water

MDPI

Article

Multi-Location Emulation of a Process-Based Salinity Model Using Machine Learning

Siyu Qi [1,*], Minxue He [2,*], Zhaojun Bai [3], Zhi Ding [1], Prabhjot Sandhu [2], Yu Zhou [2], Peyman Namadi [2], Bradley Tom [2], Raymond Hoang [2] and Jamie Anderson [2]

[1] Department of Electrical and Computer Engineering, University of California, Davis, CA 95616, USA; zding@ucdavis.edu
[2] California Department of Water Resources, 1516 9th Street, Sacramento, CA 95814, USA; prabhjot.sandhu@water.ca.gov (P.S.); yu.zhou@water.ca.gov (Y.Z.); peyman.hosseinzadehnamadi@water.ca.gov (P.N.); bradley.tom@water.ca.gov (B.T.); raymond.hoang@water.ca.gov (R.H.); jamie.anderson@water.ca.gov (J.A.)
[3] Department of Computer Science, University of California, Davis, CA 95616, USA; bai@cs.ucdavis.edu
* Correspondence: syqi@ucdavis.edu (S.Q.); kevin.he@water.ca.gov (M.H.)

Abstract: Advances in machine-learning techniques can serve practical water management needs such as salinity level estimation. This study explores machine learning, particularly deep-learning techniques in developing computer emulators for a commonly used process model, the Delta Simulation Model II (DSM2), used for salinity estimation in California's Sacramento-San Joaquin Delta (Delta). We apply historical daily input data to DSM2 and corresponding salinity simulations at 28 study locations from 1990 to 2019 to train two machine-learning models: a multi-layer perceptron (MLP) and Long-Short-Term Memory (LSTM) networks in a multi-task learning framework. We assess sensitivity of both networks to the amount of antecedent input information (memory) and training data to determine appropriate memory size and training data length. We evaluate network performance according to several statistical metrics as well as visual inspection. The study further investigates two additional networks, the Gated Recurrent Unit (GRU) and Residual Network (ResNet) in salinity modeling, and compares their efficacy against MLP and LSTM. Our results demonstrate strong performance of the four neural network models over the study period, achieving absolute bias below 4%, plus near-perfect correlation coefficients and Nash–Sutcliffe efficiency coefficients. The high complexity LSTM shows slight performance edge. We further show that deeper and wider versions of MLP and LSTM yield only marginal benefit over their baseline counterparts. We also examined issues related to potential overfitting by the proposed models, training data selection strategies, and analytical and practical implications. Overall, this new study indicates that machine-learning-based emulators can efficiently emulate DSM2 in salinity simulation. They exhibit strong potential to supplement DSM2 in salinity modeling and help guide water resource planning and management practices for the Delta region.

Keywords: Sacramento-San Joaquin Delta; salinity modeling; machine learning; multi-task learning; multi-layer perceptron; recurrent neural networks

Citation: Qi, S.; He, M.; Bai, Z.; Ding, Z.; Sandhu, P.; Zhou, Y.; Namadi, P.; Tom, B.; Hoang, R.; Anderson, J. Multi-Location Emulation of a Process-Based Salinity Model Using Machine Learning. *Water* **2022**, *14*, 2030. https://doi.org/10.3390/w14132030

Academic Editors: Xing Fang, Jiangyong Hu and Suresh Sharma

Received: 2 May 2022
Accepted: 17 June 2022
Published: 24 June 2022

1. Introduction

1.1. Background

Salinity management is the keystone of water resource management in estuarine environments due to the underlying biological significance and inherently high variations in space and time of salinity [1]. Understanding these variations and predicting variation patterns under different potential future scenarios is the foundation for informed water management decision-making. This is especially true for areas with great ecological, social, and economic importance including the Sacramento–San Joaquin Delta (Delta) in California, United States.

The Delta is the hub of California's water supply system and protecting its economic and ecological vitality is a priority of California's Water Resilience Portfolio [2]. The Delta is the confluence of freshwater inflows from upstream rivers and saline tidal flows from the Pacific Ocean. Freshwater flow releases from upstream reservoirs are managed to maintain Delta salinity at levels that support water supply and environmental needs. This requires estimates of salinity for various climate, flow, and operational conditions. Empirical or process-based models have been traditionally developed and applied for this purpose. However, running these models for long study periods under multiple scenarios can be computationally expensive.

This study explores the use of machine learning including deep-learning approaches to create emulators for an operational process-based model to estimate salinity in the Delta. The ability of machine-learning emulators to estimate Delta salinity quickly for a vast number of scenarios makes them powerful adaptive management tools. For example, salinity emulators can be used in water operations models to ensure that managed flows are sufficient to meet salinity standards downstream [3–5], or to support analyses with immense numbers of possible scenarios, such as the use of Decision Scaling to explore impacts of future climate conditions on water supplies [6].

1.2. Literature Review

Salinity in the Delta has been estimated using empirical models, process-based numerical models and machine-learning models such as Artificial Neural Networks (ANNs). Empirical models have been developed to simulate salinity in the Delta, with the goal being to inform water management practices. One of the earliest attempts is the Minimum Delta Outflow (MDO) procedure [7]. The MDO predicts the Minimum Delta Outflow required to meet the maximum allowable salinity standards given a level export and the Delta Cross Channel Gate position. Another attempt is the G-model of [8] which relates salinity at various Delta locations to the net Delta outflow, as well as the prior history of net Delta outflow. The G-model improves upon the MDO routine by including the antecedent outflow conditions. More recently, the Delta Salinity Gradient (DSG) model [9] extends the G-model to simulate the position of the low salinity zone in the Delta.

Process-based numerical models have also been developed to simulate the spatial and temporal variations of salinity across the Delta. Unlike empirical models, these models explicitly simulate the salinity transport process (via solving the salinity advection–dispersion equations) driven by hydrology, hydrodynamics, water operations, and other relevant forcing. These models include one-dimensional models (e.g., the Delta Simulation Model II (DSM2) [10]) and multi-dimensional models (e.g., TRIM2D [11], RMA10 [12], UnTrim [13], and SCHISM [14]). Compared to one-dimensional models, multi-dimensional models provide salinity simulations at a higher spatial level but require considerably more input data preparation and computing resources. For studies requiring long-term simulations of multiple potential planning (e.g., climate, operational, and structural) scenarios, simpler one-dimensional models are largely favored. When it comes to Delta salinity modeling, DSM2 has probably the best-understood performance over decades and is widely used in contemporary applications in the Delta [15].

Data-driven machine-learning approaches have also been explored for Delta salinity modeling. The MDO procedure is implemented into the first generation of the California Department of Water Resources' (DWR's) planning and simulation model (DWRSIM [16]), which simulates the coordinated operations and the State Water Project (SWP) and the federal Central Valley Project (CVP). However, MDO's performance in certain months and years is found to be less desirable. Sandhu and Finch [17] investigated ANNs as an alternative to MDO using a public domain program named Stuttgart Neural Network Simulator. They tested networks with one hidden layer and two hidden layers with different numbers of neurons. They also explored the impacts of various input variables on model results. They concluded that a structure of two hidden layers driven by multiple

inputs including flows and gate operations would yield satisfactory simulations on salinity. They also discovered that increasing the number of neurons could lead to overfitting issues.

DWRSIM is later superseded by CalSim II [18]. CalSim II uses the G-model to simulate Delta flow–salinity relationships. The G-model generally provides more desirable results than the MDO procedure. However, it does not distinguish between different flow patterns which can yield the same Net Delta Outflow. Moreover, it does not consider the salinity control gate position which affects the salinity. The ANNs of Sandhu and Finch [17] are enhanced to emulate the G-model in CalSim II [3–5]. CalSim II ANNs are further enhanced and incorporated into the latest version of the planning model, CalSim 3.0. Enhancements in CalSim 3.0 ANNs include: (a) a total number of seven daily inputs is used to drive the ANNs; (b) the ANNs are trained using DSM2 simulations in a much longer period compared to CalSim II ANNs. CalSim 3.0 ANNs are shown to be superior to CalSim II ANNs [19].

In addition to emulating flow–salinity models embedded in planning models, ANNs have also been developed to emulate other empirical or process-based models in salinity modeling. Rath et al. [20] integrate the empirical DSG with a Bayesian ANN to simulate salinity in the San Franciso Bay-Delta estuary. The hybrid model yields improved estimations of salinity compared to the DSG model alone. The hybrid approach also provides an uncertainty range on the estimated salinity. Chen et al. [21] develop ANNs to emulate DSM2 in simulating volumetric contributions (i.e., fingerprints) from different flow boundaries. ANN-simulated volumetric fingerprints are next applied to derive salinity levels at different locations in the Delta via multiplying the fingerprints by salinity concentrations at flow boundaries. He et al. [22] explore several ANN architectures in emulating another empirical model named Martinez Boundary Salinity Generator (MBSG) in simulating salinity at Martinez, the downstream boundary of the Delta. These architectures include the Long-Short-Term Memory (LSTM) networks and the one-dimensional Convolutional Neural Network (CNN). They observe that ANNs outperform the MBSG model across all salinity ranges in different years with various wetnesses.

All these previous data-driven studies apply ANNs in the single-task learning (STL) paradigm, namely one ANN per study location. Motivated by the fact that salinity levels at different locations in a region are impacted by the same set of boundary conditions, Qi et al. [23] extend the work of Jayasundara et al. [19] by developing multi-task learning (MTL) ANNs to replace STL ANNs in the latter. Specifically, Qi et al. [23] develop one MTL ANN to simultaneously simulate salinity levels at 12 study locations versus 12 STL ANNs in the work of Jayasundara et al. [19]. The hypothesis is that the region-specific information may enable one task (one study location) to eavesdrop on features learned by other tasks (other study locations), leading to improved efficacy and generalizability of the neural network [24]. The MTL ANN has the same structure (i.e., one input layer, two hidden layers, and one output layer) as any of those STL ANNs but with 12 times more neurons in each hidden layer. The MTL ANN is shown to outperform STL ANNs in salinity simulation in terms of both accuracy and efficiency [23].

1.3. Scope of the Current Work

The current study aims to develop machine-learning models that can mimic the fidelity and efficacy of the commonly used process-based DSM2 model on salinity modeling in the Delta, while requiring a much shorter run time. This study neither compares DSM2 emulation with observed salinity data nor trains the machine-learning models with observed data. However, salinity emulation of real-world data is planned as a future study. Since the goal of this study is emulation, any biases or errors in the process-based DSM2 model would also be emulated. The current work builds upon the success of those previous studies on Delta salinity modeling via machine learning but extends beyond them in terms of (1) emulating DSM2 in Delta salinity modeling directly; it is worth noting that Chen et al. [21] emulate DSM2 in flow simulation while other relevant studies do not emulate DSM2; (2) being the first study to explore deep-learning approaches (i.e.,

LSTM, Gated Recurrent Units, and Residual Networks) in emulating DSM2 for salinity modeling. He et al. [22] is the only study that explored deep learning in salinity simulation but focused on a much simpler model that can only generate salinity at a single location in the Delta; other relevant previous studies mostly apply the traditional multi-layer perceptron architecture; and (3) being capable of generating salinity at multiple locations across the Delta. Most previous machine-learning studies focus on only a small group of locations. This study aims to develop modular MTL machine-learning models and exemplify their applications at 28 study locations in the Delta. However, the modular nature of these models allows them to be readily adapted to simulate salinity at any other user-defined locations.

The rest of the paper is organized as follows. Section 2 presents the methodology consisting of study locations, study dataset, the model to be emulated, the machine-learning architectures proposed, study scenarios, and analysis metrics. Section 3 presents the results and findings of the study, including sensitivity analysis results and emulation results. Section 4 provides discussions on training strategies and overfitting potential of the proposed models, scientific and practical implications of the current work, and potential follow-up work. Section 5 concludes the study.

2. Materials and Methods

2.1. Study Locations

The capability of machine-learning models proposed in this study in simulating salinity is exemplified at various locations in the Sacramento-San Joaquin Delta (Delta), California, United States. The Delta is an estuarine system formed by the confluence of the Sacramento River from the north, the San Joaquin River from the south, and several tributaries (Consumnes, Mokelumne, and Calaveras rivers) from the east, woven together with hundreds of kilometers of waterways including channels and sloughs (Figure 1). Freshwater inflows travel westward through the Delta channels and exit through the San Francisco Bay which is bounded by the Pacific Ocean on the west. Within the Delta, consumptive uses of water include evaporation, seepage, and crop evapotranspiration. Suisun Marsh along with adjoining bays is the brackish transition between freshwater and saltwater. Salinity in the Delta is a complex function of freshwater inflows, Delta consumptive use, incoming ocean tides, Delta exports, and gate (e.g., the Delta Cross Channel Gate) operations.

The Delta serves as the hub for the SWP and CVP, which export water from the Delta to irrigate more than 3 million acres of farmland in the state and provide municipal supply for more than 20 million people. The conveyance of water relies on the movement of that water through the Delta and maintaining the right balance of saltwater and freshwater. Operations of the SWP and CVP are critical to keeping the balance as the Delta is also a diverse ecological region, providing critical habitat for many fish and wildlife species including the threatened Delta Smelt [25]. These operations are regulated by state and federal compliant requirements to meet water supply and environmental needs [26,27].

This study focuses on 28 important compliance locations, export locations, key flow junctions, and locations of ecological significance in the Delta (Figure 1). These locations are grouped into three regions: western (locations #1–10), northern (locations #11–13), and interior (locations #14–28). Salinity levels at the western locations are more impacted by seawater rather than freshwater boundaries. It is the opposite for northern locations. For interior locations, both seawater and freshwater play an important role in determining their salinity levels.

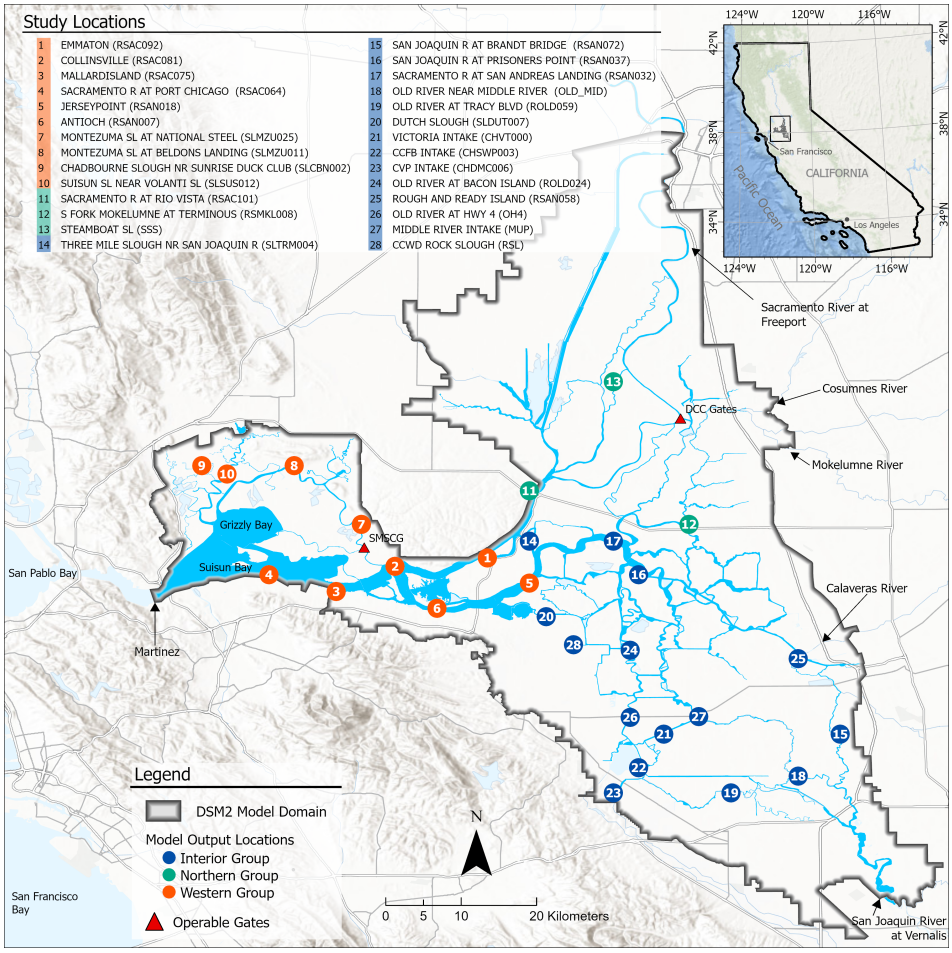

Figure 1. Schematic showing the Sacramento–San Joaquin Delta (Delta), the 28 study locations in three groups, and the model domain of the DSM2 model. The insert map illustrates the location of the Delta in the State of California, United States.

2.2. Study Model and Dataset

This study aims to emulate DSM2 in modeling salinity in the Delta. DSM2 is a process-based hydrodynamic and water quality model that is commonly used to simulate Delta salinity to inform water resource management decisions [3–5,10,21,23]. DSM2 simulates salinity, the amount of dissolved salt in water, as Electrical Conductivity (EC) since EC is how salinity is typically measured in the field. Despite the availability of other hydrodynamics and water quality models in the Delta, DSM2 is probably the most extensively used and has the best-understood performance over decades [15]. Nevertheless, it is a nontrivial effort to run, maintain, and enhance DSM2. Running the model for multiple scenarios can be time-consuming. A single 20-year historical DSM2 simulation, including hydrodynamics and water quality, takes about 3 h to complete on a desktop [21]. DSM2 needs to be recalibrated when there are significant changes to Delta channel geometry or there is a considerable amount of new flow, stage, and water quality data available. The entire recalibration can take up to months [22]. A machine-learning-based emulator of DSM2 needs to be retrained anytime DSM2 is recalibrated or when a major structural assumption is changed. With an appropriately trained emulator, it is feasible to run many plausible scenarios in a short amount of time, which is particularly appealing for real-time operations, long-term planning studies, and surrogate optimization where a model needs to be run hundreds or thousands of times.

