Invertibility aware Integration of Static and Time-series data: An application to Lake Temperature Modeling

Kshitij Tayal* Xiaowei Jia† Rahul Ghosh* Jared Willard* Jordan Read‡
Vipin Kumar*

Abstract

Accurate predictions of water temperature are the foundation for many decisions and regulations, with direct impacts on water quality, fishery yields, and power production. Building accurate broad-scale models for lake temperature prediction remains challenging in practice due to the variability in the data distribution across different lake systems monitored by static and time-series data. In this paper, to tackle the above challenges, we propose a novel machine learning based approach for integrating static and time-series data in deep recurrent models, which we call Invertibility-Aware-Long Short-Term Memory(IA-LSTM), and demonstrate its effectiveness in predicting lake temperature. Our proposed method integrates components of the Invertible Network and LSTM to better predict temperature profiles (forward modeling) and infer the static features (i.e., inverse modeling) that can eventually enhance the prediction when static variables are missing. We evaluate our method on predicting the temperature profile of 450 lakes in the Midwestern U.S. and report relative improvement of 4% to capture data heterogeneity and simultaneously outperform baseline predictions by 12% when static features are unavailable.

1 Introduction

The seasonal water temperature value of thousands of water bodies regulates the water flow and directly influences local water supply quality [1], food resources [2], and aquatic life [3]. In the context of global warming, accurate prediction of the water temperature in lakes can provide crucial information to policymakers and assist in their timely intervention.

Recently, machine learning (ML) models have shown a great promise for predicting lake temperature using meteorological drivers (weather, temperature, rainfall) [4, 5]. However, lake temperature prediction using data-driven/ML methods is challenging because the relationship between input (meteorological drivers) and target (water temperature response) is governed by inherent characteristics of different lakes. For example, the same amount of solar radiation can have different warming effects for lakes with different hypsography (e.g., depth and surface area) and other water properties (e.g., clarity and wind exposure). See Figure 1 for a demonstration of the importance of static features in predicting lake temperature. Additional complexity arises given that some of these system characteristics are unknown or difficult to measure. One intuitive approach for handling the data heterogeneity is to build individual models separately for each system [4]. However, there are hundreds of thousands of water bodies, making it costly to collect a large number of training samples for every system that will be needed to train high-quality individual models for capturing complex processes.

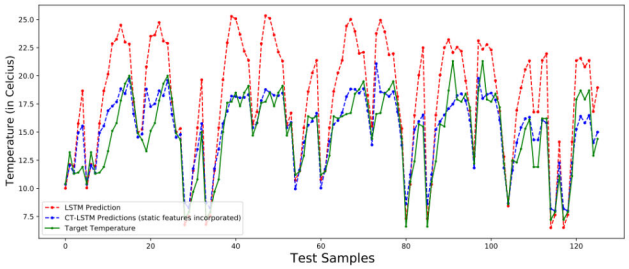


Figure 1: Demonstration of the importance of static features in learning a target lake temperature (green points). Red line shows network output trained solely using time-series data, and the Blue line shows network output trained using both static and time-series data.

Our approach in this paper is to build a global ML model for a large number of lakes. To capture the data heterogeneity, the ML model needs to incorporate both lake characteristics (which are often assumed to be static over time) and dynamic time-series of meteorological drivers [6]. As static and time-series variables often make complementary contributions to predicting water temperature, the ML model needs to effectively incorporate both types of variables to maximize its prediction power. Recurrent neural networks (RNN)

*University of Minnesota. {tayal, ghosh128, willa099, kumar001}@umn.edu
†University of Pittsburgh. {xiaowei@pitt.edu}
‡U.S. Geological Survey. {jread@usgs.gov}

provide a powerful approach to model time-series data by exploiting temporal information. In fact, LSTMs are now extensively used for environmental modeling [7], where both static and time-series variables are supplied as input (here static variables are repeated at each time step). However, original RNN models were not designed to exploit static data. Recently, researchers have explored two approaches. First approach uses two separate neural networks to learn representations of time-series data (e.g., LSTM) and static characteristics (e.g., CNN) and then concatenate them at the final layer [8, 9]. Second approach learns a representation of the static characteristics using a feed forward neural network (FFN) and then feeds this learned representation at each time-step along with time-series data into an LSTM [10, 11]. Since all these models employ a feed-forward neural network, they cannot be used when static characteristics are not present, which is common for many lakes in the U.S. This motivates us to ask the question: *Can we develop a novel neural network architecture that can accurately capture data heterogeneity through a forward modeling process, and further recover static features when they are missing through an inverse modeling process ?*

Inverse modeling [12] appears in many fields of engineering when the goal is to recover "hidden" characteristics of a system from "observed" data. In general, the forward problem, i.e., generating observations/outputs from parameters, is well-defined; while the inverse problem is generally ill-posed, i.e., one may not be able to uniquely recover the input field given noisy and incomplete observations. In recent years, deep learning techniques have shown remarkable success for solving inverse problems in various fields such as medical imaging [13], and many more. However, most methods train separate models to model forward and inverse processes. Recently flow-based invertible neural network models [14, 15] have been used to solve inverse problems that are mathematically invertible by architectural design, which allows them to model forward and inverse processes within a single network. Consequently, models can be trained on a forward process and provide the inverse for free by running them backward.

