regionalization of process-based models (PBMs), see [155, 11] for hydrology. The regionalization techniques require significant amounts of site-specific data collection and computation power to relate the parameter values of a PBM (that may already be calibrated to the data of a monitored system) to the inherent characteristics of the ss-observed or unmonitored system.

ML models are increasingly being considered as an alternative to PBM due to their ability to benefit from training data from diverse entities [120], enabling them to transfer knowledge between them. There are two primary methods of transferring this knowledge. The first approach involves incorporating ancillary characteristics of the entities as features (e.g., CTLSTM [21]) to account for their diversity and effectively transfer information to both less-observed (few-shot setting) and unobserved (zero-shot setting) entities. However, these characteristics can be difficult to measure accurately, leading to uncertainty or incomplete data. They may also be unknown, poorly understood, or absent in available entity characteristics. The second approach, termed inverse modeling, has been used to infer time-invariant entity characteristics from its driver-response data [22] in a deterministic fashion. A prominent example of these methods, Knowledge-Guided Self-supervised Learning (KGSSL) [22], offers a solution for performing entity-specific modulation for less-observed entities by conditioning the entity's response to external drivers on attributes inferred from the available few-shot responses, without requiring entity characteristics. They are shown to outperform the state-of-the-art forward model that uses the actual incomplete characteristics in a few-shot setting [22].

This Chapter introduces a new approach called Entity-aware Conditional Variational Inference (EA-CVI) for probabilistic inverse modeling. EA-CVI infers entity-specific attributes as a distribution over a latent space from driver-response data. Compared to deterministic models like KGSSL, EA-CVI has several advantages. First, the latent representations are built probabilistically, which aligns well with Bayesian reasoning and allows for principled approaches to tasks such as Bayesian inference and posterior estimation. Second, EA-CVI captures the inherent uncertainty associated with limited data, enabling more flexible generalization to new entities. Third, EA-CVI can generate new data samples by sampling from the learned distribution, thus rendering it useful even in a zero-shot setting. Fourth, the variational latent space of EA-CVI is parsimonious, with most of the variability captured in a few latent dimensions. Lastly, this latent space has a semantic meaning that produces a coherent effect on the predicted response, making the response generation mechanism controllable with physical interpretability.

Next, we provide a brief overview of the key features of our proposed model and discuss how these features enable the advantages mentioned above. Specifically, EA-CVI consists of an entity *Encoder* that uses the driver and response of an entity to infer the data-driven posterior distribution over a latent space, followed by a response *Decoder* to perform the inference and generative steps, see Figure 7.1b. The latent space holds information about entity characteristics, and embeddings sampled from this posterior distribution are used to predict responses. We derive the evidence-based lower bound (ELBO) [156] of the loss function used to train EA-CVI, which comprises two key components: prediction error, which penalizes deviations between predicted responses and ground truth, and KL Divergence, which regularizes the latent space.

The KL-divergence loss of the variational approach shapes the latent space representation by aligning the approximate posterior distribution (obtained from the *Encoder*) with a predefined prior (e.g., a multivariate Gaussian in our case). It offers a crucial advantage over the deterministic KGSSL approach in its inherent embrace of variability and uncertainty within this space, especially when dealing with limited data. By training on extensive driver-response data from diverse entities, EA-CVI's adaptable yet structured latent space allows the discovery of dominant modes within the entity distribution. Sampling the entity characteristics from this latent space facilitates the model to generate responses for unobserved entities in a zero-shot setting. In environmental science, the search for a deeper understanding of how various entity physio-graphic factors influence response generation mechanisms has long been a fundamental endeavor. Notably, EA-CVI introduces a novel perspective in identifying the physical attributes associated with different response variation modes, opening the door to exploring entity responses under diverse bio-geo-physical conditions. Associating physical attributes with each response mode significantly enhances the interpretability of variations in the output model.

Operational decisions (e.g., probabilistic characterization of design-relevant extremes) [157] often need to consider relatively rare but high-impact events. A principled method of managing this uncertainty during regionally unprecedented events can improve trust in data-driven decision-making from these methods. Deep learning approaches to inverse problems [22, 158] can return high-quality point estimates but usually do not provide uncertainty estimates, which are essential to aid decision-makers. Although techniques such as Bayesian Neural Networks [159, 160] focus on modeling uncertainty in the predictions by treating the parameters of the neural network as random variables. EA-CVI implicitly characterizes uncertainty within the latent space, making it well-suited for representation learning in inverse problems [161]. Our method is able to provide much better estimates of uncertainty, particularly during periods of high streamflow response, rendering it essential for real-world applications.

We evaluate our proposed framework for predicting streamflow using CAMELS-GB (Catchment Attributes and MEteorology for Large-sample Studies) [147], a widely used hydrology benchmark dataset, for understanding the Earth's interconnected ecosystems and how they are impacted by humans and changing environment. CAMELS input data are freely available on the website of UK Centre for Ecology & Hydrology, and the code is available at Google Drive[1] .

7.2 Related Works

7.2.1 Entity-Aware Modeling:

[21] train a long short-term memory (LSTM) network using data available for a large cross-section of diverse basins to improve streamflow prediction accuracy. In their use case, they assume that the basin characteristics are available. However, in many scenarios, characteristics are not known for any entity. [22] introduced a novel inverse framework that automatically extracts time-invariant characteristics from the driver-response data. More recently, [162] followed a similar approach using an encoder-decoder structure to obtain learned encodings that are analogous to hydrological signatures. Further, conditional generative networks have been used to produce full images conditioned on

[1] `https://drive.google.com/drive/folders/1iLC0-Wg4xRJHAHFMvCWiaYJqfYnyAzDB?usp=sharing`

sparse point measurements for modeling geology [163] and ocean currents [164].

7.2.2 Few-Shot Learning

Meta Learning [46] is a widely used approach when few observation samples are available for the out-of-sample entities. Meta-learning methods leverage the shared structure between different training tasks. This leads to better generalization and adaptation for new entities when only a small amount of labels are available. Model Agnostic Meta-Learning (MAML) [31] is a popular approach that learns a global meta-model, which can then be easily adapted to create personalized models for each entity using limited data. Along these lines several advancements [50, 51] of MAML have been proposed that essentially tackle the same problem. Another line of research is to encode tasks into low-dimensional latent embeddings, which will modulate the prediction function's behavior for diverse entities [63]. Specifically, it involves conditioning the prediction function on entity observations using an inferred embedding obtained through an encoder from input and output pairs for an entity. Recently, Ghosh et al. [22] introduced KGSSL, which infers time-invariant entity characteristics from its driver-response data. Similarly, Botterill et al. [162] used an encoder-decoder structure to obtain learned encodings similar to hydrological signatures. Further advancements include using bootstrapping [64] to have multiple latent embeddings or attention-based versions of NP [65]. It is important to note that the encoder in these approaches can be viewed as an inverse network, where the objective is to infer task/context characteristics from input and output pairs. Our work proposes a variational approach to such encoder based inverse models and thus can be easily incorporated in the above mentioned methods.

7.2.3 Uncertainty Quantification:

Several Bayesian deep learning methods have been at the forefront of computing posterior prediction distribution and providing uncertainty estimates. Dropout-based methods like Monte Carlo Dropout [165] are utilized during the testing period for approximate Bayesian inference when making predictions. Weight perturbation schemes [166] have also been adopted for weight-perturbation-based uncertainty quantification. Using variational inference makes learning in these Bayesian networks more feasible [167, 159, 160].

Stochastic variational inference has been used to estimate the predictive uncertainty of Bayesian LSTM models [168]. On the other hand, methods like mixture density networks [169] use likelihood cost as the objective function. Mixture density networks are useful for multi-modal data where each of the modalities can be captured using the mixing components. [170] investigate the use of mixture density networks and Monte Carlo Dropout for estimating the uncertainty in streamflow predictions. Ensemble modeling is another popular approach to quantifying prediction uncertainty. Ensemble modeling can include variational mode decomposition [171] or data assimilation techniques [172]. However, to our knowledge, a variational inference framework has not been explored for entity response prediction.

7.3 Problem Formulation

This work focuses on learning ML models for a set of entities. An entity can be a physical system such as a lake or river basin, a task, a person, or a domain/distribution. For each entity i, we have access to multiple driver/response pairs of time series sequences, as $\{(x_i^1, y_i^1), (x_i^2, y_i^2), \ldots, (x_i^{T_i}, y_i^{T_i})\}$, where superscripts indicate time step indices. The objective is to learn the mapping function from input variables x_i^t to target variables y_i^t. In conventional supervised machine learning, we train a predictive model $p_{\theta_i}(y_i^t|x_i^t)$, parameterized by θ, by finding the parameters that maximize the likelihood of the observed data:

$$\theta_i^* = \arg\max_{\theta_i} \log p_{\theta_i}(y_i^t|x_i^t) \tag{7.1}$$

Given sufficient training data for each entity, we can train individual ML models that capture these inherent biases in each entity within the learned parameters θ_i^*. However, this is not feasible as many entities lack sufficient training data. Hence, we consider learning a global model combining data from all the entities. 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 entities. These entities' behavior is often governed by their inherent characteristics z_i, i.e., the conditional distribution is of the form $p_\theta(y_i^t|z_i, x_i^t)$, where θ denotes the function class shared by the target systems and z_i denotes entity-specific inherent characteristics. In many scenarios, measurement of the entity characteristics may be entirely unavailable. Without these entity attributes, the global model cannot accurately predict each entity's response; thus, we present a variational inference method to address this challenge in this Chapter.

7.4 Preliminaries

Variational Autoencoder: The variational autoencoder (VAE) framework [156, 173] seeks to approximate a probability distribution $p(y)$ over observed variables y through a deep latent-variable model (DLVM) with latent variable z. The framework uses neural networks to optimize DLVMs using stochastic gradient-based optimization methods. VAEs use a DLVM that is factorized as $p_\theta(y, z) = p_\theta(y|z)p_\theta(z)$, where θ represents the learnable parameters of the decoder network. The VAE framework provides an efficient way to avoid evaluating the intractable exact posterior $p_\theta(z|y)$ by introducing an inference model $q_\phi(z|y)$, also known as an encoder model. The variational parameters ϕ are shared to infer posteriors for all observed data, a strategy known as amortized variational inference. The framework is trained using a cost function that is the negative of a lower bound on the log-likelihood of the data, known as the evidence lower bound (ELBO),

$$\text{ELBO} = \mathbb{E}_{q_\phi(z|y)}\left[\log p_\theta(y|z)\right] - \mathcal{D}_{KL}\left(q_\phi(z|y)||p_\theta(z)\right) \tag{7.2}$$

where the second term of the expression is the Kullback-Liebler (KL) divergence of the inferred posterior distribution $q_\phi(z|y)$ from a known prior distribution $p_\theta(z)$.

Conditional Variational Autoencoder: The conditional variational autoencoder (CVAE) is an extension of the VAE that allows data y to be conditioned on some input x [174, 175]. This framework seeks to maximize the conditional likelihood $p_\theta(y|x)$. The DLVM is factorized as $p_\theta(y, z|x) = p_\theta(y|z, x)p_\theta(z|x)$. Like VAE, CVAE is trained by maximizing an ELBO, as

$$\text{ELBO} = \mathbb{E}_{q_\phi(z|y,x)}\left[\log p_\theta(y|z, x)\right] - \mathcal{D}_{KL}\left(q_\phi(z|y, x)||p_\theta(z|x)\right) \tag{7.3}$$

In this equation, note that the second term is the KL divergence of the inferred posterior from a conditional prior $p_\theta(z|x)$. Often, the conditional prior is set to be equal to an unconditional known prior $p_\theta(z)$ [175].

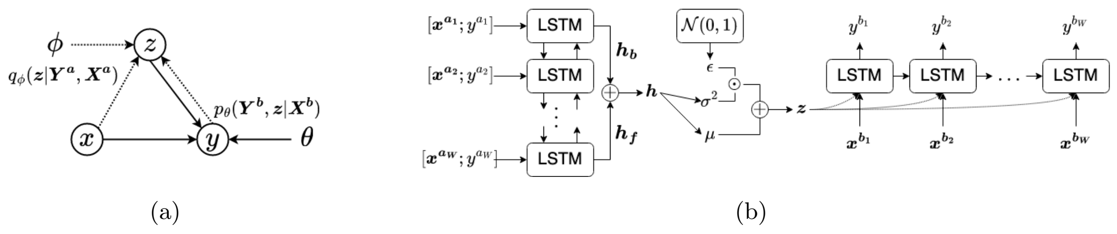


Figure 7.1: (a) *Graphical model of the proposed deep latent variable model (DLVM)*, (b) *Architectural diagram of our proposed method. We use a bidirectional-LSTM as encoder network and an LSTM as the conditional decoder.*

7.5 Methods

7.5.1 Architecture

Our proposed method infers latent entity characteristics ($\boldsymbol{z_i} \in \mathbb{R}^{D_z}$) given the time-varying driver ($\boldsymbol{X_i} = [\boldsymbol{x_i^1}, \boldsymbol{x_i^2}, \ldots, \boldsymbol{x_i^T}]$ where $\boldsymbol{x_i^t} \in \mathbb{R}^{D_x}$) and response ($\boldsymbol{Y_i} = [y_i^1, y_i^2, \ldots, y_i^T]$ where $y_i^t \in \mathbb{R}$) data and uses these latent characteristics to predict an entity's response given the drivers. We use a temporal deep latent variable model (DLVM) that comprises a sequence encoder (inference network) and a decoder (generator network), as shown in Figure. 7.1a. The inference network ($q_\phi : \mathbb{R}^{T\times(D_x+1)} \rightarrow \mathbb{R}^{D_z}$) is trained to encode entities into the latent space. The generator network ($p_\theta : \mathbb{R}^{T\times(D_x+D_z)} \rightarrow \mathbb{R}^{T\times 1}$) is trained to decode latent vectors and driver data into the response space. During training, latent vectors are encouraged to contain the minimum amount of information needed to reconstruct the entity response from latent vectors and drivers. In the following sections, we describe the choice of neural network architectures. Subsequently, we will describe the training process and the novel loss function, focusing on how they facilitate variational modeling.

Inference Network (Encoder)

Because the exact posterior inference is intractable, an inference model, $q_\phi(\boldsymbol{z}|[\boldsymbol{x^t}; y^t]_{1:T})$, that approximates the true posterior, $p_\theta(\boldsymbol{z}|[\boldsymbol{x^t}; y^t]_{1:T})$, for variational inference [176] is introduced. This can also be viewed as encoding the driver and response data interaction for an entity to learn an approximate posterior over latent variables in these sequences.

This distribution $q_\phi(\boldsymbol{z}|[x^t; y^t]_{1:T})$ is modeled using a neural network, where ϕ are the related weight parameters (Figure 7.1a). We implement this using a bidirectional RNN based sequence encoder. LSTM [151] is particularly suited for our task where long-range temporal dependencies between driver and response exist as they are designed to avoid exploding and vanishing gradient problems. The final hidden states for the forward ($\boldsymbol{h_f}$) and backward LSTM ($\boldsymbol{h_b}$) are added to get the final embeddings $\boldsymbol{h}$ as shown in Figure 7.1b. We define the posterior distribution as a function of $\boldsymbol{h}$, using multi-layer perceptrons (MLPs) to infer the parameters $(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ of a multivariate normal distribution with a diagonal covariance matrix, as

$$\begin{aligned} \boldsymbol{h} &= \text{BiLSTM}([\boldsymbol{x^t}; y^t]_{1:T}; \phi_{\boldsymbol{h}}) \\ \boldsymbol{\mu} &= \text{MLP}(h; \phi_{\boldsymbol{\mu}}) \\ \boldsymbol{\sigma}^2 &= \text{diag}(\exp(\text{MLP}(h; \phi_{\boldsymbol{\sigma}^2}))) \\ q_\phi(\boldsymbol{z}|[\boldsymbol{x^t}; y^t]_{1:T}) &= \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}, diag(\boldsymbol{\sigma}^2)) \end{aligned} \tag{7.4}$$

where the parameter set ϕ is divided into the LSTM parameters ($\phi_{\boldsymbol{h}}$) and the two MLP parameters ($\phi_{\boldsymbol{\mu}}$ and $\phi_{\boldsymbol{\sigma}^2}$). To draw a sample of $\boldsymbol{z}$, we use the reparameterization trick [156], given as $\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma}^2\boldsymbol{\epsilon}$, where $\epsilon \in \mathcal{N}(0, 1)$.

Generator Network (Decoder)

The generator network allows for the conditional generation of response data given the latent variable ($\boldsymbol{z}$) from the decoder and the driver data. The conditional generative process of the model is given in Figure 7.1a as follows: for a given sequence of driver and response data ($\boldsymbol{X^a}$ and $\boldsymbol{Y^a}$), $\boldsymbol{z}$ is drawn from the posterior distribution $q_\phi(\boldsymbol{z}|[\boldsymbol{X^a}; \boldsymbol{Y^a}])$, and the sequence of response data for another time-period is generated from the distribution $p_\theta(\boldsymbol{Y^b}|\boldsymbol{z}, \boldsymbol{X^b})$. Specifically we construct an LSTM based conditional sequence generator $y^t = LSTM(\boldsymbol{z}, [\boldsymbol{x^{1:t}}; \theta)$, where $y^t \in \boldsymbol{Y^b}$ and $\boldsymbol{x^t} \in \boldsymbol{X^b}$.