There are eight major inputs to DSM2: combined inflows from the Sacramento River and the eastern tributaries, San Joaquin River inflows, total SWP and CVP exports, Delta Cross Channel Gate operational state (open or close), net Delta consumptive use, Martinez

water stage, Sacramento River salinity, and San Joaquin River salinity. These eight variables either represent boundary conditions for DSM2 or inform important operating rules for Delta flow and salinity management. Historical daily values of these variables from 1990–2019 are used as input to train the machine-learning models investigated in this study. DSM2-simulated daily salinity, represented by EC in micro Siemens/cm (μs/cm), at the 28 study locations during the same period is used as the training target. Accessibility information of these data is provided in Appendix A.

2.3. Machine-Learning Architectures

ANNs have been extensively applied in water resource modeling, of which the conventional multi-layer perceptron (MLP) networks are probably the most popular networks being explored [28]. Recently, deep-learning approaches have gained increasing attention and seen more applications in the field of water resources [29]. In this work, we first follow the study of Qi et al. [23] and examine the applicability of the traditional MLP networks as our benchmark in emulating DSM2 in salinity modeling. Next, following the work of He et al. [22], we develop a deep-learning architecture: the Long-Short-Term Memory (LSTM) network for the emulation task and compare its performance against that of the MLP networks. Finally, as a proof-of-concept exploration, we examine two additional deep-learning architectures, the Gated Recurrent Units (GRUs) and the Residual Networks (ResNets), that have never been applied in salinity modeling before, to our best knowledge. The goal of this final exploration is to assess the capability of less complex, and thus less prone to overfitting (than LSTM), deep-learning architectures in emulating DSM2 for salinity simulation. All these architectures are implemented in the form of multi-task learning (one single network for all study locations). They are briefly described as follows.

2.3.1. Multi-Layer Perceptron (MLP) Networks

MLP networks have demonstrated success in salinity modeling in the Delta [17,19–21,23]. In the current study, we use the same MLP network architecture as proposed in [19], which consists of an input layer, two fully connected (FC) hidden layers with eight and two neurons, respectively, and an output layer with one neuron. Further, we apply the MTL strategy as in [23] and expand the scope of the salinity modeling task from 12 salinity stations to 28 stations in the Delta. The detailed model architecture is given in Figure 2. Additionally, the same pre-processing method utilized in [19,23] is adopted for input dimension reduction in the current study.

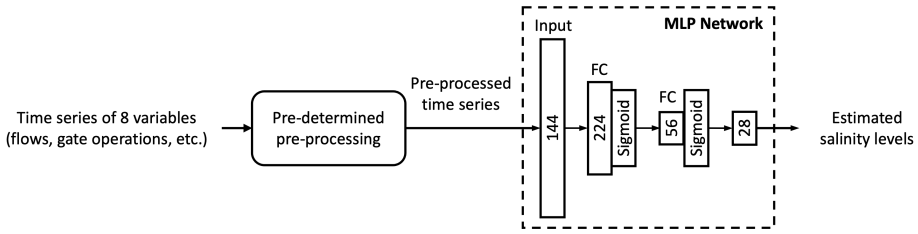

Figure 2. Architecture of the proposed MTL MLP Network. Numbers in layer blocks represent the numbers of neurons.

Based upon the STL MLP network architecture proposed in [19], the authors of [23] designed an MTL MLP architecture by increasing the numbers of neurons for all layers by a factor of 12, which is equivalent to the number of monitoring stations in their study. Following the same MTL model scaling method in [23], we increase the numbers of neurons in each layer by a factor of 28, which is equivalent to the number of study locations in this work. This gives us an MTL MLP model consisting of an input layer with 144 neurons, two hidden layers with 224 and 56 neurons, respectively, and an output layer with 28 neurons.

2.3.2. Long-Short-Term Memory (LSTM) Networks

The recurrent neural network (RNN) is a type of ANN model well suited for handling sequential data, such as the time series data in salinity modeling [30–33]. An LSTM network [34] is a special type of RNN that has been widely used in time series estimation [33,35,36], and has recently been applied to estimate salinity at one location in the Delta [22]. Inside the LSTM layer, there is an input gate, a forget gate, and an output gate. In comparison with common RNN layers, the LSTM layer stores a cell state vector and can maintain, update, or erase it when recurrently processing the input vector. A detailed diagram of an LSTM layer is provided in Figure A1 in Appendix B. To ensure a fair comparison with the MLP model in Section 2.3.1, we propose the LSTM network architecture depicted in Figure 3, which consists of an input layer, an LSTM layer with 224 units, and an output layer with 28 neurons. As the LSTM layer is naturally designed to capture the sequence dependence in the input time series, the pre-defined input data pre-processing can be skipped.

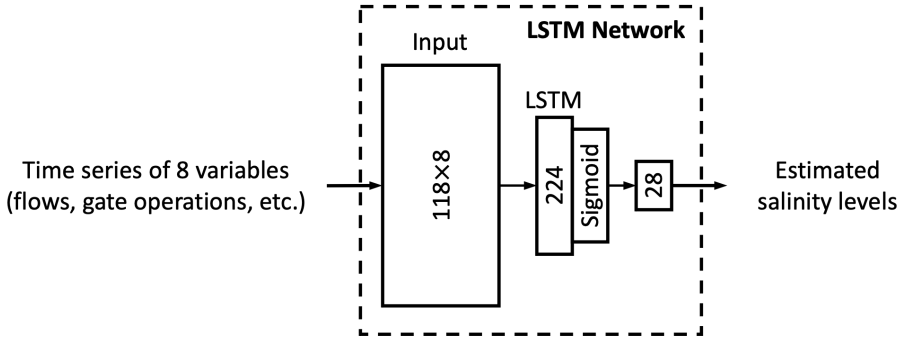

Figure 3. Architecture of the proposed MTL LSTM Network. Numbers in input layer blocks represent the input shape while numbers inside subsequent layers represent the number of units/neurons of the layer.

2.3.3. Gated Recurrent Unit (GRU) Networks

As RNNs are commonly used for time series estimation, other than the LSTM units with three gates, we further explore the less complex Gated Recurrent Unit (GRU) architecture, which is another type of RNN [37] that has less structural complexity than the LSTM network. A GRU consists of only two gates, a reset gate and an update gate, and does not have an internal memory unit as LSTM does. Similar to LSTM units, GRUs are also developed for processing long sequences and are thus suitable for salinity estimation in this work. We design the GRU-based RNN architecture as shown in Figure 4, which includes an input layer, a GRU layer with the same number of units as the LSTM architecture in Figure 3, and an output layer.

2.3.4. Residual Networks (ResNets)

The ResNet [38] is originally employed to solve the gradient-vanishing problem in deep-neural-network-based feature extraction. It implements the idea of adding skip connections across layers. It has been successfully applied in the field of time series forecasting [39–42]. With the skip connections in the ResNet, intermediate features can be passed forward from initial layers to later layers while gradients can flow backward through later layers to initial layers. In our proposed MTL MLP architecture, as input features are pre-processed in a pre-defined way, information loss is inevitable. Inspired by the powerful ResNet architecture, we suggest to skip the pre-defined pre-processing steps and attach a skip path, with a hidden layer, to the proposed MLP network to ensure that input information can be preserved. In the meantime, following the structure of Residual Blocks in ResNet, we insert two one-dimensional convolutional layers to the main path for input feature extraction. Features of the main path and side path are summed up before being sent to the second hidden layer. The detailed architecture can be found in Figure 5.

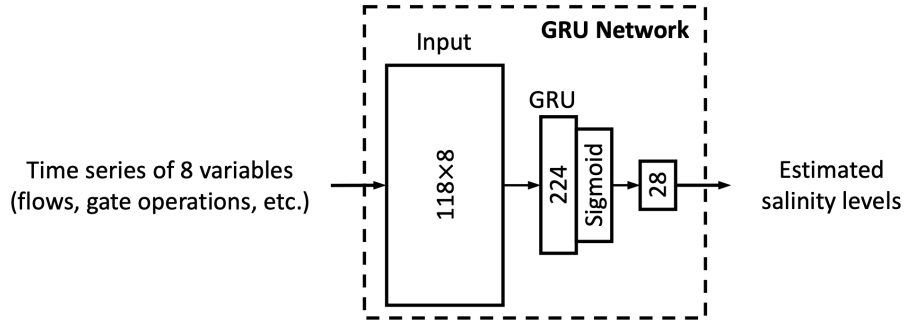

Figure 4. Architecture of the proposed MTL GRU Network. Numbers in input layer blocks represent the input shape while numbers insides subsequent layers represent the numbers of units/neurons of the layer.

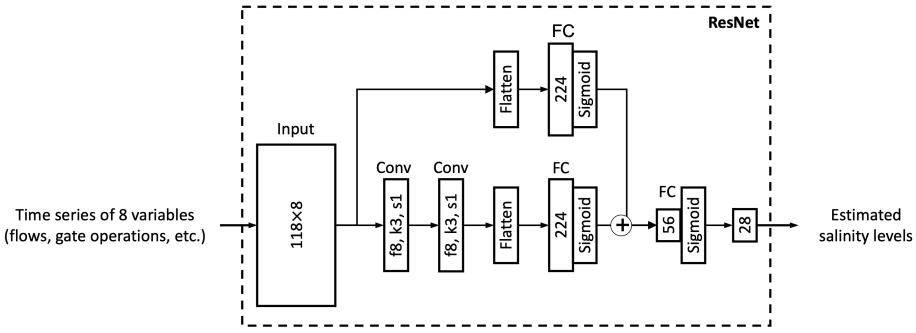

Figure 5. Architecture of the proposed MTL ResNet. In the convolutional layers, "f" represents number of filters, "k" represents kernel size, and "s" represents stride.

2.3.5. Implementation Details

We conduct our experiments on Google Colaboratory, a public online platform, which provides various GPUs for use. In Section 3.1.1, we randomly select 80% from the dataset, with a pre-defined random seed of 0 such that the training results are reproducible, as the training set and use the remaining 20% as the test set. According to the results of model sensitivity analysis on training data length in Section 3.1.2, in other sections of this work, we randomly select 70% of data as the training set and use the remaining 30% as the test set. We do not use a separate validation set. Instead, we monitor the models' performance on the test set during training. We utilize the Adam optimizer [43] with the mean squared error (MSE) as objective function to train the proposed networks. The learning rate is a fixed constant of 0.001 throughout training. The training process stops either at epoch 5000 or when the test MSE does not decrease for 50 epochs.

2.4. Study Scenarios

2.4.1. Network Inputs and Outputs

Similar to the approach described in [19], we utilize a combination of eight hydrological, water quality, and operation parameters as inputs, as detailed in Section 2.2. Note that in addition to the seven input variables employed in [19], in this work, Sacramento River salinity serves as the eighth input variable to provide extra information for the task.

As in the input pre-preprocessing method proposed in [19], we reduce the dimension of input data from 118 daily values to 18 values by extracting one value from the current day plus the most recent 7 antecedent days, along with 10 non-overlapping 11-day averages of the prior 110 days.

We use superscript for matrix and vector indexing and subscript for variable indexing. For example, x_i^t is the value for the i-th input variable on day t. We use $\overline{x_i^{t_1 \to t_2}}$ to represent the average value of the i-th input variable from t_1 to t_2, where $t_1 < t_2$:

$$\overline{x^{t_1 \to t_2}} = \frac{1}{t_2 - t_1 + 1} \sum_{t=t_1}^{t_2} x_i^t \tag{1}$$

We linearly normalize the time series of each input variable or the salinity at each station to the range of [0, 1]. For example, the input variable x_i^t can be normalized as in Equation (2), with T being the total number of samples in the dataset.

$$\hat{x}_i^t = \frac{x_i^t - (\min\limits_{k=1,\ldots,T} x_i^k)}{(\max\limits_{k=1,\ldots,T} x_i^k) - (\min\limits_{k=1,\ldots,T} x_i^k)} \tag{2}$$

For the proposed MLP network, given a set of input time series $\hat{x}_i^t$ with $1 \leq i \leq 8$ for the task of learning to estimate reference salinity levels on day t, we directly extract eight daily values $\hat{x}_i^t, \ldots, \hat{x}_i^{t-7}$ and compute a total of 10 successive non-overlapping 11-day moving averages $\overline{\hat{x}_i^{t-8 \to t-18}}, \ldots, \overline{\hat{x}_i^{t-107 \to t-117}}$. These 18 values for each of the eight input variables together serve as the $8 \times 18 = 144$ input parameters for our proposed MLP network.

2.4.2. Grouping Stations

As explained in [23], multi-tasking enables intermediate layers in an ANN to share some features of underlying inputs and thus improves the general performance of the ANN. However, as the 28 salinity stations studied in this work are located at different areas in the Delta, as shown in Figure 1, their salinity levels are dominated by different factors. For example, salinity levels in the western Delta are more likely to be affected by ocean tides. Therefore, directly treating these stations as one single group and training one ANN for all of them may not be the optimal solution for the salinity estimation task, it is natural to consider dividing these 28 stations into three groups (Figure 1) by their physical locations and train one neural network for each group.

2.4.3. Input Memory

In addition to the pre-processing method defined in [19], which we consider as the "baseline" case, we introduce five more scenarios with different length of input memories in this study to test the time memory sensitivity of proposed models. These five additional scenarios consist of: (I) no input memory; (II) eight-day memory; (III) one-week memory; (IV) three-week memory; (V) three-month memory. The detailed input vector formulation can be found in Table 1.

Table 1. Arrangement of pre-processed inputs for each case.

Cases (Memory Length)	**Prepared Inputs (for Estimating EC on Day *t*)**	
	MLP	**LSTM**
Baseline (Four-month)	Eight daily values: $\hat{x}_i^t, \ldots, \hat{x}_i^{t-7}$; Ten moving averages: $\overline{\hat{x}_i^{t-8 \to t-18}}, \ldots, \overline{\hat{x}_i^{t-107 \to t-117}}$	118 daily values: $\hat{x}_i^t, \ldots, \hat{x}_i^{t-118}$
I (No memory)	One daily value: $\hat{x}_i^t$	One daily value: $\hat{x}_i^t$
II (Eight-day)	Eights daily values: $\hat{x}_i^t, \ldots, \hat{x}_i^{t-7}$	Eights daily values: $\hat{x}_i^t, \ldots, \hat{x}_i^{t-7}$
III (One-week)	One daily values: $\hat{x}_i^t$; One moving average: $\overline{\hat{x}_i^{t-1 \to t-7}}$	Same as (II)
IV (Three-week)	One daily value: $\hat{x}_i^t$; Two moving averages: $\overline{\hat{x}_i^{t-1 \to t-11}}, \overline{\hat{x}_i^{t-12 \to t-22}}$	23 daily values: $\hat{x}_i^t, \ldots, \hat{x}_i^{t-22}$

Table 1. *Cont.*

Cases (Memory Length)	Prepared Inputs (for Estimating EC on Day *t*) MLP	LSTM
V (Three-month)	Eight daily values: $\hat{x}_i^t, \ldots, \hat{x}_i^{t-7}$; Ten moving averages: $\overline{\hat{x}_i^{t-8 \to t-18}}, \ldots, \overline{\hat{x}_i^{t-85 \to t-95}}$	96 daily values: $\hat{x}_i^t, \ldots, \hat{x}_i^{t-95}$

2.5. Study Metrics

To train the ANNs, we employ the commonly used mean squared error (MSE) as the cost function. Further, to evaluate the performance of proposed networks, we utilize four statistical metrics consisting of r^2, Bias, RMSE-observations standard deviation ratio (RSR), and the Nash–Sutcliffe Efficiency coefficient (NSE). Their detailed descriptions and formulas can be found in Table 2, where S stands for the salinity, $\bar{S}$ indicates the global average of the salinity sequence in the dataset, t represents an arbitrary day, T is the total number of days or samples in the dataset, and $_{ANN}$ and $_{ref}$ designate ANN-estimated and reference values (in this case, DSM2 simulated daily salinity represented by EC), respectively. Among the four evaluation metrics, r^2 assesses the strength of the linear relationship between ANN estimations and the reference salinity; percent bias reveals the extent to which the model overestimates or underestimates the salinity; RSR standardizesthe root mean squared error (RMSE) using the standard deviation of reference salinity; NSE quantifies the predictive skill of ANN models in comparison with the mean of reference salinity. For metrics r^2 and NSE, values closer to 1 demonstrate better performance, whereas for percent bias and RSR, values closer to 0 indicate better performance.

Table 2. Study Metrics.

Name	Definition	Formula
MSE	Mean Squared Error	$\mathrm{MSE} = \sum_{t=1}^{T} \left(S_{ref}^t - S_{ANN}^t\right)^2$
r^2	Squared Correlation Coefficient	$r^2 = \left(\frac{\sum_{t=1}^{T} \lvert(S_{ref}^t - \overline{S_{ref}}) \times (S_{ANN}^t - \overline{S_{ANN}})\rvert}{T \times \sigma_{ref} \times \sigma_{ANN}}\right)^2$
Bias	Percent Bias	$\mathrm{Bias} = \frac{\sum_{t=1}^{T}(S_{ref}^t - S_{ANN}^t)}{\sum_{t=1}^{T} S_{ref}^t} \times 100\%$
RSR	RMSE-observations standard deviation ratio	$\mathrm{RSR} = \frac{\sqrt{\sum_{t=1}^{T} \left(S_{ref}^t - S_{ANN}^t\right)^2}}{\sqrt{\sum_{t=1}^{T} \left(S_{ref}^t - \overline{S_{ref}}\right)^2}}$
NSE	Nash–Sutcliffe Efficiency coefficient	$\mathrm{NSE} = 1 - \frac{\sum_{t=1}^{T} \left(S_{ref}^t - S_{ANN}^t\right)^2}{\sum_{t=1}^{T} \left(S_{ref}^t - \overline{S_{ref}}\right)^2}$

3. Results

This section presents three types of results consisting of sensitivity analysis results, performance of the MLP and LSTM models proposed, and performance of two additional deep-learning architectures. The sensitivity analysis aims to identify the appropriate portion of the available data for training as well as the appropriate amount of antecedent input information, namely, memory to be used for training. The outcome of sensitivity analysis is utilized to train the proposed MLP and LSTM models of which the results are presented next. Finally, two additional deep-learning models are trained in the same way as the MLP and LSTM models have been trained. Their performance is compared with that of the original MLP and LSTM models developed.

3.1. Sensitivity Analysis

3.1.1. Sensitivity to Memory Length

Figure 6 shows the performance of MLP models with various memory lengths as detailed in Table 1. Both the grouped (G) and ungrouped (UG) versions of the models are

examined based on the training data and test data. From the dataset, we randomly select 80% as the training set and use the remaining 20% as the test set.

