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

Rahul Ghosh
University of Minnesota - Twin Cities
ghosh128@umn.edu

Arvind Renganathan
University of Minnesota - Twin Cities
renga016@umn.edu

Kshitij Tayal
University of Minnesota - Twin Cities
tayal007@umn.edu

Xiang Li
University of Minnesota - Twin Cities
lixx5000@umn.edu

Ankush Khandelwal
University of Minnesota - Twin Cities
khand035@umn.edu

Xiaowei Jia
University of Pittsburgh
xiaowei@pitt.edu

Christopher Duffy
Penn State University
cxd11@psu.edu

John Nieber
University of Minnesota - Twin Cities
nieber@umn.edu

Vipin Kumar
University of Minnesota - Twin Cities
kumar001@umn.edu

ABSTRACT

Machine Learning is beginning to provide state-of-the-art performance in a range of environmental applications such as streamflow prediction in a hydrologic basin. However, building accurate broad-scale models for streamflow remains challenging in practice due to the variability in the dominant hydrologic processes, which are best captured by sets of process-related basin characteristics. Existing basin characteristics suffer from noise and uncertainty, among many other things, which adversely impact model performance. To tackle the above challenges, in this paper, we propose a novel Knowledge-guided Self-Supervised Learning (KGSSL) inverse framework to extract system characteristics from driver(input) and response(output) data. This first-of-its-kind framework achieves robust performance even when characteristics are corrupted or missing. We evaluate the KGSSL framework in the context of stream flow modeling using CAMELS (Catchment Attributes and MEteorology for Large-sample Studies) which is a widely used hydrology benchmark dataset. Specifically, KGSSL outperforms baseline by 16 % in predicting missing characteristics. Furthermore, 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.

CCS CONCEPTS

• **Applied computing** → **Earth and atmospheric sciences**; • **Computing methodologies** → **Machine learning**.

KEYWORDS

Self-supervised Learning, Inverse Modeling, Forward Modeling

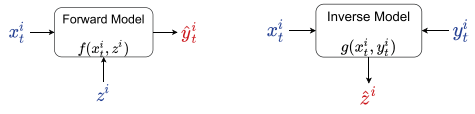
KDD '22, August 14–18, 2022, Washington, DC, USA.

ACM ISBN 978-1-4503-9385-0/22/08…$15.00
https://doi.org/10.1145/3534678.3539448

ACM Reference Format:
Rahul Ghosh, Arvind Renganathan, Kshitij Tayal, Xiang Li, Ankush Khandelwal, Xiaowei Jia, Christopher Duffy, John Nieber, and Vipin Kumar. 2022. Robust Inverse Framework using Knowledge-guided Self-Supervised Learning: An application to Hydrology. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3534678.3539448

1 INTRODUCTION

Machine learning (ML) is increasingly being used to solve challenging tasks in scientific applications such as hydrology, freshwater 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 [24]. 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 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 [19, 35]. 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 [15]. Such knowledge-guided machine learning (KGML) models have been shown to provide improved performance over black-box ML



(a) (b)

Figure 1: (a) Forward model which uses meteorological drivers (x_i^t) and entity characteristics (z_i) to predict response (y_i^t) (b) The inverse model which approximates entity characteristics (z_i) by inverting the forward process.

models even with fewer sample, and are able to generalize in unseen scenarios [13, 16, 17, 31].

In streamflow modeling (as well as many other scientific applications), entity characteristics are often surrogate variables of the true basin characteristics [4] 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 paper 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 1b shows the diagrammatic representation of this inverse problem. Inverse problems [26] 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 [33], and many more (see [26] 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 [26].

In this paper, we propose 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 [22] 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. We present 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. We call our methodology 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, we further add a pseudo-inverse loss on top of the learned embeddings to guide the learning using the known basin characteristics.

We demonstrate the usefulness of KGSSL in the context of stream flow modeling using CAMELS (Catchment Attributes and MEteorology for Large-sample Studies) [23] which is a widely used hydrology benchmark data set. Specifically, we show that our methodology can effectively impute basin characteristics (if they are missing) or reduce their uncertainty (if they are uncertain). We additionally demonstrate its usefulness in situations where static characteristics $\boldsymbol{z}$ are not known for any basins. Here, we show that the similarity between basins using learned embeddings closely follow the similarity based on the actual characteristics. Further, we show that the learned embeddings can act as an effective replacement for static characteristics in a forward model. Our main contributions are listed below:

- We demonstrate 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.