7.5.2 Learning

Consider two time periods: a period of known sequences $\boldsymbol{X^a}$ and $\boldsymbol{Y^a}$, and a period where the drivers $\boldsymbol{X^b}$ are known, but the responses $\boldsymbol{Y^b}$ are to be predicted. Since we assume that the entity attributes do not change over time, that data from each period contain similar information about the latent variables $\boldsymbol{z}$. We train the framework to extract these latent attributes by maximizing an evidence lower bound (ELBO) on the conditional log-likelihood $p_\theta(\boldsymbol{Y^b}|\boldsymbol{X^b}, \boldsymbol{Y^a}, \boldsymbol{X^a})$, given by

$$\begin{aligned}\text{ELBO} =& \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{Y^a},\boldsymbol{X^a})}\left[\log p_\theta(\boldsymbol{Y^b}|\boldsymbol{z}, \boldsymbol{X^b})\right] \\ & - \mathcal{D}_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{Y^a}, \boldsymbol{X^a})||p_\theta(\boldsymbol{z}|\boldsymbol{X^b})\right)\end{aligned} \tag{7.5}$$

Equation 7.6 results in a training approach wherein data from different periods are provided as inputs to the encoder and decoder. As suggested by [175], we let the latent variables be independent of the drivers so that the prior distribution becomes an unconditional prior, i.e., $p_\theta(\boldsymbol{z}|\boldsymbol{X^b}) = p_\theta(\boldsymbol{z})$. We choose a multivariate standard normal distribution prior, i.e. $\boldsymbol{z} \sim \mathcal{N}(0, 1)$. Note that 7.5 differs from the standard CVAE [175] objective. Rather than maximizing the log-likelihood of the observations given their corresponding drivers (as proposed by CVAE), we maximize the conditional log-likelihood of one set of observations, b, given other observations, a, and drivers for all observations.

Maximizing 7.5 increases the probability of training data under the generative model and encourages the inference model to be similar to the unknown exact posterior distribution. When the inference process is ambiguous, the inference model is incentivized to produce a wide latent distribution such that all the latent encodings are needed for the generative model to produce all possible responses. Thus, a wide range of possible responses can be produced, and uncertainty in the responses can be expressed. Note that the approximate posterior is for one period, and the true posterior is for another. If the data from each period provided the same information about the latent variables, then this term would be zero for a perfect approximate distribution $q_\phi(\boldsymbol{z}_i|\boldsymbol{Y^b}_i, \boldsymbol{X^b}_i)$. Following we justify 7.5 by showing that it still forms a valid ELBO.

Theorem 1. *With our parameterization of $\boldsymbol{z}$, 7.5 is a valid lower bound of the conditional log-likelihood $p_\theta(\boldsymbol{Y^b}|\boldsymbol{X^b}, \boldsymbol{Y^a}, \boldsymbol{X^a})$.*

Proof. We seek to maximize the conditional likelihood $p_\theta(\boldsymbol{Y^b}|\boldsymbol{X^b}, \boldsymbol{Y^a}, \boldsymbol{X^a})$ of all response sequences in the training data, conditioned on their corresponding driver sequences. We can write the conditional log-likelihood as

$$
\begin{aligned}
&\log p_\theta(\boldsymbol{Y^b}|\boldsymbol{X^b},\boldsymbol{Y^a},\boldsymbol{X^a})\\
=&\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{Y^a},\boldsymbol{X^a})}\left[\log p_\theta(\boldsymbol{Y^b}|\boldsymbol{X^b})\right]\\
=&\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{Y^a},\boldsymbol{X^a})}\left[\log\frac{p_\theta(\boldsymbol{Y^b},\boldsymbol{z}|\boldsymbol{X^b})}{p_\theta(\boldsymbol{z}|\boldsymbol{Y^b},\boldsymbol{X^b})}\right]\\
=&\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{Y^a},\boldsymbol{X^a})}\left[\log\frac{p_\theta(\boldsymbol{Y^b},\boldsymbol{z}|\boldsymbol{X^b})q_\phi(\boldsymbol{z}|\boldsymbol{Y^a},\boldsymbol{X^a})}{q_\phi(\boldsymbol{z}|\boldsymbol{Y^a},\boldsymbol{X^a})p_\theta(\boldsymbol{z}|\boldsymbol{Y^b},\boldsymbol{X^b})}\right]\\
=&\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{Y^a},\boldsymbol{X^a})}\left[\log\frac{p_\theta(\boldsymbol{Y^b},\boldsymbol{z}|\boldsymbol{X^b})}{q_\phi(\boldsymbol{z}|\boldsymbol{Y^a},\boldsymbol{X^a})}\right]\\
+&\mathcal{D}_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{Y^a},\boldsymbol{X^a})||p_\theta(\boldsymbol{z}|\boldsymbol{Y^b},\boldsymbol{X^b})\right).
\end{aligned}
\tag{7.6}
$$

The second term is the KL divergence of the approximation to the posterior distribution from the true posterior. Because the KL divergence is always non-negative, the first term of 7.6 can be the quantity to maximize during training because it is a lower bound on the conditional log-likelihood, resulting in the ELBO expression shown in 7.5.

$$
\begin{aligned}
\text{ELBO} =&\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{Y^a},\boldsymbol{X^a})}\left[\log\frac{p_\theta(\boldsymbol{Y^b},\boldsymbol{z}|\boldsymbol{X^b})}{q_\phi(\boldsymbol{z}|\boldsymbol{Y^a},\boldsymbol{X^a})}\right]\\
=&\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{Y^a},\boldsymbol{X^a})}\left[\log p_\theta(\boldsymbol{Y^b}|\boldsymbol{z},\boldsymbol{X^b})\right]\\
+&\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{Y^a},\boldsymbol{X^a})}\left[\log\frac{p_\theta(\boldsymbol{z}|\boldsymbol{X^b})}{q_\phi(\boldsymbol{z}|\boldsymbol{Y^a},\boldsymbol{X^a})}\right]\\
=&\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{Y^a},\boldsymbol{X^a})}\left[\log p_\theta(\boldsymbol{Y^b}|\boldsymbol{z},\boldsymbol{X^b})\right]\\
&-\mathcal{D}_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{Y^a},\boldsymbol{X^a})||p_\theta(\boldsymbol{z}|\boldsymbol{X^b})\right)
\end{aligned}
\tag{7.7}
$$

□

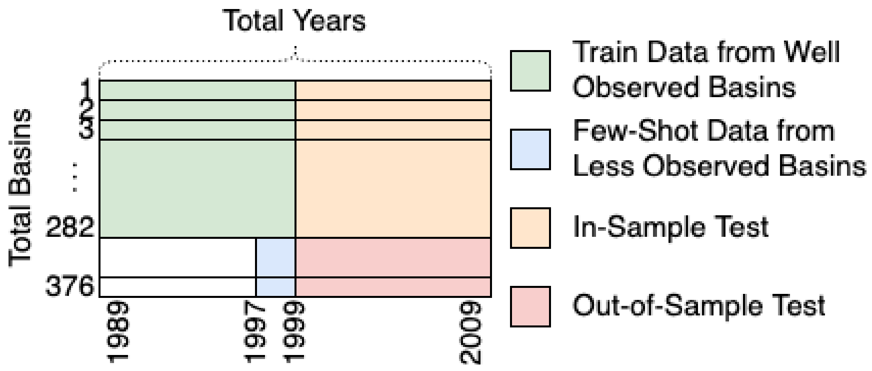


Figure 7.2: *Experimental setting followed in the Chapter for training and testing of the ML models.*

7.6 Dataset and Baselines

CAMELS-GB (Catchment Attributes and MEteorology for Large-sample Studies) [147] is part of a family of continental scale datasets that are used by the hydrology community to assess the quality of process-based and/or ML model extensively [177]. CAMELS-GB dataset provides daily meteorological forcing data (e.g., precipitation, air temperature), daily streamflow observation, and basin characteristics (we provide the complete list in the Supplementary Material) for 671 basins in the UK. Our study uses data for 376 basins (entities) from CAMELS from Oct 01, 1989, to Sep 30, 2009. Data from 1989-1999 is used for model training, and 1999-2009 is used for testing, as shown in Figure. 7.2. The basins are divided into two subsets: in-sample basins, which are used to build and train ML models, and out-of-sample basins, which are not encountered during training.

We compare the performance of EA-CVI to state-of-the-art methods in few-shot learning and inverse-modeling. **MAML$_{LSTM}$** trains a meta LSTM base model using model agnostic meta-learning (MAML) [31] approach for fast adaptation of the base LSTM model. We use the streamflow from the out-of-sample basins in a few-shot setting and five inner optimization steps to finetune the meta-model. **KGSSL** is the state-of-the-art purely deterministic inverse framework [22] for few-shot settings to infer the entity attributes in the form of embeddings and further use them to predict the streamflow. **KGSSL$_{\boldsymbol{Bayesian}}$** [178] further extends the KGSSL framework using Bayesian approach [159]. Lastly for comparison only, we also present results using **CTLSTM** [21] and **MAML$_{\boldsymbol{CTLSTM}}$**. *Note: Both have access to the actual basin characteristics, which are not used in our proposed method.*

We create input sequences of length 365 using a stride of half the sequence length, i.e., 183. All LSTMs used in the response predictor for EA-CVI (decoder) and the baselines have one hidden layer with 128 units, whereas the LSTMs used in the encoder of EA-CVI and KGSSL have a hidden layer with 32 units. The feed-forward network used to get the mean and standard deviation also has one hidden layer with 32 units. In our experiments, we perform extensive hyperparameter search with the list provided in the Supplementary Material. To reduce the randomness typically expected with network initialization, we report the result of ensemble prediction obtained by averaging predictions from five models with different weight initializations.

7.7 Experiment and Results

Table 7.1: *Mean R^2 values for streamflow modeling on CAMELS-GB for EA-CVI and the baselines in a few-shot setting. The amount of data (in years) used as few-shot are denoted as column names. CTLSTM and MAML$_{CTLSTM}$ have access to additional information about entity characteristics and thus have strictly more information.*

MODELS	Few-Shot in years			
	0.5	1	2	3
MAML$_{LSTM}$	-2.313	-0.823	-0.625	-0.288
KGSSL	-1.352	0.523	0.540	0.599
KGSSL$_{Bayesian}$	0.330	0.504	0.531	0.604
EA-CVI	**0.443**	**0.580**	**0.607**	**0.628**
CTLSTM	0.339	0.339	0.339	0.339
MAML$_{CTLSTM}$	0.451	0.532	0.554	0.578

7.7.1 Predictive performance

In Table 7.1, we evaluate the performance in terms of mean coefficient of determination (R^2) for each streamflow prediction method. Here we report the performance on the out-of-sample basins (i.e., the training and testing data are from different basins and different years) in a few-shot setting, by varying the amount of data available as few-shots. Refer to the Supplementary Material for an evaluation on the in-sample basins during test years. Testing data is exclusively from 1999-2009 (as shown in Figure. 7.2).

We observe that MAML$_{LSTM}$ has relatively lower R^2 values, indicating poorer predictive performance. This is because the whole parameters of the model are adapted for each entity using a few shots during finetuning, resulting in a suboptimal model. KGSSL performs better than MAML$_{LSTM}$ (with positive mean R^2 values across all the few-shot settings) due to efficient use of the few-shot settings. Instead of adapting the whole model parameter set, KGSSL infers the entity attributes using the few-shot samples and uses it to modulate the predictor model. KGSSL$_{Bayesian}$ performs similarly to KGSSL, showing positive mean R^2 values, and the performance tends to improve with more few-shot samples. EA-CVI outperforms the previous models, consistently exhibiting the highest mean R^2 values across all few-shot settings. This indicates that EA-CVI is more sample-efficient and effective at predicting streamflow in a few-shot setting than KGSSL. Interestingly, when few samples of observation are available for the out-of-sample entities, CTLSTM and MAML$_{CTLSTM}$ are outperformed by the inverse modeling methods (KGSSL and EA-CVI) that infer the characteristics from the driver-response data. This shows that the known characteristics present may be incomplete, and both inverse modeling methods infer the entire latent variable space as the embeddings represent known and unknown static attributes. The EA-CVI approach has added benefits of creating a semantically meaningful latent space, zero-shot prediction, and uncertainty quantification, which we discuss in the following sections.

7.7.2 Semantic Meaning of Latent Space

This section provides a semantic analysis of EA-CVI's latent embeddings. First, we demonstrate its effectiveness in encapsulating diverse entity characteristics within compact, low-dimensional representations. Second, we show how different latent components affect the streamflow generation, highlighting its ability to generate possible scenarios of entity response under different bio-geo-physical conditions. Lastly, we provide a physical interpretation of these latent dimensions with known physical characteristics of entities.

Table 7.2: *Mean R^2 values for EA-CVI and KGSSL streamflow modeling with decreasing number of latent dimensions for in-sample basins in test years*

Method	All	Top 10	Top 5	Top 2	Top 1
EA-CVI	0.7271	0.7182	0.7169	0.7010	0.6869
KGSSL	0.6957	0.2713	0.1957	0.0741	-0.4625

Efficient Latent Space

We explore the latent spaces learned by KGSSL and EA-CVI, respectively. Precisely, we measure the activity of each latent vector dimension defined by its variance over all the entities in the in-sample set. For both methods, we analyze the information in each dimension by gradually incorporating an increasing number of these latent variables ordered by their activity. Table 7.2 shows the predictive performance on test years of the in-sample entities for both the methods where we use top-k most active latent dimensions with the remaining dimensions replaced by zero values. From the results, we observe that the performance drop in EA-CVI is significantly less than KGSSL with fewer latent dimensions. This shows that EA-CVI's efficiency in encoding more information in the most active latent dimensions. The latent dimensions are thus analogous to principal component analysis (PCA), as the most active latent variables can be viewed as the dominant modes of variation in the response.

Coherent Streamflow generation

We observe independent modes of response variation under the posterior distribution by changing one latent variable at a time. In Figure 7.3, we used the top two most and least active latent variables and then tested the effect of changing each of them on the output to show that these modes can be explored using the EA-CVI inverse framework. The activity of a component in the latent vector is defined by its variance over all the entities. We change the value in the range $(-3\sigma, 3\sigma)$ with a step of 0.25σ for a selected top active latent variable. Using these values, we create a set of predictions and show them in Figure 7.3.

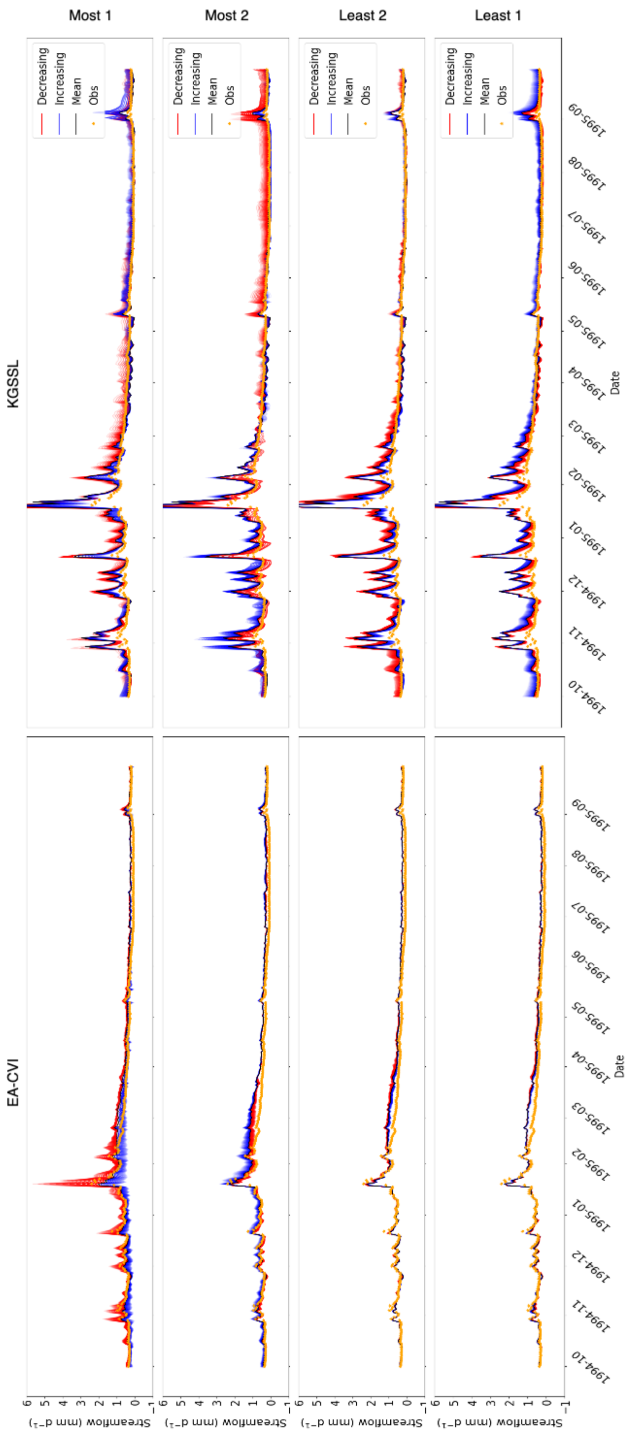


Figure 7.3: *Streamflow profiles of a basin generated by increasing (blue plots) and decreasing (red plots) the value of the top two and bottom two most active components of the latent vector for both EA-CVI and KGSSL_ CTLSTM.*

For example, the most active latent component of EA-CVI (see the top of Figure 7.3) captures high variation around peak streamflows. By contrast, the second most active latent component consistently affects the entire streamflow time series. We also observe that less active dimensions have more complex modes of variations, which supports our analogy to PCA. In contrast, we observe that for KGSSL, most and least active latent dimensions have similar effects on streamflow prediction. In addition, changing the latent variables in EA-CVI seems to have a coherent effect on streamflow, i.e., the streamflow either increases or decreases consistently throughout the time series. In contrast, we do not observe such a coherent effect in the plots of KGSSL. This further shows that the information is encoded uniformly across the dimensions of the latent vector in KGSSL and thus lacks interpretability in inferring the dominant modes of variation. Lastly, increasing the value of the most active latent variable from EA-CVI leads to a decrease in the predicted streamflow and vice-versa. An opposite effect on predicted streamflow is observed when varying the second most active latent variable. In the following section we provide a physical interpretation of this phenomenon by calculating the correlation of latent vectors and basin characteristics over all the entities.

