Enhanced Artificial Neural Networks for Salinity Estimation and Forecasting in the Sacramento-San Joaquin Delta of California

Siyu Qi*[1], Zhaojun Bai[2], Zhi Ding[3], Nimal Jayasundara[4], Minxue He[5], Prabhjot Sandhu[6], Sanjaya Seneviratne[7], and Tariq Kadir[8]

[1]Dept. of Electrical and Computer Engineering, University of California, Davis, CA 95616 (Email:syqi@ucdavis.edu)

[2]Dept. of Computer Science, University of California, Davis, CA 95616 (Email:bai@cs.ucdavis.edu)

[3]Dept. of Electrical and Computer Engineering, University of California, Davis, CA 95616 (Email:zding@ucdavis.edu)

[4]Bay Delta Office, California Dept. of Water Resources, 1416 9th St., Sacramento, CA 95614 (Email:nimal.jayasundara@water.ca.gov)

[5]Bay Delta Office, California Dept. of Water Resources, 1416 9th St., Sacramento, CA 95614 (Email:kevin.he@water.ca.gov)

[6]Bay Delta Office, California Dept. of Water Resources, 1416 9th St., Sacramento, CA 95614 (Email:prabhjot.sandhu@water.ca.gov)

[7]Bay Delta Office, California Dept. of Water Resources, 1416 9th St., Sacramento, CA 95614 (Email:sanjaya.seneviratne@water.ca.gov)

[8]Bay Delta Office, California Dept. of Water Resources, 1416 9th St., Sacramento, CA 95614 (Email:tariq.kadir@water.ca.gov)

ABSTRACT

Domain-specific architectures of artificial neural networks (ANNs) have been developed to estimate salinity levels for planning at key monitoring stations in the Sacramento-San Joaquin

Delta (Delta), California. In this work, we propose three major enhancements to existing ANN architectures for purposes of training time reduction, estimation error reduction and better feature extraction. Specifically, we design a novel multi-task ANN architecture with shared hidden layers for joint salinity estimation at multiple stations, achieving a reduction of 90% training and inference time. As another major structural redesign, we replace pre-determined pre-processing on input data by a trainable convolution layer. We further enhance the multi-task ANN design and training for salinity forecasting. Test results indicate that these enhancements substantially improve the efficiency and expand the capacity of the current salinity modeling ANNs in the Delta. Our enhanced ANN design methodologies have the potential for incorporation into the current modeling practice and provide more robust and timely information to guide water resource planning and management in the Delta.

INTRODUCTION

The Sacramento-San Joaquin Delta consists of a maze of interconnected channels that are central to California's water supply systems. Major streams like the Sacramento River, San Joaquin River, and eastside tributaries enter the Delta (Fig. 1) and the waters flow through the Delta in a complex network of intersecting channels which ultimately flow west out to the Pacific Ocean or are diverted for agricultural and municipal use inside and outside of the Delta. The salinity of water in the channels (concentration of salt measured, for example, in milligrams of salt per liter of stream water) determines the suitability for fish and wildlife, growing crops (the Delta has approximately 420,000 acres of prime agricultural lands), and urban indoor/outdoor use. Water salinities in the Delta channels are affected by many factors including ocean tides, inflows to the Delta from inland rivers and streams, and agricultural activities/practices within the Delta. Also, human actions related to water usage such as diverting to the Delta islands for Agricultural and urban use, or exports from the Delta through the State Water Project (SWP) and Central Valley Project (CVP) pumping plants would also change flows and salinities through the mixing process. To ensure safe water use, ecosystem sustainability, and economic viability, State and federal regulatory agencies have established several salinity criteria (maximum concentrations not to be exceeded) spatially and temporally within the Delta. One such regulatory example is the Water Right Decision 1641 (D1641) of the California State Water Resources Control Board ((SWRCB) 2000) which specifies the threshold salinity values at certain compliance locations during certain periods in a year. To assist in the planning and management of the water resources in the Delta, the California Department of Water Resources (CDWR) has developed two key simulation models for use in planning studies: (1) CalSim, a water allocation model of the SWP and CVP systems (Draper et al. 2004), and (2) Delta Simulation Model 2 (DSM2), a hydrodynamics and water quality model (DWR-DSM2 2019), which is developed based upon the mathematical flow-salinity relationship model presented in (Denton 1993; Denton and Sullivan 1993). We refer interested readers to an earlier paper (Jayasundara et al. 2020) on detailed discussions of CalSim and DSM2 as tools used in water resource management and their functionalities. There are 12 key water quality monitoring stations in the Delta: Emmaton, Jersey Point, Collinsville, Rock Slough, Antioch, Mallard Island, Old River at HWY 4, Martinez, Middle River Intake, Victoria Intake, CVP Intake and Clifton Court Forebay (CCFB) Intake (see Fig. 1). However, computational runtimes and other programming factors limit simulations of CalSim and DSM2 concurrently during a planning.

ANNs have been developed and applied extensively in the field of water resources engineering to model (Ranjithkumar and Robert 2021; Tung et al. 2020; Tealab 2018; Kang et al. 2017), for instance, groundwater level (Chen et al. 2011), surface runoff (Swain et al. 2017), reservoir operations (Chandramouli and Raman 2001), water demand (Bata et al. 2020), leak detection(Bohorquez et al. 2020), and water system control (Hajgató et al. 2020). ANNs have also been explored in modeling salinity in groundwater (Banerjee et al. 2011), soil (Dai et al. 2011; Jiang et al. 2019), Oceans (Bhaskaran et al. 2010; Chen and Hu 2017), rivers (Bowden et al. 2005; Hunter et al. 2018; Maier and Dandy 1999), and estuarine environments (DeSilet et al. 1992; Huang and Foo 2002; Sreekanth and Datta 2010; Le et al. 2019; Zhou et al. 2020). ANNs have only been applied recently in salinity modeling in the Delta (Chen et al. 2018; Rath et al. 2017; He et al. 2020; Jayasundara et al. 2020). Specifically, Chen et al. (2018) proposed a one-dimensional hydrodynamic model emulator to represent estuarine mixing and water quality in the northern reach of the San Francisco

Bay-Delta estuary, California, while Rath et al. (2017) developed an ANN-incorporated hybrid model of salinity in the same estuary. He et al. (2020) investigated the use of MLP ANNs in estimating boundary salinity in the Delta based on water flow and tidal stage.

Jayasundara et al. (2020), for the first time, have developed and applied individual MLP ANNs consisting of one input layer, two hidden layers, and one output layer, in simulating salinity based on seven variables in the Delta, including water control gate operations, water exports, tidal stage, as well as flow and salinity boundaries, to emulate DSM2 within CalSim 3, making runtimes much more practical. However, it is not efficient to train and inference those 12 separate ANNs. In the context of our objective for simultaneously estimating salinity levels at multiple monitoring stations based on the same set of inputs, we can view this problem as a special case of multi-task learning (MTL). This formulation is motivated by the fact that the salinities at the multiple monitoring stations are all affected by the same set of hydrological measurements within the same regional ecosystem.

MTL, in contrast to single-task learning (STL), is a machine learning strategy where multiple tasks sharing commonalities are solved simultaneously. As shown in (Caruana 1993; Caruana 1995; Ruder 2017), the domain-specific information contained in input data may allow one task to “eavesdrop" on features discovered for other related tasks and may lead the model to prefer some hypotheses over others. By leveraging the domain-specific information, MTL helps improve neural networks' efficacy and generalizability. One of the most commonly used MTL methods is known as *hard parameter sharing*, which is achieved by a joint architecture that requires multiple tasks to share some hidden layers while keeping several task-specific layers towards the end of model for each task (Caruana 1993). The idea of hard parameter sharing has been applied to time series prediction such as energy flux prediction (Guijo-Rubio et al. 2020), rainfall amount prediction(Qiu et al. 2017) and water quality forecasting(Liu et al. 2016). We design the MTL ANN for simultaneous estimation of salinity at multiple monitoring stations and this new paradigm enables the ANN to better extract the underlying data features and generate better overall performance than the current STL model individually trained and optimized for each monitoring station (Jayasundara et al. 2020).