This paper aims to build a global predictive model with both forward and inverse modeling processes. The forward modeling process aims to predict water temperature from climate drivers while also capturing the data heterogeneity. The inverse modeling process adapts the model to unmonitored lakes where static variables are missing. Specifically, we integrate static and time-series data in deep recurrent models, which we call Invertibility-Aware Long-Short Term Memory (IA-LSTM). To the best of our knowledge, this paper is the first to present an integration method that can solve both forward and inverse problem. The model consists of two components, namely an invertible network and an LSTM network, and both components are trained together in an end-to-end fashion. In particular, the invertible network is constructed using the concepts of real NVP [14] that consists of several affine coupling layers. We evaluated our proposed method on Midwestern U.S. lakes for which data are available from the U.S. Geological Survey (USGS) [16] and show that (1) our proposed method improves the prediction performance by 4% over the state of the art model when both static and time-series data is present (forward model) and (2) it can effectively infer static variables when they are missing, and the prediction using inferred variables bring 12% improvement in predicting temperature profile.

2 Literature Review

Integrating Static and Time-series Data : Recurrent Neural Network (RNN) and its variations are among the most extensively researched deep neural networks for handling sequential temporal data. While predicting future events, the majority of the RNN-based models merely utilize dynamic data. Only a few studies use both static and dynamic data for prediction by leveraging a separate algorithm, such as the feed-forward (FFN) or convolutional network [9]. These models concatenate the learned representation of static and time-series data either at the latter layers [8, 9] or feed them together into RNN at each time step [10, 11]. For example, [10] presents a deep-learning approach that takes static information (i.e., demographics, family history, blood group) into a FFN and dynamic information (patients' visits at different times) into an RNN to predict future clinical events. In a parallel study, but in musical research, authors[8] developed a neural network architecture by combining FFN and LSTM to generate drum sequences. In this design, dynamic data from three bands of the drum was fed into an LSTM layer and FFN was used to model bass information (static data). The outputs of both layers were fused to predict the final sequence. Likewise, [11] propose a multi-modal fusion technique that exploits the correlation of static and dynamic time-series data through cross-modal imputation in an integrated recurrent model for oncology early warning systems. Recently [17] combines both static and dynamic data for human design decision prediction by integrating FFN and RNN. All the above models address forward modeling. Still, because of their design, none of them are meant to infer the static variables (i.e., inverse modeling) that could be used to enhance prediction when static variables are missing.

Inverse Problems and Invertible Neural Network: Inverse problems 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. The inverse problem is generally ill-posed, i.e., multiple input values exist for the same observations [18]. Commonly used methods for inverse modeling [19] are computationally expensive and often produce inconsistent results. Recently, deep learning techniques [12] have successfully solved inverse problems; however, most methods train separate models for learning inverse processes. Training a separate inverse model ignores the connection to its forward problem, losing the potential to exploit the shared knowledge between them. Furthermore, direct optimization of those problems requires two dedicated models to be trained separately, increasing the computational requirements. To avoid training two separate models Jayaram et al [20] construct a forward model, numerically invert, and optimize it to find the most likely input to generate that output. Lately, a new class of neural networks, called Invertible Neural Networks (INNs), have been introduced by [14, 15] based on the ideas of normalizing flow. INNs are bijective functions which can be trained on a forward process $\mathscr{F} : \mathbb{R}^d \to \mathbb{R}^d$ and we get inverse mapping $\mathscr{F}^{-1} : \mathbb{R}^d \to \mathbb{R}^d$ for free by running them backwards. Their bijective architecture allows direct log-likelihood training, which makes it a popular choice for solving many inverse problems like superresolution [21]. INNs have three distinctive properties[22]: (i) it models forward/inverse mapping within a single network (shared parameters), (ii) invertible architecture allows free inverse mapping, and (iii) addressing the ill-posed problems via addition of latent variables to have a one-to-one mapping between input and output. In this work, we integrate components of INN in our LSTM model, which incorporates both static and time-series features to address heterogeneity and infer the static variables (i.e., inverse modeling) that can eventually enhance the prediction when static variables are missing/not available.

3 Problem Formulation & Preliminaries

We illustrate our approach for the problem of modeling the temperature of water in a lake at depth d and time t. Mathematically, we assume a dataset of n samples (n is the number of lakes in our case) $\{\boldsymbol{s_i}, X_i, Y_i\}_{i=1}^{n}$. For each sample, daily climate drivers are represented by X_i as a multivariate input time series for T timestamp i.e. $X_i = [\boldsymbol{x_i^1}, \boldsymbol{x_i^2}, \ldots, \boldsymbol{x_i^T}]$ where $\boldsymbol{x_i^t} \in \mathbb{R}^{D_x}$ indicates input vector at time $t \in T$ with D_x dimension. $\boldsymbol{s_i} \in \mathbb{R}^{D_s}$ denotes the static characteristic vector of the lake with D_s dimensions. $Y_i = [\boldsymbol{y_i^1}, \boldsymbol{y_i^2}, \ldots, \boldsymbol{y_i^T}]$, where $\boldsymbol{y_i^t} \in \mathbb{R}^{D_d}$ denotes the daily temperature observations at multiple depths corresponding to $(X_i, \boldsymbol{s_i})$.

Forward Modeling Here the goal is to accurately model heterogeneity by integrating the daily climate drivers (X_i) with the static characteristics $(\boldsymbol{s_i})$ of a lake to learn a forward operator $\mathscr{G}$ that predicts the temperature of water in a lake at depth d and time t i.e $\mathscr{G} : X, \boldsymbol{s} \to Y$.

Inverse Modeling The inverse problem aims to recover the static characteristics/variables $(\boldsymbol{s_i})$ for a lake given a few samples of temperature observations (Y_i) and input climate drivers (X_i) of a lake. With a known forward operator $\mathscr{G}$, the generic solution of the inverse problem can be written as:

$$\boldsymbol{s}^\star = \min_{\boldsymbol{s}} \mathcal{L}(\mathscr{G}(X, \boldsymbol{s}), Y) + \lambda \mathscr{R}(\boldsymbol{s}) \tag{3.1}$$

where $\mathcal{L}(.)$ is a distance metric, $\mathscr{R}$ is the regularizer with λ as the regularization parameter. These static variables have utility to enhance the prediction of Y_i when static variables are missing.