Correlation with Entity Characteristics

Figure 7.4 shows the relative contribution of each latent dimension for explaining the rainfall-runoff process. The vertical axis represents the actual entity Characteristics, and the horizontal axis shows the latent variables ranked from most to least activity across entities. Note that the most active latent variable correlates with attributes like soil porosity (degree of porosity of soil) and crop percentage (amount of vegetation), which are known to hydrologists to reduce the streamflow for similar weather drivers. Similar conclusions can be drawn for the other latent variables, thus leading to a knowledge-guided exploration of the latent space and providing explainability to the predictions.

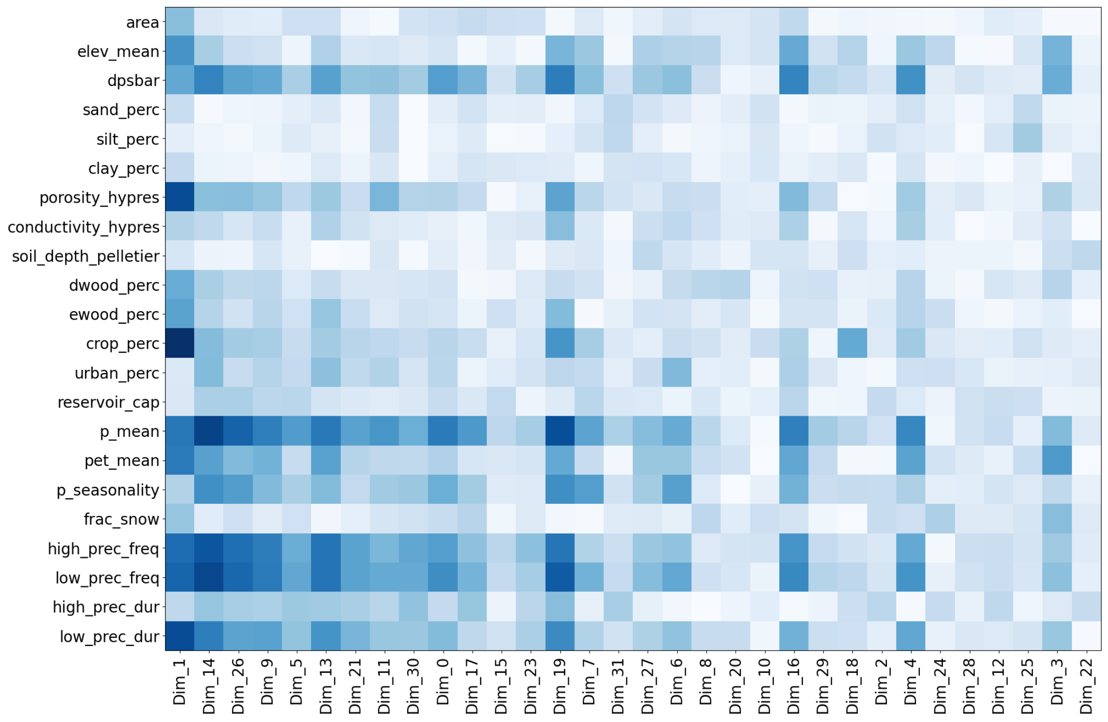


Figure 7.4: *Correlation of each dimension of the learned embeddings with each physical characteristic.*

7.7.3 Zero-Shot Streamflow generation

In many situations, building a reliable model for response generation in out-of-sample entities in zero-shot settings is necessary. The CTLSTM model cannot be used in this scenario without entity characteristics, and KGSSL cannot infer these characteristics without few-shot data. EA-CVI allows us to generate conditional streamflow based on the dominant modes of latent characteristics inferred from the entities observed during training. Specifically, we cluster the latent vectors of in-sample entities to create categories of different types of entities based on their inferred attributes. Given an out-of-sample entity, we obtain the centroid from each cluster of entities and use it in the decoder to provide conditional streamflow prediction, as shown in Figure 7.5.

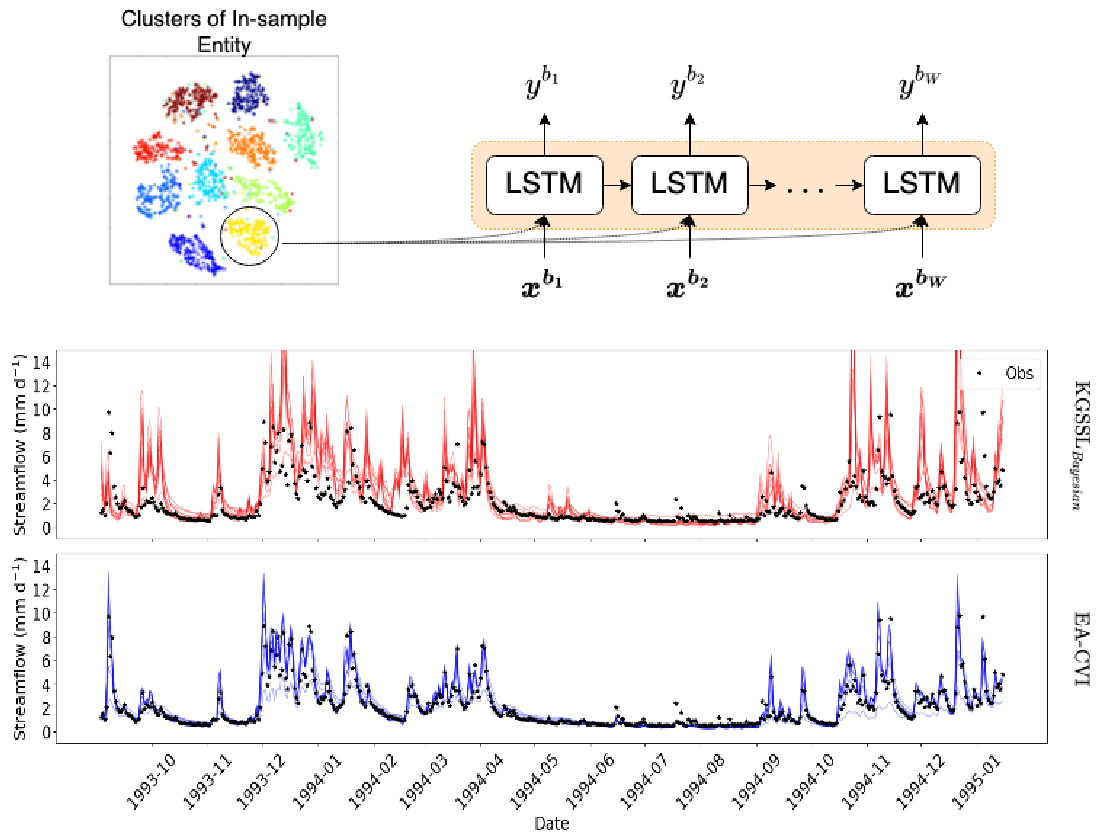


Figure 7.5: *Performance of models in Zero-shot setting. Top figure shows the architectural setup. For a randomly chosen Out-of-Sample basin KGSSL (red) predictions using cluster centroids do not match its observations. EA-CVI (blue) produces conditional predictions using cluster centroids that contain the observations.*

In the figure, many generated streamflows from EA-CVI (blue lines) overlap with the observed streamflow, showing that the cluster centroids can be used for conditional streamflow prediction as the embedding space of EA-CVI is regularized. On the other hand, none of the generated streamflow from KGSSL (red lines) lies on the observations. This is because KGSSL does not have a continuously defined latent space, and the decoder cannot use the cluster centroids as these points in the latent space have not been encountered during training.

7.7.4 Uncertainty Quantification

In this section, we aim to evaluate the uncertainty estimations of EA-CVI and compare with the Monte Carlo Dropout (MCD) [165] version of CTLSTM (CTLSTM_{MCD}) and $\text{KGSSL}_{Bayesian}$. First, we visually compare the estimated uncertainty from the two approaches. Second, we quantitatively evaluate the estimated uncertainty distributions using commonly used metrics. We generate multiple inferences by running the model 100 times.

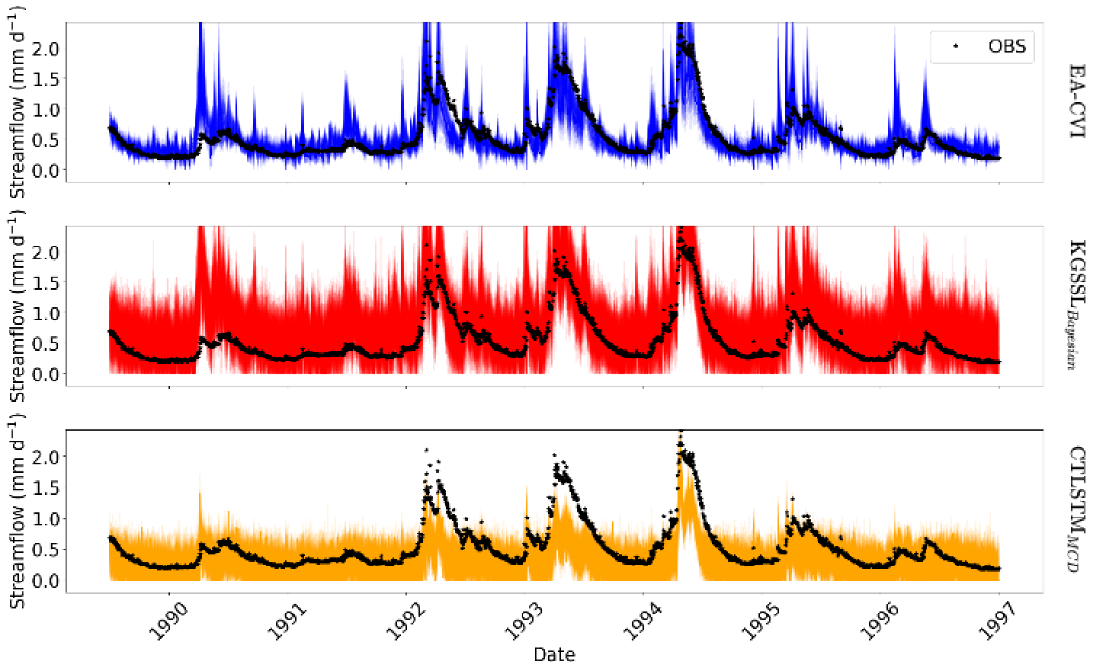


Figure 7.6: *Observed streamflow and 100 predicted streamflow realizations for EA-CVI (blue), KGSSL_ CTLSTM_{MCD} (red) and CTLSTM_{MCD} (orange) for a randomly chosen Out-of-Sample basin.*

Visualizing Predicted Uncertainty: In Figure. 7.6, we visually compare the predictions from EA-CVI, $\text{KGSSL}_{Bayesian}$ and CTLSTM_{MCD} on a randomly selected test basin during the test years. We observe that EA-CVI produces high-resolution prediction with uncertainty increasing during times with high streamflow response, whereas, the predictions predicted by $\text{KGSSL}_{Bayesian}$ and CTLSTM_{MCD} are of lower resolution with wide uncertainty bands at all times. In addition, the uncertainty bands from EA-CVI better capture the observations (especially during peak/major events) denoting that EA-CVI better estimates uncertainty while producing accurate predictions.

Table 7.3: *Resolution of the predictions from the two methods.*

Metric	CTLSTM_{MCD}	$\text{KGSSL}_{Bayesian}$	EA-CVI	ALL Basins
Mean Absolute Deviation	0.2260	0.5207	0.1366	1.3408
Standard Deviation	0.2793	0.6422	0.1721	2.0294
Variance	0.1972	0.5907	0.0784	8.3943
Quantile Distance 0.75-0.25	0.3878	0.9008	0.2262	1.6446
Quantile Distance 0.9-0.1	0.7107	1.6548	0.4302	3.5870

Estimating Predicted Uncertainty: We also evaluate the predicted distributions from two perspectives: a) measures of dispersion and b) reliability of the distributional predictions. Table 7.3 reports the measures of dispersion for the methods and the empirical distribution from the observations aggregated over all the basins as a reference ("ALL Basins"). The "ALL Basins" statistics should be used as a reference to contextualize the statistics from the modeled distributions. The table shows that EA-CVI has a higher resolution of the predicted distribution. Next, we use a probability plot [179] to evaluate how well the distributions of predictions match the true distributions of their corresponding observations. We compute the fraction of corresponding predictions that are less than the observation for each observation. Those fractions will be distributed uniformly between (0,1) if the prediction distribution matches the distribution of the observation. We evaluate whether the fractions are uniformly distributed by ranking the fractions from lowest to highest and plotting the normalized ranks against the fractions. The plotted points fall close to the 1:1 line if the fractions are distributed normally. From Figure 7.7, we can observe that the line corresponding to EA-CVI lies closer to the 1:1 line than CTLSTM_{MCD} and $\text{KGSSL}_{Bayesian}$. EA-CVI's predicted probabilities match the distribution of the observations better than the baselines. $\text{KGSSL}_{Bayesian}$ line lies below the 1:1 line, indicating a bias toward low values.

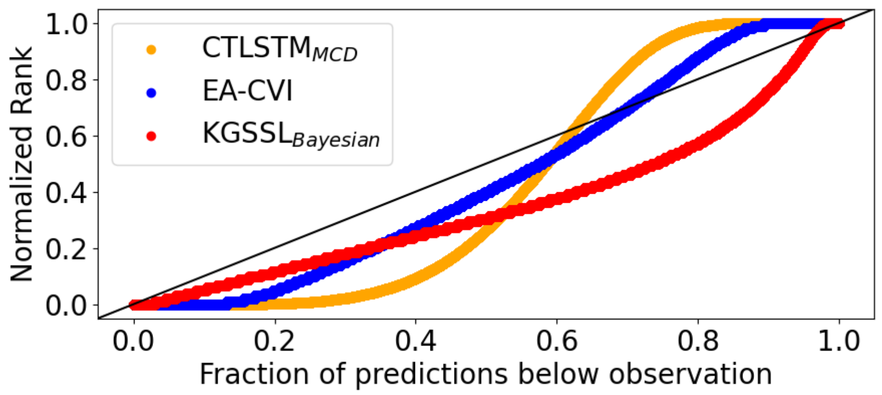


Figure 7.7: *Probability plot to show the reliability of the predictions.*

7.8 Conclusion

In this work, we presented a novel inverse model using the variational framework to infer the entity characteristics in a latent space and leverage them for the conditional prediction of responses given driver data. Extensive experiments on a hydrological benchmark dataset, showed that in a few-shot setting, EA-CVI is more sample-efficient and outperforms baseline models for less-observed entities (even models with access to the actual entity characteristics). EA-CVI's ability to identify the physical attributes associated with different response variation modes has the potential to offer deeper insights. The proposed method can add value in other applications in environmental sciences, where global models are to be learned for a diverse set of entities. Our framework can further be extended to handling missing observations in the driver or response data. In its current form, EA-CVI does not use known entity characteristics. Thus, incorporating partially known or noisy basin characteristics as prior knowledge to modulate the latent dimension is a direction of further research. Additionally, the methods presented here can be applied to other methods for task-aware modulation in machine-learning and will be considered in future work.

Chapter 8

Hierarchically Disentangled Recurrent Network for Factorizing System Dynamics

8.1 Introduction

Physical systems, whether they are natural ecosystems or engineered structures are governed by a complex interplay of internal dynamics (state) and external influences (drivers). At the heart of understanding these systems lies the concept of physical states, which encapsulate the system's internal characteristics and behaviors. These physical states dictate how the system responds to external drivers, making them a cornerstone in modeling and predicting the behavior of physical systems. For example, the impact of rainfall on runoff for a catchment is governed by several processes such as soil moisture, snow-pack and rainfall location. The ability to accurately capture and manipulate these states is paramount for a wide range of applications.

