IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 14, 2020, accepted April 24, 2020, date of publication April 28, 2020, date of current version May 11, 2020.

*Digital Object Identifier 10.1109/ACCESS.2020.2990939*

# DeepOcean: A General Deep Learning Framework for Spatio-Temporal Ocean Sensing Data Prediction

**YU GOU[1], TONG ZHANG[1], JUN LIU[2,3], LI WEI[4], AND JUN-HONG CUI[1,3], (Member, IEEE)**

[1]College of Computer Science and Technology, Jilin University, Changchun 130000, China
[2]College of Electronic Information Engineering, Beihang University, Beijing 100191, China
[3]Robotics Research Center, Peng Cheng Laboratory, Shenzhen 518066, China
[4]Department of Electrical and Computer Engineering, Michigan Technological University, Houghton, MI 49931, USA

Corresponding author: Jun Liu (liujun2019@buaa.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61631008, Grant 61971206, and Grant U1813217, in part by the Fundamental Research Funds for the Central Universities under Grant 2017TD-18, and in part by the National Key Basic Research Program under Grant 2018YFC1405800.

**ABSTRACT** The emerging Internet of Underwater Things (IoUT) and deep learning technologies are combined to provide a novel, intelligent, and efficient data processing and analyzing schema, which facilitates the sensing and computing abilities for the smart ocean. The underwater acoustic (UWA) communication network is an essential part of IoUT. The thermocline, in which temperature and density change drastically, affects the connectivity and communication performance between IoUT nodes, as well as the network topologies. In this paper, we propose DeepOcean, a deep learning framework for spatio-temporal ocean sensing data prediction, which consists of a generative module and a prediction module. We implement the generative module with a multi-layer perceptron (MLP) to capture the spatial dependencies and construct high-resolution data based on sparse observations. The prediction module is implemented with our proposed Multivariate Convolutional LSTM (MVC-LSTM) neural network, which captures both the spatio-temporal dependencies and the interactions of different oceanographic features for prediction. We evaluate the effectiveness of DeepOcean with Argo data, where the proposed framework outperforms fifteen state-of-art baselines in terms of accuracy.

**INDEX TERMS** Internet of Underwater Things (IoUT), deep learning, spatio-temporal prediction, multivariate convolutional LSTM (MVC-LSTM) neural network, thermocline.

## I. INTRODUCTION

The Internet of Underwater Things (IoUT) is the network of interconnected underwater systems, which is envisioned to facilitate a variety of applications, such as oceanographic data collection, pollution monitoring, offshore exploration, disaster prevention, assisted navigation, and tactical surveillance [1]. Due to the challenging nature of communication in the ocean environment, underwater acoustic (UWA) communication plays an essential role in the networking of various underwater systems in an IoUT. Since the speed of sound in an ocean environment is mainly affected by the temperature, salinity, and pressure, the distribution of these oceanographic features determines the attenuation, reflection, refraction, and scattering of UWA waveforms, which results in a complex convergence zone and shadow zone distribution of a UWA IoUT. The complex distribution of convergence zones and shadow zones determines the connectivity and communication performance between IoUT nodes, as well as the network topologies. Thus, learning the distribution and dynamics of these oceanographic features could be beneficial to the development of IoUT networking strategies.

According to the acoustic velocity, the ocean can be vertically divided into three layers: the surface layer (0 to ∼100m), the thermocline layer (∼100 to ∼600m), and the deep isothermal layer (600+m) [2]. The sound velocity increases with the depth in the surface layer, and then drastically decreases with the depth in the thermocline, and finally slowly increases with the depth in the isothermal layer [3]. Thus, as the refraction rate of acoustic waveform changes with the depth, and total reflection could happen when the acoustic wave is propagating at the boundary area between layers from a specific

The associate editor coordinating the review of this manuscript and approving it for publication was Hao Luo.

direction. For example, at the boundary area between the surface layer and thermocline layer, the acoustic waveform rays from a source IoUT node in a shallow water area could split into two folds. The rays with a larger pitch angle to the direction of gravity propagate along a convex curve until being reflected by the surface, and then it propagates within the surface layer and never be able to penetrate the thermocline layer. On the other hand, the rays with a smaller pitch angle to the direction of gravity propagate along a concave curve after penetrating the thermocline layer. Thus, there will be a shadow zone in the deep area around 5km away from the source, which significantly affects the topology of the IoUT, because IoUT nodes in the shadow zone will not be able to receive the UWA communication signal from the source node.

In the thermocline, temperature, salinity, and density change drastically [4], [5]. Defined by temperature gradients, the thermocline changes with geographic location (longitude, latitude), depth, and time. The design of routing and media access control strategy in underwater sensor networks can benefit a lot from the precise prediction of thermocline distributions [6]. We formulated the prediction of thermocline as a regression problem.

Existing prediction models mainly include data-driven statistical models and deep learning-based models. Mathematical and statistical models, including Kalman Filters, Support Vector Regression, and K-Nearest Neighbors, are commonly applied for prediction. These methods have better performance in prediction and ideal time complexity. However, these methods cannot capture the spatial dependencies and the evolution of the dependencies on the temporal domain simultaneously [7].

In recent years, deep learning techniques have made significant achievements in areas such as natural language processing, which encourages researchers to apply deep learning techniques to spatio-temporal prediction problems. With the development of sensing technologies, the massive volume of observations is available for model training. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are the two most popular techniques among deep learning techniques [8]. In [9], CNN is applied to traffic speed prediction problems for the capabilities of extracting abstract spatial features predicting future states. RNNs are utilized to learn the temporal dependencies for predicting the traffic speed of a single location from historical traffic time series [10]. However, CNNs and RNNs consider either the spatial dependencies or temporal dependencies, so they do not apply to spatio-temporal prediction problems. In [11], the authors propose a graph-based RNN (SRNN) for short-term traffic prediction. By incorporating feature vectors containing spatial dependencies into the RNN sequence, all the spatial and temporal dependencies are jointly learned. Zhang *et al.* propose a deep spatio-temporal residual network (ST-ResNet) in [12] to predict the human flow in the area. They summarized the spatio-temporal dependencies of regional traffic and modeled each feature using a deep residual network. In [13], the authors propose a scalable graph convolutional deep learning architecture (GCDLA) to forecast the wind-speed time series of the whole graph nodes. By modeling the wind farms as an undirected graph and using LSTM, the proposed architecture captures both temporal and spatial features. He *et al.* proposed STCNN, which employees a general encoder-decoder architecture based on ConvLSTM units [8]. The encoder is composed of ConvLSTM and Skip-ConvLSTM, which explores the global spatio-temporal dependencies. The decoder utilizes the learned spatio-temporal hidden states from the encoder to make spatial and temporal predictions.

These methods have excellent performance in spatio-temporal predictions, but most of them are designed for 2-D terrestrial scenes and do not apply to 3-D ocean scenes composed of longitude, latitude, and depth. Besides, the sparsity that arises from the difference between terrestrial IoT and IoUT of raw observations is ignored. In [14], the authors consider the 3-D structure of the ocean and propose a model of multi-layer ConvLSTM (M-convLSTM) to predict the ocean temperature. However, the authors also failed to consider the impact of data sparseness on marine-related applications. We can learn from the encoder-decoder structure to solve the problem of data sparsity in ocean time-series predictions by capturing the data dependencies [15].