In addition, as MTL has been successfully applied to time series prediction tasks in (Guijo-Rubio et al. 2020; Qiu et al. 2017; Liu et al. 2016), we test and analyze the prediction capability of our proposed ANNs.

Generally, the current study stems from the Jayasundara et al. (2020) study but extended all those previous studies in the Delta in terms of:

1. Improved ANN training efficiency by applying a joint MTL approach.
2. Exploration of ANN-based salinity forecasting efficacy for the Delta.
3. Improved ANN performance through systematic pre-processing of input time series by using a trainable convolution layer.
4. Expansion of ANN to discover the relationship between performance and their size.

METHODS

Similar to the approach described in (Jayasundara et al. 2020), we aim to improve salinity estimation by leveraging the seven hydrological, water quality and operation parameters, namely Northern Net flows (Sacramento River and East side Streams); San Joaquin river flows; Delta cross-channel gate operation; net Delta consumptive use; tidal energy; San Joaquin River inflow salinity at Vernalis; SWP and CVP exports via Banks pumping plant, Jones pumping plant, and Contra Costa canal (see Fig. 2). We will estimate salinities at a number of measurement points which include Emmaton, Jersey Point, Collinsville and Rock Slough, among others. The input data are the (pre-processed) seven input variables. Following (Jayasundara et al. 2020), each of the seven variables is pre-processed via an empirical convolution process that converts the values of the input at the current day plus the antecedent 117 days into 18 values, including one value from each of the current day plus the most recent seven antecedent days along with 10 non-overlapping 11-day averages. Fig. 3 outline the pipeline to obtain the estimated salinity levels in (Jayasundara et al. 2020).

Network Inputs and Outputs

The complete pipeline in mathematical notation is given in Fig. 4. We use subscript for matrix and vector indexing and superscript to denote the variable. For example, $\boldsymbol{x}^{(m)}_{n,t_r}$ is the t_r-th value for m-th input parameter in ANN's input vector for day n. As explained in Section 1, there are seven input parameters and 12 output parameters. For training and validation, we have access to monthly input data and daily salinity data covering water years 1941-2015. In California, each water year cycle runs from October 1 to September 30 of the following calendar year. As described in (Jayasundara et al. 2020), CalSim can refine monthly data record into daily by spline interpolation.

There is a total of N data samples (or days) in the dataset. In our problem, we select $M = 7$ observation variables. Same as in CalSim (Jayasundara et al. 2020), we pick $T = 118$ and $T_r = 18$ in the baseline case and pre-process the data as denoted in Fig. 5.

For input variable m on day n, we extract 8 daily values:

$$\boldsymbol{x}^{(m)}_{n,i} = \boldsymbol{z}^{(m)}_{n-i+1}, \tag{1}$$

where $i \in \{1, \ldots, 8\}$. We also compute a total of 10 successive but non-overlapping 11-day moving averages before the first daily data $\boldsymbol{x}^{(m)}_{n,i}$, $i \in \{1, \ldots, 8\}$ to be stored in

$$\boldsymbol{x}^{(m)}_{n,i+8} = \frac{1}{11}\sum_{j=1}^{11} \boldsymbol{z}^{(m)}_{n-11i-j+4}, \tag{2}$$

where $i \in \{1, \ldots, 10\}$. Altogether, for the M variables in each day n, we form $M \times T_r = 7 \times 18 = 126$ values as the $M \times T_r$ input matrix $\boldsymbol{x}_n$ to the ANNs.

Later for exploring a different ANN architecture to bypass this rather *ad hoc* pre-processing, we would form a trainable convolution layer instead of applying the above pre-determined pre-processing steps. In that case, those 118 daily values of each of the seven variables are directly provided to the convolution layer. The corresponding details will be described in Section 2.

The target outputs of ANNs are the salinity levels at one or more monitoring stations. Each STL

ANN's output is salinity level at one single monitoring station, while each MTL ANN's outputs are salinity levels at all 12 monitoring stations.

Different from the previous study (Jayasundara et al. 2020), the current work randomly split 80% and 20% of this dataset for training and validation, respectively.

Multi-Task Learning

The goals of ANNs are to predict salinity at multiple monitoring stations which are physically related to one another in the Delta. It is therefore natural that these ANNs can share some of the same features of the underlying data inputs. To improve the ANN for individual monitoring stations (Jayasundara et al. 2020), we explore the MTL approach for the 12 monitoring stations under study.

As described in (Caruana 1995), these inter-related multiple tasks may be learned jointly by training a single ANN. The output layers shall include more neurons whereas the hidden layers are shared by the monitoring stations. These hidden layers together serve as a joint mechanism for feature extractions that can be used more consistently to generate salinity estimates at different monitoring stations. With MTL, an ANN can show better general performance over multiple disjoint single-task ANNs. As shown in Fig. 6, the MLP architecture proposed in (Jayasundara et al. 2020) consists of two fully connected (FC) hidden layers and one output layer, with each layer containing 8 neurons, 2 neurons and 1 neuron, respectively.

Based on the model in previous successful STL ANNs in Fig. 6, we build the multi-task ANN architecture, which is an MLP network containing two hidden layers with sigmoid activation functions and one output layer with a Leaky ReLU (Maas et al. 2013) activation function. As illustrated in Fig. 7, we increase number of neurons by a factor of 12, which coincides with the number of monitoring stations, for all layers to build the multi-task ANN, that is, the two hidden layers and output layer in multi-task ANN contain 96, 24 and 12 neurons respectively. We also explored various ANN sizes in Section “ANN Size".

Trained Input Pre-processing via a Convolution Layer

As discussed earlier, the authors of (Jayasundara et al. 2020) utilized 8 newest daily values together with 10 non-overlapping moving averages of the daily values immediately before the 8 daily values as input data for salinity estimation (Fig. 5).

It should be recognized that the reported direct daily mappings and moving window averages are special cases of convolution processing, except that the existing pre-processing is not optimized through data training. Understanding the shortcomings of such a heuristic pre-processing, we propose instead to include a trainable convolution layer for data pre-processing in our novel ANN architecture. Mathematically, the convolution layer would implement the following data processing through the training weights $f_{j,i}^{(m)}$:

$$\boldsymbol{x}_{n,i}^{(m)} = \sum_{j=1}^{T} \boldsymbol{z}_{n-j+1}^{(m)} \times \boldsymbol{f}_{j,i}^{(m)}, \tag{3}$$

where $n \in \{1, \ldots, N\}$, $m \in \{1, \ldots, M\}$ and $i \in \{1, \ldots, T_r\}$. Clearly, by appropriately setting the convolution weights $f_{j,i}^{(m)}$, the convolution layer is capable of delivering daily value mapping and sliding window averaging. Moreover, this convolution layer is trainable in conjunction with the additional layers in the ANN. The inclusion of the convolution layer within the ANN allows the weights in this layer and other ANN layers be jointly optimized to achieve better overall performance.

By including the convolution layer, the two respective novel architectures of single-task and multi-task ANNs with a convolution layer are shown in Fig. 8. There are $T_r = 18$ filters in a convolution layer such that the convolution layers are able to extract at least the same 8 daily values and 10 average values in the pre-determined pre-processing. The complete pipeline with proposed convolution layer and the MTL ANN can be found in Fig. 9.