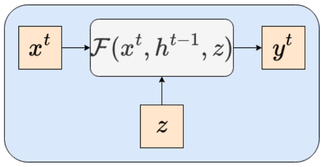


Figure 8.1: Caption

Over the years a number of physics-based models (PBMs) have been developed to model different aspects of the physical systems using physical equations. These models have been extensively used for research in understanding the system or in operational settings. For example, For streamflow modeling, the hydrological community makes extensive use of rainfall-runoff models that are built on equations which approximate the underlying biogeophysical processes of the water cycle. This model have been further used in operational setting for forecasting streamflow at NOAA's National Weather Service (NWS). However, environmental systems such as a catchment are highly complex, depending on many aspects, including snow-pack depth, soil moisture, and spatial heterogeneity in rainfall. Many of these relationships between physical characteristics and responses may not be replicated within the structure of a PBM due to incomplete understanding of the underlying physics, which impacts their ability to predict the physical quantities (fluxes) of interest. Hydrologic science has evolved considerably from the traditional linear, stationary, and static processes that historical rainfall-runoff models have utilized (Kirchner et al., 2023). Another major drawback of these models is that they require extensive effort to calibrate for any given geography of interest.

Kirchner, J.W., Benettin, P. and van Meerveld, I., 2023. Instructive Surprises in the Hydrological Functioning of Landscapes. Annual Review of Earth and Planetary Sciences, 51.

In recent years, deep learning techniques have shown tremendous success in a number of computer vision and natural language processing applications. These techniques are increasingly becoming popular in earth science applications including hydrology. Due to the temporal structure in hydrological cycle, time aware deep learning techniques such as RNNs have gained prominence as powerful tools for capturing temporal dependencies and building states. Their capacity to process sequential data and model dynamic behaviors makes them well-suited for many physical system applications that require modeling different output variables using external inputs.

Long-range dependence (LRD) is a critical concept in sequential modeling, as it describes the decay rate of past information's influence on predicting future states in various real-world time series applications, such as environmental modeling, finance, network traffic, and others. Especially in physical and environmental systems, LRD dictates modeling choices, as higher LRD implies that perturbations have a longer-lasting footprint, which requires sequential models with longer memories. While RNNs excel at constructing states, their ability to capture LRD patterns remains the subject of active research. Research has shown that introducing gates to improve stability in LSTMS has led to such gates constantly erasing information under reasonable assumptions on the input process. This heavy emphasis on recent information and inherent short-term bias poses a significant challenge when dealing with systems that exhibit diverse temporal behaviors, where capturing both short-term fluctuations and long-term trends is crucial.

The limitation of current ML models, including RNNs, becomes more evident when considering physical systems with multi-scale temporal dynamics. These systems often feature phenomena that evolve at different time scales, from rapid fluctuations to slowly changing trends. For instance, the impact of the location of rainfall on runoff is evident at hourly/daily scale. The soil-moisture state of a catchment is built up at weekly scale and thus needs a longer window of information. On the other hand snow-pack is accumulated at a monthly scale and thus takes longer time to impact runoff of a catchment (snow accumulation in December can impact the stage and discharge in April). Standard RNNs lack the ability to explicitly build states at various temporal scales, which hinders their effectiveness in capturing the full spectrum of system behavior.

To address the above challenges, we present novel framework that can create representations based on multiple scales of historical data . Specifically, we use a hierarchical recurrent neural networks (RNN) that uses a factorization technique to expand the size of the memory of RNNs. As depicted in Figure 8.3, the hierarchical RNN is composed of multi-layers, and each layer is with one or more short RNNs, by which the long input sequence is processed hierarchically. The proposed hierarchical RNN is a general architecture to build states evolving at different temporal scales and thus the framework can vary according to specific application. Based on inverse modeling, this framework can empirically resolve the system's temporal modes from data (physical model simulations, observed data, or a combination of them from the past) and use them to improve the accuracy of the forecast. By incorporating multiple levels of temporal granularity and physical interpretation of the modes, the hierarchical model gains a comprehensive understanding through a nuanced representation of the system's dynamics.

To address this critical gap, we propose a novel model in this Chapter that goes beyond the limitations of standard RNNs. Our model is designed to explicitly build states at different temporal scales, allowing it to capture both the immediate responses to external drivers and the slowly evolving, long-term trends within a physical system. By incorporating multi-scale state construction into our model, we aim to provide a more comprehensive and nuanced understanding of how physical systems evolve over time. This innovation holds the potential to significantly enhance our ability to model and predict the behavior of diverse physical systems and unlock new insights into their complex dynamics.

8.2 Problem formulation and Preliminaries

8.2.1 Problem formulation

This study focuses on learning driver-response behavior for entities. These entities can be physical systems like flux towers, river basins, tasks, people, or domains/distributions. Specifically, we focus on Understanding and predicting how precipitation transforms into streamflow at the catchment scale, which is a key part of operational hydrologic forecasting. This problem is referred to as “rainfall-runoff transform". For a basin, we have access to multiple driver/response pairs of time series sequences, as $\{(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \ldots, (\boldsymbol{x_T}, y_T)\}$, where, $\boldsymbol{x_t} \in \mathbb{R}^{D_x}$ represents the input vector at time $t \in T$ with D_x dimensions, and $y_t \in \mathbb{R}$ represents the corresponding output. Although in this study y_t is a single scalar target, in many scenarios it can be multiple target and thus is a simple extension. The goal is to learn a regression function $\mathcal{F} : X \to Y$ that maps the input drivers to the output response for an entity. The major challenge lies in ....

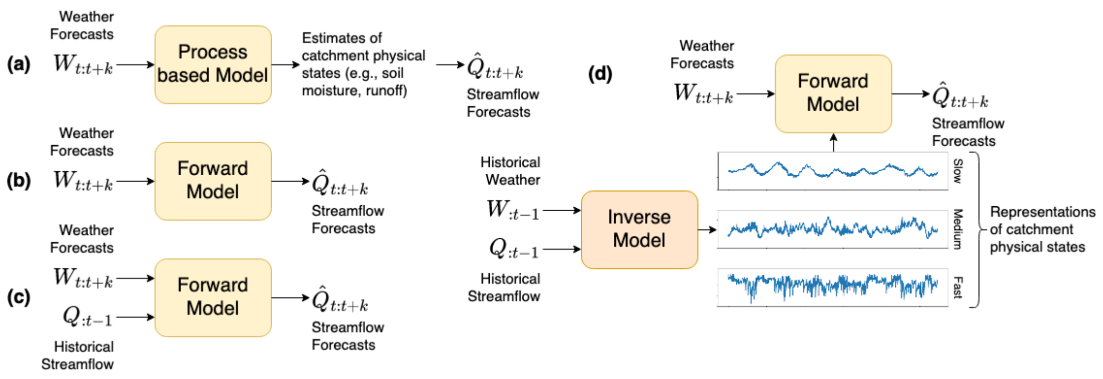


Figure 8.2: Caption

8.2.2 NWS Model and Operations

The "rainfall-runoff transform", depends on many meteorological and land-surface characteristics, such as soils, geology, vegetation, snowpack dynamics, and the pattern of rainfall inputs over the land surface. Many of the conceptual and process-based models are developed based on a set of theories (including linearization necessary for the guiding equations to become tractable). At NWS River Forecast Centers (RFCs), a suite of modeling approaches and PBMs are used, ranging from highly detailed first-principles-based models to simple conceptual models of runoff generation mechanisms. The North Central RFC, where model testing and development is ongoing, is responsible for forecasts in the Upper Mississippi River basin through St. Louis, the US side of the Hudson Bay basin, and the western Great Lakes basin. Operational river models of streamflow take catchment-scale precipitation and temperature as inputs, and run on a 6-hourly timescale. Simulated flow is transformed into stream stage, a measure of water surface elevation, according to a stage-discharge rating curve. The North Central RFC then issues stream stage (i.e., elevation) forecasts for many stream gage sites in the US Midwest when the stream water elevation crosses predefined levels categorized as Minor, Moderate, and Major flooding, which are defined by that elevation's relative impact to life, property, and the economy at those levels.

Depending on season, snowmelt and rainfall both drive streamflow at different times of year. Thus, precipitation and temperature inputs are first used in a snow model, the NWS Snow-17 model (Anderson Cite). After rainfall and snowmelt are summed together, this "total water available to the soil profile" is sent into a rainfall-runoff model, which takes in this water and returns the amount of runoff available to the stream. Once the rainfall-runoff model returns the runoff, the timing of when the runoff gets to the streamgage is accounted for by a Unit Hydrograph. Where tributaries exist, water is routed from upstream to downstream according to "Tatum" attenuation methods. This suite of models has been known as the NWS River Forecasting System, and additional documentation of modeling subcomponents can be found in Anderson, 2006, and Burnash, 1976.

The centerpiece hydrologic model within the NWS River Forecasting System used at the North Central RFC (and at most RFCs nationwide in the USA) is the Sacramento Soil Moisture Accounting Model (Sac-SMA). Sac-SMA is the model component that transforms rainfall (and/or snowmelt) into runoff. Sac-SMA is a lumped catchment-scale conceptual model with two soil zones, each with reservoirs for soil tension water and free water. Superposition of a primary and secondary lower zone allows for the simulation of more realistic baseflow, and terms of the model are allowed for interflow, effects of riparian vegetation, variable source area expansion, and losses to deep groundwater, in addition to Hortonian and saturation excess runoff (Burnash, 1976). Sac-SMA has been shown to have better performance than many rainfall-runoff models in maintaining performance between wet and dry periods (Fowler et al., 2016; John et al., 2021). There are 18 major parameters in the Snow-17 and Sac-SMA models combined, in addition to several minor parameters. Calibration is conducted manually by the methods outlined in Anderson, 2002, wherein the physical meaning of parameters and basin physical characteristics are used to constrain parameter guesses. An interative approach is taken by an experienced forecaster to vary a parameter while the effect of other parameters is expected to be minimal, assess summary statistics, then proceed. However, this process is time consuming, and inherent interactions within Sac-SMA make it difficult to truly isolate the effect of any one parameter.

A key component of operations at NWS RFCs is daily manual assimilation to align observed and simulated streamflow when they diverge. The PBM used currently are unchanged by observed streamflow: streamflow is their output, deterministically simulated from observed inputs (precipitation and temperature). Divergences between observed and simulated streamflow may be due to input uncertainty, calibration issues, or problems with the inherent physical process representation. To assimilate new streamflow data, a hydrologic forecaster must alter the underlying model states of the PBM manually, such as soil moisture, to incorporate observed streamflow data “on the fly” and to force the model to reproduce observed streamflow behavior. This manual assimilation process is time-consuming, and includes considerable uncertainty as to which among many underlying model states to change in a given operational situation. Alternatively, computationally intensive assimilation routines are necessary, which are often prohibitive for short-fuse operations over large areas.

Figure 8.2a, shows the diagramatic representation of the PBM.

8.2.3 LSTM for modeling Dynamical Systems

In the realm of machine learning, there exists a class of models dedicated to learning complex transformations from input series $\{\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_T}\}$ to target variables $\{y_1, y_2, \ldots, y_T\}$. Recent strides in deep learning have empowered the automatic extraction of meaningful patterns from multivariate temporal data, significantly enhancing predictions of the target variable. Among temporal deep learning models, Recurrent Neural Networks (RNNs) have emerged as a popular choice with successes spanning various applications. To provide some context, a standard RNN is essentially an extended version of a feed-forward neural network, enhanced with feedback connections to facilitate sequential data modeling. When modeling driver-response relationships, using an RNN equipped with Long Short-Term Memory (LSTM) cells or similar LSTM variants is quite intuitive, as shown in Figure 8.2 (b). This architecture involves a many-to-many prediction setup, allowing direct mapping of weather inputs to streamflow outputs using a single LSTM network. LSTM, or Long Short-Term Memory, was purposefully developed to combat the common problem of gradient vanishing in RNN training and has evolved into one of the most popular and effective RNN variants. A noteworthy feature of LSTM is its incorporation of an extra memory cell, pivotal for selectively retaining pertinent information from past inputs. In practical terms, it interprets an input sequence and generates a corresponding output sequence iteratively through a set of equations, as shown.

$$\begin{aligned}
\text{Forget gate :} \quad & f_t = \sigma(W_x^f x_t + W_h^f h_{t-1} + b^f) \\
\text{Input gate :} \quad & i_t = \sigma(W_x^i x_t + W_h^i h_{t-1} + b^i) \\
\text{Candidate :} \quad & \tilde{c}_t = \sigma(W_x x_t + W_h h_{t-1} + b) \\
\text{Context update :} \quad & c_t = f_t \odot \tilde{c}_t + i_t \odot c_{t-1} \\
\text{Output gate :} \quad & o_t = \sigma(W_x^o x_t + W_h^o h_{t-1} + b^o) \\
\text{Output :} \quad & h_t = o_t \odot \tanh(c_t)
\end{aligned} \tag{8.1}$$

As we wish to conduct regression for continuous values, we generate the predicted response $\hat{y}_t$ at each time step t via a linear combination of hidden units, as:

$$\hat{y}_t = W_y h_t \tag{8.2}$$

To overcome the challenge of feeding a long time series into a model, the available training data is divided into $\mathcal{T}$ sliding windows of length K during the training phase with/without overlap. The entire framework is trained using a prediction loss, like mean squared error (MSE) as the supervised loss, as shown in equation 8.3.

$$\mathcal{L} = \frac{1}{\mathcal{T}} \sum_{t=1} \sum_{t=1}^{K} (y_t - \hat{y}_t)^2 \tag{8.3}$$

8.2.4 Autoregressive LSTM

The standard lstm model cannot ingest near-real time response data and perform data assimilation. Thus a simple extension to the standard LSTM is to provide lagged response data as input to the model. Several studies [145, ?] have shown that AR improves streamflow predictions from LSTMs. A traditional way to train this sequence-to-sequence autoencoder is teacher forcing [109], 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. To avoid this the model is trained with a combination of observed and simulated lagged streamflow inputs, a solution known as scheduled sampling (Bengio et al., 2015). Another class of approaches to solving this problem are professor-forcing methods (Lamb et al., 2016), which uses adversarial learning to encourage a teacher-forcing network (i.e., trained with only observed inputs) to match the fully recursive model. This could be applied to the driver-response modeling if AR models were to exhibit divergent behavior that cannot be solved with scheduled sampling

Even though the LSTM network was proposed to overcome the difficulty in learning long-term dependence, several research have shown that LSTM do not have long memory from a statistical perspective [?], due to the presence of the gating mechanisms [?]. In the following section, we will discuss the proposed FHNN model in detail. First, we describe how to model the system dynamics at different temporal scales and use them to forecast the an LSTM to model temperature dynamics using sparse observed data. Then, we further utilize a pre-training method to improve the learning performance even with limited training data.

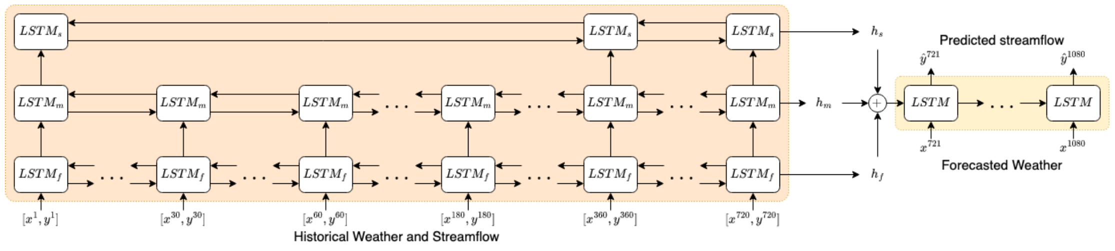


Figure 8.3: Caption

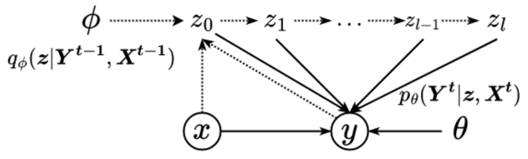

ϕ
z_0
z_1
z_{l-1}
z_l
$q_\phi(\boldsymbol{z}|\boldsymbol{Y^{t-1}}, \boldsymbol{X^{t-1}})$
$p_\theta(\boldsymbol{Y^t}|\boldsymbol{z}, \boldsymbol{X^t})$
x
y
θ

Figure 8.4: Caption

8.3 Factorised Hierarchical Neural Network

Our proposed method infers latent entity states ($\boldsymbol{h} \in \mathbb{R}^{D_h}$) given the historical time-varying driver ($\boldsymbol{X}_{hist} = [\boldsymbol{x^1}, \boldsymbol{x^2}, \ldots, \boldsymbol{x^T}]$ where $\boldsymbol{x^t} \in \mathbb{R}^{D_x}$) and response ($\boldsymbol{Y}_{hist} = [y^1, y^2, \ldots, y^T]$ where $y^t \in \mathbb{R}$) data and uses these latent states to forecast an entity's response ($\boldsymbol{Y}_{forecast} = [y^{T+1}, \ldots, y^{T+K}]$) given the forecasted drivers ($\boldsymbol{X}_{forecast} =$

$[\boldsymbol{x^{T+1}}, \ldots, \boldsymbol{x^{T+K}}])$. However, effective modelling the internal states of physical entities requires capturing dynamics with varying temporal frequency. We introduce a Factorized Hierarchical Neural Network (FHNN) that enables robust modeling of internal states by capturing and factorizing information at various temporal scales. Unlike standard RNNs, which treat all time steps equally, FHNN extracts dependencies that may be short, medium or long-term. Specifically, FHNN consists of a sequence encoder (inference network) and a decoder (generator network), as shown in Figure. 8.4. By hierarchically organizing the modeling process in the sequence encoder, FHNN can effectively capture and process information across a wide range of temporal distances, making it well-suited for tasks where understanding complex, interconnected patterns over extended sequences is paramount. The modeled internal states are then used by the generator network to forecast the response using the modeled states and future drivers. In the following sections, we will delve deeper into the architecture and mechanisms of FHNN, illustrating how it overcomes the limitations of standard RNNs and empowers us to tackle challenging problems involving long-range dependencies in sequential data.