In this paper, we propose DeepOcean, a deep learning framework for predicting oceanographic feature distributions, which consists of a generative module and a prediction module. The generative module is implemented with a multi-layer perceptron (MLP), which learns spatial dependencies and constructs high-resolution datasets based on sparse observations. The prediction module is implemented with our proposed Multivariate Convolutional LSTM (MVC-LSTM) neural network. By stacking multivariate spatio-temporal observations into fixed-dimensional representations and coupling ConvLSTMs and Conv3Ds into a single framework, MVC-LSTM further captures the hidden correlation among different features. Besides, MVC-LSTM also maintains the consistency of the input and output forms, avoiding the loss of edge information.

The effectiveness of DeepOcean is demonstrated using one representative and challenging task: thermocline prediction with raw observations. We choose the task because 1) the prediction of the thermocline is a spatio-temporal task, whose position and shape differ according to geographical location (longitude, latitude), depth, and time; 2) historical observations are generally sparse while the predictions require inputs of high-resolution, which can reflect the data sparseness of the IoUTs. We predict the accordingly temperature profiles to infer the position of thermoclines. This task, therefore, illustrates the capacity of DeepOcean to learn hidden dependencies and predict future states.

The task is evaluated with collected data or existing datasets. We compare DeepOcean to state-of-the-art

algorithms that perform the time-series predicting tasks. Experimental results reveal that DeepOcean outperforms other methods in terms of accuracy.

The main contributions are as follows:

1) Propose DeepOcean, a general time-series predicting framework that accommodates a wide range of applications based on raw, sparse historical observations.
2) Implement the generative module with a multi-layer perceptron to model both nearby and distant spatial dependencies of historical observations to build high-resolution datasets for the reason that IoUTs are by nature sparse network structures.
3) Implement the prediction module with our proposed Multivariate Convolutional LSTM (MVC-LSTM) neural network and creatively stack the observation sequences of different variables into a multivariate matrix, capturing the hidden dependencies including spatio-temporal dependencies and interactions of different features, to predict future states and changing trends. MVC-LSTM keeps all the spatial information and interactions between multivariate observations throughout the predictions.
4) Evaluate DeepOcean using Argo data. The results reveal the effectiveness of DeepOcean compared with fifteen state-of-art baselines.

The remainder of this paper is organized as follows: In Section II, we introduce related work on Recurrent Neural Networks and time-series predicting researches. We describe the proposed DeepOcean in Section III. The evaluation and discussions are presented in Section IV and section V. Finally, we conclude in Section VI.

## II. RELATED WORK

The prediction of ocean sensing data can be formulated as a regression problem. There are several conventional methods, including linear regression, logistic regression, ridge regression, and support vector regression. Jiang *et al.* exploit SVR for regressive predictive analysis [16]. Gou *et al.* apply the KNN regression algorithm to predict ocean temperature and salinity [17]. Most of these methods do not consider the spatio-temporal dependencies of the ocean, or they only consider the temporal variability of a specific position but ignore the dynamic spatial correlations.

Recently, deep learning has become one of the most popular technologies in time-series prediction tasks, such as traffic flow prediction, citywide crowd flows prediction, and weather forecasting [12], [18], [19]. Recurrent neural networks with Long Short Term Memory (LSTM) architecture have been successfully applied to various supervised sequence learning tasks [20]. Sutskever *et al.* presented a general end-to-end approach to sequence learning that makes minimal assumptions on the sequence structure [21]. Based on LSTM, Xingjian *et al.* proposed the convolutional LSTM (ConvLSTM) and used it to build an end-to-end trainable model for the precipitation nowcasting problem [22]. Sagheer and his colleagues proposed a deep learning approach capable of addressing the limitations of traditional forecasting approaches and showing accurate predictions. The proposed approach is a deep long-short term memory (DLSTM) architecture as an extension of the traditional recurrent neural network. The genetic algorithm is applied in order to configure DLSTM's optimum architecture [23] optimally. In [24], Althelaya *et al.* studied the integration of deep learning methodologies into stock market forecasting. They evaluated and compared several variants of Deep Recurrent Neural Network based on LSTM and GRU. Ghaderi and some other researchers proposed a framework to model the spatio-temporal information by a graph whose nodes are data-generating entities and its edges model how these nodes are interacting with each other [25].

Recurrent neural networks (RNNs) and LSTM architectures are employed to process sequential data. Deep learning techniques are widely utilized to predict seawater temperatures. Zhang *et al.* (2017) take sea surface temperature prediction as a time-series prediction problem. They adopt the LSTM architecture to predict sea surface temperature (SST) [26]. The method proposed by Yang *et al.* includes a fully connected LSTM layer (FC-LSTM) and a convolutional neural network layer. They combined both spatial and temporal information to improve the prediction performances [27]. This paper proposes a general deep learning framework for time-series predictions, which considers both the spatio-temporal dependencies and the interactions of different oceanographic features. As a result, it improves the prediction accuracy.

Many researchers are working on the thermocline recently. Peng *et al.* researched the responsibilities of the thermocline depth to the El Niño-Southern Oscillation (ENSO) events [28]. They found that the response of the thermocline depth in the South China Sea to the ENSO events is mainly caused by the sea surface buoyancy flux and the wind stress curl. Jiang *et al.* calculated the depths of the upper thermocline boundaries in the South China Sea based on the sea temperature profiles of China Ocean Reanalysis from 1986 to 2008. Seasonal variation characteristics of the thermocline are also revealed in their paper [29].

To the best of our knowledge, DeepOcean is the first unified time-series prediction framework that considers the specialty of ocean observations, including the spatio-temporal dependencies, the internal relationships between oceanic features(such as temperature and salinity), and the sparsity of the observations. Moreover, it is the first research focused on predicting the position of the thermocline.

## III. FRAMEWORK

In this section, we introduce DeepOcean, a general deep learning framework for ocean time-series sensing data prediction. Our proposed DeepOcean, as shown in Fig. 1, mainly consists of a generative module and a prediction module. The generative module is the same for all applications, which learns spatial dependencies and constructs high-resolution datasets based on sparse multivariate

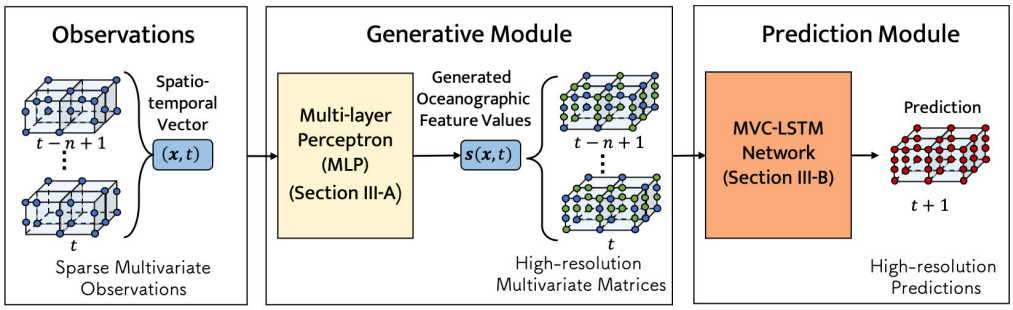



**FIGURE 1. Main architecture of DeepOcean. The generative module is implemented with a multi-layer perceptron (MLP), which consists of an input layer, an output layer, and *n* hidden layers with hidden neurons. The prediction module is implemented with our proposed Multivariate Convolutional LSTM (MVC-LSTM) neural network, including the input layer, ConvLSTM layers, Batch Normalization layers, Conv3D layer, and output layer.**

observations. The prediction module further captures both the spatio-temporal dependencies and the interactions of different oceanographic features for predictions based on the previously constructed high-resolution datasets.