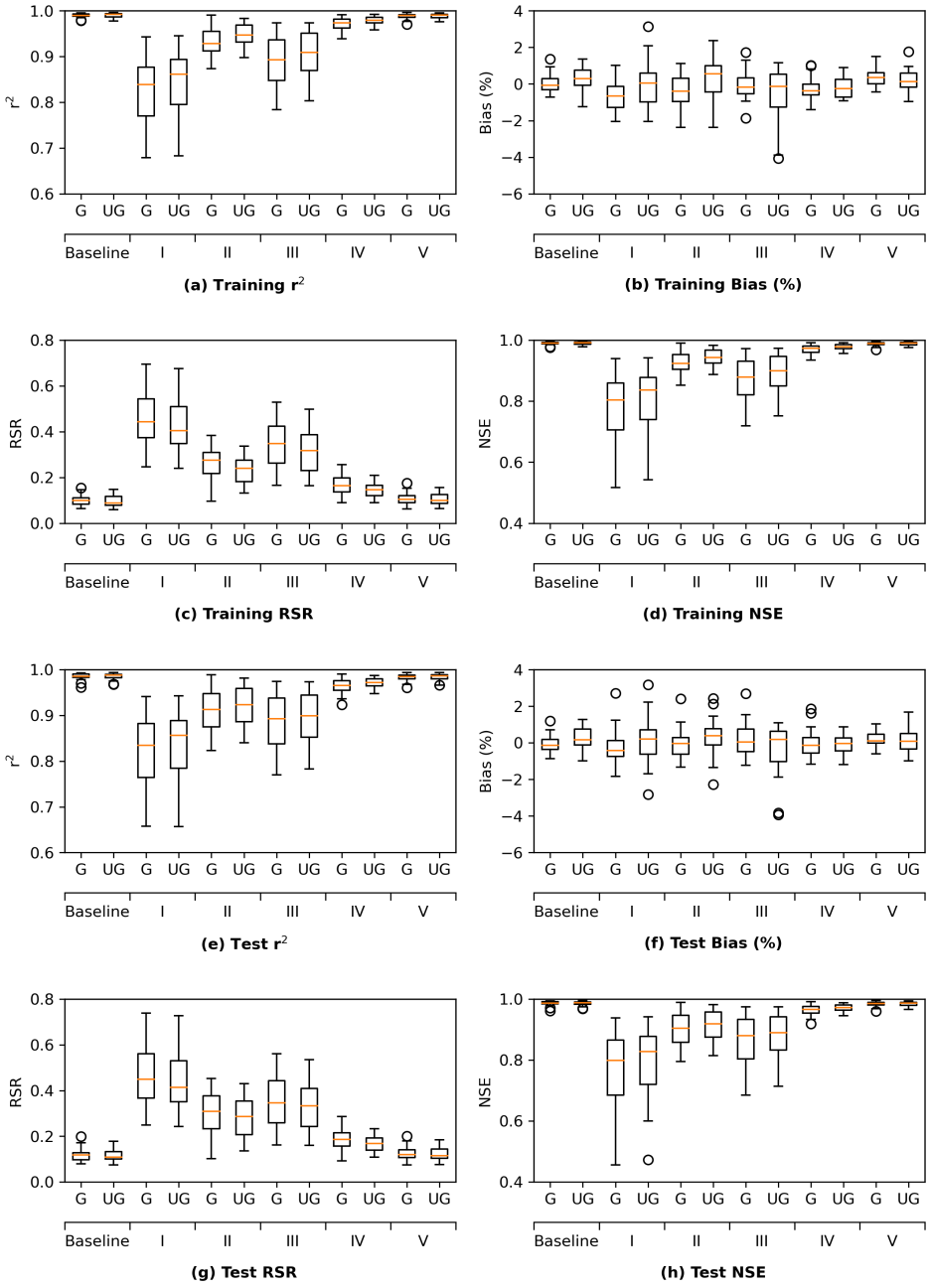

Figure 6. Box and whisker plots of four types of metrics consisting of r^2, percent bias, RSR, and NSE during the training (panels (**a**–**d**)) and test (panels (**e**–**h**)) phases of the grouped (G) and ungrouped (UG) MLP models under six scenarios: baseline and five different memory lengths as presented in Table 1. For a specific plot, the orange line represents the median value of the 28 metrics corresponding to 28 study locations; the box containing the orange line is the interquartile range from the 25th percentile up to the 75th percentile. The top bar designates the maximum metric value within 1.5 times the interquartile range above the 75th per percentile, while the bottom bar represents the minimum metric values within 1.5 times the interquartile range below the 25th percentile. The open circles designate outliers.

Incorporating any length of memory improves performance compared to the model with no memory for both training and test datasets. As memory length increases, r^2, RSR,

and NSE exhibit less variance and median values become closer to the ideal values. A significant improvement is observed when memory is increased to a week (II and III). The improved performance of case II over case III denotes that inclusion of all eight daily values is better than a one-week moving average. Another significant performance increase is observed when the memory is increased from one week to one month. The three-month memory model slightly outperforms the one-month memory model. We can conclude that (1) increasing input memory length generally leads to more desirable model performance; and (2) there is a notable improvement in using daily values over a moving average with short (i.e., one week) memory lengths. Grouping stations does not necessarily improve the model performance. On the contrary, the ungrouped models perform better with generally higher r^2 (panels (a) and (e)), lower RSR (panels (c) and (g)), higher NSE (panels (d) and (h)), as well as less variation of all three metrics. Grouping results in less variations in percent bias (panels (b) and (f)).

Figure 7 shows information similar to Figure 6, but for the LSTM models rather than the MLP models. Additionally, for LSTM models, scenario III is identical to scenario II and thus not shown in Figure 7. Looking at r^2, RSR, and NSE, increasing memory length leads to improved model performance for both the training and test datasets. This trend is not present in percent bias as it is not a normalized metric. The absolute magnitudes of the biases for all models are small, generally less than 4%. It is also clear that none of the proposed scenarios outperforms the baseline which has the longest memory, even though Scenarios IV (one-month memory) and V (three-month scenario) have comparable results to the baseline. Nevertheless, even the model with the least memory (i.e., Scenario I) performs reasonably well. The median values of test r^2 and NSE are above 0.8. The median test RSR is less than 0.5. The test bias is generally close to 0. Comparing the grouped version of LSTM to the ungrouped LSTM, the former tends to be slightly superior to the latter on average.

In brief, while both MLP and LSTM perform well even with limited antecedent data (i.e., memory) fed into them during the training, increasing the memory length clearly leads to improved model performance. The baseline MLP and LSTM models have the longest memory (118 days) and generally have the most desirable performance. For MLP, the ungrouped version tends to slightly outperform the grouped version. However, for LSTM, the grouped version seems to have the edge on the ungrouped version. The baseline memory length (118 days) will be adopted in all machine-learning models to be discussed henceforth.

3.1.2. Sensitivity to Training Data Length

Both grouped and ungrouped versions of the MLP and LSTM models are trained using randomly selected 80%, 70%, 60%, and 50% of the 30-year study dataset. Figure 8 shows the corresponding metrics associated with salinity simulations from these scenarios. The figure indicates that model performance drops as the training dataset length is reduced when judged by r^2, RSR, and NSE. For LSTM models, performance degrades steadily from 80% to 60% scenarios. A noteworthy drop is observed at 50% training length, where outliers are observed at greater magnitude and frequency. Nevertheless, in all scenarios examined, the metrics are generally satisfactory. Between 80% and 70% scenarios, the changes in metrics are generally marginal or not notable. MLP is more sensitive to reducing training length and generally shows greater variability compared to LSTM with a comparable training length. Similar to what has been shown in Figures 6 and 7, the signal in percent bias is less consistent compared to the other three metrics. The bias values in all scenarios are generally low and satisfactory. Figure 8 further shows that, within LSTM models, grouping consistently improves performance over ungrouped models. Within MLP models, the opposite is observed. Grouping either degrades or does not change model performance. Again, the percent bias metric does not clearly illustrate a consistent trend of performance between LSTM, MLP, or their grouped or ungrouped variants as the biases in all cases are

very small. Those observations are in line with what has been illustrated in the memory sensitivity analysis (Section 3.1.1).

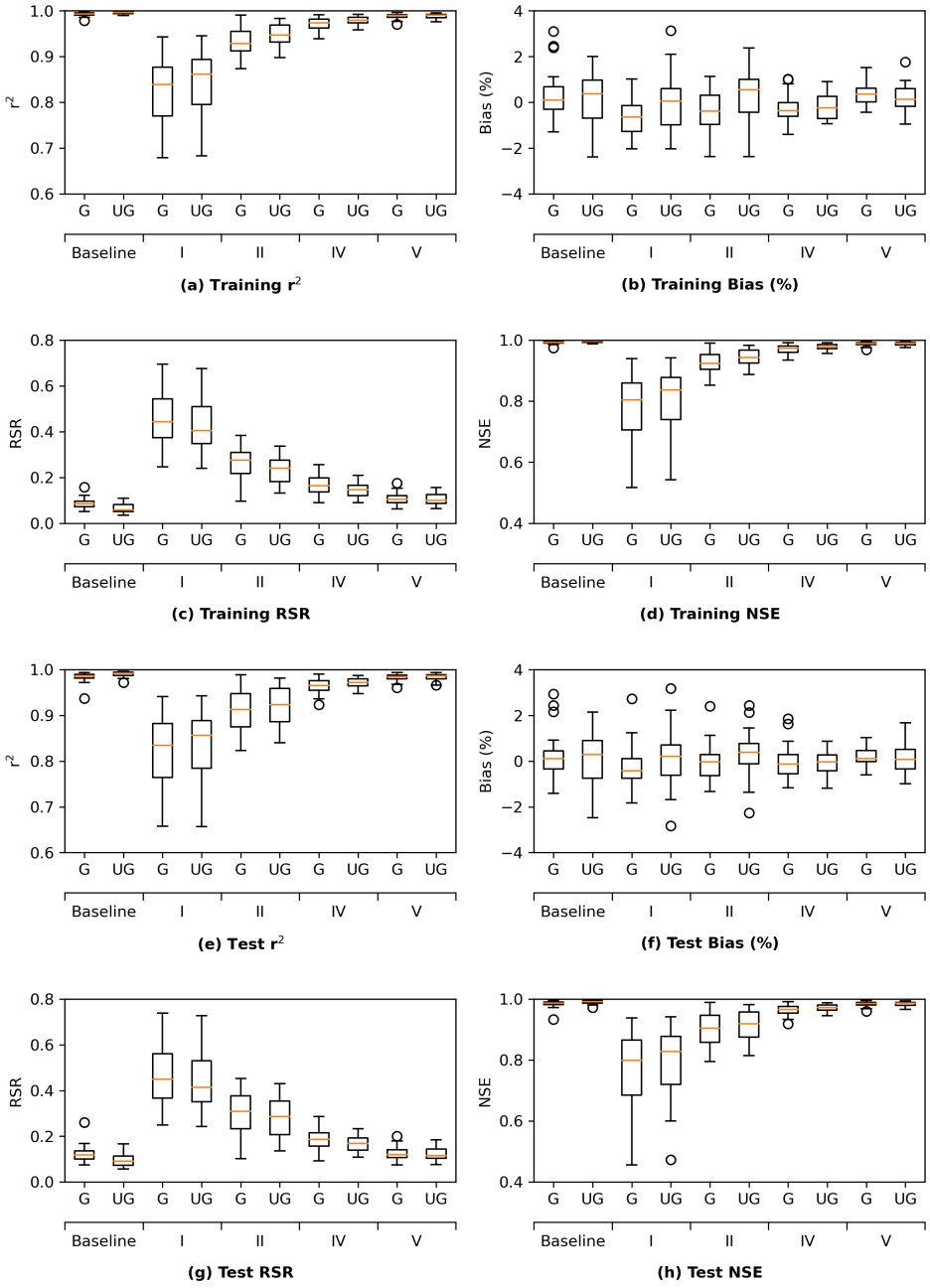

Figure 7. Box and whisker plots of four types of metrics consisting of r^2, percent bias, RSR, and NSE during the training (panels (**a**–**d**)) and test (panels (**e**–**h**)) phases of the grouped (G) and ungrouped (UG) LSTM models under five scenarios: baseline and four different memory lengths as presented in Table 1.

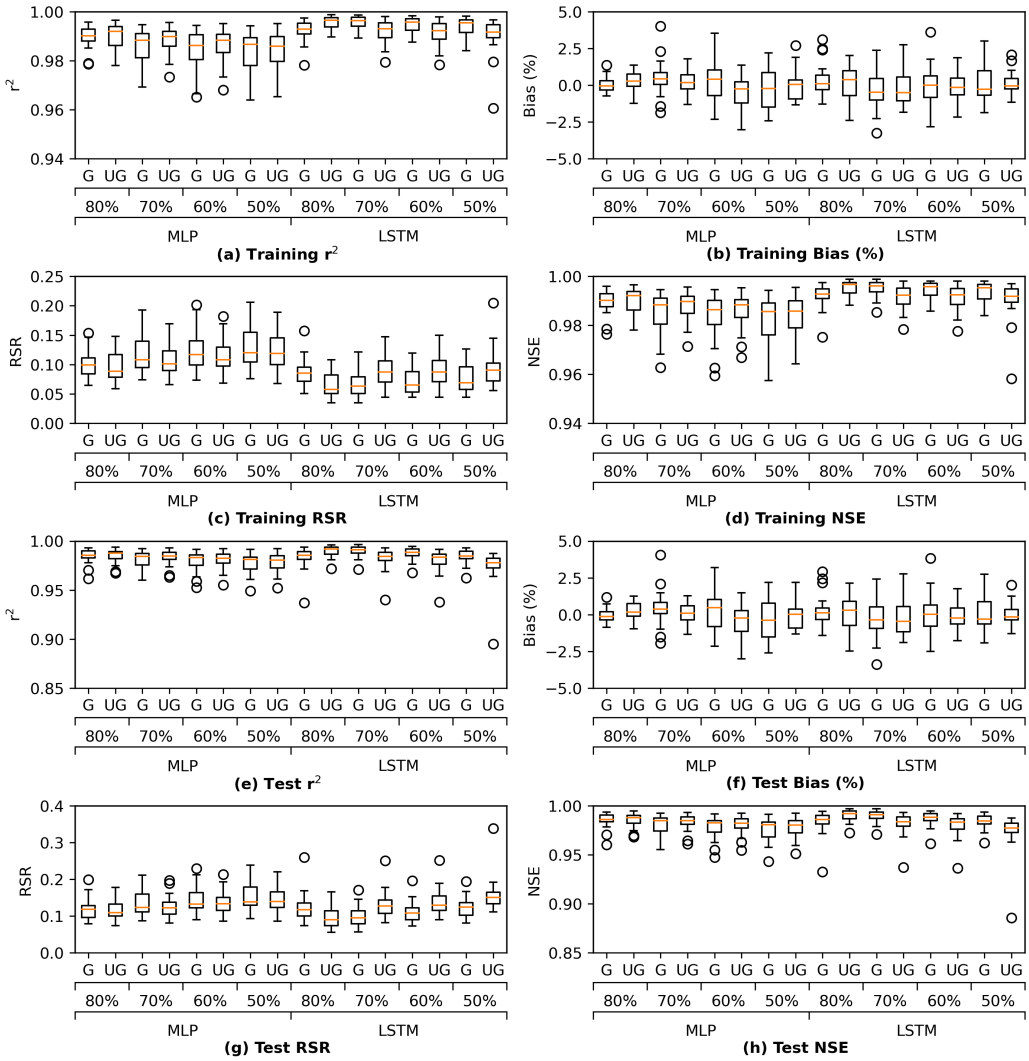

Figure 8. Box and whisker plots of four types of metrics consisting of r^2, percent bias, RSR, and NSE during the training (panels (**a**–**d**)) and test (panels (**e**–**h**)) phases of the grouped (G) as well as ungrouped (UG) MLP and LSTM models under four scenarios with varying lengths of training data: 50%, 60%, 70%, and 80% of all the data available.

In short, applying a larger portion of the available data to train MLP and LSTM models generally yields improved model performance. Nonetheless, the models trained with less data perform reasonably well. In particular, increasing the training dataset from 70% to 80% only leads to marginal or no improvements. Only results from models trained via 70% of the study dataset will be discussed hereinafter.

3.2. Salinity Simulation Results

The capability of the proposed MLP and LSTM models in emulating DSM2 in salinity simulation is examined from two aspects. Firstly, their performance quantified via the above-mentioned statistical metrics at different ranges of salinity is scrutinized. Even though it is important to capture the full range of salinity variations, high salinity is typically more of a concern from a water resource management perspective. Relevant water quality standards [26] normally prescribe a maximum salinity threshold that water operations need to comply with. Additionally, salinity (represented by EC) simulations from the proposed machine-learning models are compared to the target salinity directly. As illustrated in Section 3.1, the grouped LSTM generally outperforms its ungrouped counterpart, while the ungrouped MLP tends to yield better performance than the grouped

MLP model. For the sake of simplicity, the current section focuses on the results from the grouped LSTM (LSTM G) model. The corresponding results from the ungrouped MLP model are presented in Figures A2 and A3 of Appendix C. Figure 9 shows the statistical metrics r^2, percent bias, RSR, and NSE for each study location calculated in three ranges of salinity, low-middle (0–75%), high (75–95%), and extremely high (95–100%) range of EC. A note on the symbology of the figure is that "yellow" for r^2, NSE, and RSR reflects the satisfactory performance of the simulation models. The dark red and dark blue reflect the positive and negative percent bias, respectively, and the light color shows the near-ideal percent bias. As a general trend, model performance is most satisfactory when the salinity is in the low-middle range and decreases with higher salinity. Despite this trend, the metrics are generally desirable even for the extremely high range of salinity. Most of the metrics of extremely high salinity for location RSAC064 (location #4 in Figure 1) are distinctive from their counterparts of other locations. Its r^2 and NSE are the lowest while its RSR is the highest, even though they are all acceptable. This is further investigated by looking at the numerator and denominator components of the formulas for these metrics (Tables A1–A6 in Appendix D). It is evident that, compared to other locations, the high RSR and low NSE of RSAC064 during the extremely high range of salinity are caused by a relatively low variance in salinity at this location, while the low r^2 can be attributed to the relatively low numerator (differences between individual daily salinity and the overall mean value) when calculating this metric. The bias of RSAC064 is actually very small (Figure 9) and in line with the biases of all other locations. Another notable pattern is that the model generally overestimates low-middle EC range of salinity (referring to the formula for bias calculation in Table 2), and underestimates high and extremely high ranges of salinity for most locations. The corresponding metrics of the ungrouped MLP model share similar patterns but are slightly inferior (Figure A2 in Appendix C).