8.3.1 State Encoder

The inference network ($q_\phi : \mathbb{R}^{T\times(D_x+1)} \rightarrow \mathbb{R}^{D_z}$) is trained to encode the current state of the entity into the latent space using a hierarchical neural network. Our formulation captures the general intuition that we can separate the state of the entity into its dynamic components (described by $z_0, z_1, \ldots, z_l$) of different time-scales. Our framework can be applied to model any number of such internal states, as shown in Figure 8.4. In this study we model the internal state of a hydrological basin using three latent variables (slow, medium and fast), i.e. z_s, z_m, and, z_f. Thus, our latent state have a factorized form:

$$\begin{aligned} q(\boldsymbol{z}|\boldsymbol{y_{1:t}}, \boldsymbol{x_{1:t}}) &= q(\boldsymbol{z_s}, \boldsymbol{z_m}, \boldsymbol{z_f}|\boldsymbol{y_{1:t}}, \boldsymbol{x_{1:t}}) \\ &= q(\boldsymbol{z_s}, \boldsymbol{z_m}|\boldsymbol{z_f}, \boldsymbol{y_{1:t}}, \boldsymbol{x_{1:t}})q(\boldsymbol{z_f}|\boldsymbol{y_{1:t}}, \boldsymbol{x_{1:t}}) \\ &= q(\boldsymbol{z_s}|\boldsymbol{z_m}, \boldsymbol{z_f})q(\boldsymbol{z_m}|\boldsymbol{z_f})q(\boldsymbol{z_f}|\boldsymbol{y_{1:t}}, \boldsymbol{x_{1:t}}) \end{aligned} \tag{8.4}$$

Specifically, in this configuration, the encoder consists of three lstm networks each network processing sequences with skip connections. The lowest LSTM models the data at the most finer resolution Δt, the middle LSTM at the medium resolution $m\Delta t$, and the top most LSTM modeling the data at coarsest resolution $s\Delta t$. We implement this using a bidirectional-LSTM based sequence encoder. LSTM [151] is particularly suited for our task where long-range temporal dependencies between driver and response exist as they are designed to avoid exploding and vanishing gradient problems. The final hidden states for the forward ($\boldsymbol{h_f}$) and backward LSTM ($\boldsymbol{h_b}$) are added to get the final embeddings $\boldsymbol{h}$ for each LSTM network, as shown in Figure 8.2. The internal state (z) of the dynamical system is obtained by first concatenating the slow, medium and fast states and feeding them into an multi-layer perceptron (MLP). Thus the encoder function are of the form:

$$\begin{aligned}
[\boldsymbol{h_t^f}]_{1:T}^{\Delta T} &= \mathrm{BiLSTM}([\boldsymbol{x_t}; y_t]_{1:T}^{\Delta T}; \phi_{\boldsymbol{h_f}}) \\
[\boldsymbol{h_t^m}]_{1:T}^{m\Delta T} &= \mathrm{BiLSTM}([\boldsymbol{h_t^f}]_{1:T}^{m\Delta T}; \phi_{\boldsymbol{h_m}}) \\
[\boldsymbol{h_t^s}]_{1:T}^{s\Delta T} &= \mathrm{BiLSTM}([\boldsymbol{h_t^m}; \boldsymbol{h_t^f}]_{1:T}^{s\Delta T}; \phi_{\boldsymbol{h_s}}) \\
\boldsymbol{z} &= \mathrm{MLP}([\boldsymbol{h_T^s}; \boldsymbol{h_T^m}; \boldsymbol{h_T^f}]; \phi_{\boldsymbol{z}})
\end{aligned} \tag{8.5}$$

where the parameter set ϕ is divided into the LSTM parameters ($\phi_{\boldsymbol{h}}$) and the MLP parameters ($\phi_{\boldsymbol{z}}$). We denote the full encoder using the symbol $\mathcal{E}(\boldsymbol{y_{1:t}}, \boldsymbol{x_{1:t}}; \phi)$. Generally, the encoder can be composed of multi-layers and each layer with several RNNs. In other words, it is a general architecture that varies according to specific tasks.

8.3.2 Response Decoder

The generator network allows for the conditional generation of response data given the latent variable ($\boldsymbol{z}$) from the decoder and the driver data. The conditional generative process of the model is given in Figure 8.4 as follows: for a historical sequence of driver and response data ($\boldsymbol{X}_{hist}$) and $\boldsymbol{Y}^{hist}$), $\boldsymbol{z}$ is obtained from the hierarchical sequence encoder $q_\phi(\boldsymbol{z}|[\boldsymbol{X}_{hist}; \boldsymbol{Y}_{hist}])$, and the sequence of response data for the forecast time-period is generated from the decoder $p_\theta(\boldsymbol{Y}_{forecast}|\boldsymbol{z}, \boldsymbol{X}_{forecast})$. Specifically we construct an LSTM based conditional sequence generator $y^t = LSTM(\boldsymbol{z}, \boldsymbol{x^{T+K:t}}; \theta)$, where the encoded internal state of the system ($\boldsymbol{z}$) is used as the initial state ($\boldsymbol{h}^0$) of the LSTM model. We denote the decoder based forward model using the symbol $\mathcal{F}(\boldsymbol{z}, \boldsymbol{x^{T+K:t}}; \theta)$.

The model is trained in an end-to-end fashion on the training windows from the training data.

8.3.3 Training a single Global Model

Environmental systems are highly heterogeneous (e.g., different entities/catchments can have widely different characteristics). Hence, PB models are often calibrated for each catchment individually. Even in this age of big data, high quality and plentiful observations are available only for a limited number of entities and other entities may have limited data or no data at all. ML offers excellent potential for dealing with the high degree of heterogeneity that is always present in environmental systems by leveraging a collection of observations from a diverse set of entities to build a powerful global meta-model. The reason is that ML models can benefit from training data from diverse entities and thus can transfer knowledge across entities. Previously, multi-basin models have been shown improved prediction capabilities [44, 21] and have reduced input data needs [22, 154].

We thus develop our integrated KGML framework, called FHNN$_{global}$ to leverage information from multiple basins jointly. Similar to [44] we assign a one-hot vector to each of the basin. The dimension of the one-hot vectors equals the number of catchments. These one-hot vectors originated from the binary vectors used to encode categorical variables in regression, where in our case, the variable is catchment ID. There is one such one-hot binary vector for each basin and these vectors are orthogonal to each other. Regardless of how basins are sorted, one-hot vector assignment assures each basin will be assigned uniquely.

8.3.4 Pretraining with Simulation Data

In real-world environmental systems, observed data is limited. For example, amongst the lakes being studied by USGS, less than 1% of the lakes have 100 or more days of temperature observations and less than 5% of the lakes have 10 or more days of temperature observations [Read et al. 2017]. Given their complexity, the RNN-based models trained with limited observed data can lead to poor performance. In addition, ML models often require an initial choice of model parameters before training. Poor initialization can cause models to anchor in local minimum, which is especially true for deep neural networks. ML models can only learn (however complex) patterns in the data used for training and thus fail on unseen data that is outside the range seen in training. On the other hand, PB models can make predictions for any arbitrary input variables (e.g., heretofore unseen weather patterns that may result from changing climate). Hence, ML algorithms will need to incorporate scientific knowledge in the ML framework to enable generalization in unseen scenarios. If physical knowledge can be used to help inform the initialization of the weights, model training can accelerated (i.e., require fewer epochs for training) and also need fewer training samples to achieve good performance.

To address these issues, we propose to pre-train the FHNN model using the simulated data produced by a generic SAC-SMA model (which has been lightly caliberated). In particular, given the input drivers, we run the generic SAC-SMA model to predict the streamflow. These simulated streanmflow data are often imperfect but they provide a synthetic realization of physical responses of a basin to a given set of meteorological drivers. Hence, pre-training a neural network using simulations from the SAC-SMA model allows the network to emulate a synthetic but physically realistic phenomena. This process results in a more accurate and physically consistent initialized status for the learning model. When applying the pre-trained model to a real system, we fine-tune the model using true observations. Here our hypothesis is that the pre-trained model is much closer to the optimal solution and thus requires less observed data to train a good quality model. In our experiments, we show that such pre-trained models can achieve high accuracy given only a few observed data points.

8.4 Experiments

8.4.1 Datasets

We evaluate our method to predict streamflow for several river catchments from the National Weather Service (NWS) North Central River Forecast Center (NCRFC) region. Input driver data is limited to catchment-scale Mean Areal Precipitation (MAP) and Mean Areal Temperature (MAT) values at a six-hour time-step, just as needed for the NCRFC operational Sacramento Soil Moisture Accounting Model and Snow-17 suite of models. We select multiple validation basins in the NCRFC area. We split the observed data into "training" and "testing" periods, and assess within the testing period, generally testing between 2012 and 2019. Table 8.1 shows the training, validation and test time frames for each of the catchment.

Table 8.1: Caption

Basin ID	Training	Validation	Testing
AGYM5	2001-2009	2010-2011	2012-2019
AMEI4	2001-2009	2010-2011	2012-2019
BCHW3	1987-2009	2010-2011	2012-2019
BEAW3	1986-2009	2010-2011	2012-2019
BIFM5	1997-2009	2010-2011	2012-2019
HWYM5	1994-2009	2010-2011	2012-2019
CVTI2	1995-2009	2010-2011	2012-2019
DARW3	1986-2009	2010-2011	2012-2019
EAGM4	2001-2009	2010-2011	2012-2019
IRNM7	1996-2009	2010-2011	2012-2019
KALI4	1990-2009	2010-2011	2012-2019
MRPM4	1989-2009	2010-2011	2012-2019
RUSI2	1986-2009	2010-2011	2012-2019
STRM4	1989-2009	2010-2011	2012-2019

8.4.2 Model Setup and Baselines

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

- NWS: The National Weather Service (NWS) model is a physics-based model that simulates the rainfall-runoff process for cathcments. NWS 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.

- LSTM: We also assessed model performance of the KGML approach versus other state-of-the-art "out of the box" AI approaches becoming common in hydrology, but that are "black box" models without the added step of estimated hierarchical time-varying catchment states. These AI alternatives include a standard Long Short Term Memory (LSTM) Machine Learning model that takes in precipitation and temperature and generates a streamflow forecast.

- LSTM-AR: We train an autoregressive LSTM Machine Learning model that takes in precipitation, temperature, and the most recent observed streamflow as inputs and directly generates a streamflow forecast. This autoregressive ML approach does not generate the catchment representations via an inverse model, but is using the exact same input information as our KGML approach. The structures of the different models attempted are shown in Figure 3.

8.4.3 Hyperparameters and Architectures

We create sliding windows of 748 time-steps for the streamflow dataset, strided by the forecast length (28 steps). Here, LSTM takes 720-length sequences (6 month data) as input and generates output at a stride of 28 steps (7 days). We create input sequences of length 365 using a stride of half the sequence length, i.e., 183. All LSTMs used in the response predictor for FHNN (decoder) and the baselines have one hidden layer with 32 units, whereas the LSTMs used in the encoder of FHNN have a hidden layer with 11 units. The feed-forward network used to get the latent state also has one hidden layer with 32 units. In our experiments, we perform extensive hyperparameter search with the list provided in the Supplementary Material. To reduce the randomness typically expected with network initialization, we report the result of ensemble prediction obtained by averaging predictions from five models with different weight initializations for all architectures (ours and baselines).

We used the Nash-Sutcliffe Model Efficiency (NSE), a widely used metric to measure the predictive skill of hydrologic models. Equation 8.6 shows the NSE equals one minus the ratio of error variance (MSE) and variance of the observations. When the error variance is larger than the variance of the observations, NSE becomes negative. As a result, NSE ranges from $-\inf$ to 1, with 1 being a perfect forecast and a negative number being poorer performance in prediction than simply offering the mean historical observation as the forecast. As a measurement metric, the higher NSE means better prediction skills for the investigated models. The NSE was calculated as an average of the NSE for all 7-day forecasts generated by the KGML model, compared to the overall NSE for the NWS model.

$$NSE(Y, \hat{Y}) = 1 - \frac{\sum_{i=1}^{N} (\hat{Y}_i - Y_i)^2}{\sum_{i=1}^{N} (Y_i - \bar{Y})^2} \tag{8.6}$$

8.5 Results

8.5.1 Overall Predictive Performance

Table 8.2: Model Performance Comparison (Nash-Sutcliffe Efficiency, NSE) of Hydrologic Models Across Multiple Stations

MODELS	**AGYM5**	**AMEI4**	**BCHW3**	**BEAW3**	**BIFM5**	**HWYM5**	**CVTI2**	**DARW3**	**EAGM4**	**IRNM7**	**KALI4**	**MRPM4**	**RUSI2**	**STRM4**
NWS Sac-SMA	0.60	0.73	0.54	0.20	0.16	0.13	0.81	0.59	0.24	0.64	0.75	0.63	0.75	0.72
LSTM	0.14	0.66	0.44	0.55	0.70	0.44	0.85	0.40	0.80	0.64	0.71	0.72	0.76	0.81
LSTM-AR	0.48	0.74	0.55	0.68	0.80	0.76	0.91	0.49	0.87	0.66	0.76	0.78	0.86	0.83
FHNN	0.61	0.80	0.58	0.75	0.84	0.83	0.92	0.57	0.89	0.69	0.80	0.83	0.87	0.86

First, we aim to evaluate how creating states at different levels helps improve the forecast accuracy. We assess the performance of each model based on their forecast accuracy across several NOAA sites. The first row shows the benchmark process-based model SAC-SMA used at NWA with an average NSE of 0.53 across the basins. Standard ML models LSTM and LSTM-AR generally outperform the PBM showing that ML has the capability of using data. Specifically, LSTM model improved over the NWS SAC-SMA model by $\sim$15% on mean NSE across test periods in 14 basins. Another major takeaway from these statistics is that LSTM-AR improved over the base LSTM model on mean NSE by $\sim$18%. FHNN consistently outperforms other models (by $\sim$6% and $\sim$45% over LSTM-AR and NWS SAC-SMA, respectively, on mean NSE), achieving the highest NSE scores across most stations. This suggests that the FHNN model is highly effective for hydrologic modeling. Moreoever, we observe that for the basins that have low forecast NSE for LSTM-AR, the increase in NSE is more from FHNN.