### A. GENERATIVE MODULE

The generative module is implemented with a multi-layer perceptron (MLP), which is useful when there are (unknown) relationships between the inputs and the outputs. Fig. 2 demonstrates the structure of the MLP and algorithm 1 shows how we construct high-resolution datasets using MLP. MLP consists of several neurons, which are clustered into an input layer, an output layer, and $n$ hidden layers with $p$ hidden neurons.

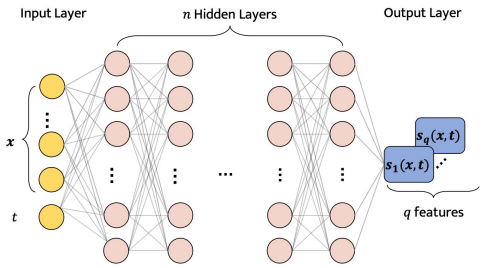



**FIGURE 2. Architecture of the generative module.**

In the generative module, we use $\boldsymbol{A}$ to denote the matrix of historical observations, which is further divided into a spatio-temporal feature submatrix $\boldsymbol{X}$ and a variable submatrix $\boldsymbol{S}$. Let $(\boldsymbol{x}, t) \in \boldsymbol{X}$ denotes a piece of space-time coordinate in $\boldsymbol{X}$, which contains spatio-temporal information and uniquely identifies an oceanographic feature vector $s(\boldsymbol{x}, t) \in \boldsymbol{S}$. Equation 1 denotes the relationship of $\boldsymbol{A}, \boldsymbol{X}$, and $\boldsymbol{S}$.

$$\boldsymbol{A} = [\boldsymbol{X}; \boldsymbol{S}] \tag{1}$$

The input matrix $\boldsymbol{X}$ to the generative module consists of $n_{samples} \times p$ elements, where $n_{samples}$ is the number of observations and $p$ represents the number of spatio-temporal features in one piece of observation. The variable submatrix $\boldsymbol{S}$ contains $n_{samples} \times q$ elements, where $q$ is the number of oceanographic features in feature vector $\boldsymbol{s}(\boldsymbol{x}, t)$. MLP learns non-linear transformation functions for each of these $p$ features to represent the spatio-temporal dependencies from input to output.

Let set $F$ denotes all non-linear transformation functions learned from input $\boldsymbol{X}$ to each vector $\boldsymbol{s_i}$ in matrix $\boldsymbol{S}$, and $\hat{f}_i$ denotes the optimal non-linear transformation function in set $F$ for feature $\boldsymbol{s_i}$. The hidden layers and neurons are trained according to these $p$ input coordinates and their corresponding output variables. We hope that the non-linear transformation function $f_i(\cdot)$: $R^p \rightarrow R$ can better represent the spatial dependencies from inputs to outputs. Through the optimal model $\hat{f}_i$ we can get a predicted output $\hat{y}$, corresponding to a piece of observation $y \in \boldsymbol{s_i}$. We use $D_H$ to denote the high-resolution dataset constructed by the generative module, which contains $n_{highResolution} \times (p + q)$ elements in total. Similarly, $n_{highResolution}$ is the number of samples in a high-resolution dataset. Noting that the generative module constructs high-resolution datasets for all variables in $V$ using MLP, which will be used in data prediction at the prediction module.

In this paper, the space-time coordinate $(\boldsymbol{x}, t) = (x_{longitude}, x_{latitude}, x_{depth}, t_{order})$, where $x_{longitude}$, $x_{latitude}$, $x_{depth}$, and $t_{order}$ represent the longitude, latitude, depth, and order of the variables to be predicted, respectively. The corresponding variable set is denoted as $\boldsymbol{s}(\boldsymbol{x}, t) = \{s_{temperature}, s_{salinity}\}$. Thus the number of input features $p$ of MLP is 4, and the number $q$ is 2. Noting that the $t_{order}$ refers to the month order starting from January 2004.

Like most neural network algorithms, more hidden layers and neurons mean better learning performance. However, more hidden layers and neurons also increase the possibility of overfitting. We adopted $L2$ regularization as a penalty and added $\theta = \frac{1}{2}\|w\|^2$ to the objective function to reduce the generalization error and the risk of overfitting.

### B. PREDICTION MODULE

In order to have more accurate and robust prediction results, we propose the Multivariate Convolutional LSTM (MVC-LSTM) neural network to implement the prediction module. As demonstrated in Fig. 3, our MVC-LSTM

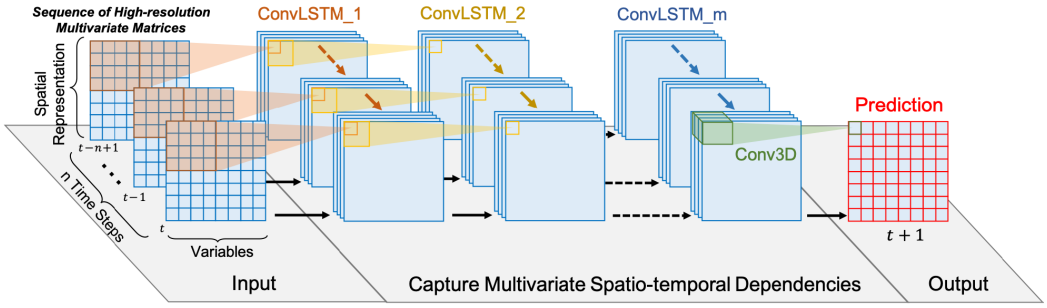



**FIGURE 3.** The structure of the proposed Multivariate Convolutional LSTM (MVC-LSTM) neural network.

can be generally divided into four components: input layer, ConvLSTM layer, Conv3D layer, and output layer. By stacking multivariate spatio-temporal observations into fixed-dimensional representations and coupling ConvLSTMs and Conv3Ds into a single framework, MVC-LSTM captures not only the spatio-temporal dependencies but also the hidden correlation among different features. We customize the filter number and size of the ConvLSTM layer and the Conv3D layer according to specific applications and different inputs.

In the leftmost input layer, sequences of specified variables at a position in the generated high-resolution dataset $D_H$ are stacked into $\boldsymbol{A_H}$, which is a matrix of $m$ variables at $n$ depths and contains $m \times n$ elements. $a_{ij} \in A_H$ is the value of the $i-th$ variable at the $j-th$ depth. Comparing $a_{ij}$ to a pixel in a single-channel image, we can view the matrix as a three-dimensional tensor that has only one channel. The adopted time step length and the number of variables jointly decided the numbers and size of the tensors. For a thermocline prediction task in this paper, the matrix $\boldsymbol{A_H}$ contains high-resolution sequences of temperature and salinity at the same longitude and latitude, while the depths of different samples are different.