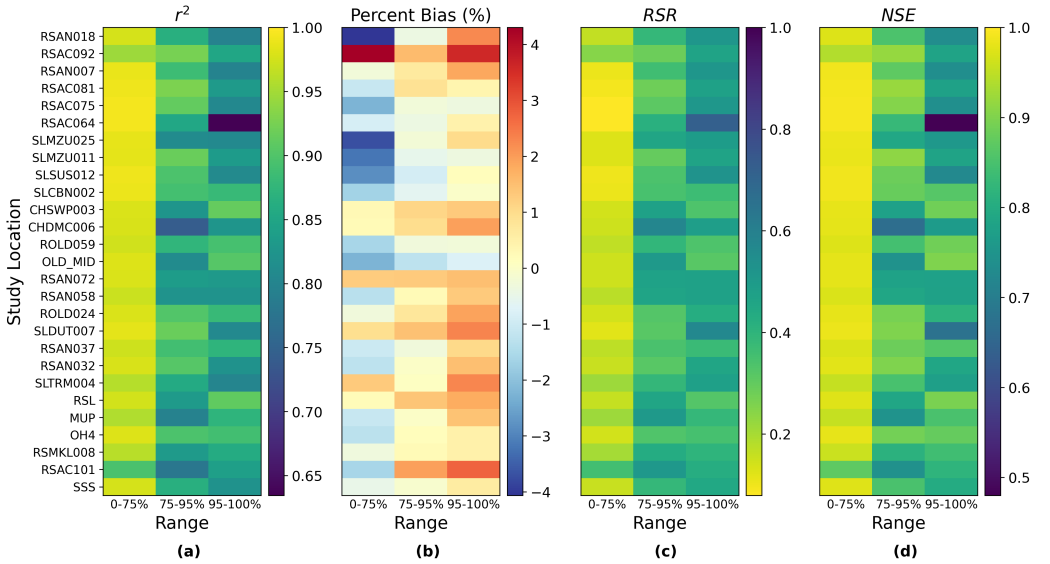

Figure 9. Model (grouped LSTM) performance measured by (**a**) r^2, (**b**) Percent Bias, (**c**) RSR, and (**d**) NSE at different salinity ranges: low-middle range (lowest 75%), high range (75 to 95 percentile, and extremely high range (highest 5%) at the study locations.

In addition to the statistical metrics, the specific daily salinity simulations from the proposed models are compared to the corresponding target salinity. For demonstration purposes, the results from the grouped LSTM model are presented at six selected locations in the form of exceedance probability curves and time series plots. These locations include three intake locations where water is pumped and transferred to various users to meet different needs: CCWD Rock Slough (RSL; location #28 in Figure 1), CVP Intake (CHDMC006; location #23), CCFB Intake (CHSWP003; location #22). Two important salinity compliance

locations: Emmaton (RSAC092; location #1) and Jersey Point (RSAN018; location #5) are also included. The last location is the Sacramento River at Port Chicago (RSAC064; location #4) of which some of the metrics for an extremely high range of salinity are not as desirable as their counterparts for other locations, as previously noted in Figure 9.

Figure 10 shows the corresponding exceedance probability curves and daily time series plots comparing DSM2 emulation with process-based DSM2 outputs. The simulations mimic the target salinity very well at all six locations across the full spectrum of exceedance probabilities. There are only marginal discrepancies between them, especially the upper tail (high salinity) at some locations. This is corroborated by the time series plots which show that the simulations generally capture the temporal variations of the target salinity but slightly underestimate the peak values. For RSAC064, though some statistical metrics of it are inferior to those of other locations, visual inspection of the exceedance curve and time series plot reveals that the grouped LSTM model can skillfully apprehend the temporal pattern and magnitude of the reference salinity at this location. The relevant results of the ungrouped MLP model (Figure A3 in Appendix C) are highly similar but slightly inferior.

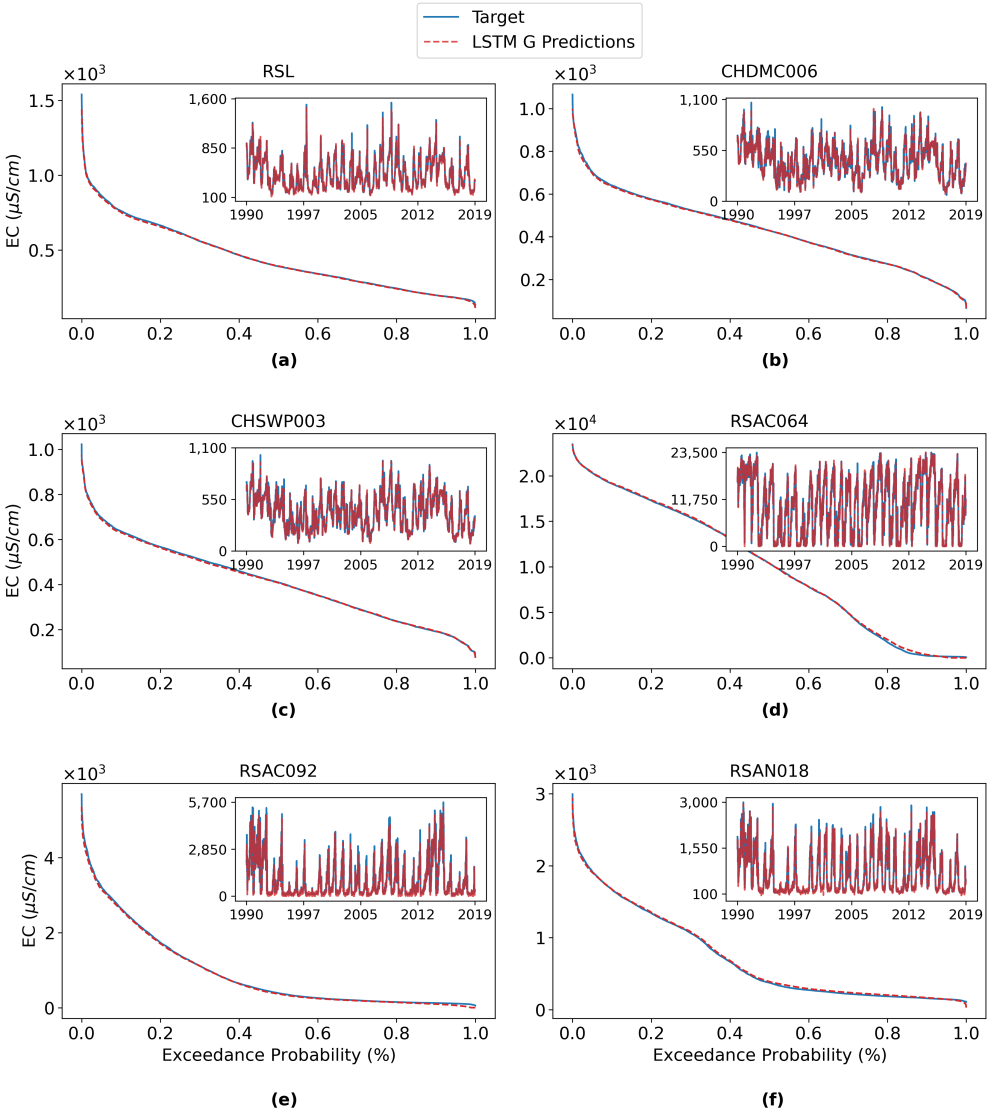

Figure 10. Exceedance probability curves and time series plots of salinity at six key study locations, including (**a**) RSL, (**b**) CHDMC006, (**c**) CHSWP003, (**d**) RSAC064, (**e**) RSAC092 and (**f**) RSAN018, comparing the grouped LSTM model with the process-based DSM2 outputs.

We further investigate the results by comparing the group-wise performance of the grouped LSTM model in Figure 11. It can be observed that the LSTM model delivers the best test performance in the western Delta group, which is likely due to the fact that this group is located the closest to the ocean and is more affected by seawater. Their salinity level patterns appear to be more regular. In addition, the LSTM model achieves the worse test performance on the northern Delta among the three groups. There are only 3 stations in the northern Delta group, while there are 10 and 15 in the western and interior Delta groups, respectively. As a result, the amount of information provided by the training set for northern Delta is much less than the other two groups, making the task more challenging. Finally, estimating salinity levels for the interior Delta group is known as the most difficult because their salinity levels can be affected by more variables, our LSTM model delivers the worse test performance of the group compared to the western Delta group, as expected. The box and whisker plots of group-wise performance of the other three proposed models, namely, MLP, ResNet, and GRU, can be found in Figures A15–A17 in Appendix I. The other three models present a similar pattern, except for ResNet, which estimates the interior Delta group mildly better than the western Delta one, indicating that the interior Delta group benefits more from the additional convolutional path attached to an MLP model to compensate for the potential estimation residuals. These comparisons lead to the conclusion that to accurately estimate the salinity levels for the interior Delta group, relying only on the pre-processed, dimension-reduced input data may not be sufficient and more detailed information from inputs is necessary.

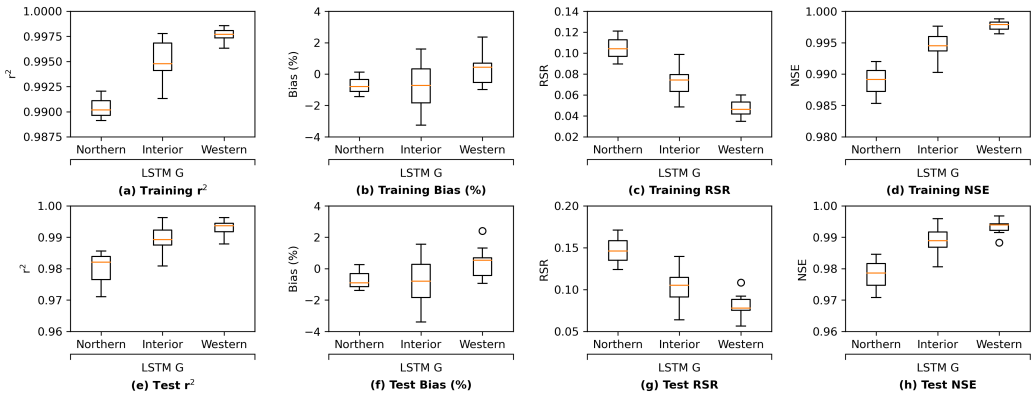

Figure 11. Box and whisker plots of four types of metrics consisting of r^2, percent bias, RSR, and NSE on stations belonging to the northern, interior, and western Delta groups during the training (panels (**a**–**d**)) and test (panels (**e**–**h**)) phases of the grouped (G) LSTM model.

Based on the observations above, we consider the interior Delta group as the key group and visualize the station-wise performance of this group in Figure 12. In general, among the 14 stations in the interior Delta group, all the four proposed models appear to show a poorer test performance on RSAN037 (location #16 in Figure 1), ROLD059 (location #19), CHWSP003 (location #22), CHDMC006 (location #23), RSAN058 (location #25), MUP (location #27), and RSL (location #28). According to Figure 1, all of them are located at the central part of interior Delta, and hence are further away from the locations where input variables are measured. We can observe the same patterns in the station-wise performance comparison plots of proposed MLP, ResNet, and GRU models, which can be found in Figures A18–A20 in Appendix J.

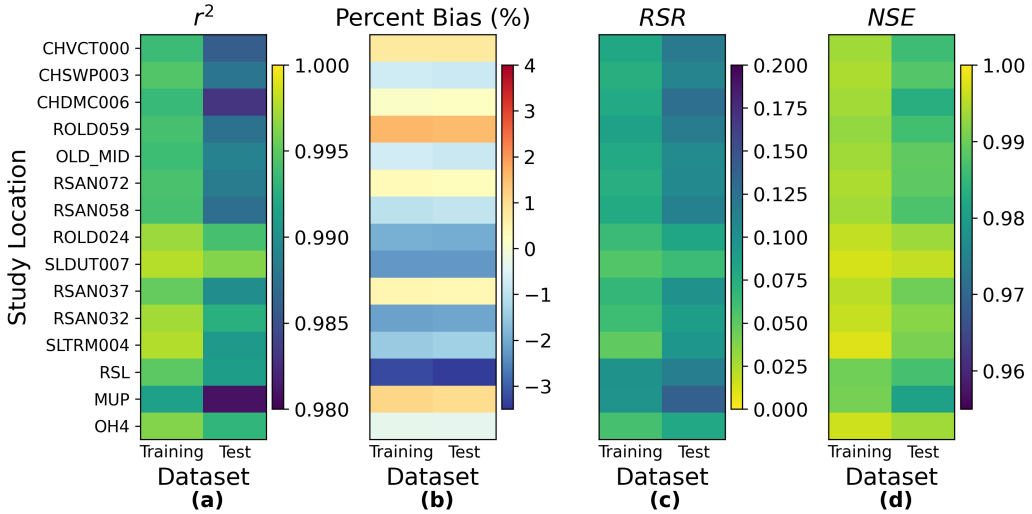

Figure 12. Model (grouped LSTM) performance measured by (**a**) r^2, (**b**) Percent Bias, (**c**) RSR, and (**d**) NSE on training and test datasets at the study locations belonging to the Interior Delta group.

In short, the proposed MLP and LSTM models are able to solidly mimic DSM2-simulated daily salinity across the study locations. The models tend to overestimate the low-middle range of salinity but underestimate the high range of salinity. However, the biases are generally small with the absolute amount being less than 4%. The grouped LSTM model has an edge over the ungrouped MLP model.

3.3. Performance of Additional Architectures

The simpler MLP and more complex LSTM models are shown to soundly emulate DSM2 in salinity simulation (Section 3.2). The current sub-section further assesses the two performance exploratory neural networks, GRU and ResNet, whose structural complexities are in between those of the MLP and LSTM models. Figure 13 illustrates the comparison of performance in terms of four study metrics. Each model architecture has its grouped (G) and ungrouped (UG) versions. The figure reveals that all the proposed architectures have satisfactory performance in both G and UG scenarios. The training results in Figure 13a–d indicate that the GRU model performs best while the MLP model is outperformed by the other three architectures, most likely because the MLP model has the simplest complexity and thus the fewest parameters. Meanwhile, the test results in Figure 13e,f suggest that LSTM and GRU models yield better results than MLP and ResNet models in the corresponding G or UG scenarios, reflecting that recurrent-based architectures are more suitable for the time series estimation task in this study. Moreover, the grouped GRU model appears to overfit more to training data than the grouped LSTM model, which is probably because it does not have an internal cell state to regularize the learned features. In addition, as the proposed ResNet model has more parameters than the MLP model, the ResNet model clearly overfits more as its training performance is better while its test performance is poorer than MLP. Potential model overfitting will be further discussed in the following section.

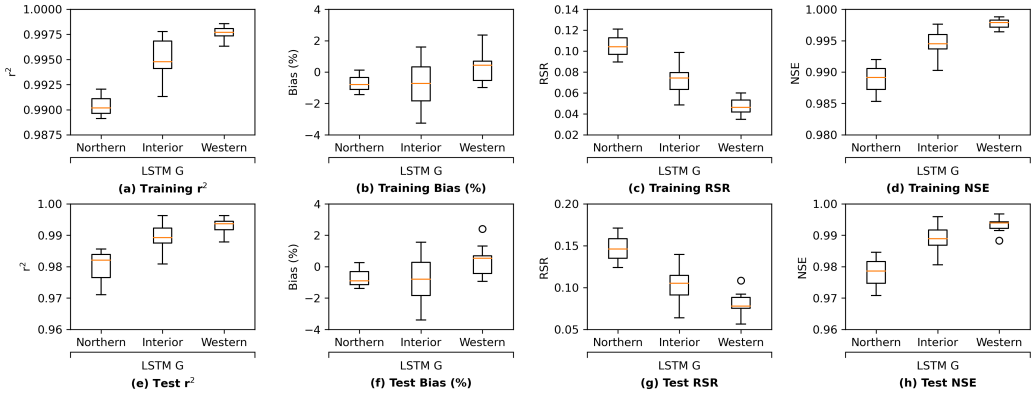

Figure 13. Box and whisker plots of four types of metrics consisting of r^2, percent bias, RSR, and NSE during the training (panels (**a**–**d**)) and test (panels (**e**–**h**)) phases of the originally proposed MLP and LSTM models as well as two additional models (GRU and ResNet).

In addition, for all architectures except for the LSTM model, not grouping (UG) the stations can lead to a slight performance improvement in comparison with grouping. This is likely due to that with more stations (information) available, it is less likely for the models to converge to a "bad" local minimum while the most complex LSTM model is less impacted by the information fed into it.

In brief, the RNN-based architectures (LSTM and GRU) achieve better statistical performance than MLP and ResNet models, which is most likely because the RNN-based models are much more complex in terms of model sizes as well as their internal feature extraction processes (Table A7 in Appendix E). The exploratory ResNet and GRU are more prone to overfitting than the originally proposed MLP and LSTM, respectively. Among all the models examined, the grouped LSTM model has the most desirable metrics.

4. Discussion and Future Work

Overfitting and generalization are two crucial and related topics in the field of machine learning. Machine-learning models are inherently prone to overfitting. Their generalization ability largely depends on how their parameters are optimized using what portion of the available training data. This section first discusses the overfitting potential of the machine-learning models proposed in this study. Next, the impact of different training data selection strategies on model performance is assessed. The section then discusses the practical and scientific implications of the current study. Finally, a few future research directions are proposed.