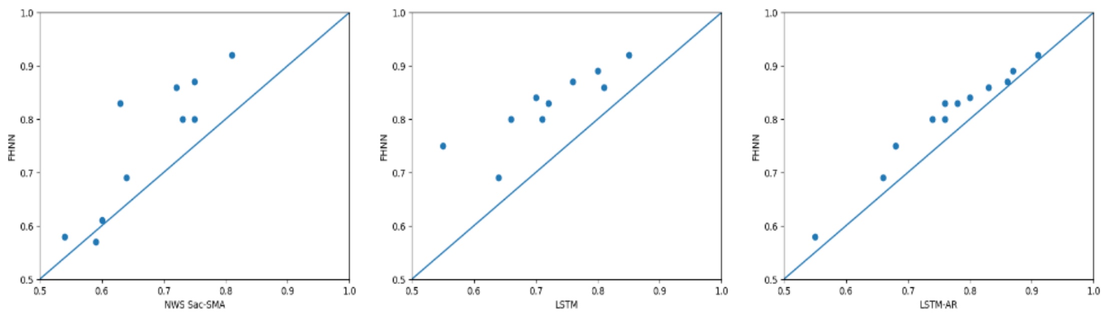


Figure 8.5: Caption

8.5.2 Pretraining with Simulation Data

Table 8.3: Simulation Data Nash–Sutcliffe model efficiency coefficient (NSE) CONTEXT: 6 Months (720 steps), FORECAST: 7 days (28 steps), STRIDE: 6 hours (1 step) FORMAT: ENSEMBLE of 5 RUNS

MODELS	AGYM5	AMEI4	BCHW3	BEAW3	BIFM5	HWYM5	CVTI2	DARW3	EAGM4	IRNM7	KALI4	MRPM4	RUSI2	STRM4
NWS Sac-SMA	0.52	0.73	0.54	0.20	0.16	0.13	0.81	0.59	0.24	0.64	0.75	0.63	0.75	0.72
Sim: No **Obs:** 2 yr	0.39	0.56	0.42	0.66	0.66	0.69	0.86	0.43	0.80	0.61	0.78	0.74	0.69	0.74
Sim: Yes **Obs:** 2 yr	**0.64**	0.76	**0.60**	**0.76**	**0.84**	**0.84**	**0.92**	**0.60**	0.84	**0.70**	**0.84**	0.78	0.85	0.85
Sim: No **Obs:** All	0.61	**0.80**	0.58	0.75	**0.84**	0.83	**0.92**	0.57	**0.89**	0.69	0.80	**0.83**	**0.87**	**0.86**

Table 8.4: NWS Sac-SMA Mean: 0.53 Median: 0.61

MODELS	**Sim: No**		**Sim: Yes**	
	mean	**Median**	**mean**	**Median**
Obs: 0.5 yr	0.18	0.30	0.68	0.75
Obs: 1 yr	0.52	0.53	0.74	0.77
Obs: 2 yr	0.65	0.68	0.77	0.81
Obs: 10 yr	0.77	0.82	0.78	0.82

Here we show the power of pre-training to improve prediction accuracy of the model even with small amounts of training data. A basic premise of pre-training our models is that NWS SAC-SMA simulations, though imperfect, provide a synthetic realization of physical responses of a cathcment to a given set of meteorological drivers. Hence, pre-training a neural network using SAC-SMA simulations allows the network to emulate a synthetic realization of physical phenomena. Our hypothesis is that such a pre-trained model requires fewer labeled samples to achieve good generalization performance, even if the SAC-SMA simulations do not match the observations. To test this hypothesis, we pre-train FHNN on the simulated data by minimizing the loss function defined in Equation **??**. We further fine-tune the pre-trained FHNN models with two years of observed data and report the performance in Table 8.3. This model is denoted using the notation "**Sim**: Yes **Obs**: 2yr" in Table 8.3. For comparison, we also report the performance of FHNN models that has been trained from scratch using only 2 years of observed data, denoted by "**Sim**: No **Obs**: 2yr". Finally we report the FHNN model that has been trained using all the available data denoted by "**Sim**: No **Obs**: All". Table 8.3 shows that pre-training can significantly improve the performance. The improvement is relatively much larger given a small amount of observed data. Even with two years of observed data, a pretrained FHNN achieves NSE similar to that obtained by the model when using all of observed data.

Thus, FHNN has shown that it can learn from a physically-based model and only a small amount of observed data, thereby acting as a model assimilator. These insights from process-based modeling can be used in regions with limited streamflow data. The service ramifications of this result are clear – often, communities have particular stream reaches of concern that cause localized flooding, and desire RFC and NWS services to produce forecasts. However, funding is also often limiting, and these communities perhaps fund a stream-gage for a year or two hoping it will be enough for NWS to begin forecast services. Calibrated process-based models, however, often need at least 5 years of data to ensure robust calibration for NWS RFC forecast services. The ability of the KGML to learn from a minimal amount of data with the help of model globalization and the use of imperfect process-based models means that potentially years will be taken off the time required to develop robust RFC forecast services. Although additional verification of model robustness in different conditions is underway, initial results show that this is a vision for the not-too-distant future.

8.5.3 Learning Global Model

Table 8.5: Comparison of all the models to their global counterparts using Nash–Sutcliffe model efficiency coefficient (NSE)

Metric	LSTM	LSTM-AR	FHNN	LSTM$_{global}$	LSTM-AR$_{global}$	FHNN$_{global}$
Mean	0.63	0.75	0.80	0.69	0.79	**0.83**
Min	0.14	0.48	0.61	0.45	0.68	**0.71**
25%	0.60	0.71	0.77	0.62	0.74	**0.79**
Median	0.70	0.76	0.83	0.72	0.78	**0.84**
75%	0.74	0.83	0.85	0.77	0.84	**0.87**
Max	0.85	0.91	**0.92**	0.87	**0.92**	**0.92**

Table 8.5 provides a comprehensive comparison of several models and their global counterparts based on their performance using the Nash–Sutcliffe model efficiency coefficient (NSE) as the evaluation metric. For this study we used a subset of 11 basins out of the available basins. The seleceted basins are AGYM5, AMEI4, BEAW3, BIFM5, HWYM5, CVTI2, EAGM4, IRNM7, KALI4, MRPM4, and, RUSI2. The models are evaluated using various statistics, including the mean, minimum, 25th percentile, median, 75th percentile, and maximum NSE values. The table provides several key observations. FHNN$_{global}$ exhibits the highest mean NSE value of 0.83 among all the models, indicating that it, on average, performs the best in accurately predicting the target variable. LSTM has the lowest minimum NSE value of 0.14 among all the models, suggesting that it has the poorest performance in some cases, resulting in very low NSE scores. LSTM-AR and FHNN have higher minimum NSE values, indicating better worst-case performance. FHNN$_{global}$ again performs the best at the 25th percentile, with a value of 0.79. It is followed closely by LSTM-AR$_{global}$ with 0.74. LSTM$_{global}$ is slightly better than its standard counterpart, LSTM. FHNN$_{global}$ has the highest median NSE value of 0.84, indicating its strong overall performance in predicting the target variable. LSTM-AR$_{global}$ also outperforms its standard version. FHNN$_{global}$ continues to lead in performance at the 75th percentile with a value of 0.87. LSTM-AR$_{global}$ and LSTM$_{global}$ also perform well in this regard. FHNN$_{global}$ achieves the highest maximum NSE value of 0.92, indicating that it can perform exceptionally well in specific cases. LSTM-AR$_{global}$ and LSTM$_{global}$ also reach high maximum NSE values. Thus, the table clearly illustrates that the global configurations of these models generally lead to improved NSE

performance compared to their standard counterparts.

8.5.4 Only Driver

Table 8.6

MODELS	HWYM5	EAGM4	KALI4	SPLI4
LSTM	0.426	0.763	0.682	0.571
FHNN-driver	0.498	0.804	0.756	0.571

The proposed FHNN model offers a unique advantage beyond its traditional data-assimilation capabilities. While originally designed to excel in data-assimilation tasks, its architecture's ability to construct states at various temporal scales makes it versatile and applicable even in non data-assimilation scenarios. By building states at multiple temporal scales, the FHNN model demonstrates an intrinsic capacity to capture intricate dependencies in sequential data. This feature extends its utility to a wide range of applications beyond data assimilation. In scenarios where understanding and modeling temporal dynamics are paramount, FHNN's flexibility in capturing information at different time resolutions can enhance predictive accuracy and temporal understanding. Table 8.6 shows the result when using the model (FHNN-driver) in a non data assimilation setting. Specifically, we feed only the sequence of drivers ($\boldsymbol{X}_{hist} = [\boldsymbol{x}^{\boldsymbol{1}}, \boldsymbol{x}^{\boldsymbol{2}}, \ldots, \boldsymbol{x}^{\boldsymbol{T}}]$ where $\boldsymbol{x}^{\boldsymbol{t}} \in \mathbb{R}^{D_x}$) into the encoder to infer the latent entity states. From the table we observe that even in this setting, modelling the inherent states at multiple scales leads to better performance compared to a standard LSTM model.

8.5.5 Visualizing States

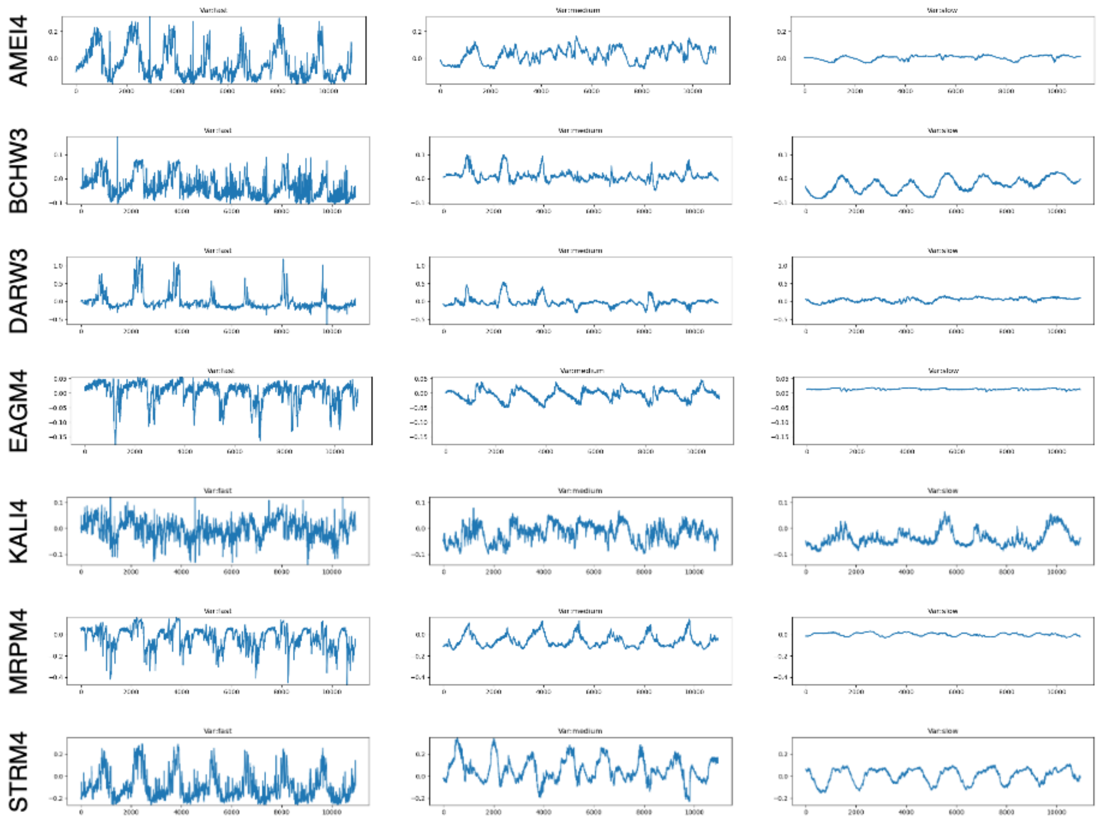


Figure 8.6: Caption

8.5.6 Operational Results

To show the promise of FHNN, we compare its performance to actual NWS-issued forecasts in a pseudo-operational context. The NWS-issued forecasts are essentially created by a human forecaster adjusting the states of the Sac-SMA model and thus can add considerable skill to the base hydrologic model. During such operational setting, the forecaster only have access to forecasted precipitation limited to values within the first 24-48 hours of the forecast initialization time. To test FHNN in the presence of the same uncertain information an NWS forecaster would have during operations, we forced the FHNN model with NCRFC-generated archived Quantitative Precipitation Forecasts (QPF) at each timestep, sourced generally from the NOAA NWS Weather Prediction

Center with minimal alterations. This is the precipitation forecast that would have been issued at the time of generating the hydrologic forecast. QPF is limited to 24 hours in the future in the warm season, and 48 hours in advance in the cold season, and is assumed to be equal to 0 outside of this window. We also used perfectly known temperature into the future. We used perfectly known temperature forecasts as usually the temperature forecasts within the first 7-days are closer than the precipitation forecasts, and due to archive issues where we were not able to access archived temperature forecasts. We should note that temperature only influences NCRFC operational models during snowmelt events. We utilized the archived USGS streamflow and stream stage to make the validation comparison, and to "feed" the FHNN model with assimilation data. Finally, the FHNN model is currently developed for streamflow, but the archived NWS forecasts are issued in stage (i.e., elevation). For this test, we used observed USGS flow and stage values during the event to reconstruct the rating curve using a LOESS regression with a span of 0.2, and we transformed the FHNN flow forecast into stage to compare directly to the NWS forecast.

Table 8.7

Event Date	**Duration**	**NWS**	**FHNN**
March 2013	5	0.71	**0.85**
April 2013	5	0.81	**0.92**
May 2013	6	0.65	**0.83**
June 2014	8	-0.07	**0.02**
September 2014	5	0.28	**0.81**
June 2015	4	0.55	**0.83**
August 2015	5	**0.31**	-0.65
December 2015	6	**0.95**	0.93
September 2018	10	**0.11**	0
October 2018	10	**0.37**	0.21
March 2019	10	0.66	**0.73**
May 2019	16	**-1.09**	-1.26

We test the FHNN model on for several major floods in the testing period on a basin in Central Iowa (NWS KALI4). This basin provides a chance to test snowmelt as well as rainfall-dominated flooding. This basin was chosen through consultation with NCRFC forecasters as a basin that can be difficult to forecast; although, the base Sac-SMA model has good performance with an NSE of 0.75. Table 8.7 shows that the overall forecast skill is higher using FHNN in 7 out of the 12 Major Floods tested. Some low-performance events are due to imperfect QPF, particularly where it was limited to QPF values within the first 24-48 hours of the forecast initialization time. This is particularly apparent in the multi-peak May 2019 event, where multiple rounds of rain would have not been captured in initial QPF estimates due to the limitation of 24 hours of "best estimate" QPF, and forcing QPF to be 0 beyond that.

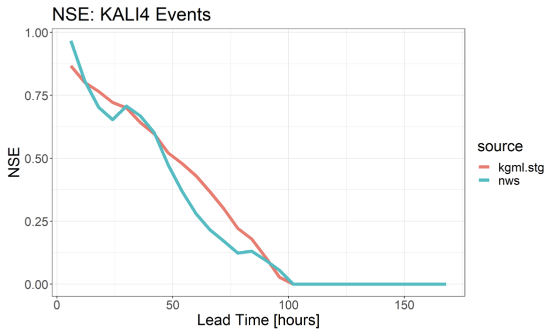


Figure 8.7: Figure 4: Major floods on KALI4 see NWS Sac-SMA based forecasts, adjusted by human forecasters, outperforming KGML in the first 12 hours; approximate parity for KGML through the end of Day 2; and KGML outperforming NWS beyond that.

Next, in Figure 8.7, we show the goodness of prediction versus lead-time by aggregating the foreasts for all events and viewing the NSE versus time. We observe that within the first 12 hours, the human NWS forecaster has more skill than the FHNN approach (NWS NSE = 0.89; FHNN NSE = 0.83). The FHNN approach then has approximate parity with NWS forecaster-issued stages through about Day 2, beyond which (even with imperfectly known QPF) the FHNN has higher skill than the NWS issued forecasts. Therefore, the NWS human forecaster is able to add significant value within the first 12 hours; beyond that, the FHNN offers higher skill because of the forecaster's reliance on the base Sac-SMA model in that time period, which does not perform as well as FHNN in general.

Chapter 9

Conclusion

9.1 Further research topics

9.1.1 Incorporating additional entity level information

Several applications exist where auxiliary information about entities can be accessed. This supplementary information can be available in primarily two forms: a) process understanding of the entities and b) additional independent observations of entity states. ML models, being data-driven, are not impacted by our limited understanding of the underlying processes. However, ML models can only learn (however complex) patterns in the data used for training and thus fail on unseen data that is outside the range seen in training. Most real-world systems consist of multiple physical processes interacting in a hierarchical order. Moreover, these processes are often highly nonlinear and exhibit complex behavior encompassing multiple inputs and outputs. There is an opportunity to advance the EAM framework further by leveraging prior physical knowledge of the hierarchical structure. The hierarchical structure provides a principled way to share parts of the entity characteristics across diverse processes through joint optimization. Apart from advances in physics-guided machine learning that utilize physical equations, boundary conditions, and other inductive biases, entity-specific physical descriptors and physical processes can also be incorporated in modeling framework to enable generalization in unseen scenarios [180]. Another opportunity unique to many environmental problems is the availability of ancillary information about the system beyond the standard input and output variables. For example, streamflow in a river catchment is modeled as a function of weather drivers, but auxiliary information such as soil moisture data from in-situ sensors or earth observing satellites [181] can provide valuable information related to underlying processes such as evapotranspiration and base flow. New EAM methods are required that can readily incorporate such diverse sources of data and has the potential to represent complex physical relationships between multiple bio-geo-physical processes.

9.1.2 Identifiability of Characteristics/Equifinality

When characteristics are unknown in EAM, a central problem is how to correctly identify those factors. Although methods like NP, SSM, and disentangled/causal representation learning show potential to learn entity-related representations, there is no guarantee the learned latent representation corresponds to the real characteristics (latent causal factors) [71]. The intuition is that given observational variables, there could be infinitely many generative models yielding the same observations, and those algorithms cannot discriminate the true causal model from other equivalent generative models. Recent progress have shown that it's impossible to to recover latent causal variables without inductive biases both on models and data sets [71, 182]. This problem is known as identifiability of causal models. Existing works established identifiability results based on the independent component analysis (ICA) [183]. The identifiability and uniqueness of linear ICA models have be well studied [184]. For nonlinear ICA model, researchers argue that the latent causal variables are unidentifiable without temporal structure [185]. Recent advances focus on extending the identifiability of linear ICA to non-linear ICA, using the nonstationary structure of time series or auxiliary variables [185]. However, this line of work doesn't assume the causal relationship or generative process between latent variables and observed variables, which limits its use. How to correctly identify latent causal variables and structure is still a open problem. Current attempt makes strong condition on measurement model, noise type, or require nonstationary time series [186, 187, 188], and those methods are only tested on synthetic dataset or simple scenarios. Thus, there is an opportunity to identify latent causal variables in complex system, especially in the scenario where people have good domain knowledge, which is more informative than auxiliary variables.