2 RELATED WORKS

Inverse problems [26] 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 [37], geophysics [18], oceanography [36], meteorology [29], remote sensing [8], etc. For example, an inverse problem arises when we reconstruct Earth's interior by modeling the physical propagation of seismic waves [32]. Similarly, in reservoir engineering [10], 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 [3] model forward/inverse mapping within a single network. However, in hydrology, the initial physical parameters are not known reliably [20] for the basins/catchment due to temporal and spatial heterogeneity.

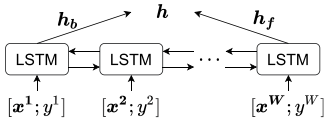


Figure 2: Bidirectional LSTM based Sequence Encoder

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 [14] for model training. In self-supervised learning, the models are trained using pretext tasks instead of an actual task. For example, image colorization [21], image inpainting [28], solving image-jigsaw [25], predicting rotations [11], etc. For a comprehensive understanding of self-supervised representation learning, we would like to redirect the reader to a survey by Jing et al. [14]. Our work utilizes a self-supervised loss called InfoNCE loss [6] 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 [30], 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.

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

$$\mathcal{L} = \lambda_1 \mathcal{L}_{Rec} + \lambda_2 \mathcal{L}_{Cont} + \lambda_3 \mathcal{L}_{Inv} \tag{1}$$

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 ($[\boldsymbol{x_i^t}; y_i^t]$).

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 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,

$$\begin{aligned}
\boldsymbol{i_t} &= \sigma(\boldsymbol{W_i}\left[[\boldsymbol{x^t}; y^t]; \boldsymbol{h^{t-1}}\right] + \boldsymbol{b_i}) \\
\boldsymbol{f_t} &= \sigma(\boldsymbol{W_f}\left[[\boldsymbol{x^t}; y^t]; \boldsymbol{h^{t-1}}\right] + \boldsymbol{b_f}) \\
\boldsymbol{g_t} &= \sigma(\boldsymbol{W_g}\left[[\boldsymbol{x^t}; y^t]; \boldsymbol{h^{t-1}}\right] + \boldsymbol{b_g}) \\
\boldsymbol{o_t} &= \sigma(\boldsymbol{W_o}\left[[\boldsymbol{x^t}; y^t]; \boldsymbol{h^{t-1}}\right] + \boldsymbol{b_o}) \\
\boldsymbol{c_t} &= \boldsymbol{f_t} \odot \boldsymbol{c_{t-1}} + \boldsymbol{i} \odot \boldsymbol{g_t} \\
\boldsymbol{h_t} &= \boldsymbol{o_t} \odot \tanh(\boldsymbol{c_t})
\end{aligned} \tag{2}$$

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 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.

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,

$$\mathcal{L}_{Rec} = \frac{1}{2N} \sum_{e \in \{a,p\}} \sum_{i=1}^{N} MSE(\hat{S}_{e_i}, S_{e_i}) \tag{3}$$

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.

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 [6, 27]. 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,

$$\begin{aligned} l(a_i, p_i) = & \frac{\exp\left(sim(\boldsymbol{h_{a_i}}, \boldsymbol{h_{p_i}})/\tau\right)}{\sum_{e \in \{a,p\}} \sum_{j=1}^{N} \exp\left(sim(\boldsymbol{h_{a_i}}, \boldsymbol{h_{e_j}})/\tau\right)} \\ & + \frac{\exp\left(sim(\boldsymbol{h_{p_i}}, \boldsymbol{h_{a_i}})/\tau\right)}{\sum_{e \in \{a,p\}} \sum_{j=1}^{N} \exp\left(sim(\boldsymbol{h_{p_i}}, \boldsymbol{h_{e_j}})/\tau\right)} \end{aligned} \tag{4}$$

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,

$$\mathcal{L}_{Cont} = \frac{1}{2N} \sum_{i=1}^{N} l(a_i, p_i) \tag{5}$$

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.