Besides are the ConvLSTM layers. Batch normalization is applied at each layer to reduce the internal covariate shift [30]. Convolutional LSTMs (ConvLSTM) are the main components of our proposed MVC-LSTM. ConvLSTM has convolutional structures in both the input-to-state and state-to-state [22] connections. We can regard all the inputs $\mathcal{S}_1, \ldots, \mathcal{S}_t$, cell outputs $\mathcal{C}_1, \ldots, \mathcal{C}_t$, hidden states $\mathcal{H}_1, \ldots, \mathcal{H}_t$, and gates $i_t, f_t, o_t$ of the ConvLSTM as 3D tensors whose first two dimensions are spatial and variable dimensions, the last dimension is the temporal dimension. The outputs of ConvLSTM cells depend on the inputs and actual states of local neighbors. The key equations of ConvLSTM are as follows, where '$*$' denotes the convolution operator, '$\circ$' denotes the Hadamard product, and '$\sigma(\cdot)$' denotes the logistic sigmoid function.

$$i_t = \sigma(W_{xi} * \mathcal{S}_t + W_{hi} * \mathcal{H}_{t-1} + W_{ci} \circ \mathcal{C}_{t-1} + b_i) \quad (2)$$

$$f_t = \sigma(W_{sf} * \mathcal{S}_t + W_{hf} * \mathcal{H}_{t-1} + W_{cf} \circ \mathcal{C}_{t-1} + b_f) \quad (3)$$

$$\mathcal{C}_t = f_t \circ \mathcal{C}_{t-1} + i_t \circ \tanh(W_{sc} * \mathcal{S}_t + W_{hc} * \mathcal{H}_{t-1} + b_c) \quad (4)$$

$$\mathcal{H}_t = f_t \bigodot m_{t-1} + i_t \bigodot c_t \quad (5)$$

$$h_t = o_t \circ \tanh(\mathcal{C}_t) \quad (6)$$

We use convolution filters of different sizes to capture the interactions among variables and spatio-temporal dependencies at different scales (spatial and temporal). A larger filter can learn the changing trends of a larger area and in a more extended period. Meanwhile, a larger convolution filter means that more information about the interaction between variables can be learned each time. While the smaller filters mainly capture the closeness in both spatial and temporal dimensions. The network depth, filter number, and filter sizes are customized in different applications — the customizable representative ability of our MVC-LSTM better prediction accuracy.

After multiple convolution operations, the input size becomes smaller and smaller. Besides, the marginal data have fewer impacts on the output than those central data, because the convolution operation ends when it moves to the edge. The data in the center will participate in multiple operations, but the data at the edge may only participate in one operation, which leads to the loss of edge information. To ensure the output of the $i-th$ layer, that is, the input of the $(i+1)-th$ layer, has the same size as the input tensor, and retain the edge information, padding is employed before applying the convolution operation, which allows the filter to go outside the border of its input [12], padding each area outside the border with a zero.

MVC-LSTM structure also includes a 3D convolution component (Conv3D). The Conv3D layer takes the multivariate spatiotemporal features learned by the ConvLSTM layer as input, and further extracts more global spatio-temporal relationships between different features. Besides, the 3D convolution transforms the number of output channels and maps the prediction results to an output space that has the same shape as the input.

The output layer returns the prediction result of 3D convolution, which has the same size as input tensor. MVC-LSTM keeps all the spatial information and interactions between multivariate observations throughout the predictions. By stacking multiple ConvLSTM and Conv3D layers, the entire structure has strong abilities in representing

spatio-temporal dependencies and interactions of various features. Therefore, our proposed MVC-LSTM has good performances in complex spatio-temporal data predictions. Section IV-C gives the empirical evaluations and discussions.

## IV. EXPERIMENTS

In this section, we first introduce the datasets used in the thermocline prediction tasks. Subsequently, the method of how to select the optimal model is introduced. Finally, we compare DeepOcean with several baseline algorithms in the thermocline predicting tasks. The experimental results show that the proposed DeepOcean architecture outperforms the state-of-art time-series prediction methods in this task.

Experiments are mainly run on a single GTX 1080Ti GPU. We implement our models in Python with the help of TensorFlow and Keras libraries.

The proposed DeepOcean framework is implemented using TensorFlow and Keras with GPU acceleration to speed up the training process. The models run on a single-GPU computer system with an NVIDIA GTX 1080Ti GPU and a 3.5 GHz CPU.

### A. DATASETS

The data used in this experiment is from the Global Ocean Argo Grid Data Set (BOA_Argo) [31]. The grid dataset provides monthly average temperature and salinity data from January 2004 to December 2017 covering the global ocean (180°W–180°E, 80°S–80°N). The spatial resolution is $1° \times 1°$ horizontally and is unevenly divided into 58 standard layers from 0–1975m in vertical.

The experimental area (165.5°$E$–179.5°E, 0.5 °N–9.5°N, 0–500 m), locates at the tropics and is suitable for the thermocline research and analyzation, is selected to speed up the training of DeepOcean. That is, the experimental area is a 15(Longitude)×10(Latitude)×35(Depth) space-time grid. In this area, the large temperature difference between the sea surface and the deep sea leads to an evident thermocline phenomenon. Each sample contains features of longitude, latitude, depth, acquisition time, temperature, and salinity. There are 35 raw historical observations from 0-500 m. By the generative module, we construct a high-resolution dataset of the 1-meter interval from 0 to 500 m based on the raw, unevenly distributed 35 samples, including the raw observations.

### B. GENERATIVE MODULE PERFORMANCES ANALYSIS

#### 1) TRAINING AND VALIDATION DATASETS

We select the optimal model for each algorithm by cross-validation. In this paper, we randomly divide the historical observations into training set and test set at a ratio of 6: 4. We further divide the training set into five mutually exclusive subsets of similar size to minimize the structural risk and prevent overfitting. Each time we select a different subset as validation subset and the other four sets as training subsets, which provide five groups of training and validation sets. The final result of the *cross-validation* is the mean of these five evaluation values($R^2$) for each model. The parameters that make the optimal model are selected based on the means. At last, we use the test set to evaluate the optimal model of each algorithm for comparison.

#### 2) GENERATIVE MODULE CONFIGURATIONS

In the evaluations, the number of hidden layers is set to five, and each layer has 200 neurons. The Rectified Linear Unit (ReLU) is employed as the activation function of all neurons. The Adam optimizer, with learning rate equals to 0.001, is used for gradient descent learning. The batch size is set to 200, and the model stops training when the loss is not improving by at least ten iterations or at maximum iterations of 200.

The inputs to MLP contain spatio-temporal information, including longitude, latitude, depth, and order as demonstrated in section III-A. The outputs are the predictions of corresponding oceanographic feature values. Generative module trains models for both temperature and salinity, respectively. By applying algorithm 1 to the models, the previous 15 (longitude)×10 (latitude)×35 (depth)×168 (time) sparse grid data is constructed into a 15 × 10×501 × 168 high-resolution datasets.

**Algorithm 1** Constructing High-Resolution Datasets Using Generative Module

**Require:** Historical observations: $A$, $X$ and $S$; optimal generative model $\hat{f}_i \in F$ for variable $\boldsymbol{s_i} \in \boldsymbol{S}$; raw depth $o_r^i \in O_r$; high-resolution depth $o_h^j \in O_h$;corresponding spatial information: $x_{lon}^i$, $x_{lat}^i$, and temporal information $t^i$.