4.1. Overfitting

Theoretically speaking, complex machine-learning models can perform better on training data than simpler ones as they have more flexibility to adjust their parameters to fit the training data. However, an over-parameterized model with insufficient training data may end up with overfitting, meaning its training performance is good while the test performance is poor. This study proposed two types of machine-learning models, the classic multi-layer perceptron (MLP) models that have been dominantly applied in relevant previous studies as well as a deep-learning architecture: Long-Short-Term Memory (LSTM) network. An enhanced version of MLP, the Residual Network (ResNet), and a lite version of LSTM, the Gated Recurrent Unit (GRU), were also explored in this study. LSTM was the most complex in terms of model structure using the number of parameters (Table A7 in Appendix E) or model training time (Table A8 in Appendix F) as a proxy, followed by GRU and then ResNet.

To reduce the overfitting potential when developing machine-learning models, a common practice is to apply a two-step procedure where a subset of the data is used to train the models while the remaining subset of data is utilized to monitor the performance of

the models being trained [44]. The current study adopted this procedure when developing the proposed models. The early stopping strategy triggered by the loss function (i.e., mean squared error (MSE) in this case) of the test set during training was used to monitor the training process. Specifically, when the test MSE did not decrease for 50 epochs, the training would be stopped. The strategy has shown to reduce the overfitting potential of the proposed models in this study. This was reflected in Figure 13 which illustrates that the training metrics of the proposed models are better than the test metrics while the test metrics themselves are very desirable.

To further assess the overfitting potential, we increase the width and depth of the original MLP and LSTM models proposed. For demonstration purposes, we selected the ungrouped MLP (MLP UG) model and grouped LSTM (LSTM G) model. As illustrated in Figure A4 in Appendix G, to build a deeper MLP model, we added an additional fully connected layer between the two hidden layers; to build a wider MLP model, we doubled the number of neurons of the two hidden layers. These additions increased the number of MLP parameters roughly by 40% and 150% (Table A9 in Appendix G), respectively. Similarly, as shown in Figure A5 in Appendix G, to build a deeper LSTM model, we added an additional LSTM layer; to build a wider LSTM model, we doubled the number of LSTM units of the LSTM layer. Those changes also largely increased the number of LSTM parameters (Table A9 in Appendix G).

Figure 14 shows the results of the deeper and wider versions of the ungrouped MLP along with their counterpart where the original architecture (in Figure 2) remains unchanged. These enlarged models clearly outperform the original baseline model in both training and test phases, with superior metrics for the study locations. This indicates that increasing the depth or width of the MLP model does lead to improved model performance while not necessarily increasing the overfitting potential. Comparing the deeper model against the wider model, their associated metrics are very close to each other, suggesting the depth and width of the architecture has similar impacts on model performance in this case.

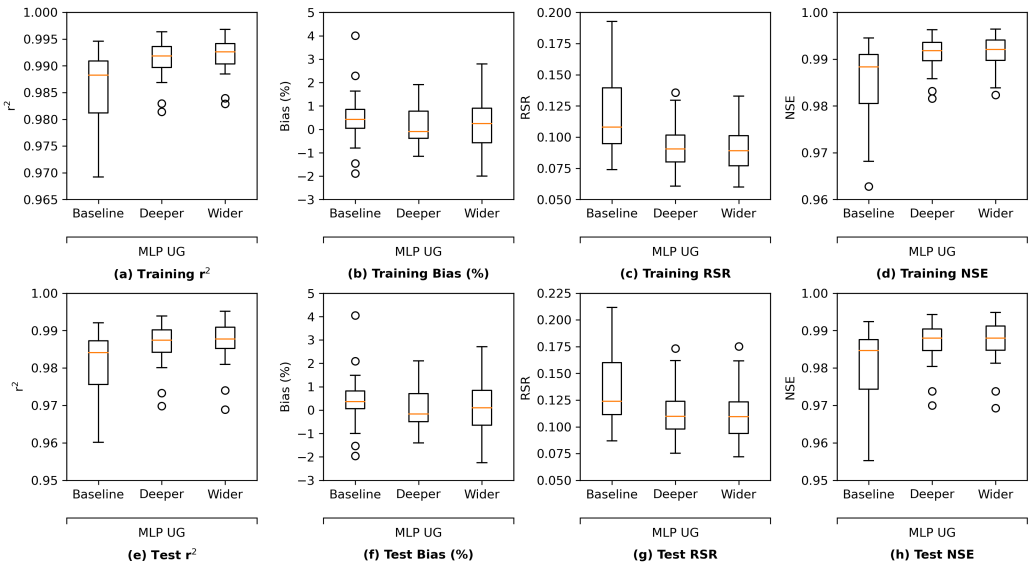

Figure 14. Performance comparison of the baseline MLP (ungrouped) versus its deeper and wider versions illustrated in Figure A4.

Figure 15 shows the metrics of the deeper and wider versions of the grouped LSTM model along with those of their base architecture (in Figure 3). On average, the enlarged models have more desirable metrics during the training for most study locations. However, their test metrics are close to the baseline test metrics. These observations suggest that employing a most complex version of LSTM does not yield evidently better results while increasing the likelihood of overfitting.

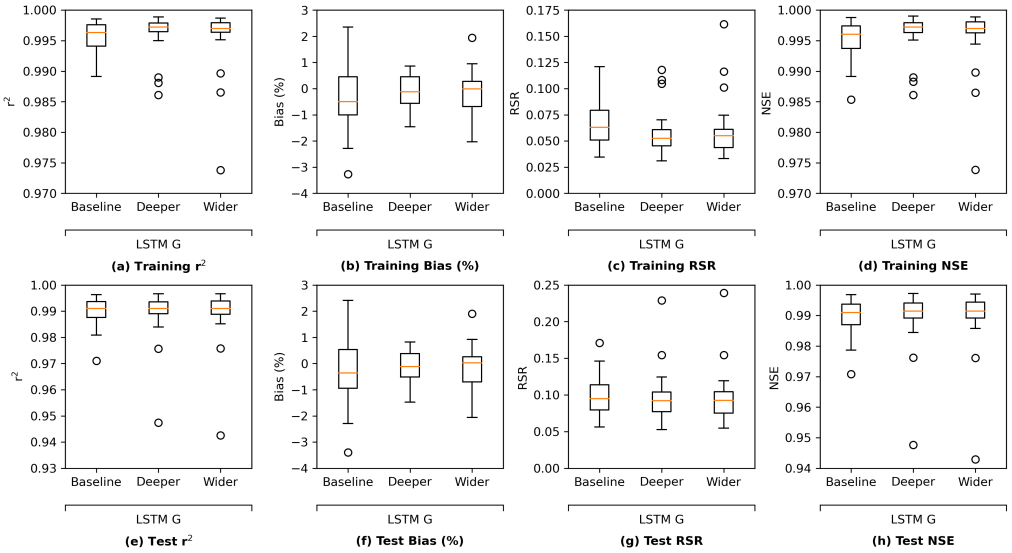

Figure 15. Performance comparison of the baseline LSTM (grouped) versus its deeper and wider versions illustrated in Figure A5.

In summary, the training process of the machine-learning models proposed in this study is monitored via the test metrics. Early stopping is implemented to terminate the training when the test metrics does not improve for a certain number of iterations. The overfitting potential is thus low, as confirmed by the test metrics shown in Section 3. Additional assessment of more complex models reveals that for the simpler MLP model, there is room to improve model performance by increasing its structural complexity (i.e., by adding additional layers or neurons) while the overfitting potential is still low. For the LSTM model which already has a complex structure, however, neither increasing the depth nor width of the architecture would lead to notable improvement in model performance but would tend to increase the overfitting potential.

4.2. Data Split

In line with relevant previous studies [20,21,23], the current study randomly split the dataset into a training subset and a test subset. There are other ways of splitting the dataset including chronological split and manual split. This sub-section compares the results of these two methods against their counterparts in the random split method applied, aiming to assess the impact of different data-split strategies on model performance.

Specifically, we select the grouped LSTM (LSTM G) model for demonstration purposes. For chronological split, data from the last 21 years (which are more representative of the current status of the Delta compared to the first 21 years) out of the available 30-year period are selected to train the LSTM G model. For manual split, we manually pick 21 years out of the study period based on the overall wetness of each year. There are five types of water years defined in California to facilitate water resource management practices in the State: wet, above normal, below normal, dry, and critical, with decreasing wetness quantified by the amount of runoff from major water supply watersheds [22,45]. To ensure that the training data contains representative years from each category, four years of each type are selected from the study period, which yields 20 years of data. Given that there are more wet years than any other type of year in the study period (referring to the data source provided in Appendix H), a fifth wet year is added as the 21st year. These years chosen are tabulated in Table A10 in Appendix H. Comparing these three methods, the chronological selection uses a fixed period of data for training with no randomness involved. The manual selection has increased randomness as there are no other constraints during the selection process other than including evenly distributed types of water years to the maximum extent possible. The random split is completely random with no constraining criteria to follow.

Figure 16 illustrates the four study metrics of those three data-split methods. The metrics associated with the chronological method and the manual method are generally satisfactory and comparable to each other. As expected, the training metrics are better than the test metrics for both methods. In comparison, the chronological method has slightly better metrics during the training process. However, the manual method has comparable or marginally better performance during the testing compared to the chronological method. This indicates that the manual method is less prone to overfitting as its test performance is closer to its training performance. In contrast, the metrics of the random split method are considerably superior to those of the chronological and manual methods. Both training and test r^2 and NSE are near the perfect value 1 for most of the study locations. The bias and RSR values are also very small. The variation range of each metric is also tight.

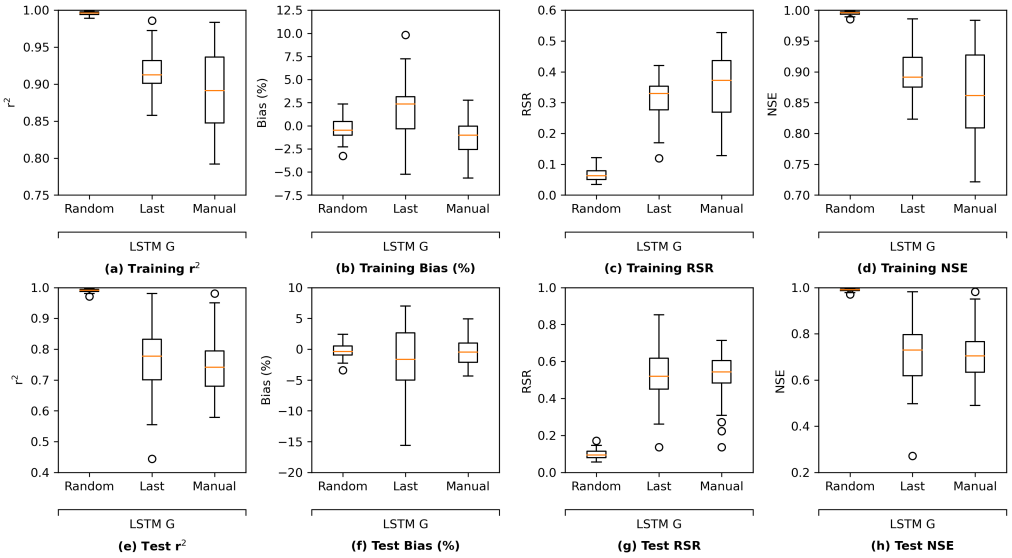

Figure 16. Training and test metrics of the grouped LSTM model under three data-split methods.

All in all, these observations suggest that training data splitting can play an important role when developing machine-learning models. The methods with increasing randomness tend to yield more robust models that have increased test performance.

4.3. Implications

The current study has important practical and scientific implications. From the practical point of view, this study demonstrates that the proposed machine-learning models can faithfully emulate the operational process-based model DSM2 in salinity modeling. The absolute bias is generally less than 4% and the correlation between them is close to the perfect value of 1 for all study locations. In addition to accuracy, efficiency in the context of computational burden is another strength of the proposed machine-learning models. Given the same set of inputs, the trained machine-learning models take only a few seconds to generate salinity simulations for even very long periods. In comparison, the running time for the calibrated DSM2 model would be hours [21] and becomes longer as the simulation period increases. Although the MTL machine-learning emulators proposed in the current study need to be retrained when a major structural assumption has changed or when its parent DSM2 model is recalibrated if new data become available or new model features are developed [46–49], they usually require less than an hour to converge (Table A8 in Appendix F). Once an emulator is appropriately trained for the range of expected conditions, it can be used in applications that require quick turnaround time (e.g., real-time forecasting) or long-term simulations of multiple planning scenarios (e.g., climate change, operational change, etc.). It is worth noting that machine-learning models presented in this study are trained with the process-based DSM2 outputs for historical conditions from 1990 to 2019, and both the process-based DSM2 and the machine-learning models are limited

in the capability to extrapolate beyond the conditions they are calibrated/trained for. In future work, we plan on training ANNs using augmented datasets that include a wider range of possible boundary conditions for DSM2 and gate operation scenarios. We will report the results in our future work.

From the scientific point of view, machine learning as a new scientific exploratory tool has the potential to supplement and improve upon process-based models in terms of (a) recognizing patterns or processes that are inadequately represented by the models due to knowledge gaps or capability limitations; (b) identifying the appropriate amount (less than the current data utilized to run process-based models) of data required to generate results that are comparable to simulations of process-based models; and (c) bypassing long-standing issues (e.g., equifinality, regionalization, scaling) associated with process-based models, among other things. The current study exemplifies the feasibility and capability of machine learning, particularly deep-learning architectures, in emulating an operational process-based model, DSM2, in salinity simulation in the Delta for the first time. This successful pioneering attempt lays the foundation for further testing of different hypotheses on the adequacy and representativeness of the input requirement, structure, and parameterization of DSM2 and other process-based models via the herein proposed or new machine-learning techniques. For example, dimension reduction techniques (e.g., Principal Component Analysis) can be applied to identify the primary predictors for salinity, further reducing the data requirement of machine-learning models. As another example, machine-learning approaches can be applied to solve the partial differential equations (PDEs) of DSM2 or other process-based models directly to circumvent the issues with the conventional numerical solvers for these equations. There have been successful applications on this front. Reference [50] proposed physics-informed neural networks with specifically designed loss functions to solve nonlinear PDEs. During training, the neural network learns to approximate the target function in the PDEs. Reference [51] designed a convolutional LSTM network called "FiniteNet" to solve PDEs, which can capture the temporal behavior with LSTM structure and the local spatial behavior by convolution. Moreover, their training strategy mimics conventional PDE-solving algorithms. They claim that the FiniteNet can reduce the root mean squared error by a factor of 2 to 3 compared to conventional algorithms. Similarly, reference [52] proposed to discretize the PDEs and train an ANN for grid-wise estimation. Generally speaking, the physics-informed neural networks can learn the underlying physical laws embedded in the training data in an interpretable way and are less prone to overfitting [50].

4.4. Future Work

Despite its practical and scientific values, the current study has a few limitations. Firstly, DSM2 is capable of providing salinity simulations on 15-min time steps. While in line with all previous relevant studies [19–23], the temporal scale of the current study is daily. The coarse-scale daily salinity simulations are aggregated from the fine-scale DSM2 simulations by averaging. The sub-daily tidal variations in salinity which can be important to certain salinity management practices are not captured. One follow-up study of the current work is to emulate DSM2 in terms of simulating sub-daily salinity variations. In that case, the 15-min salinity simulations from DSM2 will be used as the emulation target. Input features and model structures for the finer-scale emulation will be explored. We plan to use the same architectures explored in the current study but increase the output layer size by a factor of 96 (daily to 15-min conversion). Our preliminary test shows that the training time is expected to double, which is still reasonably efficient. Secondly, the current study adopts early stopping and random split of training and testing datasets to prevent the proposed machine-learning models from overfitting. In our follow-up work, we plan to explore other regularization techniques including dropout and data augmentation. Dropout is the process of randomly dropping some layer outputs in neural networks during the training. It has been shown to reduce overfitting in many applications [53–55]. Data augmentation is a set of techniques that artificially corrupt the training data to enlarge

and diversity them to better represent extreme conditions and possible future conditions, aiming to reduce overfitting and thus improve the generalization ability of the trained neural networks. Traditionally, perturbation of training data (e.g., increase or decrease by a certain percentage) and inclusion of synthetic but plausible operational scenarios have been applied to augment the historical simulation scenarios [5,21]. In another follow-up work, we will explore these perturbation techniques. Additional techniques including temporal shifting and jittering will also be investigated.

Lastly, the current study focuses on DSM2-simulation salinity which is in obedience to the advection–dispersion process defined in DSM2 and lacks noisy variations in real-world salinity. Therefore, the machine-learning models proposed in the study are capable of mimicking DSM2's advection–dispersion procedure in generating salinity, but not the actual salinity level in the field. To address this issue, we will conduct transfer learning [56,57] in a follow-up study. Transfer learning is a popular machine-learning method where a model gains knowledge from one task (i.e., DSM2-simulated salinity in this case) and applies the knowledge to a different yet related task (i.e., observed salinity). Based on the basic salinity response learned from the simulated dataset, we can augment the pre-trained model with additional observed dataset to capture the additional information without starting from scratch. As a result, the training time can be reduced and the transferred model is less prone to overfitting. In this follow-up work, we consider that (1) DSM2 salinity simulations after the augmentation mentioned above are large and general enough; and (2) the neural networks trained on them are reliable and can effectively serve as generic models for real-world salinity estimation. To be specific, we will reuse the intermediate layers straight from the neural networks trained on DSM2 simulations while replacing the output layer with a new randomly initialized fully connected layer. We will explore two different ways of customizing the pre-trained models: (1) training the output layer only; during training, all the hidden layers are fixed and directly use the extracted features for real-world salinity emulation; and (2) end-to-end fine-tuning. During training, both the hidden layers and the newly added output layer are optimized together for the new task, while a higher learning rate is applied to the output layer as it is being trained from scratch while other layers are pre-trained. The findings will be reported in a follow-up paper.

5. Conclusions

With the ever-increasing number of water management options, operational scenarios, and climate conditions to be considered in the future, there is a growing need to emulate complex process-based models using machine-learning techniques to facilitate the running of a vast number of scenario permutations for screening purposes. This study exemplifies the development and application of machine-learning-based emulators for a process-based model, DSM2, in salinity estimation in the Sacramento-San Joaquin Delta of California. The investigation shows that the emulators can efficiently and effectively mimic DSM2 in salinity simulation across the Delta. The low computational requirement of the trained emulators makes these emulators appealing tools to supplement the existing process-based model in accurately estimating salinity to inform both real-time and long-term water management decision-making.