Salinity Forecasting

The ability to forecast salinity at key monitoring stations with lead time up to several days can present an important opportunity and advantage to the water operations in the Delta. This can be especially important for real-time operators of the SWP and CVP reservoirs to ensure that adequate released water reaches the Delta to ensure regulatory compliance at the salinity monitoring stations; for example, it takes approximately five days for water released from the Shasta reservoir (a CVP facility) and three days for water released from the Lake Oroville reservoir (a SWP facility) to reach the Delta. The existing ANN studies have not tackled this challenging problem. From a physical point of view, the dynamics between the input hydrological parameters and the salinity level measurements provide a strong motivation to suggest the possible success of salinity forecasting. Successful forecasting can also provide vital insight into the development of future models.

In this work, we investigate and explore the proposed MTL ANN for salinity forecasting at the 12 monitoring stations. We utilize the same architecture to test the performance on salinity prediction tasks. In this case, with a set of inputs for day n, a MTL ANN learns to predict the salinity levels $\boldsymbol{y}_{n+i}$, $i \in \{1, \ldots, 7\}$ on day $n + i$.

Optimizer

In (Jayasundara et al. 2020), the authors adopt both Levenberg-Marquardt (LM) (Marquardt 1963; Levenberg 1944) optimization algorithm and Bayesian regularization (Foresee and Hagan 1997) to update weights and biases in ANNs. However, as mentioned in (Wilamowski et al. 2001), the demand for large memory to compute Jacobian matrices and the need for inverting matrices are the major drawbacks of the LM algorithm. When the number of trainable parameters in ANN increases, the computational complexity of LM algorithm grows exponentially. In this paper, we utilize one of the modern optimizers, the Adam optimizer (Kingma and Ba 2014), to train the ANNs. The Adam optimizer is computationally efficient and requires little memory. As a result, the Adam optimizer is well suited for machine learning problems with complex network architecture (as proposed in this paper) and/or large datasets.

ANN Size

The number of hidden layers and neurons in these layers determines the number of trainable parameters and the potential capability of an ANN. There is a trade-off between ANN complexity and performance. Generally, performance on the training dataset usually improves with the increase of the ANN size before the problem of overfitting occurs due to limited data, because a larger ANN is capable of learning a more complex non-linear function. Meanwhile, as the ANN gets deeper and/or wider, the probability of overfitting increases, and the computation complexity grows. To find the architecture that fits this specific problem, we vary the depth and width of multi-task ANNs and observe how their performance changes on the test dataset.

RESULTS AND ANALYSIS

Implementation

We implement the newly developed ANNs using the popular open source library, Tensorflow 2.2.0 (Abadi et al. 2015), with Python 3.6.9. We conduct the experiments through web-browser on Google Colaboratory, which is a cloud-based Jupyter notebook environment with a Tesla T4 GPU. We normalize inputs and outputs to the range [0.1, 0.9] by linearly converting the i-th daily value of the k-th input variable in the n-th data sample $\boldsymbol{x}_{n,i}^{(m)}$ to

$$\widehat{\boldsymbol{x}}_{n,i}^{(m)} = \frac{\boldsymbol{x}_{n,i}^{(m)} - \left(\min\limits_{k=1,\dots,N} \boldsymbol{x}_{k,i}^{(m)}\right)}{\left(\max\limits_{k=1,\dots,N} \boldsymbol{x}_{k,i}^{(m)}\right) - \left(\min\limits_{k=1,\dots,N} \boldsymbol{x}_{k,i}^{(m)}\right)} \times 0.8 + 0.1. \tag{4}$$

We apply the same normalization to outputs $\boldsymbol{y}_n$ representing the salinity at a monitoring station on day n.

$$\widehat{\boldsymbol{y}}_n = \frac{\boldsymbol{y}_n - \left(\min\limits_{k=1,\dots,N} \boldsymbol{y}_k\right)}{\left(\max\limits_{k=1,\dots,N} \boldsymbol{y}_k\right) - \left(\min\limits_{k=1,\dots,N} \boldsymbol{y}_k\right)} \times 0.8 + 0.1. \tag{5}$$

The cost function used for training is the Mean Squared Error (MSE). For the LM optimizer, we adopt the same settings as (Jayasundara et al. 2020), where the starting learning rate is 0.005 and decay factor is 10 and the training takes 150 epochs. For the Adam optimizer, the learning rate is determined using a grid search. The starting learning rate is 0.01, and it is scaled by 0.1, 0.01, 0.001 and 0.0005 at epochs 80, 120, 160 and 180, respectively, and the training takes 200 epochs.

Experimental Results and Discussions

We evaluate the performance of the newly proposed ANN models by calculating the unitless normalized mean square error (NMSE), which is computed on the normalized salinity outputs $\hat{y}_n$ based on the validation dataset. We compare the performance of several ANN architectures.

To begin, the basic model is a 3-layer STL ANN with pre-processed input data, consisting of two hidden layers and one output layer, as shown in Fig. 6. We train this baseline ANN using both the LM algorithm (STL-LM) and the Adam optimizer (STL-Adam), to illustrate the effects of optimizers. Both "STL-LM" and "STL-Adam" configurations are used as baseline results for comparison.

In our proposed ANNs based on the novel MTL strategy, we consider two different architectures: (a) a basic 3-layer MTL ANN with the pre-determined data pre-processing used in the baseline model using the Adam optimizer (3-MTL) for training; and (b) a 4-layer MTL ANN with a replacement of fixed data pre-processing by a trainable convolution layer. We consider two initializations for the trainable convolution layer parameters: random (4-MTL-R) and using the pre-determined pre-processing parameters (4-MTL-P) according to equations 1, 2 and 3.

Results from each of the five configurations are labeled, respectively, by "STL-LM", "STL-Adam", "3-MTL", "4-MTL-R", and "4-MTL-P". Table 1 presents the NMSE results of the five different ANN configurations. Correspondingly, Table 2 evaluates their respective training and inference time (complexity). From the performance comparison, we make the following observations.

- With pre-determined data processing, the LM algorithm outperforms the Adam optimizer in training STL ANN to generate lower NMSE values than STL-Adam and 3-MTL do at all study stations, as shown in Table 1. However, the LM algorithm requires 8 times longer training time (complexity) when compared with both STL and MTL trained with the Adam optimizer as shown in Table 2.
- Using our newly proposed MTL architectures with a trainable convolution layer, training with the Adam optimizer can substantially improve the NMSE performance over STL. In particular, the 4-MTL-P results are distinctly better (with smaller NMSE values) when comparing with STL-Adam at all 12 stations. The 4-MTL-P scenario outperforms STL-LM at 9 out of the 12 stations.
- The proposed 4-MTL architecture not only improves the salinity estimation performance in

providing generally lower NMSE values, but also requires much shorter training time (from 8.31 hours of STL-LM to 319 seconds of 4-MTL-P) as well as much faster inference (from 8.52 ms to 1.3 ms). Therefore, applying MTL to multi-station salinity estimation tasks can clearly improve training and inference efficiency.

- In 4-MTL, a trainable convolution layer significantly reduces NMSE as this data processing layer can learn to extract data features and adapt to wider MTL ANN architecture through training. Our pre-determined initialization helps reduce the probability of being trapped in a local minimum.
- From Table 1, Antioch, Mallard Island and Martinez are the 3 outliers in 4-MTL-P with slightly higher NMSE values than their counterparts from the STL-LM scenario. The reason is that stations located further west are more influenced by ocean tides of high salinity and are less effected by the input flows. Indeed, we can see from Fig. 1 that all these three stations are in the western part of the Delta.

Further ANN Structure Considerations

We further investigate the effect of the proposed MTL ANN size and depth. Given the success of the 4-layer MTL ANNs, we increase its depth and width.