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 $\mathcal{I}$ on sequence encoder output h to estimate $\hat{\boldsymbol{z}} = \mathcal{I}(\boldsymbol{h})$ and then we

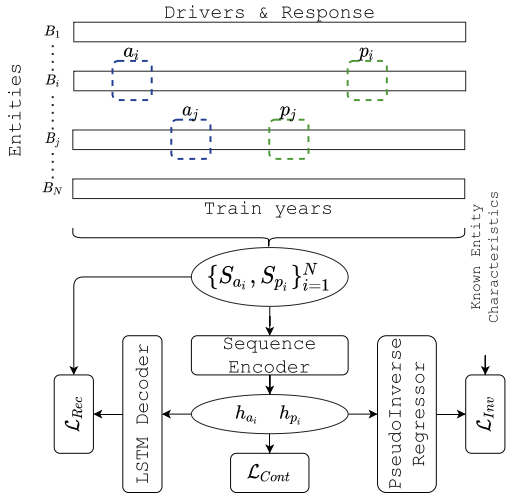


Figure 3: Proposed inverse model generates embeddings for a basin from the LSTM Encoder (Figure 2) and is trained in a self-supervised manner. Strong supervision ($\mathcal{L}_{Inv}$) is added when ground-truth characteristics are available for a limited number of entities

define regression loss with the available set of entity characteristics (z) as shown in Figure 3 . Pseudoinverse loss is defined as follows:

$$\mathcal{L}_{Inv} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{z} \sum_{j=1}^{z} (z_i^j - \hat{z}_i^j)^2 \tag{6}$$

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 $\mathcal{I}$ 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,

$$\boldsymbol{h_i} = \frac{W}{T} \sum_{j=1}^{T/W} \boldsymbol{h_i^j} \quad \hat{\boldsymbol{z}}_i = \frac{W}{T} \sum_{j=1}^{T/W} \hat{\boldsymbol{z}}_i^j \quad \boldsymbol{unc_i} = \sqrt{\frac{W}{T} \sum_{j=1}^{T/W} (\hat{\boldsymbol{z}}_i^j - \hat{\boldsymbol{z}}_i)^2} \tag{7}$$

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.

4 EXPERIMENTAL RESULTS

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,

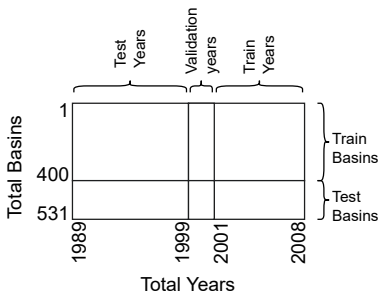


Figure 4: Experimental setting followed in the paper for training and testing of the ML models

streamflow prediction [1]. 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 [9, 19]. In particular, using the CAMELS dataset, Kratzert et al. [19] 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.

Following the set up used by Kratzert et al. [19], 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. 4 show the experimental setting followed in this paper. 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 4.2,4.3,4.4 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 4.4 considers the case where the catchment characteristics are not available during training. In addition, section 4.6 shows the ability of KGSSL to improve the forward modeling task.

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

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 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 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. 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 1 and Figure 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,*