9.1.3 Uncertainty Quantification

Uncertainty estimation in EAM enables the quantification of uncertainty stemming from the model structure and input/output data and improves our understanding of different scientific processes and inherent entity characteristics. Uncertainty estimates can be used to establish the usability of an entity-aware model for operational decision-making in real worl applications [189, 190]. Finally, uncertainty quantification (UQ) methods also allow domain scientists to encode prior knowledge as model structure [191] for robust generalization. Uncertainty can be introduced in EAM due to several sources. EAM methods may be simplification or approximation of the real-world physical systems leading to a model structure-based uncertainty. Second, imperfections, measurement errors, interpolation, or noise in entity characteristics can also lead to uncertainty in the known characteristics. Finally, more recently, there has also been a focus on estimating distributional uncertainty that arises because of differences in the data distribution between training and test set.

Existing UQ methods include Bayesian methods that compute posterior prediction distribution and provide uncertainty estimates. Dropout-based methods like Monte Carlo Dropout [165] are utilized during the testing period for approximate Bayesian inference when making predictions. Weight perturbation schemes [166] have been adopted for weight-perturbation-based uncertainty quantification. Using variational inference makes learning in these Bayesian networks more feasible [159]. Other approaches, such as Mixture density networks [169], have been used for multi-modal data where each of the modalities can be captured using the mixing components. More comparisons of uncertainty estimation methods can be found in [192]. Recent studies have also attempted to decompose different sources of uncertainties [193]. Principles of evidential theory have further been used to learn other sources of uncertainty [194].

Several of these UQ methods can be used to improve EAM methods discussed in Section **??**. First, several variational Gaussian processes methods [63] use inducing points to estimate posterior function from few-shot data. Thus the uncertainty due to the use of different approximation mechanisms and different subsampled datasets can be estimated and used to study the difference in generalization capabilities of these methods. Second, the decomposition of uncertainty estimates can be pivotal in decision-making - understanding if the current EAM can help adapt the model to specific use cases or determine if we need to build better models and use different datasets for our analysis. Third, most UQ methods develop Bayesian frameworks that use Gaussian distribution as function priors. A direction that would be useful for practical applications is looking at other prior distributions for model parameter sampling. For instance, where a target or outcome variable (e.g., extreme temperature modeling) can take extreme values, approximating the prediction function using a Gumbel or t-distribution prior can enable more accuracy. Finally, existing EAM methods consider that all entities are independent. However, in many scenarios, the entity can also be a mixture of base entities, such as a community of people or a category of micro-organisms. While several multi-modal EAM methods exist, formulating the prediction function as a mixture of components also allows for multi-modal modeling.

9.1.4 Fairness

In EAM, the imbalance in training data collected from multiple entities can naturally introduce bias for some entities or groups. Such entity-related bias can adversely affect both individual's opportunities and the inequity over the whole population. Another source of unfairness could be bias in measurement error of input features. For example, phenomena of datasets having higher error profile in emerging economies occurs in several applications [195, 196]. Fairness over multiple entities can be commonly formulated in three different ways. First, individual fairness follows the philosophy that similar entities should yield similar predictions with respect to a particular task, regardless of sensitive attributes (e.g., gender, income, and race). The second type of fairness (e.g., equal opportunity [197] and statistical parity [198]) aims to ensure that the model output distribution is fair across entities. Third, fairness can also be measured in terms of performance disparity across different entities, especially to identify biased predictions for entities in disadvantaged groups or low-resource environments. For these settings, fairness can also be defined over groups of entities formed by certain attributes. For example, fair flow prediction amongst river-streams groups that are grouped according to the local information of annual income and business type can reduce the chance of flood risks being underestimated for low-income areas.

Amongst existing fairness-enforcing methods, the most common strategy is to include additional fairness-related losses during the training process [199]. Another major direction is to learn group-invariant features [200], in which discriminators are introduced to penalize learned features with the discriminative information of certain sensitive attributes (e.g., gender). Sensitive category de-correlation also employs the adversarial learning regime, but it tries to mitigate the polarization of predictions [201, 200]. To alleviate the competition between the predictive accuracy and fairness, a bi-level model refinement is proposed to disentangle model prediction and fairness objective [202]. Another benefit of this method is that it allows non-differentiable fairness measures. On the other hand, new data collection and filtering methods are developed to reduce bias in downstream learning tasks [203]. These methods have been applied to tasks related to face detection [199], text analysis [201], land cover mapping [202], etc.

Existing fairness-enforcing methods in EAM face several challenges. First, although many definitions of fairness have been proposed in existing literature, fairness needs to be carefully formulated depending on the nature of the target problems. Second, fairness metrics are fragile or sensitive to the grouping of entities, i.e., conclusions on “fair" or “unfair" can be easily altered by simple changes grouping of entities. Third, in real-world EAM problems, the deployment environments may differ from the training environment. As a result, a fairness-enforced model learned from training samples may fail to preserve fairness in target testing scenarios.

9.2 Conclusion

In this Chapter, we proposed a structured review of entity-aware modelling (EAM) research. As shown by this paper, many different research efforts have the potential to advance EAM. We organized the existing research based on the availability of entity characteristics and training samples. We hope that this structure will help in providing an organized view of this rapidly evolving field of research. This Chapter will also be valuable for domain scientists interested in exploring the use of ML to enhance EAM in their respective applications. Furthermore, we presented additional research directions that will improve the performance and usability of EAM in operational decision-making.

References

[1] Babak Mohammadi, Roozbeh Moazenzadeh, Kevin Christian, and Zheng Duan. Improving streamflow simulation by combining hydrological process-driven and artificial intelligence-based models. *Environmental Science and Pollution Research*, 28(46):65752–65768, December 2021.

[2] Brent D Newman et al. Ecohydrology of water-limited environments: A scientific vision. *WRR*, 42(6), 2006.

[3] Ahmed M Alaa et al. Attentive state-space modeling of disease progression. *NeurIPS*, 32, 2019.

[4] Sahraoui Dhelim et al. A survey on personality-aware recommendation systems. *Artificial Intelligence Review*, 2022.

[5] Huaxiu Yao et al. Learning from multiple cities: A meta-learning approach for spatial-temporal prediction. *The World Wide Web Conference*, pages 2181–2191, 2019.

[6] Gopal Bhatt et al. A tightly coupled gis and distributed hydrologic modeling framework. *Environmental modelling & software*, 62:70–84, 2014.

[7] Andrew J. Newman et al. Gridded ensemble precipitation and temperature estimates for the contiguous united states. *Journal of Hydrometeorology*, 16(6):2481–2500, 2015.

[8] Sheri R Colberg et al. Physical activity/exercise and diabetes: a position statement of the american diabetes association. *Diabetes care*, 39(11):2065–2079, 2016.

[9] Babak Alipanahi et al. Predicting the sequence specificities of dna-and rna-binding proteins by deep learning. *Nature biotechnology*, 33(8):831–838, 2015.

[10] Pierre Baldi et al. Jet substructure classification in high-energy physics with deep neural networks. *Physical Review D*, 93(9):094034, 2016.

[11] Xiaowei Jia et al. Physics-guided recurrent graph model for predicting flow and temperature in river networks. *SDM*, 2(3):1–26, 2021.

[12] Steven K Kauwe et al. Machine learning prediction of heat capacity for solid inorganics. *Integrating Materials and Manufacturing Innovation*, 7(2):43–51, 2018.

[13] Peer Nowack et al. Using machine learning to build temperature-based ozone parameterizations for climate sensitivity simulations. *Environmental Research Letters*, 13(10):104016, 2018.

[14] Christian Tesche et al. Coronary ct angiography–derived fractional flow reserve: machine learning algorithm versus computational fluid dynamics modeling. *Radiology*, 288(1):64–72, 2018.

[15] Susan L Neitsch, Jeffrey G Arnold, Jim R Kiniry, and Jimmy R Williams. Soil and water assessment tool theoretical documentation version 2009. Technical report, Texas Water Resources Institute, 2011.

[16] 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.

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

[18] 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.

[19] Yuning Shi, Kenneth J Davis, Fuqing Zhang, Christopher J Duffy, and Xuan Yu. Parameter estimation of a physically based land surface hydrologic model using the ensemble kalman filter: A synthetic experiment. *Water Resources Research*, 50(1):706–724, 2014.

[20] M Khaki, H-J Hendricks Franssen, and SC Han. Multi-mission satellite remote sensing data for improving land hydrological models via data assimilation. *Scientific reports*, 10(1):1–23, 2020.

[21] Frederik Kratzert et al. Towards learning universal, regional, and local hydrological behaviors via machine learning applied to large-sample datasets. *HESS*, 23(12):5089–5110, 2019.

[22] Rahul Ghosh et al. Robust inverse framework using knowledge-guided self-supervised learning: An application to hydrology. *KDD*, 2022.

[23] James H Faghmous and Vipin Kumar. A big data guide to understanding climate change: The case for theory-guided data science. *Big data*, 2(3):155–163, 2014.

[24] A Karpatne et al. Theory-guided data science: A new paradigm for scientific discovery from data. *IEEE TKDE*, 29(10):2318–2331, 2017.

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

[26] Yanchao Tan et al. Metacare++: Meta-learning with hierarchical subtyping for cold-start diagnosis prediction in healthcare data. *SIGIR*, page 449–459, 2022.

[27] Zeshan M Hussain et al. Neural pharmacodynamic state space modeling. *ICML*, pages 4500–4510, 2021.

[28] Young-Min Kim et al. Predictive modeling for machining power based on multi-source transfer learning in metal cutting. *IJPEM - Green Tech*, 9(1):107–125, 2022.

[29] Yan Liu et al. Metastore: a task-adaptative meta-learning model for optimal store placement with multi-city knowledge transfer. *ACM TIST*, 12(3):1–23, 2021.

[30] Yash-yee Logan et al. Patient aware active learning for fine-grained oct classification. *ICIP*, pages 3908–3912, 2022.

[31] Chelsea Finn et al. Model-agnostic meta-learning for fast adaptation of deep networks. *ICML*, pages 1126–1135, 2017.

[32] Risto Vuorio et al. Multimodal model-agnostic meta-learning via task-aware modulation. *NeurIPS*, 32, 2019.

[33] Qiuling Suo et al. Tadanet: Task-adaptive network for graph-enriched meta-learning. *KDD*, pages 1789–1799, 2020.

[34] Xixun Lin et al. Task-adaptive neural process for user cold-start recommendation. *WWW*, pages 1306–1316, 2021.

[35] Sara Taylor et al. Personalized multitask learning for predicting tomorrow's mood, stress, and health. *IEEE Transactions on Affective Computing*, 2017.

[36] Tong Xia et al. Deepapp: Predicting personalized smartphone app usage via context-aware multi-task learning. *ACM TIST*, 11(6):1–12, 2020.

[37] Hu Wang et al. Human mobility prediction using sparse trajectory data. *IEEE Trans. on Vehicular Technology*, 69(9):10155–10166, 2020.

[38] Arun Ravindranath, Naresh Devineni, and Peter Kolesar. An environmental perspective on the water management policies of the upper delaware river basin. *Water Policy*, 18(6):1399–1419, 2016.

[39] Fred D Theurer, Kenneth A Voos, and William J Miller. Instream water temperature model. instream flow information paper 16. Technical report, US Fish and Wildlife Service, 1984.

[40] Ethan Perez et al. Film: Visual reasoning with a general conditioning layer. *AAAI*, 32, 2018.

[41] Ruoxi Wang et al. Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. *The Web Conference*, pages 1785–1797, 2021.

[42] Haoxiang Wang et al. Bridging multi-task learning and meta-learning: Towards efficient training and effective adaptation. *ICML*, pages 10991–11002, 2021.

[43] Luchen Liu et al. Multi-task learning via adaptation to similar tasks for mortality prediction of diverse rare diseases. *AMIA Annual Symposium*, 2020:763, 2020.

[44] Xiang Li et al. Regionalization in a global hydrologic deep learning model: from physical descriptors to random vectors. *WRR*, 58(8):e2021WR031794, 2022.

[45] Yaqing Wang et al. Generalizing from a few examples: A survey on few-shot learning. *ACM CSUR*, 53(3):1–34, 2020.

[46] Timothy Hospedales et al. Meta-learning in neural networks: A survey. *TPAMI*, 44(9):5149–5169, 2021.

[47] Eunjung Lee et al. Individualized short-term electric load forecasting with deep neural network based transfer learning and meta learning. *IEEE*, 9:15413–15425, 2021.

[48] Samuel A Ajila et al. Multilayer meta-learning approach to forecasting air pollutants. *IEEE IRI*, pages 69–74, 2022.

[49] Xi Sheryl Zhang et al. Metapred: Meta-learning for clinical risk prediction with limited patient electronic health records. *KDD*, pages 2487–2495, 2019.

[50] Aniruddh Raghu et al. Rapid learning or feature reuse? towards understanding the effectiveness of maml. *arXiv*, 2019.

[51] Liang Zhao et al. Learning to recommend via meta parameter partition. *arXiv*, 2019.

[52] Shuaiqiang Liu et al. On calibration neural networks for extracting implied information from american options. *arXiv*, 2020.

[53] Luisa Zintgraf et al. Fast context adaptation via meta-learning. *ICML*, pages 7693–7702, 2019.

[54] Kshitij Tayal et al. Invertibility aware integration of static and time-series data: An application to lake temperature modeling. *SDM*, pages 702–710, 2022.

[55] Junyoung Park et al. Meta-sysid: A meta-learning approach for simultaneous identification and prediction. *arXiv*, 2022.

[56] Antreas Antoniou et al. How to train your maml. *arXiv*, 2018.

[57] Jiayi Chen et al. Hetmaml: Task-heterogeneous model-agnostic meta-learning for few-shot learning across modalities. *CIKM*, pages 191–200, 2021.

[58] Giulia Denevi et al. Conditional meta-learning of linear representations. *arXiv*, 2021.

[59] Ghassen Jerfel et al. Reconciling meta-learning and continual learning with online mixtures of tasks. *NeurIPS*, 32, 2019.

[60] Shengyu Chen et al. Physics-guided graph meta learning for predicting water temperature and streamflow in stream networks. *KDD*, pages 2752–2761, 2022.

[61] Saurav Jha et al. The neural process family: Survey, applications and perspectives. *arXiv*, 2022.

[62] Marta Garnelo et al. Conditional neural processes. *ICML*, pages 1704–1713, 2018.

[63] Marta Garnelo et al. Neural processes. *arXiv*, abs/1807.01622, 2018.

[64] Juho Lee et al. Bootstrapping neural processes. *NeurIPS*, 33:6606–6615, 2020.

[65] Hyunjik Kim et al. Attentive neural processes. *arXiv*, 2019.

[66] Ronald James Cotton et al. Factorized neural processes for neural processes: K-shot prediction of neural responses. *NeurIPS*, 33:11368–11379, 2020.

[67] Ari Pakman et al. Neural clustering processes. *ICML*, pages 7455–7465, 2020.

[68] Dongxia Wu et al. Multi-fidelity hierarchical neural processes. *arXiv*, 2022.

[69] James Requeima et al. Fast and flexible multi-task classification using conditional neural adaptive processes. *NeurIPS*, 32, 2019.

[70] Jaesik Yoon et al. Robustifying sequential neural processes. *ICML*, pages 10861–10870, 2020.

[71] Francesco Locatello et al. Challenging common assumptions in the unsupervised learning of disentangled representations. *ICML*, pages 4114–4124, 2019.

[72] Xi Chen et al. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *NeurIPS*, 29, 2016.

[73] Irina Higgins et al. beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR*, 2017.

[74] Zhiwei Deng et al. Factorized variational autoencoders for modeling audience reactions to movies. *CVPR*, pages 2577–2586, 2017.

[75] Diane Bouchacourt et al. Multi-level variational autoencoder: Learning disentangled representations from grouped observations. *AAAI*, 32, 2018.

[76] Chanho Eom and Bumsub Ham. Learning disentangled representation for robust person re-identification. *NeurIPS*, 32, 2019.

[77] Li Yingzhen et al. Disentangled sequential autoencoder. *ICML*, pages 5670–5679, 2018.

[78] James Durbin and Siem Jan Koopman. *Time series analysis by state space methods*, volume 38. OUP Oxford, 2012.

[79] Rahul Krishnan et al. Structured inference networks for nonlinear state space models. *AAAI*, 31, 2017.

[80] Binh Tang and David S Matteson. Probabilistic transformer for time series analysis. *NeurIPS*, 34:23592–23608, 2021.

[81] Marco Fraccaro et al. A disentangled recognition and nonlinear dynamics model for unsupervised learning. *NeurIPS*, 30, 2017.

[82] Yuan Xue et al. Deep state-space generative model for correlated time-to-event predictions. *KDD*, pages 1552–1562, 2020.