Starting with the 4-MTL-P ANN consisting of 1 convolution layer, 2 hidden layers and 1 output layer, we design 10 sets of neuron partitions in the hidden layers, while output layer contains 12 neurons in all cases. For the 4-layer configuration, the 10 sets of neuron partitions for the two hidden layers are provided in Fig. 10.

We further increased the number of hidden layers to the MTL ANN by one to obtain a 5-layer ANN architecture (5-MTL-P) and select 9 sets of partitions for the 3 hidden layers in 5-MTL-P. In another test, we add yet another hidden layer to form a 6-layer MTL ANN (6-MTL-P). We select 3 sets of neuron partitions for the 4 hidden layers in 6-MTL-P. The detailed neuron partitions among the hidden layers are also shown in Fig. 10.

The NMSE results for the 4-MTL-P, 5-MTL-P and 6-MTL-P ANNs are illustrated in Fig. 11. We observe that, in general, the NMSE performance improves with increasing neural network size.

However, for the 12 monitoring stations, deeper ANNs with more fully connected layers in both 5-layer and 6-layer ANNs do not necessarily outperform a 4-layer ANN, using similar number of parameters. Fig. 11 also illustrates that an expanded MTL model achieves comparable performance to the STL-LM baseline model for all monitoring stations.

Salinity Forecasting

Physically, the salinity levels at the monitoring stations are impacted by antecedent (up to months) Delta inflows. Therefore, it would be probable that one can forecast the salinity levels based on current and antecedent inflow data. Hence, in addition to the same day salinity estimation results obtained thus far, we further explore the efficacy of our ANN model for salinity forecasting days ahead in time at the monitoring stations.

To investigate the prediction accuracy of our proposed 4-MTL ANN, we train seven forecasting models based on the 4-MTL-P architecture as in Fig. **??**. The implementation is similar to the salinity estimation model and is very simple. We apply the same MTL ANN architectures except that their training is based on the forecasting error. Specifically, to train a MTL ANN to forecast salinity levels by i day(s), we simply train the ANN model by using the same input measurement data but by advancing the output salinity level by i day(s) when calculating the MSE cost function. Changing the same simple training model by advancing the output measurement by i day(s) for $i = 1, \ldots, 7$, we can test the efficacy of an $i-$day forecasting model.

The best experimental results are obtained using the 4-MTL-P configuration. The forecasting results from 4-MTL-P are depicted in Fig. 12 as we vary $i = 0, 1, \ldots, 7$ using the red line. The baseline $i = 0$ estimation results from 4-MTL-P is also highlighted as a blue dashed line for comparison. From the results, we make the following observations respecting the forecasting ANN.

- Our 4-MTL-P ANN forecasting tends to show more accurate forecasting in short term, typically in 1-day or 2-day forecasting. Such result implies that there is a decent correlation between upstream inflows and Delta salinity up to two days. After that, the correlation tends to weaken. This is most likely because of physical distance between where flows are measured and salinity monitoring stations.

- In most cases with the exception of Emmaton, the NMSE values of forecasts with lead time one day are smaller than or fairly close to their corresponding counterparts of the same day salinity estimation. Emmaton differs from other stations in that it is located on the Sacramento River which has significantly high runoff than other rivers (e.g., San Joaquin river and eastern tributaries). The lasting impacts of flow on salinity in the Sacramento River is not as obvious as those on other rivers.

Adaptive Estimation and Forecasting Models

Our test results suggest a novel adaptive hybrid ANN model in which the forecasting objectives can be set uniquely for different monitoring stations, according to their physical response times. In other words, depending on the geophysical distance between input and output locations, the training objective function should consist of forecasting errors defined with different forecasting lead times for different monitoring stations.

Such an adaptive ANN model can be fully incorporated within the proposed MTL framework. In fact, the only adaptive parameters that we need to adjust are the lead times used in the MSE cost function when training the ANN. Based on the results in Fig. 12, for each monitoring station location, we vary the number of days to forecast based on their base forecasting performance. With the 4-MTL-P ANN architecture as defined in Fig. **??**, we manually initialize the convolution filters in an ANN model during training to estimate salinity at Emmaton and forecast the remaining 11 stations. The final test results of the adaptive hybrid ANN model are given in Table 3. As expected, 8 of the 12 monitoring stations achieved improved performance. The sum NMSE of all 12 monitoring stations is also lower than the pure estimation.

It would be natural to adjust the forecasting lead times for different monitoring stations to drive down the sum NMSE further. Such fine-tuning would require a large number of experiments but do not change the basic principles and the contributions of our work reported here.

DISCUSSIONS AND CONCLUSIONS

Implications

This study has both scientific and practical implications. From a scientific standpoint, we propose the following outcomes:

1. The study introduces the concept of MTL into salinity modeling via ANNs in the Delta for the first time. This enables estimation of salinity at multiple locations in a single ANN model, with no need to develop different STL ANNs for different locations as in (Jayasundara et al. 2020).
2. In addition, this study is the first to examine and demonstrate the capability of ANNs in salinity forecasting in the Delta. This lays the foundation for further methodical exploration on this front.
3. This study proposes a novel way of pre-processing ANN input data via a trainable convolution layer. Compared to the current empirical pre-processing method in (Jayasundara et al. 2020), this new method is modular and thus portable to additional input data.

Those scientific advances are not only applicable in modeling salinity, but also other important environmental variables in the Delta including turbidity, dissolved oxygen concentration, water temperature, among others. Additionally, their potential applications are not only limited to the Delta area, but also to other estuarine environments worldwide.

Meanwhile, from a practical point of view, we present the following implications:

1. The study indicates that the MTL-based ANN proposed in this study is much more efficient compared to the traditional STL-based ANNs in terms of training time and inference time (Table 2). This is particularly appealing to CDWR's current modeling practice in water resources planning studies. In a specific planning scenario, the current modeling practice involves a process of iteratively running the planning model and the salinity emulator (i.e., current ANNs) till all salinity compliance objectives are met. Using a faster emulator instead is expected to expedite the modeling process and allows more inclusive planning scenarios to be assessed.

2. The study exemplifies the feasibility applying the proposed ANNs in salinity forecasting. In practice, DSM2 is routinely utilized in forecasting salinity in the Delta to inform decision-making. The proposed ANNs have the potential to supplement the current forecasting practice for that purpose.

Future Work

This study indicates that the proposed MTL-based ANN outperforms the current STL-based ANNs in most cases (Table 1). However, for three stations in western Delta (Martinez, Mallard Island and Antioch), the STL-based ANNs yield slightly better estimation. We attribute the probable cause to that the tide plays a more important role than upstream freshwater inflows at these stations. Currently, the tidal energy (the difference between daily maximum and minimum stages at Martinez) serves as the proxy for tidal impact in the input data. However, it is not a direct measurement of the salinity level. As illustrated previously (He et al. 2020), sea level at the Golden Gate Bridge (the downstream end of the Bay-Delta Estuary in Figure 1) is a better surrogate for the salinity source of the Delta. It is also shown that incorporating sea level as an additional input feature to ANNs can improve salinity estimation at Martinez (He et al. 2020). One potential future enhancement to the proposed ANN is to incorporate sea level at the Golden Gate Bridge as an additional input.

The study also shows that the forecasting skill of the proposed ANN decreases with increasing lead time (Fig. 12). This is expected as the lasting influence of current day's input data (predictors) on salinity (predictand) becomes weaker further into the future. To improve forecasting skills, forecasted input information (e.g., forecasts on flows, tidal energy, and gate operations) can be applied to drive the proposed ANN. This is a potential future direction to be explored.

Additionally, the ANNs examined in the current study use input data in the past 118 days since salinity relates to antecedent (up to months) flows in the Delta. The deep learning architecture Long Short-Term Memory (LSTM) architecture has shown special potential in simulating variables with such a long memory with their predictors (He et al. 2020). This type of deep learning networks will be considered in our future work.