4 Method

4.1 The architecture of IA-LSTM

Our proposed IA-LSTM is built upon LSTM and Invertible Neural Networks (INNs). We first describe these two components and then combine them to build the IA-LSTM model.

LSTM is a type of recurrent neural network designed to avoid exploding and vanishing gradient problems. Therefore it is particularly suited for tasks where long range temporal dependencies between events exist such as ours. Each LSTM cell has a cell state $\boldsymbol{c}^t$, and the following three gates: Forget gate $\boldsymbol{f}^t$, output gate $\boldsymbol{o}^t$ and input gate $\boldsymbol{i}^t$ which serves as a memory and allows preserving information from the past. The value of each gate is computed as follows, where $W_{[c,f,i,o]}$ are learnable weight matrices and $\boldsymbol{b}_{[c,f,i,o]}$ are learnable bias vectors.

$$\begin{aligned}
\bar{\boldsymbol{c}}^t &= \sigma(W_c.[\boldsymbol{h}^{t-1}, \boldsymbol{x}^t] + \boldsymbol{b}_v) \\
\boldsymbol{f}^t &= \sigma(W_f.[\boldsymbol{h}^{t-1}, \boldsymbol{x}^t] + \boldsymbol{b}_f) \\
\boldsymbol{i}^t &= \sigma(W_i.[\boldsymbol{h}^{t-1}, \boldsymbol{x}^t] + \boldsymbol{b}_i) \\
\boldsymbol{o}^t &= \sigma(W_o.[\boldsymbol{h}^{t-1}, \boldsymbol{x}^t] + \boldsymbol{b}_o)
\end{aligned} \tag{4.2}$$

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

$$\begin{aligned}
\boldsymbol{c}^t &= \boldsymbol{f}^t \otimes \boldsymbol{c}^{t-1} + \boldsymbol{i}^t \otimes \bar{\boldsymbol{c}}^t \\
\boldsymbol{h}^t &= \boldsymbol{o}^t \otimes \tanh(\boldsymbol{c}^t)
\end{aligned} \tag{4.3}$$

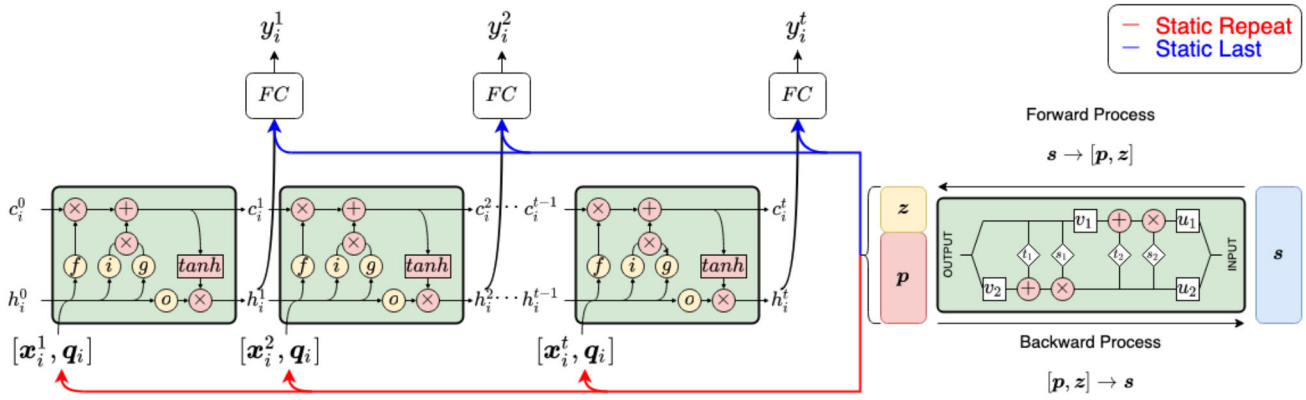


Figure 2: In Static-repeat IA-LSTM, the output of INN is concatenated with the input time series of LSTM at every time step, whereas in Static-last IA-LSTM the output of INN is concatenated with the output layer of LSTM (best seen in color).

where $\otimes$ denotes the element-wise product. According to the above equations, we can observe that the computation of $\boldsymbol{h}^t$ combines the information at current time step $\boldsymbol{x}^t$ and previous time step $\boldsymbol{h}^{t-1}$ and $\boldsymbol{c}^{t-1}$, and thus encodes the temporal patterns learned from data.

Invertible Neural Network (INNs) are bijective functions with a forward mapping $\mathscr{F} : \mathbb{R}^d \to \mathbb{R}^d$ and an inverse mapping $\mathscr{F}^{-1} : \mathbb{R}^d \to \mathbb{R}^d$. This inverse mapping can be computed in closed-form [23, 14, 15]. To create a fully invertible neural network, we follow real non-volume preserving architecture proposed by [14]. The basic unit of this network is a reversible bijective network consisting of two complementary affine coupling layers. In the forward process, the input is split into two parts . Hereby (equation 4.4), the input vector $\boldsymbol{u}$ is split into two halves $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$, which are transformed by an affine function with coefficients $\exp(s_i)$ and t_i $(i \in 1, 2)$, using element-wise multiplication $(\otimes)$ and addition. $s_i(\cdot)$ and $t_i(\cdot)$ are scale and translation functions, respectively, which are modeled by neural networks trained in the forward pass.