[83] Bernhard Schölkopf et al. Toward causal representation learning. *IEEE*, 109(5):612–634, 2021.

[84] Mengyue Yang et al. Causalvae: Disentangled representation learning via neural structural causal models. *CVPR*, pages 9593–9602, 2021.

[85] Abbavaram Gowtham Reddy et al. On causally disentangled representations. *AAAI*, 36:8089–8097, 2022.

[86] Xinwei Shen et al. Weakly supervised disentangled generative causal representation learning. *JMLR*, 23:1–55, 2022.

[87] Peter Bühlmann et al. Invariance, causality and robustness. *Statistical Science*, 35(3):404–426, 2020.

[88] Chaochao Lu et al. Invariant causal representation learning for out-of-distribution generalization. *ICLR*, 2021.

[89] Tian-Zuo Wang et al. Sound and complete causal identification with latent variables given local background knowledge. *NeurIPS*, 2022.

[90] Jie Bu and Anuj Karpatne. Quadratic residual networks: A new class of neural networks for solving forward and inverse problems in physics involving pdes. SIAM SDM (21), 2021.

[91] Steffen Rendle et al. Neural collaborative filtering vs. matrix factorization revisited. *Fourteenth ACM Conference on Recommender Systems*, 2020.

[92] Jianxun Lian et al. Xdeepfm: Combining explicit and implicit feature interactions for recommender systems. KDD '18, 2018.

[93] Nans Addor et al. The CAMELS data set: Catchment attributes and meteorology for large-sample studies. *HESS*, 21(10):5293–5313, 2017.

[94] Markus Hrachowitz, HHG Savenije, G Blöschl, JJ McDonnell, M Sivapalan, JW Pomeroy, Berit Arheimer, Theresa Blume, MP Clark, U Ehret, et al. A decade of predictions in ungauged basins (pub)—a review. *Hydrological sciences journal*, 58(6):1198–1255, 2013.

[95] Martin Erlandsson, Ishi Buffam, Jens Fölster, Hjalmar Laudon, Johan Temnerud, Gesa A Weyhenmeyer, and Kevin Bishop. Thirty-five years of synchrony in the

organic matter concentrations of swedish rivers explained by variation in flow and sulphate. *Global Change Biology*, 14(5):1191–1198, 2008.

[96] Sinno Jialin Pan et al. A survey on transfer learning. *IEEE TKDE*, 22(10):1345–1359, 2009.

[97] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pages 97–105. PMLR, 2015.

[98] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128, 2006.

[99] Jared D Willard et al. Predicting water temperature dynamics of unmonitored lakes with meta-transfer learning. *WRR*, 57(7):e2021WR029579, 2021.

[100] Wei Ying, Yu Zhang, Junzhou Huang, and Qiang Yang. Transfer learning via learning to transfer. In *International Conference on Machine Learning*, pages 5085–5094. PMLR, 2018.

[101] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

[102] Antreas Antoniou and Amos J Storkey. Learning to learn by self-critique. *Advances in Neural Information Processing Systems*, 32, 2019.

[103] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1199–1208, 2018.

[104] Garrett W Meigs, Robert E Kennedy, and Warren B Cohen. 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*, 115(12):3707–3718, 2011.

[105] Mengdi Huai, Chenglin Miao, Qiuling Suo, Yaliang Li, Jing Gao, and Aidong Zhang. Uncorrelated patient similarity learning. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, pages 270–278. SIAM, 2018.

[106] Ying Wei, Yu Zhang, Junzhou Huang, and Qiang Yang. Transfer learning via learning to transfer: Supplementary material. In *35th International Conference on Machine Learning, ICML 2018*, volume 11, page 8069. International Machine Learning Society (IMLS), 2018.

[107] Deguang Kong, Ryohei Fujimaki, Ji Liu, Feiping Nie, and Chris Ding. Exclusive feature learning on arbitrary structures via $\ell_{1,2}$ norm. *Advances in neural information processing systems*, 27, 2014.

[108] FB Schwing, T Murphree, and PM Green. The northern oscillation index (noi): a new climate index for the northeast pacific. *Progress in oceanography*, 53(2-4):115–139, 2002.

[109] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

[110] Lior Rokach. Ensemble-based classifiers. *Artificial intelligence review*, 33(1):1–39, 2010.

[111] Ludmila I Kuncheva and Christopher J Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine learning*, 51(2):181–207, 2003.

[112] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487. PMLR, 2016.

[113] 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.

[114] Emily K Read, Lindsay Carr, Laura De Cicco, Hilary A Dugan, Paul C Hanson, Julia A Hart, James Kreft, Jordan S Read, and Luke A Winslow. Water quality

data for national-scale aquatic research: The water quality portal. *Water Resources Research*, 53(2):1735–1745, 2017.

[115] Tanja N Williamson, Jeremiah G Lant, Peter Claggett, Elizabeth A Nystrom, Paul C Milly, Hugh L Nelson, Scott A Hoffman, et al. Summary of hydrologic modeling for the delaware river basin using the water availability tool for environmental resources (water). 2015.

[116] R Steven Regan, Steven L Markstrom, Lauren E Hay, Roland J Viger, Parker A Norton, Jessica M Driscoll, and Jacob H LaFontaine. Description of the national hydrologic model for use with the precipitation-runoff modeling system (prms). Technical report, US Geological Survey, 2018.

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

[118] Steven L Markstrom, Robert S Regan, Lauren E Hay, Roland J Viger, Richard MT Webb, Robert A Payn, and Jacob H LaFontaine. Prms-iv, the precipitation-runoff modeling system, version 4. *US Geological Survey Techniques and Methods*, 6:B7, 2015.

[119] Diederik P. Kingma et al. Adam: A method for stochastic optimization. *ICLR*, 2015.

[120] Jared D Willard et al. Daily surface temperatures for 185,549 lakes in the conterminous united states estimated using deep learning (1980–2020). *Limnology and Oceanography Letters*, 2022.

[121] Ankush Khandelwal et al. Physics guided machine learning methods for hydrology. *arXiv:2012.02854*, 2020.

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

[123] Anuj Karpatne et al. Knowledge guided machine learning: Accelerating discovery using scientific knowledge and data. *CRC Press*, 2022.

[124] Keith Beven. Deep learning, hydrological processes and the uniqueness of place. *Hydrological Processes*, 34(16):3608–3613, 2020.

[125] Gregory Ongie et al. Deep learning techniques for inverse problems in imaging. *IEEE Journal on Selected Areas in Information Theory*, 1(1):39–56, 2020.

[126] Ortal Senouf et al. Self-supervised learning of inverse problem solvers in medical imaging. In *Domain adaptation and representation transfer and medical image learning with less labels and imperfect data*, pages 111–119. Springer, 2019.

[127] Wei-Chiu Ma et al. Deep feedback inverse problem solver. *ECCV*, pages 229–246, 2020.

[128] Haiyan Zhou et al. Inverse methods in hydrogeology: Evolution and recent trends. *Advances in Water Resources*, 63:22–37, 2014.

[129] Yuji Kim et al. Geophysical inversion versus machine learning in inverse problems. *The Leading Edge*, 37(12):894–901, 2018.

[130] R Iestyn Woolway et al. Winter inverse lake stratification under historic and future climate change. *Limnology and Oceanography Letters*, 2021.

[131] Petr Pecha et al. Determination of radiological background fields designated for inverse modelling during atypical low wind speed meteorological episode. *Atmospheric Environment*, 246:118105, 2021.

[132] Phuong D Dao et al. Improving hyperspectral image segmentation by applying inverse noise weighting and outlier removal for optimal scale selection. *ISPRS Journal of Photogrammetry and Remote Sensing*, 171:348–366, 2021.

[133] Jeroen Ritsema et al. Seismic imaging of structural heterogeneity in earth's mantle: evidence for large-scale mantle flow. *Science Progress (1933-)*, pages 243–259, 2000.

[134] Alexandre Ganachaud et al. Improved estimates of global ocean circulation, heat transport and mixing from hydrographic data. *Nature*, 408(6811):453–457, 2000.

[135] Lynton Ardizzone et al. Analyzing inverse problems with invertible neural networks. *arXiv:1808.04730*, 2018.

[136] Karthik Kumarasamy et al. Calibration parameter selection and watershed hydrology model evaluation in time and frequency domains. *Water*, 10(6):710, 2018.

[137] Longlong Jing et al. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE TPAMI*, 2020.

[138] Gustav Larsson et al. Colorization as a proxy task for visual understanding. *CVPR*, pages 6874–6883, 2017.

[139] Deepak Pathak et al. Context encoders: Feature learning by inpainting. *CVPR*, pages 2536–2544, 2016.

[140] Mehdi Noroozi et al. Unsupervised learning of visual representations by solving jigsaw puzzles. *ECCV*, pages 69–84, 2016.

[141] Spyros Gidaris et al. Unsupervised representation learning by predicting image rotations. *arXiv:1803.07728*, 2018.

[142] Ting Chen et al. A simple framework for contrastive learning of visual representations. *ICML*, pages 1597–1607, 2020.

[143] Sriram Ravula et al. Inverse problems leveraging pre-trained contrastive representations. *NeurIPS*, 34, 2021.

[144] Aaron van den Oord et al. Representation learning with contrastive predictive coding. *arXiv:1807.03748*, 2018.

[145] Dapeng Feng et al. Enhancing streamflow forecast and extracting insights using long-short term memory networks with data integration at continental scales. *WRR*, 56(9):e2019WR026793, 2020.

[146] Zhen Hao et al. CCAM: China Catchment Attributes and Meteorology dataset. *Earth System Science Data*, 13(12):5591–5616, 2021.

[147] Gemma Coxon et al. CAMELS-GB: hydrometeorological time series and landscape attributes for 671 catchments in Great Britain. *Earth System Science Data*, 2020.

[148] Camila Alvarez-Garreton et al. The CAMELS-CL dataset: Catchment attributes and meteorology for large sample studies-Chile dataset. *HESS*, 22(11):5817–5846, 2018.

[149] V. B. P. Chagas et al. Camels-br: hydrometeorological time series and landscape attributes for 897 catchments in brazil. *Earth System Science Data*, 12(3):2075–2096, 2020.

[150] Frederik Kratzert et al. Benchmarking a catchment-aware long short-term memory network (lstm) for large-scale hydrological modeling. *HESS*, 2019.

[151] Alex Graves et al. Framewise phoneme classification with bidirectional lstm networks. *IJCNN*, 2005.

[152] Frederik Kratzert, Grey Nearing, Nans Addor, Tyler Erickson, Martin Gauch, Oren Gilon, Lukas Gudmundsson, Avinatan Hassidim, Daniel Klotz, Sella Nevo, et al. Caravan-a global community dataset for large-sample hydrology. *Scientific Data*, 10(1):61, 2023.

[153] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[154] Rahul Ghosh et al. Entity aware modelling: A survey. *arXiv:2302.08406*, 2023.

[155] Jung-Hun Song et al. Regionalization of a rainfall-runoff model: Limitations and potentials. *Water*, 2019.

[156] Diederik P Kingma et al. Auto-encoding variational bayes. *ICLR*, 2014, http://arxiv.org/abs/1312.6114v10.

[157] Keith Beven. Facets of uncertainty: epistemic uncertainty, non-stationarity, likelihood, hypothesis testing, and communication. *Hydrological Sciences Journal*, 2016.

[158] Patrick Putzky and Max Welling. Recurrent inference machines for solving inverse problems. *arXiv:1706.04008*, 2017.

[159] Charles Blundell et al. Weight uncertainty in neural network. *ICML*, 2015.

[160] Yeming Wen et al. Flipout: Efficient pseudo-independent weight perturbations on mini-batches. *arXiv*, 2018.

[161] Vanessa Böhm et al. Uncertainty quantification with generative models. *arXiv:1910.10046*, 2019.

[162] Tom E Botterill et al. Using machine learning to identify hydrologic signatures with an encoder-decoder framework. *Authorea Preprints*, 2022.

[163] Tuan-Feng Zhang et al. Generating geologically realistic 3D reservoir facies models using deep learning of sedimentary architecture with generative adversarial networks. *Petroleum Science*, 16(3):541–549, 2019.

[164] Kristian Gundersen et al. Semi-conditional variational auto-encoder for flow reconstruction and uncertainty quantification from limited observations. *Physics of Fluids*, 33(1):017119, 2021, https://doi.org/10.1063/5.0025779.

[165] Yarin Gal et al. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *ICML*, 2016.

[166] Aryan Mobiny et al. Dropconnect is effective in modeling uncertainty of bayesian deep networks. *Scientific reports*, 2021.

[167] Alex Graves et al. Practical variational inference for neural networks. *NeurIPS*, 2011.

[168] Dayang Li et al. Bayesian LSTM with stochastic variational inference for estimating model uncertainty in process-based hydrological models. *WRR*, 57(9):e2021WR029772, 2021.

[169] Christopher M Bishop et al. Mixture density networks. 1994.

[170] Daniel Klotz et al. Uncertainty estimation with deep learning for rainfall–runoff modeling. *HESS*, 2022.

[171] Ganggang Zuo et al. Decomposition ensemble model based on variational mode decomposition and long short-term memory for streamflow forecasting. *Journal of Hydrology*, 585:124776, 2020.

[172] Peyman Abbaszadeh et al. The quest for model uncertainty quantification: A hybrid ensemble and variational data assimilation framework. *WRR*, 2019.

[173] Danilo Jimenez Rezende et al. Stochastic backpropagation and approximate inference in deep generative models. *ICML*, page II–1278–II–1286, 2014.

[174] Durk P Kingma et al. Semi-supervised learning with deep generative models. *NeurIPS*, 27:3581–3589, 2014.

[175] Kihyuk Sohn et al. Learning structured output representation using deep conditional generative models. *NeurIPS*, 2015.

[176] Michael I Jordan et al. An introduction to variational methods for graphical models. *Machine Learning*, 1999.

[177] Thomas Lees et al. Benchmarking data-driven rainfall–runoff models in Great Britain: a comparison of long short-term memory (LSTM)-based models with four lumped conceptual models. *HESS*, 2021.

[178] Somya Sharma et al. Probabilistic inverse modeling: An application in hydrology. *SDM*, 2023.

[179] Francesco Laio et al. Verification tools for probabilistic forecasts of continuous hydrological variables. *HESS*, 2007.

[180] Jared Willard et al. Integrating scientific knowledge with machine learning for engineering and environmental systems. *ACM CSUR*, 2022.

[181] Dara Entekhabi et al. The soil moisture active passive (smap) mission. *Proceedings of the IEEE*, 2010.

[182] Ilyes Khemakhem et al. Variational autoencoders and nonlinear ica: A unifying framework. *AISTATS*, 2020.

[183] Alaa Tharwat et al. Independent component analysis: An introduction. *Applied Computing and Informatics*, 2020.

[184] Jan Eriksson and Visa Koivunen. Identifiability, separability, and uniqueness of linear ica models. *IEEE signal processing letters*, 2004.

[185] Aapo Hyvärinen et al. Nonlinear ica using auxiliary variables and generalized contrastive learning. *AISTATS*, 2019.

[186] Feng Xie et al. Generalized independent noise condition for estimating latent variable causal graphs. *NeurIPS*, 2020.

[187] Bohdan Kivva et al. Learning latent causal graphs via mixture oracles. *NeurIPS*, 2021.

[188] Weiran Yao et al. Learning temporally causal latent processes from general temporal data. *arXiv*, 2021.

[189] Björn Lütjens et al. Physically-consistent generative adversarial networks for coastal flood visualization. *arXiv*, 2021.

[190] Paris Perdikaris et al. Multiscale modeling and simulation of brain blood flow. *Physics of Fluids*, 2016.

[191] Alex Lavin et al. Simulation intelligence: Towards a new generation of scientific methods. *arXiv*, 2021.

[192] Apostolos F Psaros et al. Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons. *arXiv*, 2022.

[193] Jonathan Wenger et al. Posterior and computational uncertainty in gaussian processes. *arXiv*, 2022.

[194] Murat Sensoy et al. Evidential deep learning to quantify classification uncertainty. *NeurIPS*, 2018.

[195] World Health Organization et al. Improving data quality: a guide for developing countries. 2003.

[196] Aimé Patrice Koumamba et al. Health information systems in developing countries: case of african countries. *BMC Medical Informatics and Decision Making*, 2021.

[197] Moritz Hardt et al. Equality of opportunity in supervised learning. *NeurIPS*, 2016.

[198] Cynthia Dwork et al. Fairness through awareness. *ITCS*, 2012.

[199] Ignacio Serna et al. Sensitive loss: Improving accuracy and fairness of face representations with discrimination-aware deep learning. *Artificial Intelligence*, 2022.

[200] Jamal Alasadi et al. Toward fairness in face matching algorithms. *FAccT*, 2019.

[201] Chris Sweeney et al. Reducing sentiment polarity for demographic attributes in word embeddings using adversarial learning. *FAccT*, 2020.

[202] Yiqun Xie et al. Fairness by "where": A statistically-robust and model-agnostic bi-level learning framework. *AAAI*, 2022.

[203] Eun Seo Jo et al. Lessons from archives: Strategies for collecting sociocultural data in machine learning. *FAccT*, 2020.