**Ensure:** High-resolution datasets $D_H$.

$O \leftarrow O_h - O_r$;
**repeat**Randomly select a variable $\boldsymbol{s_i}$ from $\boldsymbol{S}$;
  **for** $i = 1 \rightarrow n$ **do**
    $h^i = f(x_{lon}^i, x_{lat}^i, o_r^i, t^i)$;
    Put $(\{x_{lon}^i, x_{lat}^i, o_r^i, t^i\}, h^i)$ into $D_H^{s_i}$;
    $D_H = D_H \cup D_H^{s_i}$;
  **end for**
  Remove $\boldsymbol{s_i}$ from $\boldsymbol{S}$
**until** $\boldsymbol{S} = \emptyset$
$D_H = D_H \cup A$;
**return** $D_H$ for all variables;

#### 3) EVALUATION METRICS

We employ Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and coefficient of determination $R^2$ as the performance metrics to evaluate the models.

$$\mathrm{MAE}(s, \hat{s}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} \left| s_i - \hat{s}_i \right| \tag{7}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n_{samples}} (s_i - \hat{s}_i)^2}{n_{samples}}} \tag{8}$$

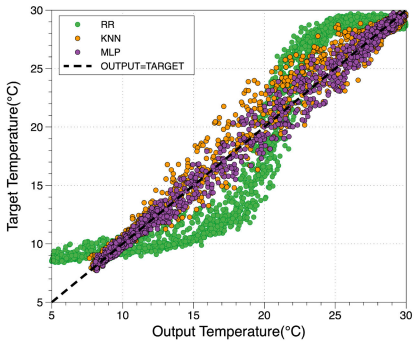
$$R^2(s,\hat{s}) = 1 - \frac{\sum_{i=1}^{n_{samples}}(s_i - \hat{s}_i)^2}{\sum_{i=1}^{n_{samples}}(s_i - \bar{s})^2} \tag{9}$$

where $s_i$ and $\hat{s}_i$ denote the real value and generated value of the $i-th$ sample, while $\bar{s}$ is the mean value of all samples.

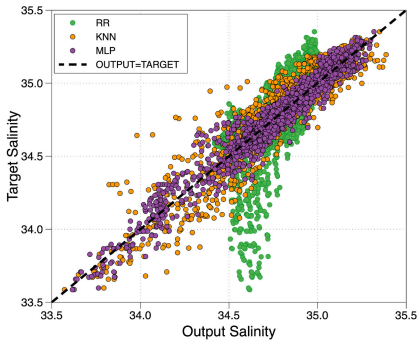
#### 4) COMPARISON WITH BASELINES

Here we compare the performance of the proposed MLP generative module with two other baseline methods, namely Ridge Regression (RR) and K-Nearest Neighbors (KNN).

- **Ridge Regression(RR)**: Ridge Regression is a technique for analyzing multiple regression data that suffer from multicollinearity.
- **K-Nearest Neighbors(KNN)**: Regression-based k-nearest neighbors (KNN) method predicts the target according to the $k$ nearest neighbors. By cross-validation, the parameter $k$ for temperature and salinity analyzations are 8 and 6, respectively.



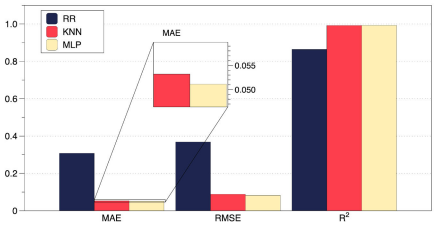

(a) Temperature(°C)



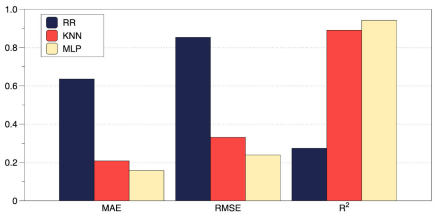

(b) Salinity

**FIGURE 4.** Comparisons of predicted values and real values.

Fig. 4(a) and Fig. 4(b) show comparisons of the real values (Target) and predicted values (Output) obtained through these three methods. The $x$-axis and $y$-axis represents predicted temperature and salinity, and real temperature and salinity, respectively. The black dotted line represents the best case where the predicted value is the same as the real value. The points in the figure are the values predicted through different methods. The distance between the point and the black dotted line represents the magnitude of the prediction error. We can observe that KNN and MLP are better fitted to the rules of data variation. The linear regression method RR failed to fit the data accurately, and the predicted data results deviated significantly from the real values.




(a) Temperature




(b) Salinity

**FIGURE 5.** Performances of our generative module and the baselines. A better model should have lower MAE and RMSE, while the $R^2$ is closer to 1.

**TABLE 1.** Evaluations of different methods in predicting temperature.

| Method | MAE | RMSE | $R^2$ |
|---|---|---|---|
| RR | 0.3073 | 0.3686 | 0.8642 |
| KNN | 0.0532 | 0.0883 | 0.9922 |
| MLP | 0.0511 | 0.0825 | 0.9932 |

**TABLE 2.** Evaluations of different methods in predicting salinity.

| Method | MAE | RMSE | $R^2$ |
|---|---|---|---|
| RR | 0.6355 | 0.8533 | 0.2738 |
| KNN | 0.2082 | 0.3312 | 0.8906 |
| MLP | 0.1577 | 0.2394 | 0.9429 |

Fig. 5(a) and Fig. 5(b) show how different algorithms perform in predicting temperature data and salinity data, respectively. Table 1 and 2 show the detail evaluation results. In order to ensure the fairness of the experimental results, we select the parameters through cross-validation before comparing the algorithms to construct models that can reflect the optimal performances of the algorithms. The models used for comparison can reflect the optimal performance of the algorithm. Fig. 5(a) is a comparison of spatial predicting abilities for temperature data. Compared with the other three algorithms, DeepOcean achieves the best performance (MAE: 0.0511, RMSE: 0.0825, and $R^2$: 0.9932). Fig. 5(b) is a comparison of spatial predicting abilities for salinity data. Similar to the results of the temperature prediction task, MLP and KNN achieved better performance than the other two algorithms in the salinity predicting tasks. At the same time, DeepOcean still achieves the best performance

(MAE: 0.1577, RMSE: 0.2394, and $R^2$: 0.9429). The comparison results reinforce the effectiveness of our generative module in the DeepOcean architecture.

### C. PREDICTION MODULE PERFORMANCES ANALYSIS

#### 1) GENERATING HIGH-RESOLUTION INPUT SEQUENCES

The input sequence to the prediction module consists of the high-resolution multivariate matrix $A_H$ of different time step length at the same longitude and latitude, and different depth. The selected data is from the depth of $0 \sim 300$ m. Each single data sample has the size of $301(depth) \times 2(temperature, salinity) \times n(time\ step\ length)$. The time interval between adjacent time steps is a month. From 2004 to 2017, the constructed high-resolution data sequence has a size of $301 \times 2 \times 168$, and the 301 high-resolution values are generated from 26 raw observations by our generative module.