$$\begin{aligned} \boldsymbol{v}_1 &= \boldsymbol{u}_1 \otimes exp(s_2(\boldsymbol{u}_2)) + t_2(\boldsymbol{u}_2) \\ \boldsymbol{v}_2 &= \boldsymbol{u}_2 \otimes exp(s_1(\boldsymbol{v}_1)) + t_1(\boldsymbol{v}_1) \end{aligned} \tag{4.4}$$

Given the output $\boldsymbol{v} = [\boldsymbol{v}_1, \boldsymbol{v}_2]$, the above expressions are easily invertible as follows:

$$\begin{aligned} \boldsymbol{u}_2 &= (\boldsymbol{v}_2 - t_1(\boldsymbol{v}_1)) \otimes exp(-s_1(\boldsymbol{v}_1)) \\ \boldsymbol{u}_1 &= (\boldsymbol{v}_1 - t_2(\boldsymbol{u}_2)) \otimes exp(-s_2(\boldsymbol{u}_2)) \end{aligned} \tag{4.5}$$

Importantly, the mappings s_i and t_i can be arbitrarily complicated functions and need not themselves be invertible. In our implementation, we implement bijectivity by a succession of several fully connected layers with leaky ReLU activations. A deep INN is composed of a sequence of these reversible blocks. We also insert permutation layers between reversible blocks, which shuffle the elements of the subsequent layers's input in a randomized but fixed way. This causes the splits $[\boldsymbol{u}_1, \boldsymbol{u}_2]$ to vary between layers and enhances interaction among the individual variables. One advantage of INNs is that can we get the inverse $\mathscr{F}^{-1}$ for free once we train them on the well-understood forward process $\mathscr{F}$. This offers a unique opportunity for building a unified model that integrates both the forward and inverse modeling processes.

Invertibility-Aware-LSTM (IA-LSTM) is an integrated RNN-based model for modeling the lake temperature (y_i^t) using static characteristics $(\boldsymbol{s_i})$ and historic climate drivers $(X_i^{[t-w:t]})$, where w is the value of the historic window used. We use INN for modeling static information and LSTM for modeling time series data.

In particular, we wish to use INN to extract the information from static variables $\boldsymbol{s}$ that accounts for the data heterogeneity across multiple lakes and embed such information into a hidden representation $\boldsymbol{p}$. It is noteworthy that the representation $\boldsymbol{p}$ is intrinsically in a lower dimension compared to $\boldsymbol{s}$ due to the irrelevant and redundant information in static variables. This poses a challenge for training INN because it needs equivalent information between input and output so as to learn both the forward and inverse processes. To counteract the inherent information loss of the forward process [22], we introduce Gaussian latent variables $\boldsymbol{z}$ to augment the hidden representation $\boldsymbol{p}$, and use INN to learn a mapping $\boldsymbol{s} \to \boldsymbol{q}$, where the representation $\boldsymbol{q}$ is comprised of two parts $\boldsymbol{q} = [\boldsymbol{p}, \boldsymbol{z}]$. The training of the

forward process optimizes the mapping $\mathscr{F}(\boldsymbol{s}) = [\boldsymbol{p}, \boldsymbol{z}]$ and implicitly determines the inverse $\boldsymbol{s} = \mathscr{F}^{-1}([\boldsymbol{p}, \boldsymbol{z}])$.

To combine both models (INN and LSTM), we use two different approaches. In our first approach (Figure 2 – red line), the static information is first processed by INN and then concatenated with input time series at every time step. We call this approach as Static-repeat IA-LSTM. Our second approach (Figure 2 – blue line) also employs INN to process static information and then concatenates the INN output with the output layer of LSTM (i.e., the hidden representation) before using them for prediction. We denote the second approach as Static-last IA-LSTM. A fully-connected (FC) layer is used to map these features to lake temperature.

We train our method with two loss functions: one for predicting true outcome $\mathcal{L}_y$ and the other to encourage $\mathbf{z}$ to follow Gaussian distribution $\mathcal{L}_z$. Depending on the problem, $\mathcal{L}_y$ can be any supervised loss. In our case, we train our model with mean square loss. We implement $\mathcal{L}_z$ using Maximum Mean Discrepancy (MMD) loss [22] with the Inverse Multiquadratic kernel $k(x, \acute{x}) = \frac{1}{1+||(x-\acute{x})/c||_2^2}$, where $c > 0$ and only requires samples from the distributions to be compared. MMD enforces that $\boldsymbol{p}$ and $\boldsymbol{z}$ are independent upon convergence (i.e. $p(\boldsymbol{z}|\boldsymbol{p}) = p(\boldsymbol{z})$) and does not encode the same information twice. Since the magnitude of the MMD depends upon the kernel choice, the relative weight of the losses $\mathcal{L}_y$ and $\mathcal{L}_z$ are adjusted as hyperparameteres such that their effect is approximately equal. To increase model capacity [22], we also pad the input and output of the network with an equal number of zeros. Specifically, given the dimension of the static variables is small and the learning task for transformation is complex, we find it beneficial to pad zeros. This does not change the intrinsic dimensions of static variables and their representation but enables the IA-LSTM interior layers to embed the data into a larger representation space.

4.2 Inverse modeling with IA-LSTM Once IA-LSTM is trained, it can be easily applied to forward modeling (i.e., given static features and time-series climate drivers, it can predict lake temperatures). However, it is difficult to find accurate solutions for equation 3.1 because this is usually a non-convex problem due to the non-convexity of $\mathscr{G}(X, \boldsymbol{s})$. Existing solutions of inverse problems are often addressed using the Bayesian formulation:

$$p(\boldsymbol{s}|Y, X) \propto \frac{p(Y|X, \boldsymbol{s})p(\boldsymbol{s}|X)}{p(Y|X)}, \tag{4.6}$$

There are two challenges associated with this approach. First, the likelihood is computationally expensive since the inference of the static variables requires multiple forward model evaluations. Second, if the dimension of the quantity of interest is high, then obtaining posterior samples is not a trivial task. In this present work, we aim to bypass Bayes' rule and directly build an inverse surrogate model for the posterior distribution $p(\boldsymbol{s}|Y, X)$ using a finite set of forward evaluation of the time-series, static variables and corresponding temperature from various lakes.