Finally, this study showcases the success of applying proposed ANNs in salinity modeling in the Delta. There are a wide range of other variables (e.g., precipitation, runoff volume, snow melt, river stage, water temperature, turbidity) elsewhere that are critical to water resources planning and management practices. The ANNs developed in the current study can be readily adapted to simulate or forecast those variables in the future.

Concluding Remarks

This study develops enhancements to the Delta salinity modeling ANNs for the purposes of training time reduction, estimation error reduction, and better feature extraction. The enhancements include structural redesign on two fronts: 1) incorporation of the MTL architecture and 2) addition of a convolution layer in input data pre-processing. The updated ANNs are further adapted to conduct salinity forecasting which is rarely investigated previously. The enhanced ANNs have the potential to be incorporated into the current modeling practice and provide more robust and timely information to guide water resources planning and management in the Delta.

DATA AVAILABILITY STATEMENTS

The following code and data that support the findings of this study are available from the corresponding author by request: Python code for training and evaluating the ANNs; input and output data used in ANN training and evaluation.

ACKNOWLEDGMENTS

This work was supported in part by the research agreement 4600013184 from the California Department of Water Resources. The views and opinions expressed in this article are those of the authors and do not reflect the policy or position of their employers.

NOTATIONS

The following symbols are used in this paper:

M = Number of input hydrological variables denoted in Fig. 2.

N = Number of data samples, or days, in dataset.

T = Number of days' data used for estimation.

T_r = Dimension of data after pre-processing.

z_n = Time series used for estimating salinity level on day n, size is $\mathbb{R}^{M\times T}$.

$\boldsymbol{x}_n$ = Pre-processed time series with size $\mathbb{R}^{M\times T_r}$ for day n.

$\boldsymbol{y}_n$ = ANN-estimated salinity level for one or more locations on day n.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). "TensorFlow: Large-scale machine learning on heterogeneous systems, <https://www.tensorflow.org/>. Software available from tensorflow.org.

Banerjee, P., Singh, V., Chatttopadhyay, K., Chandra, P., and Singh, B. (2011). "Artificial neural network model as a potential alternative for groundwater salinity forecasting." *Journal of Hydrology*, 398(3-4), 212–220.

Bata, M. H., Carriveau, R., and Ting, D. S.-K. (2020). "Short-term water demand forecasting using nonlinear autoregressive artificial neural networks." *Journal of Water Resources Planning and Management*, 146(3), 04020008.

Bhaskaran, P. K., Kumar, R. R., Barman, R., and Muthalagu, R. (2010). "A new approach for deriving temperature and salinity fields in the indian ocean using artificial neural networks." *Journal of marine science and technology*, 15(2), 160–175.

Bohorquez, J., Alexander, B., Simpson, A. R., and Lambert, M. F. (2020). "Leak detection and topology identification in pipelines using fluid transients and artificial neural networks." *Journal of Water Resources Planning and Management*, 146(6), 04020040.

Bowden, G. J., Maier, H. R., and Dandy, G. C. (2005). "Input determination for neural network models in water resources applications. part 2. case study: forecasting salinity in a river." *Journal of Hydrology*, 301(1-4), 93–107.

Caruana, R. (1995). "Learning many related tasks at the same time with backpropagation." *Advances in neural information processing systems*, 657–664.

Caruana, R. A. (1993). "Multitask connectionist learning." *In Proceedings of the 1993 Connectionist Models Summer School*, Citeseer.

Chandramouli, V. and Raman, H. (2001). "Multireservoir modeling with dynamic programming and neural networks." *Journal of Water Resources Planning and Management*, 127(2), 89–98.

Chen, L., Roy, S. B., and Hutton, P. H. (2018). "Emulation of a process-based estuarine hydrodynamic model." *Hydrological Sciences Journal*, 63(5), 783–802.

Chen, L.-H., Chen, C.-T., and Lin, D.-W. (2011). "Application of integrated back-propagation network and self-organizing map for groundwater level forecasting." *Journal of Water Resources Planning and Management*, 137(4), 352–365.

Chen, S. and Hu, C. (2017). "Estimating sea surface salinity in the northern gulf of mexico from satellite ocean color measurements." *Remote sensing of environment*, 201, 115–132.

Dai, X., Huo, Z., and Wang, H. (2011). "Simulation for response of crop yield to soil moisture and salinity with artificial neural network." *Field Crops Research*, 121(3), 441–449.

Denton, R. and Sullivan, G. (1993). "Antecedent flow-salinity relations: Application to delta planning models." *Contra Costa Water District. Concord, California.*

Denton, R. A. (1993). "Accounting for antecedent conditions in seawater intrusion modeling—applications for the san francisco bay-delta." *Hydraulic Engineering*, ASCE, 448–453.

DeSilet, L., Golden, B., Wang, Q., and Kumar, R. (1992). "Predicting salinity in the chesapeake bay using backpropagation." *Computers & Operations Research*, 19(3-4), 277–285.

Draper, A. J., Munevar, A., Arora, S. K., Reyes, E., Parker, N. L., Chung, F. I., and Peterson, L. E. (2004). “CalSim: Generalized model for reservoir system analysis.” *Journal of Water Resources Planning and Management*, 130(6), 480–489.

DWR-DSM2 (2019). “DSM2: Delta Simulation Model II.” *Bay Delta Office, California Dept. of Water Resources, Sacramento, CA.*, <https://water.ca.gov/Library/Modeling-and-Analysis/Bay-Delta-Region-models-and-tools/Delta-Simulation-Model-II>.

Foresee, F. D. and Hagan, M. T. (1997). “Gauss-Newton approximation to Bayesian learning.” *Proceedings of International Conference on Neural Networks (ICNN’97)*, Vol. 3, IEEE, 1930–1935.

Guijo-Rubio, D., Gómez-Orellana, A. M., Gutiérrez, P. A., and Hervás-Martínez, C. (2020). “Short- and long-term energy flux prediction using multi-task evolutionary artificial neural networks.” *Ocean Engineering*, 216, 108089.

Hajgató, G., Paál, G., and Gyires-Tóth, B. (2020). “Deep reinforcement learning for real-time optimization of pumps in water distribution systems.” *Journal of Water Resources Planning and Management*, 146(11), 04020079.

He, M., Zhong, L., Sandhu, P., and Zhou, Y. (2020). “Emulation of a process-based salinity generator for the sacramento–san joaquin delta of california via deep learning.” *Water*, 12(8), 2088.

Huang, W. and Foo, S. (2002). “Neural network modeling of salinity variation in apalachicola river.” *Water Research*, 36(1), 356–362.

Hunter, J. M., Maier, H. R., Gibbs, M. S., Foale, E. R., Grosvenor, N. A., Harders, N. P., and Kikuchi-Miller, T. C. (2018). “Framework for developing hybrid process-driven, artificial neural network and regression models for salinity prediction in river systems.” *Hydrology and Earth System Sciences*, 22(5), 2987–3006.

Jayasundara, N. C., Seneviratne, S. A., Reyes, E., and Chung, F. I. (2020). “Artificial neural network for Sacramento–San Joaquin Delta flow–salinity relationship for CalSim 3.0.” *Journal of Water Resources Planning and Management*, 146(4), 04020015.

Jiang, H., Rusuli, Y., Amuti, T., and He, Q. (2019). "Quantitative assessment of soil salinity using multi-source remote sensing data based on the support vector machine and artificial neural network." *International journal of remote sensing*, 40(1), 284–306.

Kang, G., Gao, J. Z., and Xie, G. (2017). "Data-driven water quality analysis and prediction: A survey." *2017 IEEE Third International Conference on Big Data Computing Service and Applications (BigDataService)*, IEEE, 224–232.