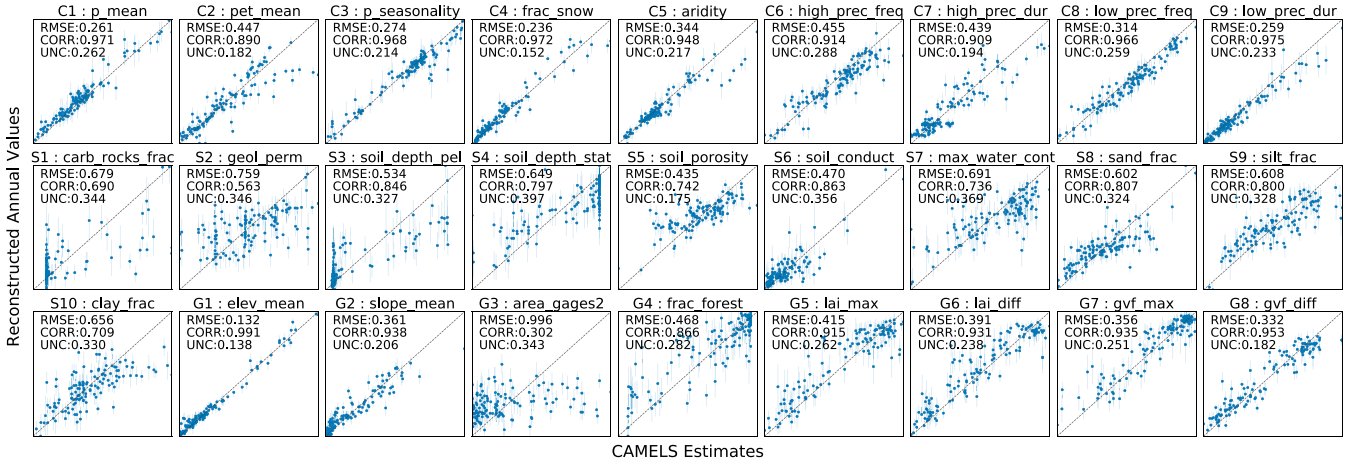


Figure 5: Scatter plot of the CAMELS Estimates for static characteristics (x-axis) for the testing basins vs. reconstructed annual characteristics (y-axis). The error bar across the points show the variation of the reconstructed characteristics annually across test years.

Group	Indexes	Sec 4.2	Sec 4.3			Sec 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 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.

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 [1], 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.

4.3 Robustness to Corruption in available characteristics

As highlighted in Sec.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

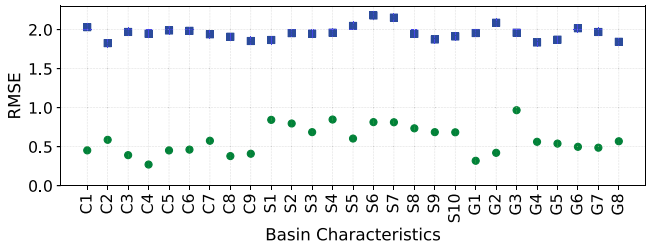
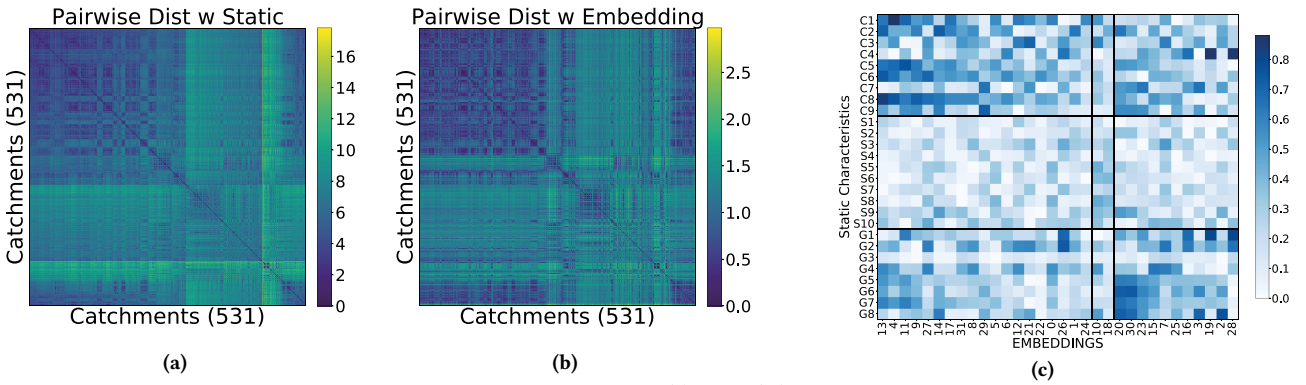


Figure 6: Comparison of the RMSE of the corrupted values (in blue) generated by adding Gaussian noise ($\mathcal{N}(0, 2\sigma_i)$) to 50% of basins and the reconstructed values (in green).

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 2 shows the performance of various methods in terms of correlation of the predicted characteristics with the original



(a) (b) (c)

Figure 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.

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 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.

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) [12] are not included in the characteristics set of the catchments in the CAMELS dataset being used in this paper (which only contains basins from USA) [1]. The same scenario is also present in CAMELS version for Great Britain [7], Chili [2], and Brazil [5]. 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. 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 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 4.2).

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 7a) and compare them with the distances computed using learned embeddings (Figure 7b). We generate (Figure 7a) by reordering the rows in the distance matrix

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 3: Model performance for different percentages of missing values.

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 7a) is then further applied to the distance matrix computed using learned embeddings to generate (Figure 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 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.

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 [19] 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.

4.6.1 *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 3 (first two rows) report performance of state-of-the-art EA-LSTM (baseline) trained with

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 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.