Algorithm 1 Algorithm for inferring $\boldsymbol{s} \in \mathbb{R}^{D_s}$, given $\mathscr{G} = [\mathscr{G}_{LSTM}, \mathscr{G}_{INN}]$ and pairs of (X,Y)

procedure INFER($\boldsymbol{s}$) ▷ Infer $\boldsymbol{s}^\star \in \mathbb{R}^{D_s}$
  Initialize $\boldsymbol{q}$
  while NOT converged **do**
    $\mathcal{L} \leftarrow \left\| \mathcal{G}_{\mathcal{LSTM}}(X, \boldsymbol{q}) - Y \right\|$ ▷ Calculate error
    $\boldsymbol{q}^\star \leftarrow \boldsymbol{q}^\star - \alpha \nabla_{\boldsymbol{q}} \mathcal{L} - \|\boldsymbol{z}\|$ ▷ Gradient Descent
  end while
  $\boldsymbol{s}^\star = \mathcal{G}_{\mathcal{INN}}(\boldsymbol{q}^\star)$ ▷ Run INN backward
  return $\boldsymbol{s}^\star$
end procedure

We construct the inverse surrogate as follows: In the first step, an efficient IA-LSTM model $\mathscr{G}$ is built for the forward process. We denote the LSTM part of the model as $\mathscr{G}_{LSTM}$ and the INN part of the model as $\mathscr{G}_{INN}$. To obtain static characteristics, we need some observation pairs of (X, Y). In the second stage, we fix $\mathscr{G}_{LSTM}$ and we calculate $\boldsymbol{q} = [\boldsymbol{p}, \boldsymbol{z}]$ via gradient descent by taking the gradient of $\mathscr{G}_{LSTM}$ with respect to $\boldsymbol{q}$. This is detailed in Algorithm 1. As with any other inverse problem, the same pair of (X, Y) can map to several $\boldsymbol{q}^\star$ representations, particularly if the model is trained on limited data points. To reduce variance in the final result, we use prior knowledge of $\boldsymbol{z}$. Our Forward model $\mathscr{G}_{LSTM}$ is trained to predict Y from $[\boldsymbol{p}, \boldsymbol{z}]$, where the distribution of $\boldsymbol{z}$ is chosen as Gaussian. We utilize this information to add regularization term to the cost function, penalizing samples that have statistics that are not consistent with $\boldsymbol{z} \sim \mathcal{N}(0, 1)$. Lastly, after recovering $q^\star$, we use invertible property of INN and run $\mathscr{G}_{INN}$ in reverse mode according to equation 4.5 and recover the estimated values of the static variables $\boldsymbol{s}$. These static variables can then be used to model heterogeneity along with climate drivers X.

5 Experiments and Results

5.1 Dataset Description In this study, we used data from 450 lakes located in the Midwestern U.S., where in-situ temperature data was collected for each lake between 1980 and 2019. The dataset is available through a data release on the U.S. Geological Survey's

ScienceBase platform [5, 16]. All the lakes differed in depth, size, and location. The temperature measurements are highly variable for different depths in the lake. Lake temperature dynamics is complex, with lakes in this region freezing in winter and thawing in the spring, with many - but not all - lakes developing density-based stratification during the warmer seasons with warmer waters overlying colder deeper waters. The seasonal variation in water temperature can range from 0 in the winter to nearly 35 degrees C in the summer. As such, temperature measurements vary across depth and through time, making the prediction problem even harder.

Static features for lakes consist of water clarity (light attenuation coefficient), shoreline development factor (SDF), latitude, longitude, maximum depth, log transformed surface area, mean temperature, and percent of the time a General Lake Model simulation stratified for a given lake. In total we have 8 static features used in this study. Further explanation of these variables can be found in the corresponding data release [16]. The static features remain constant over time and depth. A lake's dynamic time-series consists of air temperature, shortwave radiation, longwave radiation, relative humidity, wind speed, rain, and snow. All-time-series data are inferred from meteorological datasets and thus remain constant across depth but changes across time t.

Out of 450 lake, 200 were used to build the model, and the rest are considered "artificially unmonitored", where data is only used for final evaluation. For 200 lakes, we partitioned the data into three contiguous time windows for training (70%), validation(10%), and testing(20%). We used earlier years for training and later years for testing. In total, we have 428,656 samples. We pool the training, validation, and test subsets of each lake to create a comprehensive set. Finally, we choose the model maximizing the accuracy of the broad validation set.

5.2 Learning Models In this section, we provide a brief discussion of the models used in our evaluation. For the models that combined static and dynamic features, we use them in two variants, i.e. *Static-repeat* [SR]: static variables or their representations are concatenated with climate drivers before they are fed into LSTM at every time step and *Static-last* [SL]: static variables or their representations are concatenated with hidden representation extracted by LSTM. We trained and tested the following model configurations:

LSTM[24]: LSTM model was trained using only the dynamic time-series input features and ignore the static variables. It has 64 memory units followed by a dense layer. The LSTMs were run in sequence-to-sequence mode and use the same configuration of LSTM in all the below models. **CT-LSTM**[25]: LSTM model trained using both the dynamic time-series input features and static variables. We refer to this model as concatenating LSTM. The system characteristics do not undergo any transformation and simply concatenated using static repeat and static last configurations. **FFN-LSTM**[9]: A feed-forward network (FFN) is used to learn the representation of static variables, and a LSTM model is used to model time-series input data. FFN has four dense layers with eight hidden units each. Both models are combined using static repeat and static last arrangements and trained together in an end-to-end manner. **IA-LSTM**: An Invertible Neural Network (INN) is used to learn the representation of static variables, and a LSTM model is used to model time-series input data. In our INN implementation, we incorporated three coupling layers with sixty-four hidden units each. Also, we pad eight zeros to eight static features, in total sixteen input features to embed the data. Consequently, we were able to increase model capacity without changing the intrinsic dimension of static features. Additionally, to counteract the information loss in the bidirectional process, we incorporated a two dimensional Gaussian latent variable $\boldsymbol{z}$.