Kingma, D. P. and Ba, J. (2014). "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980*.

Le, D., Huang, W., and Johnson, E. (2019). "Neural network modeling of monthly salinity variations in oyster reef in apalachicola bay in response to freshwater inflow and winds." *Neural Computing and Applications*, 31(10), 6249–6259.

Levenberg, K. (1944). "A method for the solution of certain non-linear problems in least squares." *Quarterly of applied mathematics*, 2(2), 164–168.

Liu, Y., Liang, Y., Liu, S., Rosenblum, D. S., and Zheng, Y. (2016). "Predicting urban water quality with ubiquitous data." *arXiv preprint arXiv:1610.09462*.

Maas, A. L., Hannun, A. Y., and Ng, A. Y. (2013). "Rectifier nonlinearities improve neural network acoustic models." *Proc. icml*, Vol. 30, 3.

Maier, H. R. and Dandy, G. C. (1999). "Empirical comparison of various methods for training feed-forward neural networks for salinity forecasting." *Water Resources Research*, 35(8), 2591–2596.

Marquardt, D. W. (1963). "An algorithm for least-squares estimation of nonlinear parameters." *Journal of the society for Industrial and Applied Mathematics*, 11(2), 431–441.

Qiu, M., Zhao, P., Zhang, K., Huang, J., Shi, X., Wang, X., and Chu, W. (2017). "A short-term rainfall prediction model using multi-task convolutional neural networks." *2017 IEEE International Conference on Data Mining (ICDM)*, IEEE, 395–404.

Ranjithkumar, M. and Robert, L. (2021). "Machine learning techniques and cloud computing to estimate river water quality—survey." *Inventive Communication and Computational Technologies*, Springer, 387–396.

Rath, J. S., Hutton, P. H., Chen, L., and Roy, S. B. (2017). "A hybrid empirical-Bayesian artificial neural network model of salinity in the San Francisco Bay-Delta estuary." *Environmental Modelling & Software*, 93, 193–208.

Ruder, S. (2017). "An overview of multi-task learning in deep neural networks." *arXiv preprint arXiv:1706.05098*.

Sreekanth, J. and Datta, B. (2010). "Multi-objective management of saltwater intrusion in coastal aquifers using genetic programming and modular neural network based surrogate models." *Journal of Hydrology*, 393(3-4), 245–256.

Swain, E. D., Gómez-Fragoso, J., and Torres-Gonzalez, S. (2017). "Projecting impacts of climate change on water availability using artificial neural network techniques." *Journal of Water Resources Planning and Management*, 143(12), 04017068.

(SWRCB), S. W. R. C. B. (2000). "Revised water right decision 1641.

Tealab, A. (2018). "Time series forecasting using artificial neural networks methodologies: A systematic review." *Future Computing and Informatics Journal*, 3(2), 334–340.

Tung, T. M., Yaseen, Z. M., et al. (2020). "A survey on river water quality modelling using artificial intelligence models: 2000–2020." *Journal of Hydrology*, 585, 124670.

Wilamowski, B. M., Iplikci, S., Kaynak, O., and Efe, M. O. (2001). "An algorithm for fast convergence in training neural networks." *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222)*, Vol. 3, Ieee, 1778–1782.

Zhou, F., Liu, B., and Duan, K. (2020). "Coupling wavelet transform and artificial neural network for forecasting estuarine salinity." *Journal of Hydrology*, 588, 125127.

List of Tables

TABLE 1. Resulting NMSE$\times 10^4$ of different ANN architectures for salinity estimation. Both inputs and outputs of ANNs are normalized.

	STL-LM (baseline)	STL-Adam (baseline)	3-MTL	4-MTL-R	4-MTL-P
Optimizer	LM	Adam	Adam	Adam	Adam
Emmaton	3.2	9.03	10.03	4.25	**2.63**
Jersey Point	5.35	14.78	16.18	5.74	**3.28**
Collinsville	5.09	15.92	15.56	6.20	**3.86**
Rock Slough	5.34	13.33	17.95	6.55	**3.69**
Antioch	**1.84**	7.85	9.73	3.50	2.60
Mallard Island	**2.18**	8.25	9.59	3.28	2.68
Old River at HWY 4	5.01	18.99	21.03	5.27	**2.71**
Martinez	**0.61**	3.15	6.66	2.53	1.63
Middle River Intake	5.21	16.71	17.72	5.20	**2.66**
Victoria Intake	6.12	15.41	16.47	5.33	**2.88**
CVP Intake	5.11	21.32	18.95	6.97	**3.94**
CCFB Intake Gate	5.64	20.57	19.38	6.23	**3.32**

TABLE 2. Training time and inference time of different ANN architectures

Model Information	Architecture		
	STL-LM	STL-Adam	4-MTL-P
Number of parameters	981	981	16962
Optimizer	LM	Adam	Adam
Training time (sec. per model)	2493	315	319
Inference time (ms. per sample)	0.71	0.71	1.3
Number of models needed	12	12	1
Total training time	8.31 hrs	1.05 hrs	**319 secs**
Total inference time for all 12 stations (ms. per day)	8.52	8.52	**1.3**

TABLE 3. Resulting NMSE$\times 10^4$ of 4-MTL-P ANNs for salinity estimation with/without forecasting, numbers in parentheses represent the forecast time in days for that station.

Monitoring Station	Estimation only	Joint estimation and forecast
Emmaton	**2.63** (0)	2.69 (0)
Jersey Point	3.28 (0)	**3.25** (1)
Collinsville	**3.86** (0)	3.94 (1)
Rock Slough	3.69 (0)	**3.52** (1)
Antioch	**2.60** (0)	2.68 (2)
Mallard Island	2.68 (0)	**2.65** (1)
Old River at HWY 4	2.71 (0)	**2.68** (1)
Martinez	1.63 (0)	**1.54** (1)
Middle River Intake	2.66 (0)	**2.65** (1)
Victoria Intake	2.88 (0)	**2.79** (2)
CVP Intake	3.94 (0)	**3.90** (1)
CCFB Intake Gate	**3.32** (0)	3.34 (1)
Total	35.87	**35.64**