Different time step length affects the prediction performance of the model. We construct data sequences with different time step lengths. When setting time step length to $n$, the size of the data sequence from time $t$ to $t - n + 1$ is $301 \times 2 \times n$, and the size of the predicted value at time $t+1$ is $301 \times 2 \times 1$. We stack $n+1$ matrix into a sequence as a group of sample, and its size is $301 \times 2 \times (n+1)$. In a slide-window manner, $(168-n+1)$ samples with the size of $301 \times 2 \times (n+1)$ can be obtained from a $301 \times 2 \times 168$ high-resolution data sequence in total. In the following experiments, we use the last 36 samples as the test sets each time.

#### 2) PREDICTION MODULE CONFIGURATIONS

There is an input layer, two ConvLSTM layers, two Batch Normalization (BN) layers, a Conv3D layer, and an output layer in our proposed MVC-LSTM model. Settings of the filter size, filter number, and network depth will be discussed in section IV-C.5. Conv3D layer has the filter size of $2 \times 2 \times 2$. Batch Normalization (called BN) layers normalize the activations of the previous layer at each batch and accelerating network training [32]. In the training process, the momentum of BN is set to 0.99. Adam optimizer, which has a learning rate of 0.001, is used for gradient descent learning. The batch size is set to 32, and the model is trained in epochs of 600.

#### 3) EVALUATION METRIC

We measure the proposed method and the baselines by Root Mean Square Error (RMSE) as shown in equation 8.

#### 4) COMPARISON WITH BASELINES

In this section, we predict the temperature at time $t$ based on previously constructed high-resolution sequences. Table 3 shows the RMSE of all baselines and our proposed MVC-LSTM. Intuitively, we display the evaluations of these methods and the time steps they need in Fig. 6, respectively. Note that we choose different time step lengths for different methods for comparison because different model attains its best performance with the lowest RMSE at different time step length.

**TABLE 3. Comparisons with baselines on historical observations.**

| Model | Optimal Time Step Length | RMSE |
|---|---|---|
| RNN | 2 | 0.0264 |
| GRU | 2 | 0.0234 |
| LSTM | 2 | 0.0251 |
| DRNN | 2 | 0.0239 |
| DGRU | 6 | 0.0209 |
| DLSTM | 9 | 0.0205 |
| B-RNN | 4 | 0.0204 |
| B-GRU | 5 | 0.0205 |
| B-LSTM | 7 | 0.0247 |
| M-RNN | 4 | 0.0327 |
| M-GRU | 2 | 0.0249 |
| M-LSTM | 3 | 0.0202 |
| M-DRNN | 4 | 0.0310 |
| M-DGRU | 11 | 0.0216 |
| M-DLSTM | 9 | 0.0275 |
| MVC-LSTM | 2 | **0.0079** |

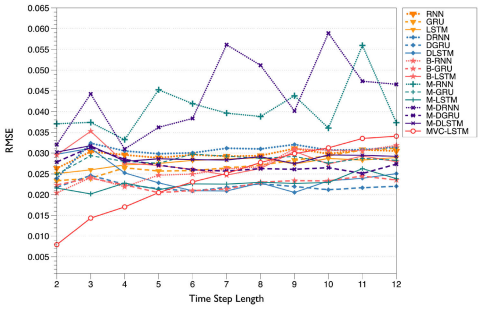



**FIGURE 6. Comparison of MVC-LSTM and other baselines with different time step length using RMSE. The smaller the better.**

The lines in Fig. 6 represent the 15 baselines and our MVC-LSTM. The $x-axis$ denotes the length of the time step. The $y-axis$ is the evaluations. For different input lengths, we observe that M-DGRU and M-DLSTM are worse than other methods. For the same predicting task, a smaller time step refers to better learning ability and less dependence on input length. As can be seen in Fig.6, RNN, GRU, LSTM, B-RNN, M-GRU, MVC-LSTM require shorter time steps to achieve their best performances. However, the RMSE of these methods is 2.58 times to 3.34 times than that of MVC-LSTM. This result reveals the role of a reasonable network structure in reducing computational network complexity. In summary, with a more reasonable network structure, MVC-LSTM further releases the storage and computing resources under the premise of improving the prediction accuracy.

#### 5) RESULTS OF DIFFERENT MVC-LSTM VARIANTS

We here present the results of different MVC-LSTM variants, including changing filter size, filter number, and network depth.

- **Different Filter Size.** The filter size determines the receptive field of a convolution. In this experiment,

we change the filter size at the first convolution layer from $4\times4$ to $6\times6$, and $2\times2$ to $4\times4$ in the second layer. The rightmost 3D convolution layer has a filter size of $2\times2\times2$ in all variants. Table 4 implies that a structure with both lager filters for long-term dependencies and smaller filters for short-term dependencies has lower RMSE.

**TABLE 4. Comparisons of MVC-LSTM Variants with Different Filter Size. Each of them has two ConvLSTM layers with 50 filters, one Conv3D layer with (2 × 2×2) 3D filters, and two BN layers.**

| ConvLSTM 1 | ConvLSTM 2 | RMSE |
|---|---|---|
| 6×6 | 2×2 | **0.0079** |
| 6×6 | 4×4 | 0.0093 |
| 4×4 | 2×2 | 0.0103 |

- **Different Filter Number.** In the previous comparisons of different size, we found that the '$(6\times6)$-50-$(2\times2)$-50-$(2\times2\times2)$-1' variant has a lower RMSE. Therefore, we compare the variants with different filter numbers based on this structure. From table 5, we can find that more filters better result.

**TABLE 5. Comparisons of MVC-LSTM Variants with Different Filter Number. '10', '30', and '50' represent the number of filters at each layer. The filter size is 6 × 6 in ConvLSTM 1, 2 × 2 in ConvLSTM 2, and 2 × 2 × 2 in Conv3D.**

| ConvLSTM1 | ConvLSTM2 | RMSE |
|---|---|---|
| 10 | 10 | 0.0142 |
| 30 | 30 | 0.0102 |
| 50 | 50 | **0.0079** |

- **Different Network Depth.** Table 6 shows the RMSE of variants with different network depths. As the number of convolution layers increases, the RMSEs of the MVC-LSTM variants first decrease and then increase. The changes imply that a deeper structure can better capture the temporal dependencies and the interactions between multiple variables. However, as the network keeps going deep, training becomes more difficult and confronted with a higher possibility of overfitting (although the BN unit is adopted).

**TABLE 6. Comparisons of MVC-LSTM Variants with Different Network Depth. All three ConvLSTM layers (if exist) have 50 filters, two BN layers, and a Conv3D layer with a 3D filter of 2 × 2 × 2.**

| ConvLSTM1 | ConvLSTM2 | ConvLSTM3 | RMSE |
|---|---|---|---|
| 6×6 | \ | \ | 0.0085 |
| 4×4 | \ | \ | 0.0143 |
| 2×2 | \ | \ | 0.0149 |
| 6×6 | 2×2 | \ | **0.0079** |
| 6×6 | 4×4 | 2×2 | 0.0082 |

### D. PREDICTION OF THE THERMOCLINE

Thermoclines change with geographic location (longitude, latitude), depth, and time. We define thermoclines when their temperature gradients are higher than the critical value $\delta$, as shown in equation 10 and 12. Equation 11 shows how to calculate the thickness, $H_{span}$, of a thermocline where $H_{top}$ is the upper bound of the thermocline and $H_{bottom}$ is the bottom of the thermocline.