We trained all models using mean square error for a maximum of 300 epochs using Adam optimizers. We terminated training if the validation loss did not reduce for 20 continuous epochs. The train, validation, and test set are kept consistent for all models to remove bias between different model runs.

	In-Distribution		Out-Distribution	
	Per Sample	*Mean*	*Per Sample*	*Mean*
LSTM	3.64 ± 0.15	3.42 ± 0.18	4.03 ± 0.22	3.92 ± 0.23
SL. CT-LSTM	2.51 ± 0.16	2.30 ± 0.21	2.97 ± 0.21	2.73 ± 0.28
SL. FFN-LSTM	2.48 ± 0.19	2.24 ± 0.12	2.83 ± 0.22	2.66 ± 0.23
SL. IA-LSTM	2.45 ± 0.08	2.21 ± 0.19	2.81 ± 0.22	2.59 ± 0.21
SR. CT-LSTM	2.42 ± 0.12	2.25 ± 0.14	2.49 ± 0.20	2.34 ± 0.13
SR. FFN-LSTM	1.85 ± 0.09	1.65 ± 0.06	2.48 ± 0.18	2.32 ± 0.15
SR. IA-LSTM	$\mathbf{1.78 \pm 0.09}$	$\mathbf{1.59 \pm 0.04}$	$\mathbf{2.42 \pm 0.12}$	$\mathbf{2.28 \pm 0.11}$

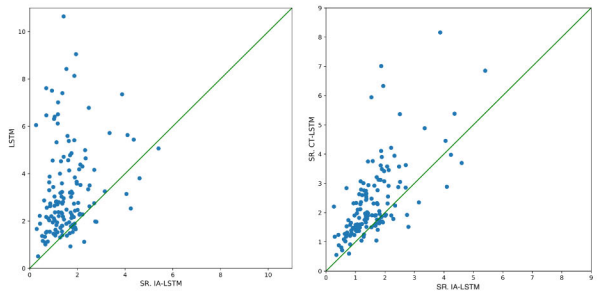
Table 1: Test root mean square error (RMSE; in $^{\circ}C$) when static features are provided for the various methods compared here. *Per sample* is the average RMSE for individual observation over multiple dates, multiple depths, and lakes. *Mean* is the mean of per-lake RMSE values of multiple lakes. *In-distribution* samples come from training lakes (years of test samples are different) and *out-distribution* samples are from lakes not in the training set.

5.3 Results

5.3.1 Forward Modeling
Table 1 reports the root mean square error (RMSE)

for each model when static variables are provided. We train all our models with 5 different random seeds to account for uncertainty in network initialization and report mean prediction along with variance. We refer *In-distribution* to the test samples which comes from the 200 lakes used in training (years of test samples are different than used in training) and refer *out-distribution* to the test samples from the lakes that were not seen during training (the remaining 250 lakes). The lake sizes vary, and subsequently, there are a different number of observations for each lake. Therefore, we report two metrics, the average RMSE for individual observation over multiple dates, multiple depths, and lakes (referred to as *Per sample*) and the mean of per-lake RMSE values of multiple lakes (referred to as *Mean*). We highlight best-performing methods for each column and make the following high level observations from our results: a) All trained models that combined static and dynamic time-series data had superior performance compared to LSTM, which was trained using only time-series data. This confirms the value of static features for handling heterogeneity and improving the prediction. b) All the models performed much better for in-distribution samples compared to out-distribution samples. This reinforces our earlier observation that data is highly heterogeneous, and it can be challenging to build a geoscientific global model for all lakes [26]. c) Models that concatenated the learned representation of static characteristics at the input layer performed better than the models where concatenation is done at the LSTM output layer. For example, IA-LSTM and FFN-LSTM performance improved by roughly 26% when the static variable representation was concatenated with the climate drivers before they are fed at every time step (i.e., SR vs SL, respectively). Our results differ from existing literature [9, 11], where authors report better performance using the SL approach.

More importantly, we observe that for static repeat (SR) and in-distribution data, both FFN-LSTM and IA-LSTM outperform CT-LSTM by a margin of 23% and 25%, respectively. This implies that static features transformation produces meaningful combinations of static features (e.g., the combination of lake surface area and lake depth indicating the volume of water) that can better capture the heterogeneity of different lakes. In particular, IA-LSTM outperformed plain LSTM by roughly 100%. Figure 3a show a scatter plot between the RMSE values of LSTM and IA-LSTM for individual lakes. We observe that most of the value lies in the left region of the diagonal line, suggesting that IA-LSTM did excellent on most of the lakes compared to LSTM. We also note that the difference of the RMSE is enormous in some of the lakes, for e.g. if we observe the



(a) (b)

Figure 3: (a) Comparison of root mean square error (RMSE) on each test lake from Long Short-Term Memory (LSTM; y-axis) and Invertibility-Aware-Long Short-Term Memory (IA-LSTM; x axis) predictions, (b) Comparison of RMSE on each test lake from concatenating LSTM (CT-LSTM; y-axis) and IA-LSTM (x axis) predictions.

top left region of the scatter plot, LSTM RMSE is 10, while IA-LSTM RMSE is 2. Likewise, Figure 3b shows a scatter plot between the RMSE values of SR CT LSTM and SR IA-LSTM. As with LSTM, we observe that most of the values lie on the diagonal's left, although not as far as they were for LSTM.