Author Contributions: Conceptualization, P.S., Z.B., Z.D. and M.H.; Data curation, B.T., R.H., P.N. and Y.Z.; Funding acquisition, P.S.; Investigation, S.Q. and M.H.; Methodology, S.Q., M.H., Z.B., Z.D., P.S. and J.A.; Project administration, P.S.; Software, S.Q.; Validation, M.H., R.H., P.N. and Y.Z.; Visualization, S.Q., M.H.; Writing—original draft, S.Q, M.H., Y.Z., B.T., R.H., P.N. and J.A.; Writing—review and editing, Z.B., P.S. and J.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the California Department of Water Resources and the University of California, Davis.

Data Availability Statement: Data availability is described in Appendix A.

Acknowledgments: The authors would like to thank their colleague Francis Chung for his comments on this work. The authors would also like to thank the three anonymous reviewers for providing thoughtful and insightful comments that helped to improve the quality of this study. The views expressed in this paper are those of the authors, and not of the State of California.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Data Sources

The data presented in this study are available online on Google Drive (accessed on 9 June 2022) on request from the corresponding author. The data are not publicly available due to privacy.

Appendix B. LSTM Layer

Figure A1 shows the detailed architecture of a typical LSTM layer used in the LSTM model in Figure 3, proposed in Section 2.3.2.

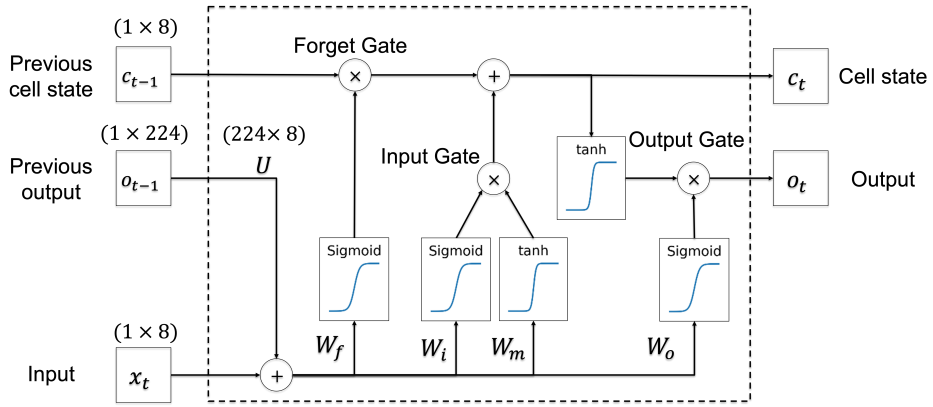

Figure A1. Details inside an LSTM layer.

Appendix C. Performance of the MLP (Ungrouped) Model

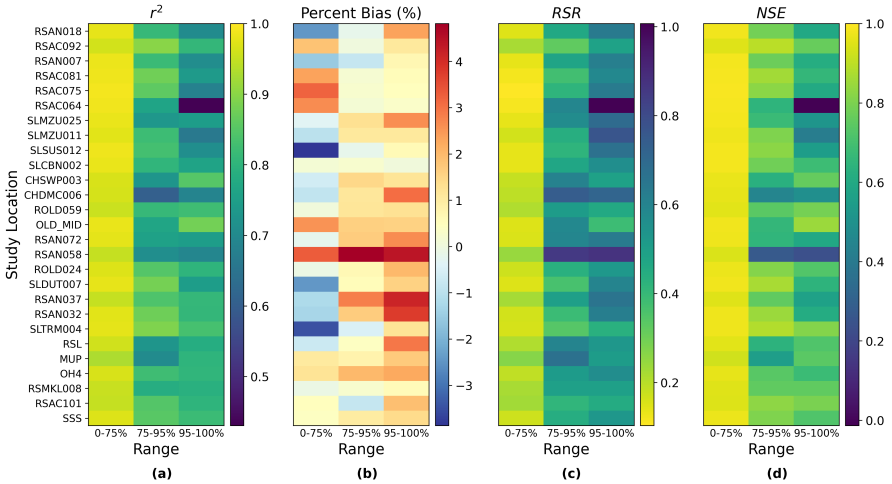

Figure A2. Model (ungrouped MLP) performance measured by (**a**) r^2, (**b**) Percent Bias, (**c**) RSR, and (**d**) NSE at different salinity ranges: low-middle range (lowest 75%), high range (75 to 95 percentile), and extreme high range (highest 5%) at the study locations.

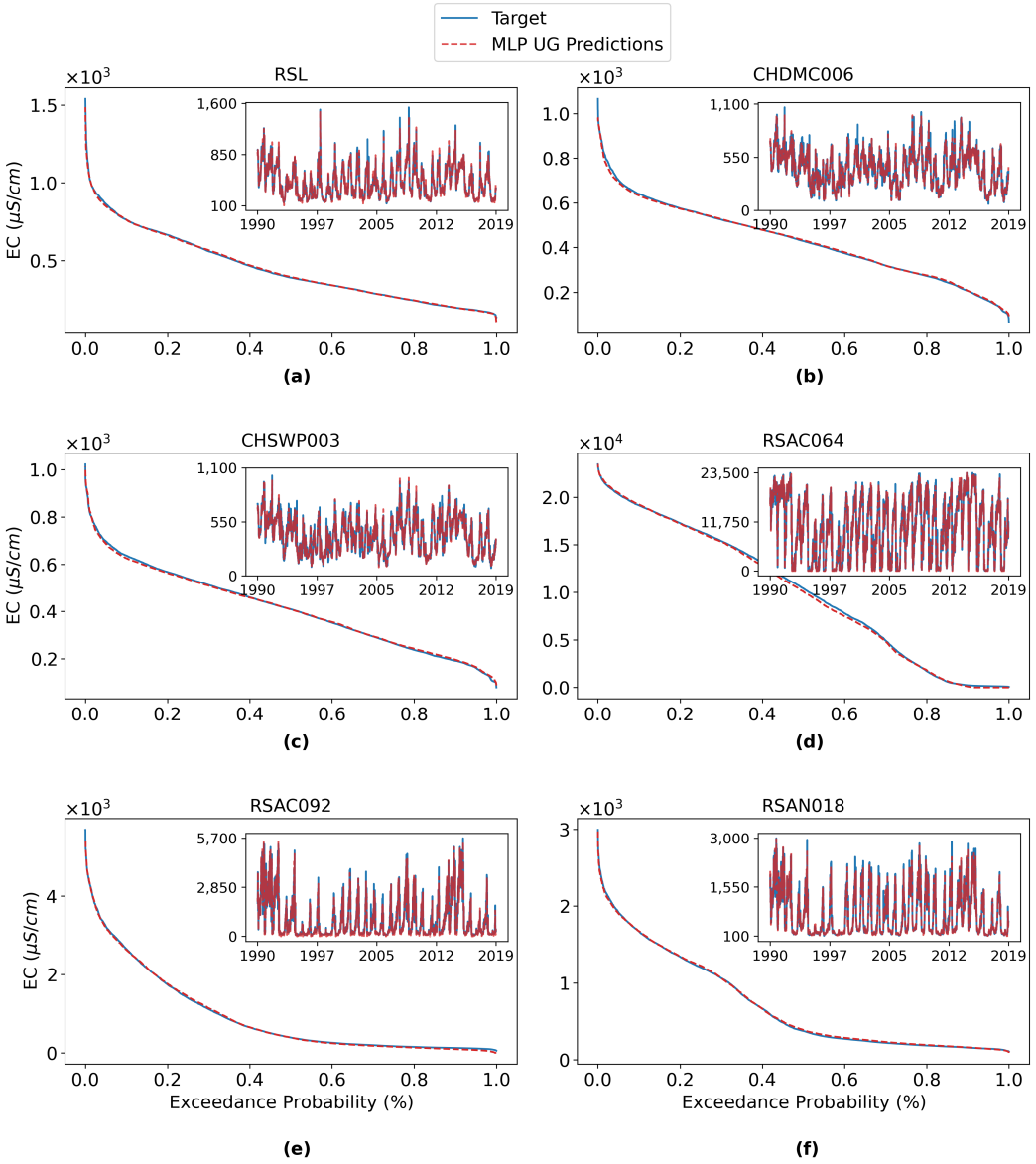

Figure A3. Exceedance probability curves and time series plots of salinity at six key study locations, including (**a**) RSL, (**b**) CHDMC006, (**c**) CHSWP003, (**d**) RSAC064, (**e**) RSAC092 and (**f**) RSAN018, comparing the ungrouped MLP model with the process-based DSM2 outputs.

Appendix D. Detailed Values for Metric Computation

Table A1. Values of $\sum_{t=1}^{T} |(S_{ref}^{t} - \overline{S_{ref}}) \times (S_{ANN}^{t} - \overline{S_{ANN}})|$, numerator in r^2, for six key stations. S_{ref}^{t} and S_{ANN}^{t} are linearly normalized according to Equation (2).

Key Station Name	0∼75%	75∼95%	95∼100%
RSL	53.74	3.72	3.33
CHDMC006	102.04	2.89	2.56
CHSWP003	108.83	3.67	2.43
RSAC064	438.44	4.77	0.47
RSAC092	29.18	18.39	4.07
RSAN018	86.00	9.41	2.38

Table A2. Values of $\sum_{t=1}^{T}(S_{ref}^{t} - S_{ANN}^{t})$ for six key stations. S_{ref}^{t} and S_{ANN}^{t} are linearly normalized according to Equation (2).

Key Station Name	0∼75%	75∼95%	95∼100%
RSL	−4.43	−14.04	−6.24
CHDMC006	−7.00	−12.39	−7.95
CHSWP003	−8.22	−14.25	−5.43
RSAC064	22.09	6.72	−2.31
RSAC092	−24.10	−13.12	−13.32
RSAN018	44.06	4.27	−8.91

Table A3. Values of $\sum_{t=1}^{T} S_{ref}^{t}$ for six key stations. S_{ref}^{t} is linearly normalized according to Equation (2).

Key Station Name	0∼75%	75∼95%	95∼100%
RSL	1814.60	1007.23	345.78
CHDMC006	2720.57	1241.83	405.82
CHSWP003	2671.24	1281.44	411.13
RSAC064	2477.64	1686.43	490.15
RSAC092	560.01	840.55	369.71
RSAN018	1083.53	1093.23	393.68

Table A4. Values of MSE ($\sum_{t=1}^{T}(S_{ref}^{t} - S_{ANN}^{t})^{2}$) for six key stations. S_{ref}^{t} and S_{ANN}^{t} are linearly normalized according to Equation (2).

Key Station Name	0∼75%	75∼95%	95∼100%
RSL	0.97	0.79	0.89
CHDMC006	0.98	0.66	0.76
CHSWP003	0.98	0.78	0.89
RSAC064	1.00	0.83	0.48
RSAC092	0.94	0.92	0.78
RSAN018	0.97	0.85	0.73

Table A5. Standard deviation of target (σ_{ref}) for six key stations. S_{ref}^{t} is linearly normalized according to Equation (2).

Key Station Name	0∼75%	75∼95%	95∼100%
RSL	0.08	0.04	0.08
CHDMC006	0.11	0.04	0.07
CHSWP003	0.12	0.04	0.07
RSAC064	0.23	0.05	0.03
RSAC092	0.06	0.09	0.09
RSAN018	0.10	0.07	0.07

Table A6. Standard deviation of LSTM G model predictions (σ_{ANN}) for six key stations. S_{ANN}^{t} is linearly normalized according to Equation (2).

Key Station Name	0∼75%	75∼95%	95∼100%
RSL	0.08	0.04	0.08
CHDMC006	0.11	0.04	0.07
CHSWP003	0.12	0.05	0.07
RSAC064	0.23	0.05	0.04
RSAC092	0.06	0.10	0.09
RSAN018	0.10	0.07	0.07

Appendix E. Numbers of Parameters in Proposed Architectures

Table A7. Total numbers of parameters of each architecture proposed, including ungrouped version (one model for 28 stations) and grouped version (three models, one for each station group).

Architecture	Ungrouped Version	Grouped Version		
		Northern Delta	Interior Delta	Western Delta
MLP	46,676	3651	19,810	14,971
LSTM	224,028	4203	60,270	38,643
GRU	170,268	3243	45,934	29,491
ResNet	79,604	7579	36,498	28,179

Appendix F. Training Time of Proposed Architectures

We train the models on the Google Colaboratory, a public platform, which provides various GPUs for use. We count the training time of each model and summarize them in Table A8. Note that the training time can be impacted by the randomness in model initialization due to the early stopping strategy we used, and thus may differ considerably from the data in Table A8, which are collected from a single test.

Table A8. Total training time (in minutes) of each architecture proposed, including ungrouped version (one model for 28 stations) and grouped version (three models, one for each station group).

Architecture	Ungrouped Version	Grouped Version		
		Northern Delta	Interior Delta	Western Delta
MLP	3.7	4.0	2.6	2.5
LSTM	22.5	23.1	51.1	47.2
GRU	10.0	14.0	26.6	24.9
ResNet	2.0	2.2	2.5	2.5

Appendix G. Architecture of Deeper and Wider MLP and LSTM Networks

Figure A4 shows the detailed structures of deeper and wider MLP ANNs and Figure A5 shows the detailed structures of deeper and wider LSTM ANNs explored in Section 4.1. Table A9 lists the total number of parameters in each model.

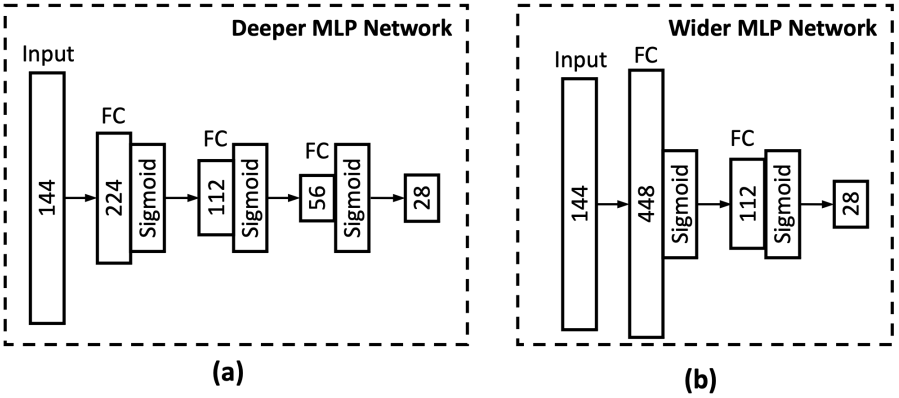

Figure A4. Proposed MLP ANN with increased (**a**) depth; (**b**) width.

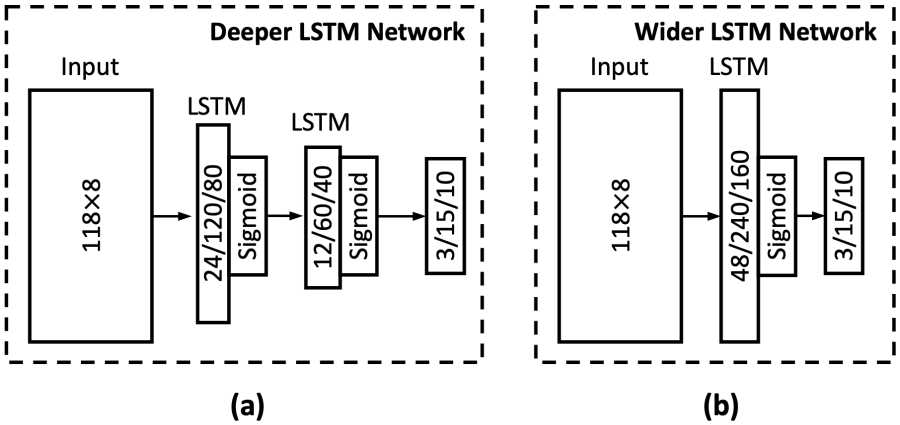

Figure A5. Proposed LSTM Network with increased (**a**) depth; (**b**) width. The three numbers in the LSTM layers and the output layer correspond to proposed LSTM model for northern, interior, and western Delta groups, respectively.

Table A9. Total numbers of parameters in the deeper and wider MLP model (ungrouped) and LSTM network (grouped) tested in the study.

	MLP, Ungrouped Ver.	LSTM, Grouped Ver.		
		Northern Delta	Interior Delta	Western Delta
Deeper Structure	65,604	5943	97,342	61,567
Wider Structure	118,412	13,011	220,878	139,227

Appendix H. Years Manually Picked for Training

A detailed summary of Water Year Type can be found here. Table A10 lists the water years picked for training in "Manual" case of Section 4.2.

Table A10. Years of data manually selected for training.

Water Year Type	Year
Wet	1995, 1997, 1998, 2006, 2019
Above Normal	1993, 2000, 2003, 2005
Below Normal	2010, 2012, 2016, 2018
Dry	2001, 2002, 2009, 2013
Critical	1991, 1994, 2008, 2015

Appendix I. Detailed Group-Wise Performance of Each Experiment

Overall, according to Figures A6–A11, with a shorter input memory length, both our proposed MLP and LSTM models show the best average test performance in the northern Delta group and poorest performance in the interior Delta group. A shorter input memory length indicates a less complex model with fewer parameters and hence lower potential of overfitting. In this case, although the amount of information provided by the dataset of the northern Delta group is less than the other groups, it seems to be sufficient to guide such simpler MLP and LSTM models. For the stations in the interior (south) Delta group, similar to what we concluded according to Figure 11, the span of their locations is the largest and their salinity levels are influenced by more variables, including both seawater and freshwater, making it harder to accurately estimate their salinity levels. Moreover, the group-wise results in Figures A6–A11 reveal the fact that the salinity levels of stations belonging to the interior Delta are influenced by the inputs in a longer term.