$$G(^\circ C/m) = \frac{\triangle temperature(^\circ C)}{\triangle depth(m)} \tag{10}$$

$$H_{span}(m) = H_{bottom} - H_{top} \tag{11}$$

$$|G_H|(^\circ C/m) = \left|\frac{T_{H_{bottom}} - T_{H_{top}}}{H_{span}}\right| > \delta \tag{12}$$

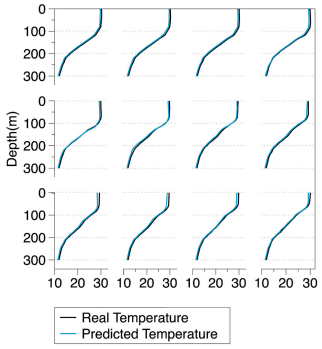



**FIGURE 7. Comparison of predicted temperature and real value in the past 12 months. The blue and black solid lines represent predicted temperature and real temperature, respectively.**

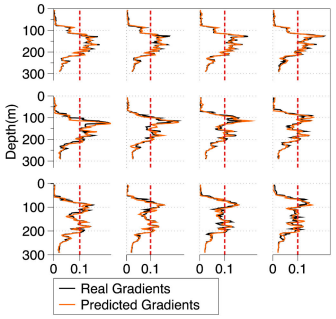



**FIGURE 8. Comparison of predicted temperature gradients *G* and real ones in the past 12 months. The orange and black solid lines represent predicted *G* and real *G*, respectively.**

Based on historical observations, DeepOcean captures temporal and spatial dependencies and the interactions between variables to predict future temperature. Fig. 7 and Fig. 8 use data at the position of 165.5°E and 0.5°N, and from 0∼300 m in the past 36 months (from January 2015 to December 2015). In Fig. 7, the blue line is the predicted values, and the black line represents real temperatures. In Fig. 8, the orange line represents corresponding predictions, and the black line shows the changes in temperature gradients ($\triangle depth = 1\ m$). We can observe that the shapes of temperature profiles change with time in Fig. 7. The temperature of the first month (the first of the first row) changes drastically at a depth of 100 m, while the temperature of the 12th month (the last of the third row) at a depth of 40 m. Also, the changes

in temperature (temperature gradient) at different times are different. Assuming that the temperature gradient critical value $\delta = 0.1$, the thickness of the thermocline changes from 50 m to 100 m, and the upper bounds vary from 40 m to 150m. The results show that the data predicted by DeepOcean fit the real values well and predict different trends with higher accuracy.

## V. DISCUSSION

In section IV-C.5, we have discussed the impact of different filter numbers in MVC-LSTM. Here, we mainly discuss the impact of network complexity on accuracy and training time, as shown in Fig. 9. All variants have the same network structure but different filter numbers in ConvLSTM layers. The black dotted line represents the RMSEs of the variants, while the bars show the training time. The red dotted line refers to the lowest RMSE of the other 15 baseline methods.

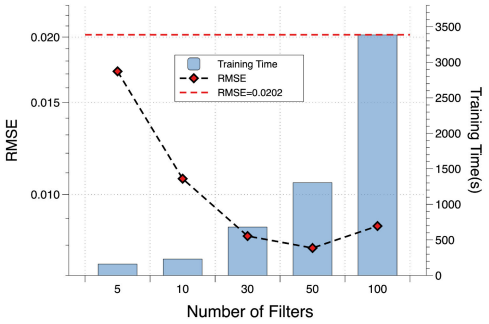



**FIGURE 9. Impact of network complexity. In this experiments, '5', '10', '30', '50', and '100' represent the number of filters at each layer. The filter size is 6 × 6 in ConvLSTM 1, 2 × 2 in ConvLSTM 2, and 2 × 2 × 2 in Conv3D.**

In comparison, all variants perform better than other baseline methods (RMSE < 0.0202). Networks with more filters can better learn input features from more perspectives. As the number of filters increases, the RMSE of the model first decreases and then increases. The model achieves its optimal performance at '50' filters. After that, training becomes more difficult, and the possibility of overfitting increases.

Nevertheless, the model still performs better than the other baselines. On the premise of acceptable prediction accuracy, we can customize the network complexity according to the particular requirements of training time in different applications. Some components can be removed to trade accuracy for training time. Evaluation results show that some variants take acceptable degradation on accuracy with less training time.

## VI. CONCLUSION

In this paper, we propose DeepOcean, a general deep learning framework for ocean timeseries sensing data prediction. The proposed DeepOcean is capable of capturing all spatial (horizontal and vertical) and temporal (long-term and recent) dependencies as well as interactions of different features (e.g., temperature, salinity). We demonstrate the effectiveness of DeepOcean using the thermocline prediction task on BOA_Argo, where DeepOcean outperforms the other fifteen baselines in terms of accuracy and structural flexibility, confirming that DeepOcean is better and more applicable to the time-series prediction. We also compared the predictive performance of different MVC-LSTM variants under the same generative module. The experimental results provide valuable insights and promising guidelines for future research to improve the universality of the framework. For future work, we will improve the framework and apply it to the classification and prediction of underwater sonar images. One possible solution is to involve the convolution-based network structure into the generative model for better capturing the spatial dependencies.

## REFERENCES

[1] I. F. Akyildiz, D. Pompili, and T. Melodia, "Underwater acoustic sensor networks: Research challenges," *Ad Hoc Netw.*, vol. 3, no. 3, pp. 257–279, May 2005.

[2] M. Stojanovic and J. Preisig, "Underwater acoustic communication channels: Propagation models and statistical characterization," *IEEE Commun. Mag.*, vol. 47, no. 1, pp. 84–89, Jan. 2009.

[3] M. Chitre, S. Shahabudeen, and M. Stojanovic, "Underwater acoustic communications and networking: Recent advances and future challenges," *Mar. Technol. Soc. J.*, vol. 42, no. 1, pp. 103–116, Mar. 2008.

[4] R. H. Stewart, *Introduction to Physical Oceanography*. College Station, TX, USA: Texas A&M Univ., 2008.

[5] M.-N. Zhang, J. Liu, K. Mao, Y. Li, X. Zhang, and Y. Shi, "The general distribution characteristics of thermocline of China sea," *Mar. Forecast*, vol. 23, pp. 51–58, Apr. 2006. [Online]. Available: http://en.cnki.com.cn/Article_en/CJFDTotal-HYYB200604007.htm

[6] S. Nguyen, E. Cayirci, L. Yan, and C. Rong, "A shadow zone aware routing protocol for acoustic underwater sensor networks," *IEEE Commun. Lett.*, vol. 13, no. 5, pp. 366–368, May 2009.

[7] D. Bacciu, F. Errica, A. Micheli, and M. Podda, "A gentle introduction to deep learning for graphs," 2019, *arXiv:1912.12693*. [Online]. Available: http://arxiv.org/abs/1912.12693

[8] Z. He, C.-Y. Chow, and J.-D. Zhang, "STCNN: A spatio-temporal convolutional neural network for long-term traffic prediction," in *Proc. 20th IEEE Int. Conf. Mobile Data Manage. (MDM)*, Jun. 2019, pp. 226–233.

[9] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang, "Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction," *Sensors*, vol. 17, no. 4, p. 818, 2017.

[10] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data," *Transp. Res. C, Emerg. Technol.*, vol. 54, pp. 187–197, May 2015.