We also note that IA-LSTM outperformed FFN-LSTM with a smaller margin of 4%. Furthermore, we note that ours is the first attempt to show that transformation of static features can better capture heterogeneity of different lakes.

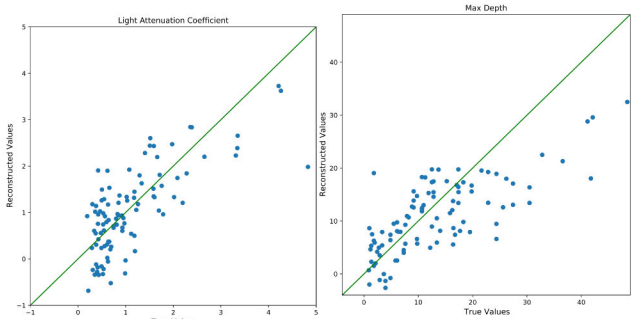
5.3.2 Inverse Modeling

This section studies how to carry forward the gain made by incorporating predicted static features for lakes where some or all of these features are missing.

We note that we need some observations (X, Y) for inverse modeling to determine the correct static features. To artificially create a case for "unmonitored" lakes, we hide static features and partition the data into two contiguous time windows for inference (10%) and testing (90%) (no temporal auto-correlation). We use observations from the inference period to recover static features and subsequently evaluate the performance of these values in the testing period. We fix the weights of the best-performing model from forward modeling and use gradient descent to recover the static input values for each lake. For CT-LSTM and FFN-LSTM, we recover the static variables directly. However, for IA-LSTM, we recover the LSTM input space according to lines 3-6 in Algorithm 1. After recovery, we run INN backward and get static variables instead of using gradient-based techniques. Figure 4 shows the scatter plot between the true values (x-axis) and reconstructed

Methods	*Per Sample*	*Mean*
LSTM	3.89 ± 0.22	3.80 ± 0.22
SR. CT-LSTM	3.17 ± 0.38	3.03 ± 0.39
SR. FFN-LSTM	4.25 ± 0.63	4.16 ± 0.65
SR. IA-LSTM	$\mathbf{2.80 \pm 0.40}$	$\mathbf{2.76 \pm 0.41}$

Table 2: Test root mean square error (RMSE; in $^{\circ}C$) when static features are missing for the various methods compared here. *Per sample* is the average RMSE for individual observation over multiple dates, multiple depths, and lakes. *Mean* is the mean of per-lake RMSE values of multiple lakes.



(a) (b)

Figure 4: Features reconstructed from the Invertibility-Aware-Long Short-Term Memory inverse modeling approach (y-axis) vs true feature values (x-axis) for light attenuation coefficient (a, in m^{-1}) and maximum lake depth (b, in m) for the 100 unmonitored lakes.

values (y-axis) for two static features, namely the light attenuation coefficient (a measure of water clarity) and maximum lake depth. For both features, the points on the scatter plot lie reasonably close to the diagonal, showing the closeness of the reconstructed values to the actual values. For lake depth, this accuracy of these reconstructed features is similar to alternative models where the primary objective was to predict maximum lake depth [27].

Next, we compare the performance of the models on test samples in the absence of static features. We note that gradient descent procedure is highly dependent on the prior (lines 2 in Algorithm 1). To account for stochasticity in the initialization, we ran the experiment 10 times with different seed values and selected the best run. We repeated the overall process five times for a total of 50 runs. Due to limitations in GPU resources, this experiment used 100 unmonitored lakes, and we reported mean RMSE in Table 2 for each model when static features were not available. We note that the LSTM method in Table 2 denote the performance without any static inference (just using (X) and is put for comparison purposes. We observe that when static characteristics are missing, IA-LSTM achieved lower RMSEs compared to the other approaches. FFN-LSTM, whose performance was second-best in forward modeling, fails badly, and CT-LSTM performs better than FFN-LSTM. We attribute this breakdown to the presence of an additional neural network in FFN-LSTM, i.e., linkage of two networks and the difficulty of gradient descent in optimizing inputs for deep networks as studied by Jayaram et al. [20]. We connect the success of IA-LSTM to an invertible neural network that learns accurate forward modeling of static characteristics and enables it to recover static variables using invertible architectures that provide the inverse mapping for free.

6 Discussion and Conclusion

In this study, we propose IA-LSTM, a unified forward/inverse framework to predict water temperature in lakes. This framework integrates static features of lake properties with the time series weather input for effectively capturing the heterogeneity across different lakes. Our method differs from prior work in combining static and time series data in that IA-LSTM can be employed for both forward and backward processes. This enables IA-LSTM to be used in many lake systems for which we do not have their measured static variables. To the best of our knowledge, this is the first attempt to develop a unified model with the aim to recover static features in situations where they are missing. Although we have shown improvements for lake temperature modeling, the method is general and can be applied to many machine learning tasks. In a clinical setting, it has been shown [11, 9] that integrating static characteristics (like patient demographics) greatly helps in the early detection of disease. Our approach can be used in these applications when static information is not available and thus serve as an alternative to approaches discussed in [10, 28], where the authors have devised methods to bring static details from other sources.

In some scientific applications, static features may be partially missing and the available static features could be different for different systems. Our method can be further extended to leverage the available static features from each system to improve the inference of missing features. Future work will also explore the development of invertible bijective functions for time-series data to avoid expensive gradient steps.