List of Figures

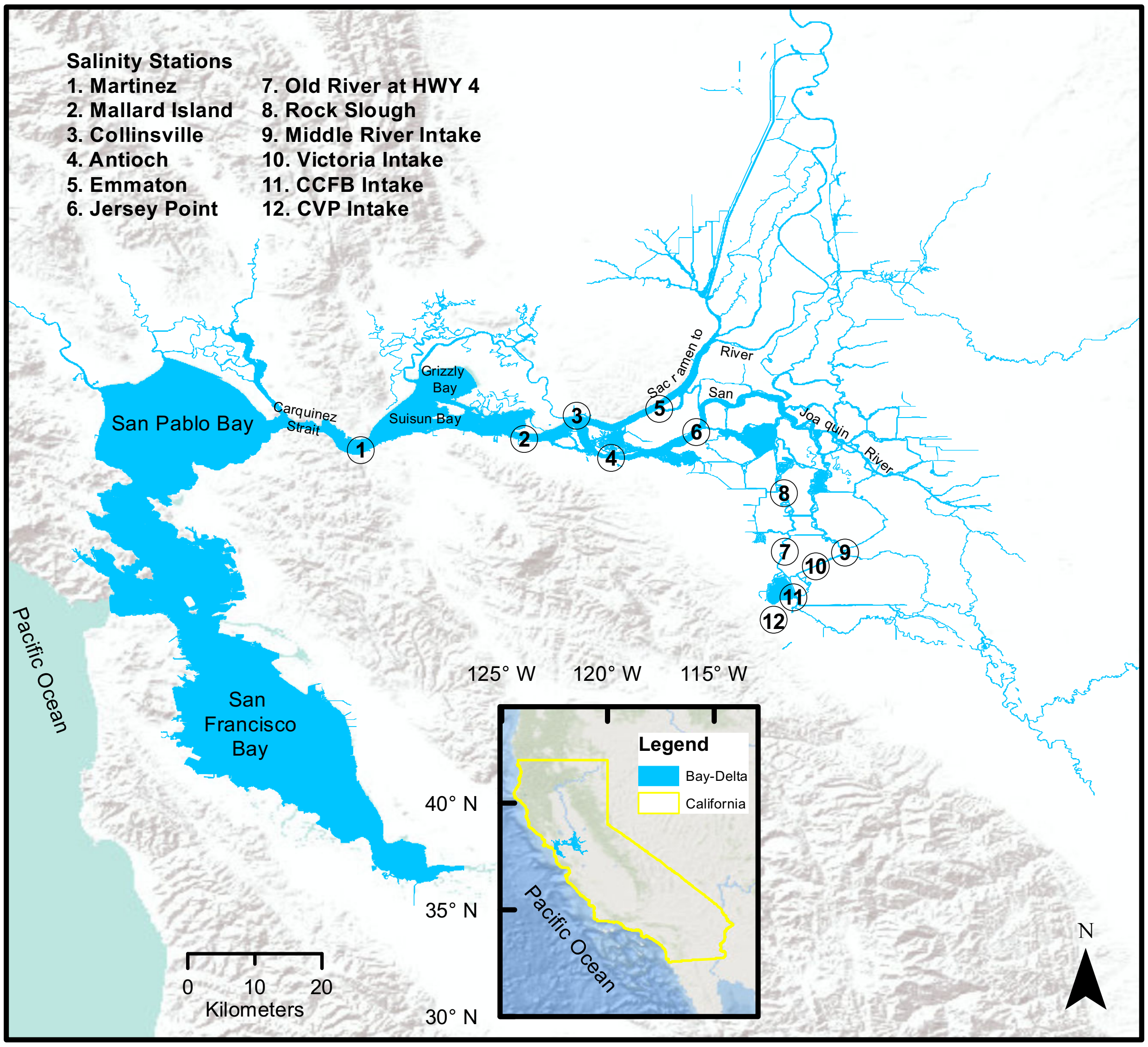


Fig. 1. Locations of the 12 Study Salinity Stations in the San Francisco Bay and Sacramento-San Joaquin Delta Estuary (Bay-Delta Estuary). The insert map shows the location of the Bay-Delta Estuary in California. (Sources: Esri, DeLorme, HERE, TomTom, Intermap, increment P Corp., GEBCO, USGS, FAO, NPS, NRCAN, GeoBase, IGN, Kadaster NL, Ordnance Survey, Esri Japan, METI, Esri China (Hong Kong), swisstopo, MapmyIndia, and the GIS User Community; Data from ((SWRCB) 2000).)

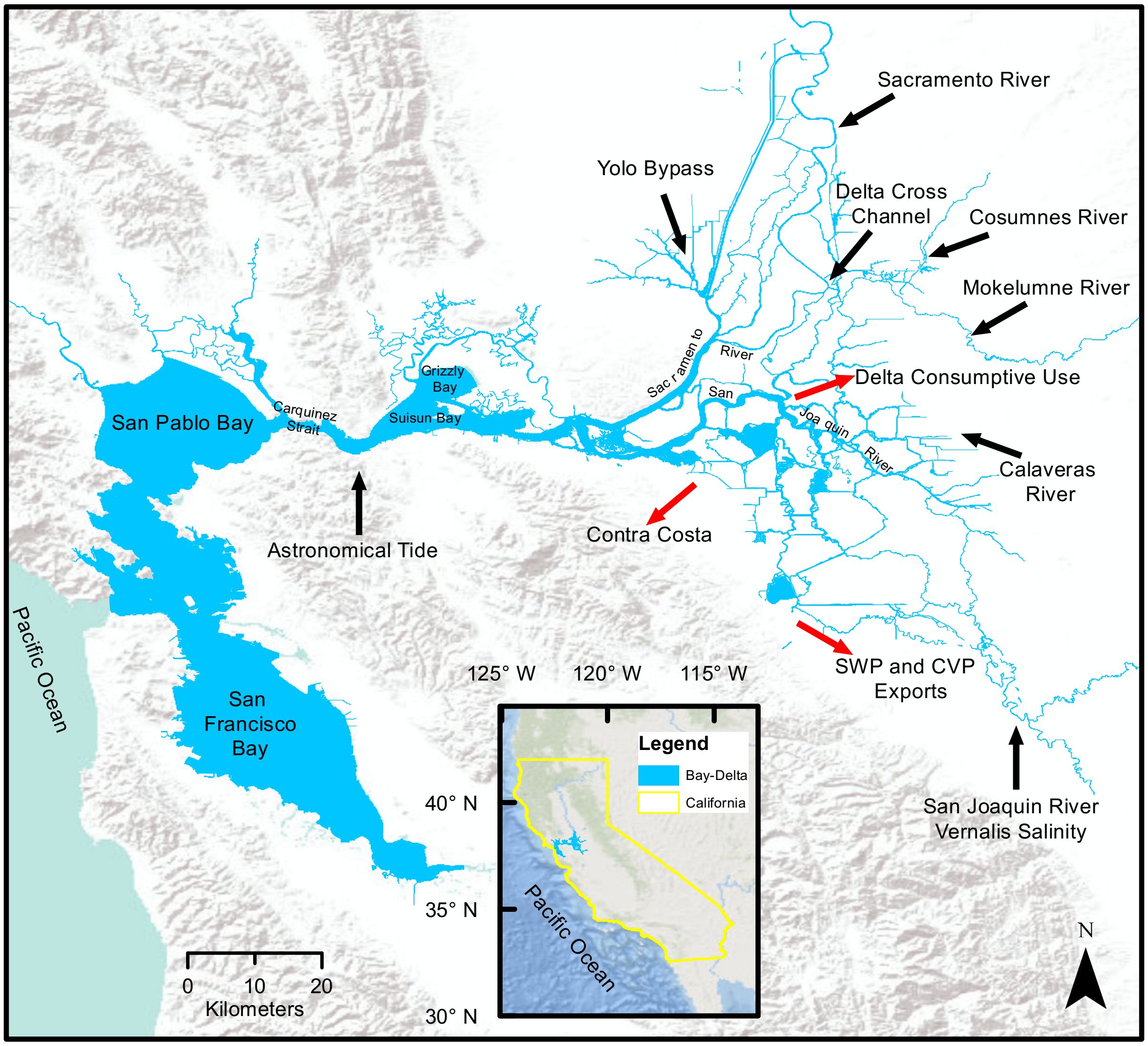
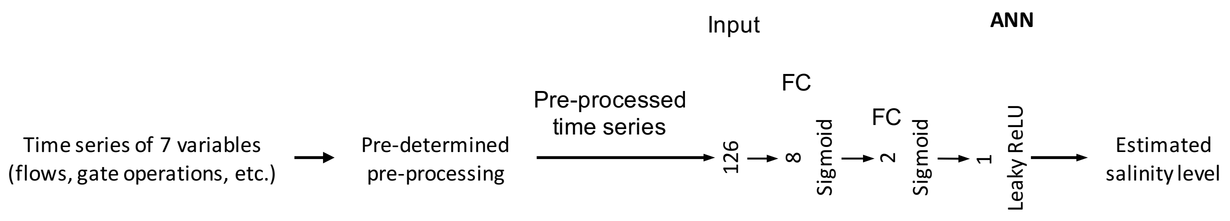


Fig. 2. ANN inputs and input locations. (Sources: Esri, DeLorme, HERE, TomTom, Intermap, increment P Corp., GEBCO, USGS, FAO, NPS, NRCAN, GeoBase, IGN, Kadaster NL, Ordnance Survey, Esri Japan, METI, Esri China (Hong Kong), swisstopo, MapmyIndia, and the GIS User Community; Data from ((SWRCB) 2000).)