[11] Y. Kim, P. Wang, and L. Mihaylova, "Structural recurrent neural network for traffic speed prediction," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 5207–5211.

[12] J. Zhang, Y. Zheng, D. Qi, R. Li, X. Yi, and T. Li, "Predicting citywide crowd flows using deep spatio-temporal residual networks," *Artif. Intell.*, vol. 259, pp. 147–166, Jun. 2018.

[13] M. Khodayar and J. Wang, "Spatio-temporal graph deep neural network for short-term wind speed forecasting," *IEEE Trans. Sustain. Energy*, vol. 10, no. 2, pp. 670–681, Apr. 2019.

[14] K. Zhang, X. Geng, and X.-H. Yan, "Prediction of 3-D ocean temperature by multilayer convolutional LSTM," *IEEE Geosci. Remote Sens. Lett.*, early access, Jan. 16, 2020, doi: 10.1109/LGRS.2019.2947170.

[15] D. Yu and L. Deng, "Deep learning and its applications to signal and information processing [Exploratory DSP]," *IEEE Signal Process. Mag.*, vol. 28, no. 1, pp. 145–154, Jan. 2011.

[16] Y. Jiang, T. Zhang, Y. Gou, L. He, H. Bai, and C. Hu, "High-resolution temperature and salinity model analysis using support vector regression," *J. Ambient Intell. Humanized Comput.*, vol. 9, pp. 1–9, Jun. 2018. [Online]. Available: https://link.springer.com/article/10.1007/s12652-018-0896-y

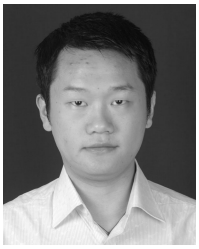
[17] Y. Gou, J. Liu, and T. Zhang, "KNN regression model-based refinement of thermohaline data," in *Proc. 13th ACM Int. Conf. Underwater Netw. Syst.*, Dec. 2018, p. 44.

[18] E. Elhariri and S. A. Taie, "H-ahead multivariate microclimate forecasting system based on deep learning," in *Proc. Int. Conf. Innov. Trends Comput. Eng. (ITCE)*, Feb. 2019, pp. 168–173.
[19] Z. Zhao, W. Chen, X. Wu, P. C. Y. Chen, and J. Liu, "LSTM network: A deep learning approach for short-term traffic forecast," *IET Intell. Transp. Syst.*, vol. 11, no. 2, pp. 68–75, Mar. 2017.
[20] N. Srivastava, E. Mansimov, and R. Salakhudinov, "Unsupervised learning of video representations using LSTMs," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 843–852.
[21] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.
[22] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-C. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 802–810.
[23] A. Sagheer and M. Kotb, "Time series forecasting of petroleum production using deep LSTM recurrent networks," *Neurocomputing*, vol. 323, pp. 203–213, Jan. 2019.
[24] K. A. Althelaya, E.-S.-M. El-Alfy, and S. Mohammed, "Stock market forecast using multivariate analysis with bidirectional and stacked (LSTM, GRU)," in *Proc. 21st Saudi Comput. Soc. Nat. Comput. Conf. (NCC)*, Apr. 2018, pp. 1–7.
[25] A. Ghaderi, B. M. Sanandaji, and F. Ghaderi, "Deep forecast: Deep learning-based spatio-temporal forecasting," 2017, *arXiv:1707.08110*. [Online]. Available: http://arxiv.org/abs/1707.08110
[26] Q. Zhang, H. Wang, J. Dong, G. Zhong, and X. Sun, "Prediction of sea surface temperature using long short-term memory," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 10, pp. 1745–1749, Oct. 2017.
[27] Y. Yang, J. Dong, X. Sun, E. Lima, Q. Mu, and X. Wang, "A CFCC-LSTM model for sea surface temperature prediction," *IEEE Geosci. Remote Sens. Lett.*, vol. 15, no. 2, pp. 207–211, Feb. 2018.
[28] H. Peng, A. Pan, Q. Zheng, and J. Hu, "A study of response of thermocline in the south China sea to ENSO events," *J. Oceanol. Limnol.*, vol. 36, no. 4, pp. 1166–1177, Jul. 2018.
[29] B. Jiang, W. U. Xin-Rong, J. Ding, and R. Zhang, "Comparison on the methods of determining the depths of thermocline in the South China sea," *Mar. Sci. Bull.*, vol. 35, no. 1, pp. 64–73, 2016.
[30] S. Yao, S. Hu, Y. Zhao, A. Zhang, and T. Abdelzaher, "DeepSense: A unified deep learning framework for time-series mobile sensing data processing," in *Proc. 26th Int. Conf. World Wide Web*, Apr. 2017, pp. 351–360.
[31] H. Li, F. Xu, W. Zhou, D. Wang, J. S. Wright, Z. Liu, and Y. Lin, "Development of a global gridded argo data set with barnes successive corrections," *J. Geophys. Res., Oceans*, vol. 122, no. 2, pp. 866–889, Feb. 2017.
[32] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015, *arXiv:1502.03167*. [Online]. Available: http://arxiv.org/abs/1502.03167

**YU GOU** received the B.S. and M.S. degrees from the College of Computer Science and Technology, Jilin University, Changchun, China, where she is currently pursuing the Ph.D. degree. Her research interests include underwater acoustic sensor networks and artificial intelligence.

**TONG ZHANG** received the B.S. degree from the College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China, and the M.S. degree from the College of Computer Science and Technology, Jilin University, Changchun, China, where he is currently pursuing the Ph.D. degree. His research interests include artificial intelligence and ocean engineering.

**JUN LIU** received the B.Eng. degree in computer science from Wuhan University, China, in 2002, and the Ph.D. degree in computer science and engineering from the University of Connecticut, USA, in 2013. He is currently a Professor with the School of Electronic and Information Engineering, BeiHang University, China, also a part-time Professor with the College of Computer Science and Technology, Jilin University and Robotics Research Center, Peng Cheng Laboratory, Shenzhen, China. His major research focuses on underwater acoustic networking, time synchronization, localization, network deployment, and also interested in operating system, cross layer design. He is a member of the IEEE Computer Society.



**LI WEI** received the B.E. degree in electrical engineering and automation from Xi'an Jiaotong University, Xi'an, China, and the M.S. degree in computer science and engineering from the University of Connecticut, Storrs, CT, USA. He is currently pursuing the Ph.D. degree in electrical and electronics engineering with the Michigan Technological University, Houghton, MI, USA. His research interests include in the area of underwater acoustic communication and networking with deep learning algorithms.

**JUN-HONG CUI** (Member, IEEE) received the B.S. degree in computer science from Jilin University, China, in 1995, the M.S. degree in computer engineering from the Chinese Academy of Sciences, in 1998, and the Ph.D. degree in computer science from UCLA, in 2003. She is currently on the Faculty with the College of Computer Science and Technology, Jilin University, Changchun City, China, also a part-time Professor of and Robotics Research Center, Peng Cheng Laboratory, Shenzhen, China. Recently, her research mainly focuses on exploiting the spatial properties in the modeling of network topology, network mobility, and group membership, scalable and efficient communication support in overlay and peer-to-peer networks, algorithm and protocol design in underwater sensor networks. She is a member of the IEEE Computer Society.