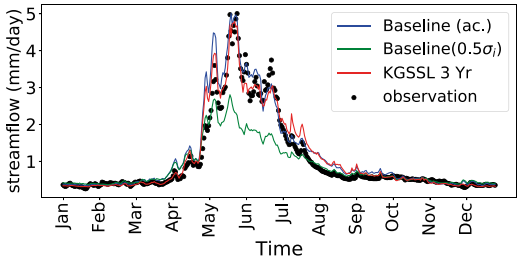


Figure 8: Basin at year 1992 (Best seen in color)

all and no static characteristics [19]. 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 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.

4.6.2 *Forward modeling with corrupted entity characteristics.*

As shown by Kratzert et al. [19], 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 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 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.

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 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 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 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 paper 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 [34, 38], and will be considered in future work.

6 ACKNOWLEDGEMENT

This work was funded by the NSF HDR Grant 1934721 and NSF FAI Grant 2147195. Access to computing facilities was provided by the Minnesota Supercomputing Institute. John Nieber's effort on this project was partially supported by the USDA National Institute of Food and Agriculture, Hatch/Multistate Project MN 12-109.

REFERENCES

[1] Nans Addor et al. 2017. The CAMELS data set: Catchment attributes and meteorology for large-sample studies. *Hydrology and Earth System Sciences* (2017).
[2] Camila Alvarez-Garreton et al. 2018. The CAMELS-CL dataset: Catchment attributes and meteorology for large sample studies-Chile dataset. *Hydrology and Earth System Sciences* (2018).
[3] Lynton Ardizzone et al. 2018. Analyzing inverse problems with invertible neural networks. *arXiv preprint arXiv:1808.04730* (2018).
[4] Keith Beven. 2020. Deep learning, hydrological processes and the uniqueness of place. *Hydrological Processes* (2020).
[5] Vinicius B.P. Chagas et al. 2020. CAMELS-BR: Hydrometeorological time series and landscape attributes for 897 catchments in Brazil. *Earth System Science Data* (2020).
[6] Ting Chen et al. 2020. A simple framework for contrastive learning of visual representations. *ICML* (2020).
[7] Gemma Coxon et al. 2020. CAMELS-GB: hydrometeorological time series and landscape attributes for 671 catchments in Great Britain. *Earth System Science Data* (2020).
[8] Phuong D Dao et al. 2021. Improving hyperspectral image segmentation by applying inverse noise weighting and outlier removal for optimal scale selection. *ISPRS Journal of Photogrammetry and Remote Sensing* (2021).
[9] Dapeng Feng et al. 2020. Enhancing streamflow forecast and extracting insights using long-short term memory networks with data integration at continental scales. *Water Resources Research* (2020).
[10] Alexandre Ganachaud et al. 2000. Improved estimates of global ocean circulation, heat transport and mixing from hydrographic data. *Nature* (2000).
[11] Spyros Gidaris et al. 2018. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728* (2018).
[12] Zhen Hao et al. 2021. CCAM: China Catchment Attributes and Meteorology dataset. *Earth System Science Data* (2021).
[13] Xiaowei Jia et al. 2019. Physics guided RNNs for modeling dynamical systems: A case study in simulating lake temperature profiles. *SDM* (2019).
[14] Longlong Jing et al. 2020. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE TPAMI* (2020).
[15] Anuj Karpatne et al. 2017. Theory-guided data science: A new paradigm for scientific discovery from data. *IEEE Transactions on knowledge and data engineering* (2017).
[16] Anuj Karpatne et al. 2022. *Knowledge Guided Machine Learning: Accelerating Discovery using Scientific Knowledge and Data.* CRC Press.
[17] Ankush Khandelwal et al. 2020. Physics guided machine learning methods for hydrology. *arXiv preprint arXiv:2012.02854* (2020).
[18] Yuji Kim et al. 2018. Geophysical inversion versus machine learning in inverse problems. *The Leading Edge* (2018).
[19] Frederik Kratzert et al. 2019. Towards learning universal, regional, and local hydrological behaviors via machine learning applied to large-sample datasets. *Hydrology and Earth System Sciences* (2019).
[20] Karthik Kumarasamy et al. 2018. Calibration parameter selection and watershed hydrology model evaluation in time and frequency domains. *Water* (2018).
[21] Gustav Larsson et al. 2017. Colorization as a proxy task for visual understanding. *CVPR* (2017).
[22] Wei-Chiu Ma et al. 2020. Deep feedback inverse problem solver. *European Conference on Computer Vision* (2020).
[23] Andrew J Newman et al. 2015. Gridded ensemble precipitation and temperature estimates for the contiguous United States. *Journal of Hydrometeorology* (2015).
[24] Brent D Newman et al. 2006. Ecohydrology of water-limited environments: A scientific vision. *Water resources research* (2006).
[25] Mehdi Noroozi et al. 2016. Unsupervised learning of visual representations by solving jigsaw puzzles. *ECCV* (2016).
[26] Gregory Ongie et al. 2020. Deep learning techniques for inverse problems in imaging. *IEEE Journal on Selected Areas in Information Theory* (2020).
[27] Aaron van den Oord et al. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).
[28] Deepak Pathak et al. 2016. Context encoders: Feature learning by inpainting. *CVPR* (2016).
[29] Petr Pecha et al. 2021. Determination of radiological background fields designated for inverse modelling during atypical low wind speed meteorological episode. *Atmospheric Environment* (2021).
[30] Sriram Ravula et al. 2021. Inverse Problems Leveraging Pre-trained Contrastive Representations. *NeurIPS* (2021).
[31] Jordan S Read et al. 2019. Process-guided deep learning predictions of lake water temperature. *Water Resources Research* (2019).
[32] Jeroen Ritsema et al. 2000. Seismic imaging of structural heterogeneity in Earth's mantle: evidence for large-scale mantle flow. *Science Progress (1933-)* (2000).
[33] Ortal Senouf et al. 2019. Self-supervised learning of inverse problem solvers in medical imaging.
[34] Risto Vuorio et al. 2019. Multimodal model-agnostic meta-learning via task-aware modulation. *arXiv preprint arXiv:1910.13616* (2019).
[35] Jared D Willard et al. 2022. Daily surface temperatures for 185,549 lakes in the conterminous United States estimated using deep learning (1980–2020). *Limnology and Oceanography Letters* (2022).
[36] R Iestyn Woolway et al. 2021. Winter inverse lake stratification under historic and future climate change. *Limnology and Oceanography Letters* (2021).
[37] Haiyan Zhou et al. 2014. Inverse methods in hydrogeology: Evolution and recent trends. *Advances in Water Resources* (2014).
[38] Luisa Zintgraf et al. 2019. Fast context adaptation via meta-learning. *ICML* (2019).