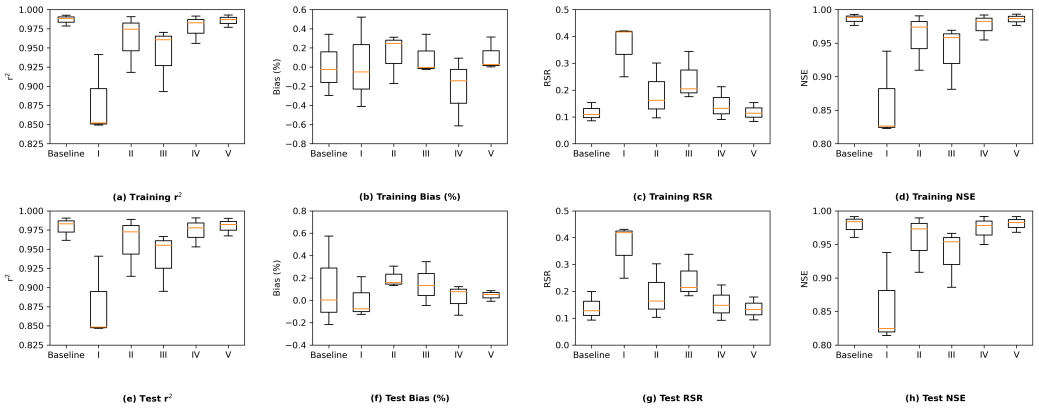

Figure A6. Box and whisker plots of four types of metrics consisting of r^2, percent bias, RSR, and NSE on stations belonging to the northern Delta group during the training (panels (**a**–**d**)) and test (panels (**e**–**h**)) phases of the grouped MLP model under five scenarios: baseline and four different memory lengths as presented in Table 1.

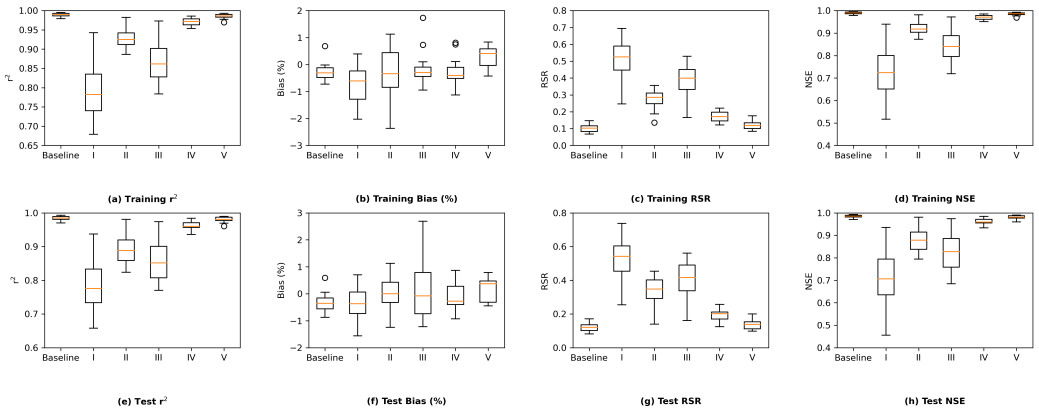

Figure A7. Box and whisker plots of four types of metrics consisting of r^2, percent bias, RSR, and NSE on stations belonging to the interior Delta group during the training (panels (**a**–**d**)) and test (panels (**e**–**h**)) phases of the grouped (G) MLP model under five scenarios: baseline and four different memory lengths as presented in Table 1.

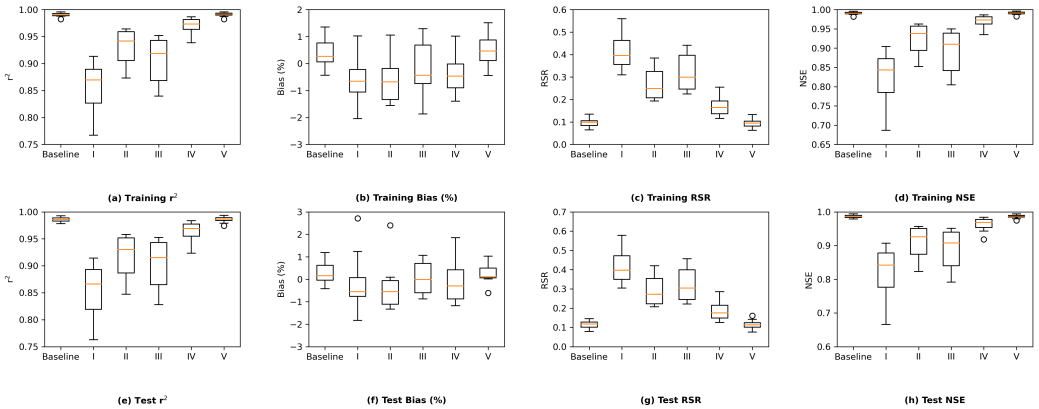

Figure A8. Box and whisker plots of four types of metrics consisting of r^2, percent bias, RSR, and NSE on stations belonging to the western Delta group during the training (panels (**a**–**d**)) and test (panels (**e**–**h**)) phases of the grouped (G) MLP model under five scenarios: baseline and four different memory lengths as presented in Table 1.

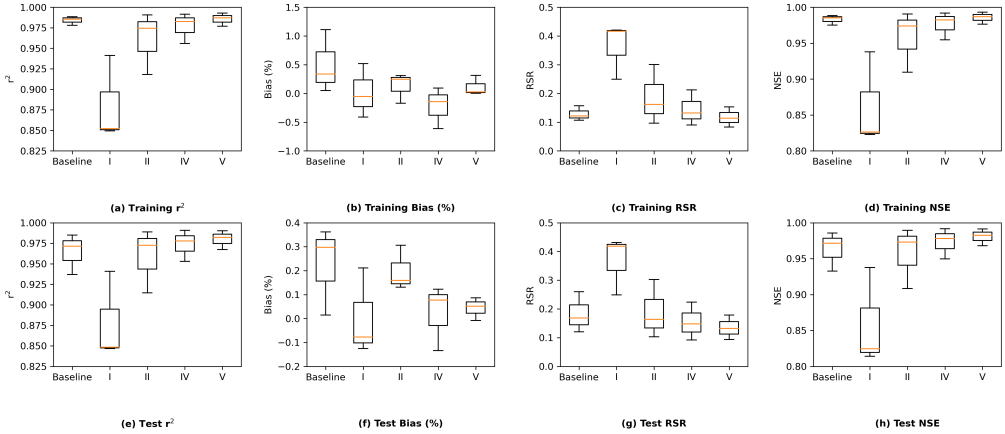

Figure A9. Box and whisker plots of four types of metrics consisting of r^2, percent bias, RSR, and NSE on stations belonging to the northern Delta group during the training (panels (**a**–**d**)) and test (panels (**e**–**h**)) phases of the grouped (G) LSTM model under five scenarios: baseline and four different memory lengths as presented in Table 1.

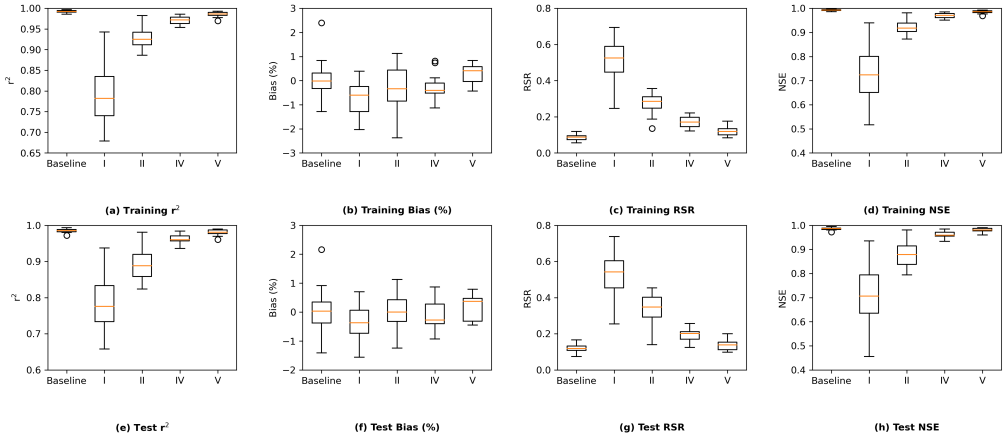

Figure A10. Box and whisker plots of four types of metrics consisting of r^2, percent bias, RSR, and NSE on stations belonging to the interior Delta group during the training (panels (**a**–**d**)) and test (panels (**e**–**h**)) phases of the grouped (G) LSTM model under five scenarios: baseline and four different memory lengths as presented in Table 1.

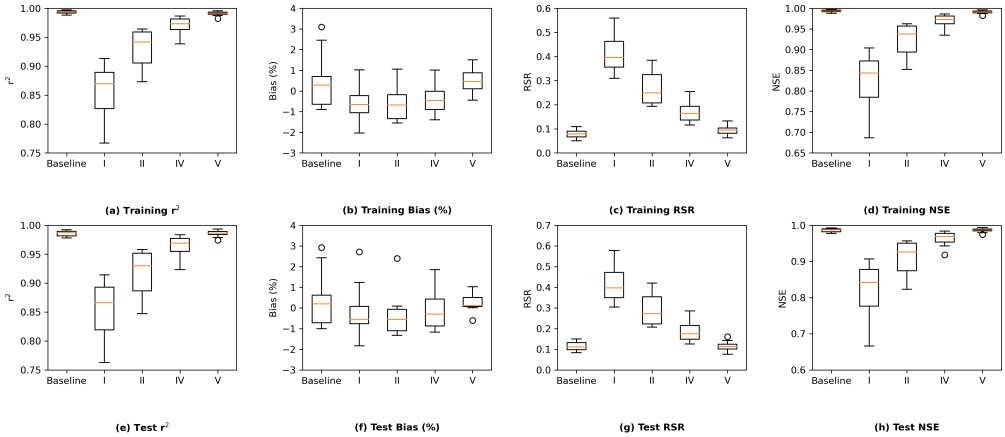

Figure A11. Box and whisker plots of four types of metrics consisting of r^2, percent bias, RSR, and NSE on stations belonging to the western Delta group during the training (panels (**a**–**d**)) and test (panels (**e**–**h**)) phases of the grouped (G) LSTM model under five scenarios: baseline and four different memory lengths as presented in Table 1.

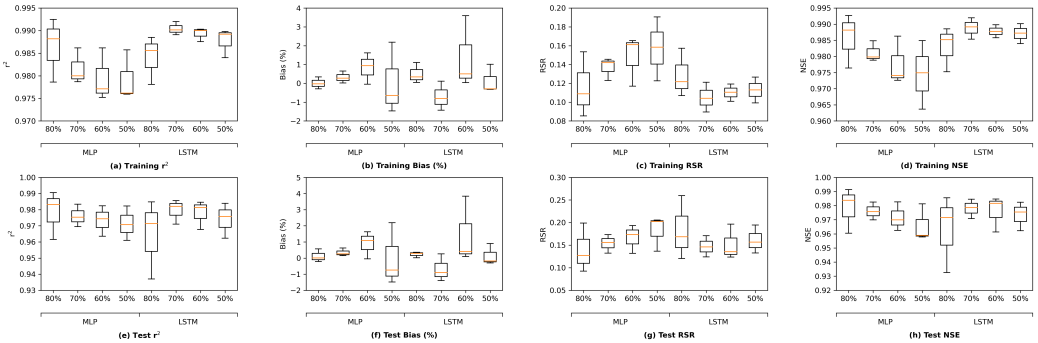

Figure A12. Box and whisker plots of four types of metrics consisting of r^2, percent bias, RSR, and NSE on stations belonging to the northern Delta group during the training (panels (**a**–**d**)) and test (panels (**e**–**h**)) phases of the grouped (G) MLP and LSTM models under four scenarios with varying lengths of training data: 50%, 60%, 70%, and 80% of all the data available.

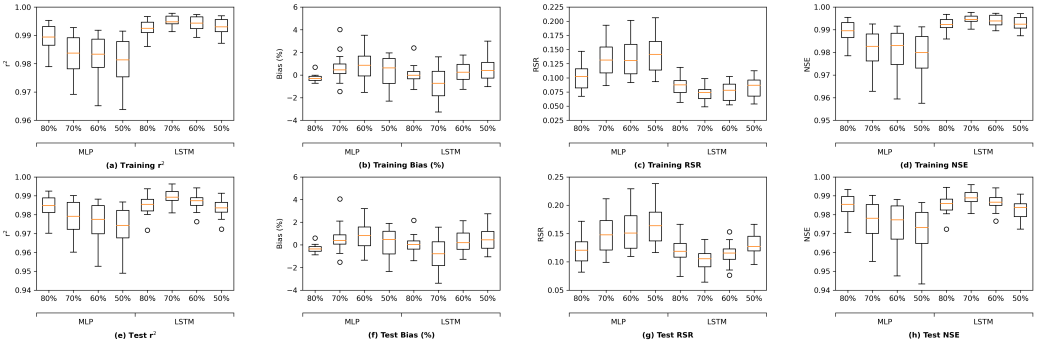

Figure A13. Box and whisker plots of four types of metrics consisting of r^2, percent bias, RSR, and NSE on stations belonging to the interior Delta group during the training (panels (**a**–**d**)) and test (panels (**e**–**h**)) phases of the grouped (G) MLP and LSTM models under four scenarios with varying lengths of training data: 50%, 60%, 70%, and 80% of all the data available.

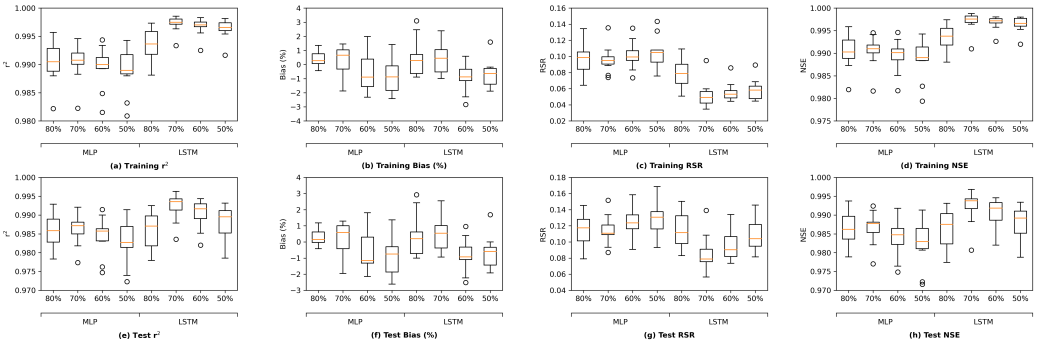

Figure A14. Box and whisker plots of four types of metrics consisting of r^2, percent bias, RSR, and NSE on stations belonging to the western Delta group during the training (panels (**a**–**d**)) and test (panels (**e**–**h**)) phases of the grouped (G) MLP and LSTM models under four scenarios with varying lengths of training data: 50%, 60%, 70%, and 80% of all the data available.

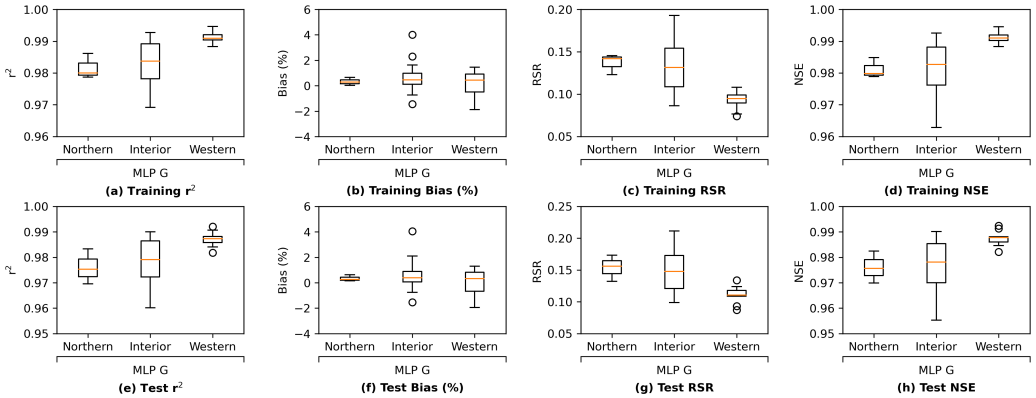

Figure A15. Box and whisker plots of four types of metrics consisting of r^2, percent bias, RSR, and NSE on stations belonging to the northern, interior, and western Delta groups during the training (panels (**a**–**d**)) and test (panels (**e**–**h**)) phases of the grouped (G) MLP model.

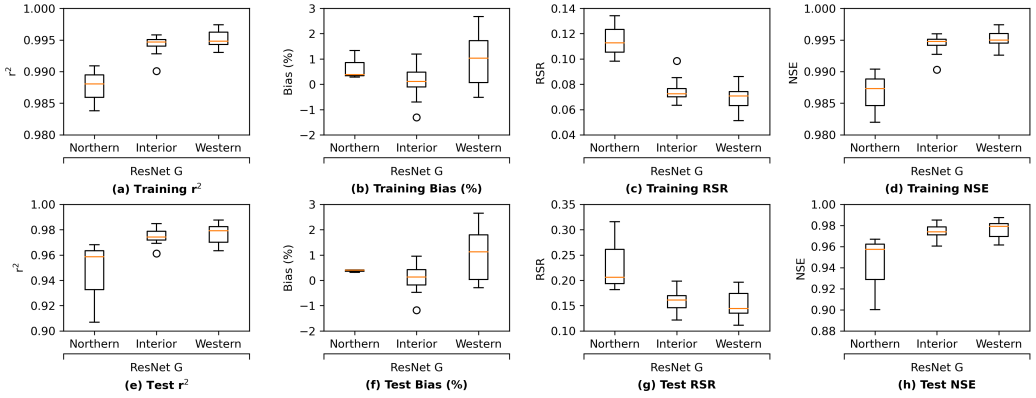

Figure A16. Box and whisker plots of four types of metrics consisting of r^2, percent bias, RSR, and NSE on stations belonging to the northern, interior, and western Delta groups during the training (panels (**a**–**d**)) and test (panels (**e**–**h**)) phases of the grouped (G) ResNet model.