Acknowledgment

This work is supported by the NSF award 1934721 under the Harnessing the Data Revolution (HDR) programe. Additional support provided by Department of the Interior Midwest Climate Adaptation Science Center. Any use of trade, firm, or product names is for descriptive purposes only and does not imply

endorsement by the U.S. Government.

References

[1] K. Yang *et al.*, "Spatial-temporal variation of lake surface water temperature and its driving factors in yunnan-guizhou plateau," *WRR*, 2019.

[2] D.-P. Häder *et al.*, "Comparing the impacts of climate change on the responses and linkages between terrestrial and aquatic ecosystems," *Science of the Total Environment*, 2019.

[3] K. E. Havens *et al.*, "Temperature effects on body size of freshwater crustacean zooplankton from greenland to the tropics," *Hydrobiologia*, 2015.

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

[5] J. D. Willard *et al.*, "Predicting water temperature dynamics of unmonitored lakes with meta-transfer learning," *WRR*, 2021.

[6] A. Karpatne *et al.*, "Machine learning for the geosciences: Challenges and opportunities," *TKDE*, 2018.

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

[8] D. Makris *et al.*, "Combining lstm and feed forward neural networks for conditional rhythm composition," in *International conference on engineering applications of neural networks*. Springer, 2017.

[9] C. Lin, Y. Zhang, J. Ivy, M. Capan, R. Arnold, J. M. Huddleston, and M. Chi, "Early diagnosis and prediction of sepsis shock by combining static and dynamic information using convolutional-lstm," in *2018 IEEE International Conference on Healthcare Informatics (ICHI)*. IEEE, 2018, pp. 219–228.

[10] C. Esteban, O. Staeck, S. Baier, Y. Yang, and V. Tresp, "Predicting clinical events by combining static and dynamic information using recurrent neural networks," in *2016 IEEE International Conference on Healthcare Informatics (ICHI)*. IEEE, 2016, pp. 93–101.

[11] D. Li, P. Lyons, J. Klaus, B. Gage, M. Kollef, and C. Lu, "Integrating static and time-series data in deep recurrent models for oncology early warning systems," 2021.

[12] G. Ongie, A. Jalal, C. A. Metzler, R. G. Baraniuk, A. G. Dimakis, and R. Willett, "Deep learning techniques for inverse problems in imaging," *IEEE Journal on Selected Areas in Information Theory*, vol. 1, no. 1, pp. 39–56, 2020.

[13] O. Senouf, S. Vedula, T. Weiss, A. Bronstein, O. Michailovich, and M. Zibulevsky, "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*. Springer, 2019, pp. 111–119.

[14] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," *arXiv preprint arXiv:1605.08803*, 2016.

[15] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," *arXiv preprint arXiv:1807.03039*, 2018.

[16] J. D. Willard, J. S. Read, A. P. Appling, and S. K. Oliver, "Data release: Predicting water temperature dynamics of unmonitored lakes with meta transfer learning," U.S. Geological Survey - ScienceBase 2020, 10.5066/P9I00WFR.

[17] M. H. Rahman, S. Yuan, C. Xie, and Z. Sha, "Predicting human design decisions with deep recurrent neural network combining static and dynamic data," *Design Science*, vol. 6, 2020.

[18] K. Tayal, C.-H. Lai, V. Kumar, and J. Sun, "Inverse problems, deep learning, and symmetry breaking," *arXiv preprint arXiv:2003.09077*, 2020.

[19] S. Lan, T. Bui-Thanh, M. Christie, and M. Girolami, "Emulation of higher-order tensors in manifold monte carlo methods for bayesian inverse problems," *Journal of Computational Physics*, vol. 308, pp. 81–101, 2016.

[20] R. Jayaram, D. P. Woodruff, and Q. Zhang, "Span recovery for deep neural networks with applications to input obfuscation," *arXiv preprint arXiv:2002.08202*, 2020.

[21] G. A. Padmanabha and N. Zabaras, "Solving inverse problems using conditional invertible neural networks," *Journal of Computational Physics*, vol. 433, p. 110194, 2021.

[22] L. Ardizzone, J. Kruse, S. Wirkert, D. Rahner, E. W. Pellegrini, R. S. Klessen, L. Maier-Hein, C. Rother, and U. Köthe, "Analyzing inverse problems with invertible neural networks," *arXiv preprint arXiv:1808.04730*, 2018.

[23] L. Dinh, D. Krueger, and Y. Bengio, "Nice: Non-linear independent components estimation," *arXiv preprint arXiv:1410.8516*, 2014.

[24] I. Sutskever *et al.*, "Sequence to sequence learning with neural networks," in *NeurIPS*, 2014, pp. 3104–3112.

[25] A. Leontjeva and I. Kuzovkin, "Combining static and dynamic features for multivariate sequence classification," in *DSAA*. IEEE, 2016.

[26] W.-P. Tsai, D. Feng, M. Pan, H. Beck, K. Lawson, Y. Yang, J. Liu, and C. Shen, "From calibration to parameter learning: Harnessing the scaling effects of big data in geoscientific modeling," *Nature communications*, vol. 12, no. 1, pp. 1–13, 2021.

[27] S. K. Oliver, P. A. Soranno, C. E. Fergus, T. Wagner, L. A. Winslow, C. E. Scott, K. E. Webster, J. A. Downing, and E. H. Stanley, "Prediction of lake depth across a 17-state region in the united states," *Inland Waters*, vol. 6, no. 3, pp. 314–324, 2016.

[28] C. Xiao *et al.*, "Opportunities and challenges in developing deep learning models using electronic health records data: a systematic review," *Journal of the American Medical Informatics Association*, 2018.