A APPENDIX

A.1 Index of Catchment attributes used in our paper

Group	index	Name
Climate	C1	p mean
	C2	pet mean
	C3	p seasonality
	C4	frac snow
	C5	aridity
	C6	high prec freq
	C7	high prec dur
	C8	low prec freq
	C9	low prec dur
Soil geology	S1	carbonate rocks frac
	S2	geol permeability
	S3	soil depth pelletier
	S4	soil depth statsgo
	S5	soil porosity
	S6	soil conductivity
	S7	max water content
	S8	sand frac
	S9	silt frac
	S10	clay frac
Geomorphology	G1	elev mean
	G2	slope mean
	G3	area gages2
	G4	frac forest
	G5	lai max
	G6	lai diff
	G7	gvf max
	G8	gvf diff

Table 5: Table of catchment characteristics used in this experiment, description of the characteristics is available and defined in [1]

A.2 Hyperparameter Tuning

We used grid search over a range of parameter values to find the hyperparameters, i.e., λ_1, λ_2, λ_3, batch_size, the temperature of contrastive loss, learning rate, dimension of embedding. Specifically, we considered the following possible parameter values.

- Dimension of embedding: 32, 64, 128,256
- Learning_rate : 0.0005, 0.001, 0.003, 0.005, 0.05
- λ_1 :0.01, 0.1, 1, 10
- λ_2 :1
- λ_3 : 0.1, 1, 10
- Batch_size: 100, 200
- Temp : 0.1, 0.5, 0.7, 1

We trained our model on train basins in the training period and tested the model in the testing period on testing basins for each combination in the parameter set. We chose the parameter set with the least avg RMSE in the train basins during the validation years as the final parameter configuration

A.3 Reproducibility

CAMELS input data is freely available on the homepage of the NCAR at https://ral.ucar.edu/solutions/products/camels. The code is available at https://tinyurl.com/bdny7fk6 (KGSSL Code Link).