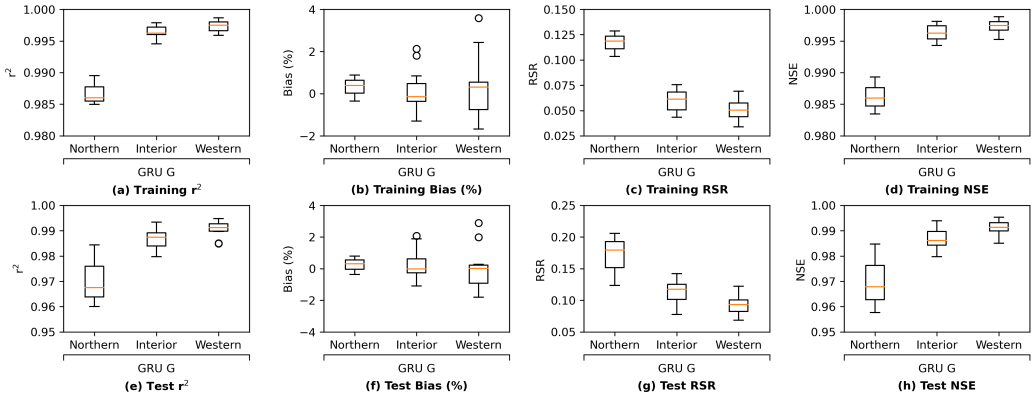

Figure A17. Box and whisker plots of four types of metrics consisting of r^2, percent bias, RSR, and NSE on stations belonging to the northern, interior, and western Delta groups during the training (panels (**a**–**d**)) and test (panels (**e**–**h**)) phases of the grouped (G) GRU model.

Appendix J. Detailed Station-Wise Performance of the Proposed MLP, ResNet, and GRU Models for Interior Delta

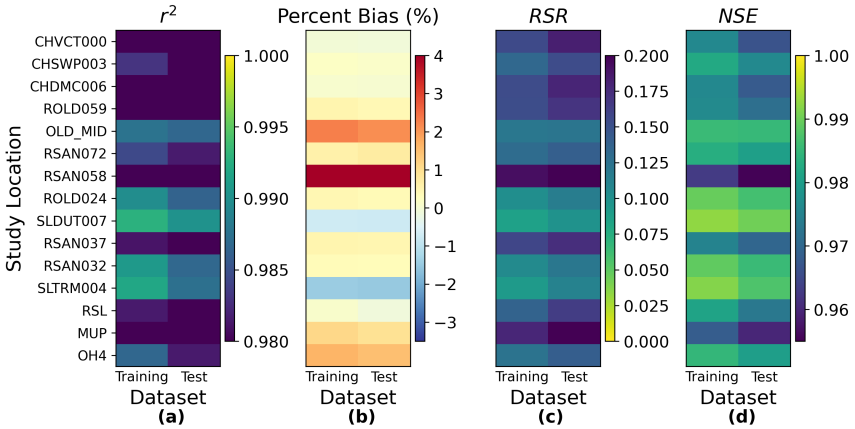

Figure A18. Model (grouped MLP) performance measured by (**a**) r^2, (**b**) Percent Bias, (**c**) RSR, and (**d**) NSE on training and test set at the study locations belonging to the interior Delta group

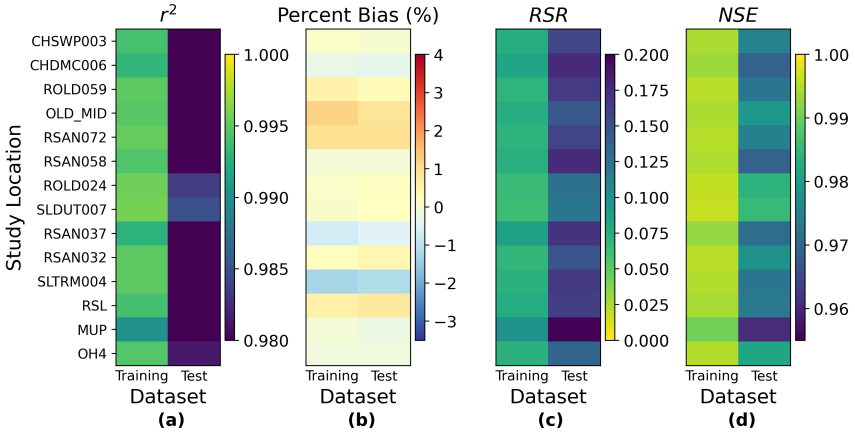

Figure A19. Model (grouped ResNet) performance measured by (**a**) r^2, (**b**) Percent Bias, (**c**) RSR, and (**d**) NSE on training and test set at the study locations belonging to the interior Delta group.

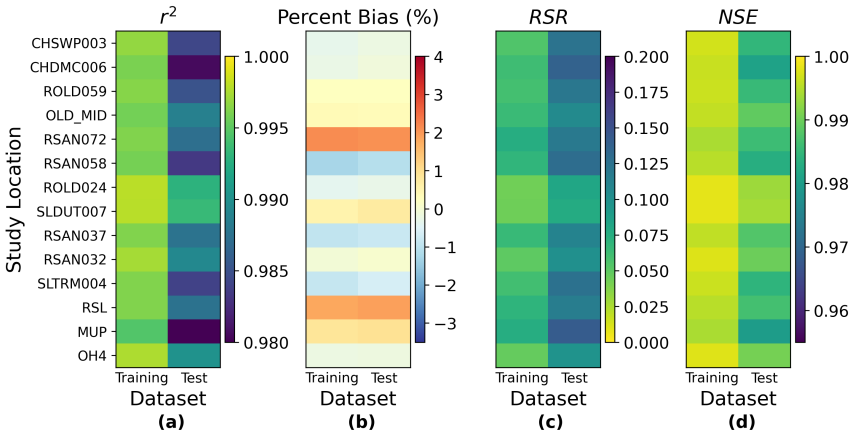

Figure A20. Model (grouped GRU) performance measured by (**a**) r^2, (**b**) Percent Bias, (**c**) RSR, and (**d**) NSE on training and test set at the study locations belonging to the interior Delta group.

References

1. Alber, M. A conceptual model of estuarine freshwater inflow management. *Estuaries* **2002**, *25*, 1246–1261. [CrossRef]
2. CNRA. *Water Resilience Portfolio*; California Natural Resources Agency: Sacramento, CA, USA, 2020; p. 141.
3. Wilbur, R.; Munevar, A. Integration of CALSIM and Artificial Neural Networks Models for Sacramento-San Joaquin Delta Flow-Salinity Relationships. In *Methodology for Flow and Salinity Estimates in the Sacramento-San Joaquin Delta and Suisun Marsh: 22th Annual Progress Report*; U.S. Fish and Wildlife Service: Washington, DC, USA, 2001.
4. Mierzwa, M. CALSIM versus DSM2 ANN and G-model Comparisons. In *Methodology for Flow and Salinity Estimates in the Sacramento-San Joaquin Delta and Suisun Marsh: 23rd Annual Progress Report*; U.S. Fish and Wildlife Service: Washington, DC, USA, 2002.
5. Seneviratne, S.; Wu, S. Enhanced Development of Flow-Salinity Relationships in the Delta Using Artificial Neural Networks: Incorporating Tidal Influence. In *Methodology for Flow and Salinity Estimates in the Sacramento-San Joaquin Delta and Suisun Marsh: 28th Annual Progress Report*; U.S. Fish and Wildlife Service: Washington, DC, USA, 2007.
6. Ray, P.; Wi, S.; Schwarz, A.; Correa, M.; He, M.; Brown, C. Vulnerability and risk: Climate change and water supply from California's Central Valley water system. *Clim. Chang.* **2020**, *161*, 177–199. [CrossRef]
7. CDWR. Minimum Delta Outflow Program. In *Methodology for Flow and Salinity Estimates in the Sacramento-San Joaquin Delta and Suisun Marsh: 11th Annual Progress Report*; U.S. Fish and Wildlife Service: Washington, DC, USA, 1990.
8. Denton, R.A. Accounting for antecedent conditions in seawater intrusion modeling—Applications for the San Francisco Bay-Delta. In Proceedings of the Hydraulic Engineering, San Francisco, CA, USA, 25–30 July 1993; ASCE: Reston, VA, USA, 1993; pp. 448–453.
9. Hutton, P.H.; Rath, J.S.; Chen, L.; Ungs, M.J.; Roy, S.B. Nine decades of salinity observations in the San Francisco Bay and Delta: Modeling and trend evaluations. *J. Water Resour. Plan. Manag.* **2016**, *142*, 04015069. [CrossRef]
10. CDWR. Calibration and verification of DWRDSM. In *Methodology for Flow and Salinity Estimates in the Sacramento-San Joaquin Delta and Suisun Marsh: 12th Annual Progress Report*; U.S. Fish and Wildlife Service: Washington, DC, USA, 1991.
11. Cheng, R.T.; Casulli, V.; Gartner, J.W. Tidal, residual, intertidal mudflat (TRIM) model and its applications to San Francisco Bay, California. *Estuar. Coast. Shelf Sci.* **1993**, *36*, 235–280. [CrossRef]
12. DeGeorge, J.F. *A Multi-Dimensional Finite Element Transport Model Utilizing a Characteristic-Galerkin Algorithm*; University of California: Davis, CA, USA, 1996.
13. MacWilliams, M.; Bever, A.J.; Foresman, E. 3-D simulations of the San Francisco Estuary with subgrid bathymetry to explore long-term trends in salinity distribution and fish abundance. *San Fr. Estuary Watershed Sci.* **2016**, *14*, 3. [CrossRef]
14. Chao, Y.; Farrara, J.D.; Zhang, H.; Zhang, Y.J.; Ateljevich, E.; Chai, F.; Davis, C.O.; Dugdale, R.; Wilkerson, F. Development, implementation, and validation of a modeling system for the San Francisco Bay and Estuary. *Estuar. Coast. Shelf Sci.* **2017**, *194*, 40–56. [CrossRef]
15. MacWilliams, M.L.; Ateljevich, E.S.; Monismith, S.G.; Enright, C. An overview of multi-dimensional models of the Sacramento–San Joaquin Delta. *San Fr. Estuary Watershed Sci.* **2016**, *14*, 2.
16. Barnes, G.W., Jr.; Chung, F.I. Operational planning for California water system. *J. Water Resour. Plan. Manag.* **1986**, *112*, 71–86. [CrossRef]
17. Sandhu, N.; Finch, R. Application of Artificial Neural Networks to the Sacramento-San Joaquin Delta. In Proceedings of the Estuarine and Coastal Modeling, San Diego, CA, USA, 26–28 October 1995; ASCE: Reston, VA, USA, 1995. pp. 490–504.
18. Draper, A.J.; Munévar, A.; Arora, S.K.; Reyes, E.; Parker, N.L.; Chung, F.I.; Peterson, L.E. CalSim: Generalized model for reservoir system analysis. *J. Water Resour. Plan. Manag.* **2004**, *130*, 480–489. [CrossRef]
19. Jayasundara, N.C.; Seneviratne, S.A.; Reyes, E.; Chung, F.I. Artificial Neural Network for Sacramento–San Joaquin Delta flow–salinity relationship for CalSim 3.0. *J. Water Resour. Plan. Manag.* **2020**, *146*, 04020015. [CrossRef]
20. Rath, J.S.; Hutton, P.H.; Chen, L.; Roy, S.B. A hybrid empirical-Bayesian Artificial Neural Network model of salinity in the San Francisco Bay-Delta estuary. *Environ. Model. Softw.* **2017**, *93*, 193–208. [CrossRef]
21. Chen, L.; Roy, S.B.; Hutton, P.H. Emulation of a process-based estuarine hydrodynamic model. *Hydrol. Sci. J.* **2018**, *63*, 783–802. [CrossRef]
22. He, M.; Zhong, L.; Sandhu, P.; Zhou, Y. Emulation of a process-based salinity generator for the sacramento–san joaquin delta of california via deep learning. *Water* **2020**, *12*, 2088. [CrossRef]
23. Qi, S.; Bai, Z.; Ding, Z.; Jayasundara, N.; He, M.; Sandhu, P.; Seneviratne, S.; Kadir, T. Enhanced Artificial Neural Networks for Salinity Estimation and Forecasting in the Sacramento-San Joaquin Delta of California. *J. Water Resour. Plan. Manag.* **2021**, *147*, 04021069. [CrossRef]
24. Caruana, R. Learning many related tasks at the same time with backpropagation. *Adv. Neural Inf. Process. Syst.* **1994**, *7*, 658–664.
25. Sommer, T.; Mejia, F. A place to call home: a synthesis of Delta Smelt habitat in the upper San Francisco Estuary. *San Franc. Estuary Watershed Sci.* **2013**, *11*, 1–25. [CrossRef]
26. CSWRCB. *Water Right Decision 1641*; California State Water Resources Control Board: Sacramento, CA, USA, 1999.
27. U.S. Fish and Wildlife Service. *Formal Endangered Species Act Consultation on the Proposed Coordinated Operations of the Central Valley Project (CVP) and State Water Project (SWP)*; U.S. Fish and Wildlife Service: Washington, DC, USA, 2008; p. 410.
28. Maier, H.R.; Dandy, G.C. Neural networks for the prediction and forecasting of water resources variables: a review of modelling issues and applications. *Environ. Model. Softw.* **2000**, *15*, 101–124. [CrossRef]

29. Shen, C. A transdisciplinary review of deep learning research and its relevance for water resources scientists. *Water Resour. Res.* **2018**, *54*, 8558–8593. [CrossRef]
30. Buongiorno Nardelli, B. A deep learning network to retrieve ocean hydrographic profiles from combined satellite and in situ measurements. *Remote Sens.* **2020**, *12*, 3151. [CrossRef]
31. Song, T.; Wang, Z.; Xie, P.; Han, N.; Jiang, J.; Xu, D. A novel dual path gated recurrent unit model for sea surface salinity prediction. *J. Atmos. Ocean. Technol.* **2020**, *37*, 317–325. [CrossRef]
32. Zhang, D.; Peng, Q.; Lin, J.; Wang, D.; Liu, X.; Zhuang, J. Simulating reservoir operation using a recurrent neural network algorithm. *Water* **2019**, *11*, 865. [CrossRef]
33. Thai-Nghe, N.; Thanh-Hai, N.; Chi Ngon, N. Deep learning approach for forecasting water quality in IoT systems. *Int. J. Adv. Comput. Sci. Appl.* **2020**, *11*, 686–693. [CrossRef]
34. Hochreiter, S.; Schmidhuber, J. Long-Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
35. Sagheer, A.; Kotb, M. Time series forecasting of petroleum production using deep LSTM recurrent networks. *Neurocomputing* **2019**, *323*, 203–213. [CrossRef]
36. Sun, Q.; Wan, J.; Liu, S. Estimation of sea level variability in the China Sea and its vicinity using the SARIMA and LSTM models. *IEEE J. Sel. Top. Appl. Earth Obs. Remote. Sens.* **2020**, *13*, 3317–3326. [CrossRef]
37. Cho, K.; Van Merriënboer, B.; Bahdanau, D.; Bengio, Y. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv* **2014**, arXiv:1409.1259.
38. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
39. Smyl, S.; Kuber, K. Data preprocessing and augmentation for multiple short time series forecasting with recurrent neural networks. In Proceedings of the 36th International Symposium on Forecasting, Santander, Spain, 19–22 June 2016.
40. Chen, K.; Chen, K.; Wang, Q.; He, Z.; Hu, J.; He, J. Short-term load forecasting with deep residual networks. *IEEE Trans. Smart Grid* **2018**, *10*, 3943–3952. [CrossRef]
41. Choi, H.; Ryu, S.; Kim, H. Short-term load forecasting based on ResNet and LSTM. In Proceedings of the 2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), Aalborg, Denmark, 29–31 October 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–6.
42. Wan, R.; Mei, S.; Wang, J.; Liu, M.; Yang, F. Multivariate temporal convolutional network: A deep neural networks approach for multivariate time series forecasting. *Electronics* **2019**, *8*, 876. [CrossRef]
43. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
44. Kawaguchi, K.; Kaelbling, L.P.; Bengio, Y. Generalization in deep learning. *arXiv* **2017**, arXiv:1710.05468.
45. Null, S.E.; Viers, J.H. In bad waters: Water year classification in nonstationary climates. *Water Resour. Res.* **2013**, *49*, 1137–1148. [CrossRef]
46. CDWR. DSM2 Model Development. In *Methodology for Flow and Salinity Estimates in the Sacramento-San Joaquin Delta and Suisun Marsh: 18th Annual Progress Report*; U.S. Department of Education: Washington, DC, USA, 1997.
47. Nader-Tehrani, P.; Shrestha, B. DSM2 Calibration. In *Methodology for Flow and Salinity Estimates in the Sacramento-San Joaquin Delta and Suisun Marsh: 21st Annual Progress Report*; U.S. Department of Education: Washington, DC, USA, 2000.
48. CDWR. *DSM2 Recalibration*; Technical Report; CDWR: Sacramento, CA, USA, 2009.
49. He, M.; Zhou, Y.; Bradley, T.; Nader-Tehrani, P.; Sandhu, P. DSM2 v8.2.0 Calibration. In *Methodology for Flow and Salinity Estimates in the Sacramento-San Joaquin Delta and Suisun Marsh: 43rd Annual Progress Report*; U.S. Department of Education: Washington, DC, USA, 2022.
50. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **2019**, *378*, 686–707. [CrossRef]
51. Stevens, B.; Colonius, T. Finitenet: A fully convolutional lstm network architecture for time-dependent partial differential equations. *arXiv* **2020**, arXiv:2002.03014.
52. Mishra, S. A machine learning framework for data driven acceleration of computations of differential equations. *arXiv* **2018**, arXiv:1807.09519.
53. Baldi, P.; Sadowski, P.J. Understanding dropout. *Adv. Neural Inf. Process. Syst.* **2013**, *26*, 1–9.
54. Wager, S.; Wang, S.; Liang, P.S. Dropout training as adaptive regularization. *Adv. Neural Inf. Process. Syst.* **2013**, *26*, 1–9.
55. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
56. Bozinovski, S.; Fulgosi, A. The influence of pattern similarity and transfer of learning upon training of a base perceptron B2. (original in Croatian: Utjecaj slicnosti likova i transfera ucenja na obucavanje baznog perceptrona B2). In Proceedings of the Symposium Informatica, Gdansk, Poland, 6–10 September 1976; pp. 3–121.
57. Tan, C.; Sun, F.; Kong, T.; Zhang, W.; Yang, C.; Liu, C. A survey on deep transfer learning. In Proceedings of the International Conference on Artificial Neural Networks, Rhodes, Greece, 4–7 October 2018; Springer: Berlin/Heidelberg, Germany, 2018; pp. 270–279.