Input
ANN
FC
FC
Pre-processed time series
Time series of 7 variables (flows, gate operations, etc.)
Pre-determined pre-processing
126
8
Sigmoid
2
Sigmoid
1
Leaky ReLU
Estimated salinity level

Fig. 3. Complete pipeline for ANNs according to (Jayasundara et al. 2020)

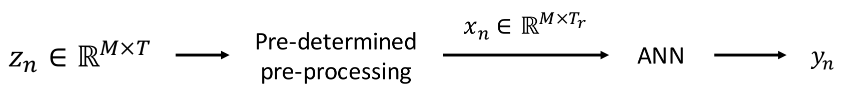

$z_n \in \mathbb{R}^{M\times T}$
Pre-determined pre-processing
$x_n \in \mathbb{R}^{M\times T_r}$
ANN
y_n

Fig. 4. Pipeline for ANNs with mathematical notations according to (Jayasundara et al. 2020)

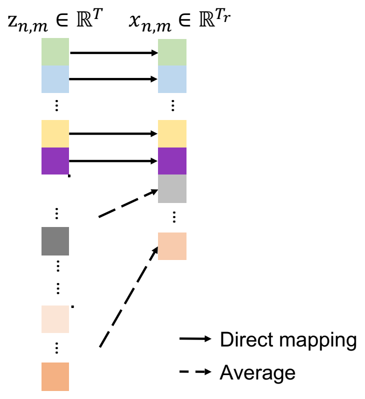


Fig. 5. Pre-processing diagram

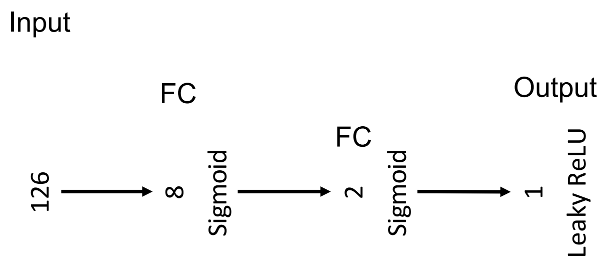


Fig. 6. Architecture of a single-task ANN

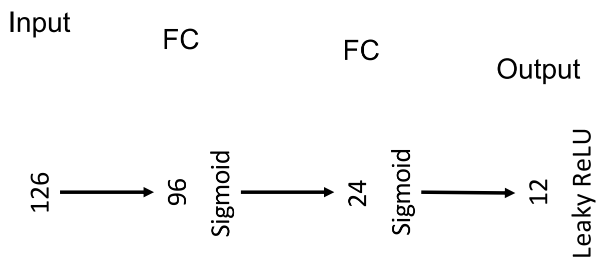


Fig. 7. Architecture of a multi-task learning ANN

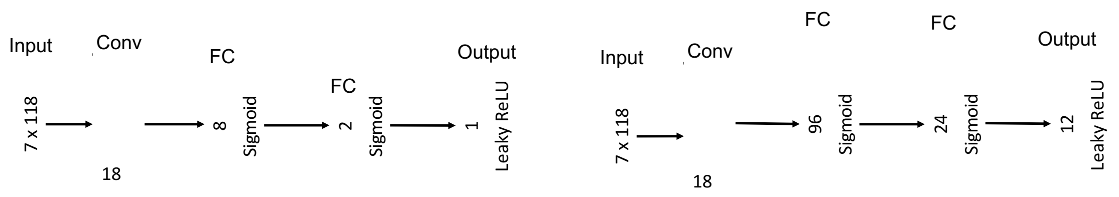


Fig. 8. STL (left) and MTL (right) ANN architectures with a convolution layer.

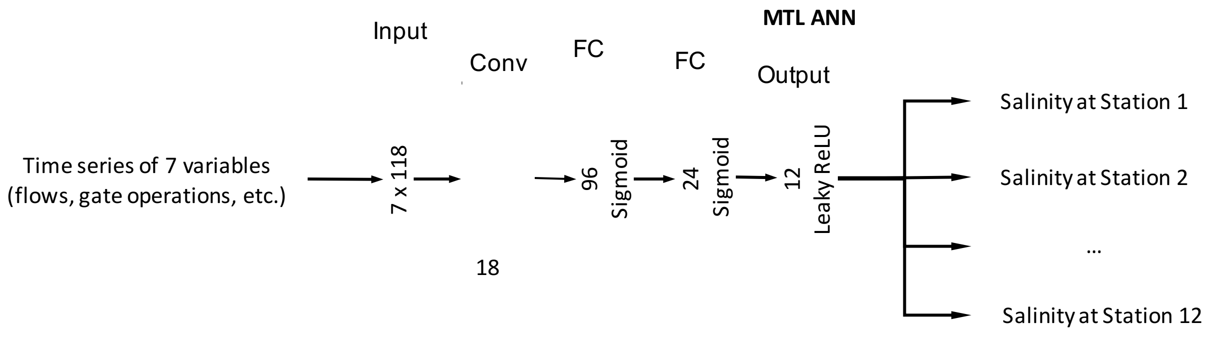


Fig. 9. Complete pipeline for proposed MTL ANNs

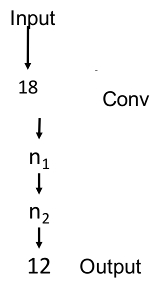
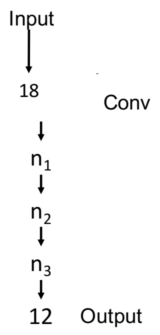
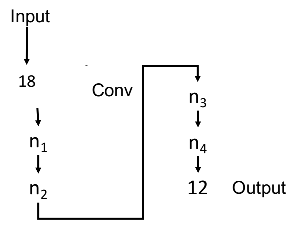
ANN Depth (Including Conv Layer)	Numbers of Neurons in hidden layers	Diagram
4 Layers	$n_1 = 96,\ n_2 = 24$	
	$n_1 = 96,\ n_2 = 48$	
	$n_1 = 126,\ n_2 = 48$	
	$n_1 = 192,\ n_2 = 24$	
	$n_1 = 192,\ n_2 = 48$	
	$n_1 = 192,\ n_2 = 96$	
	$n_1 = 384,\ n_2 = 24$	
	$n_1 = 384,\ n_2 = 48$	
	$n_1 = 384,\ n_2 = 96$	
	$n_1 = 384,\ n_2 = 192$	
5 Layers	$n_1 = 96,\ n_2 = 48,\ n_3 = 24$	
	$n_1 = 192,\ n_2 = 96,\ n_3 = 24$	
	$n_1 = 192,\ n_2 = 96,\ n_3 = 48$	
	$n_1 = 384,\ n_2 = 48,\ n_3 = 24$	
	$n_1 = 384,\ n_2 = 96,\ n_3 = 24$	
	$n_1 = 384,\ n_2 = 96,\ n_3 = 48$	
	$n_1 = 384,\ n_2 = 192,\ n_3 = 24$	
	$n_1 = 384,\ n_2 = 192,\ n_3 = 48$	
	$n_1 = 384,\ n_2 = 192,\ n_3 = 96$	
6 Layers	$n_1 = 192,\ n_2 = 96,\ n_3 = 48,\ n_4 = 24$	
	$n_1 = 384,\ n_2 = 192,\ n_3 = 96,\ n_4 = 24$	
	$n_1 = 384,\ n_2 = 192,\ n_3 = 96,\ n_4 = 48$	

Fig. 10. Numbers of neurons for expanded ANNs. Activation functions are not included in the diagrams.

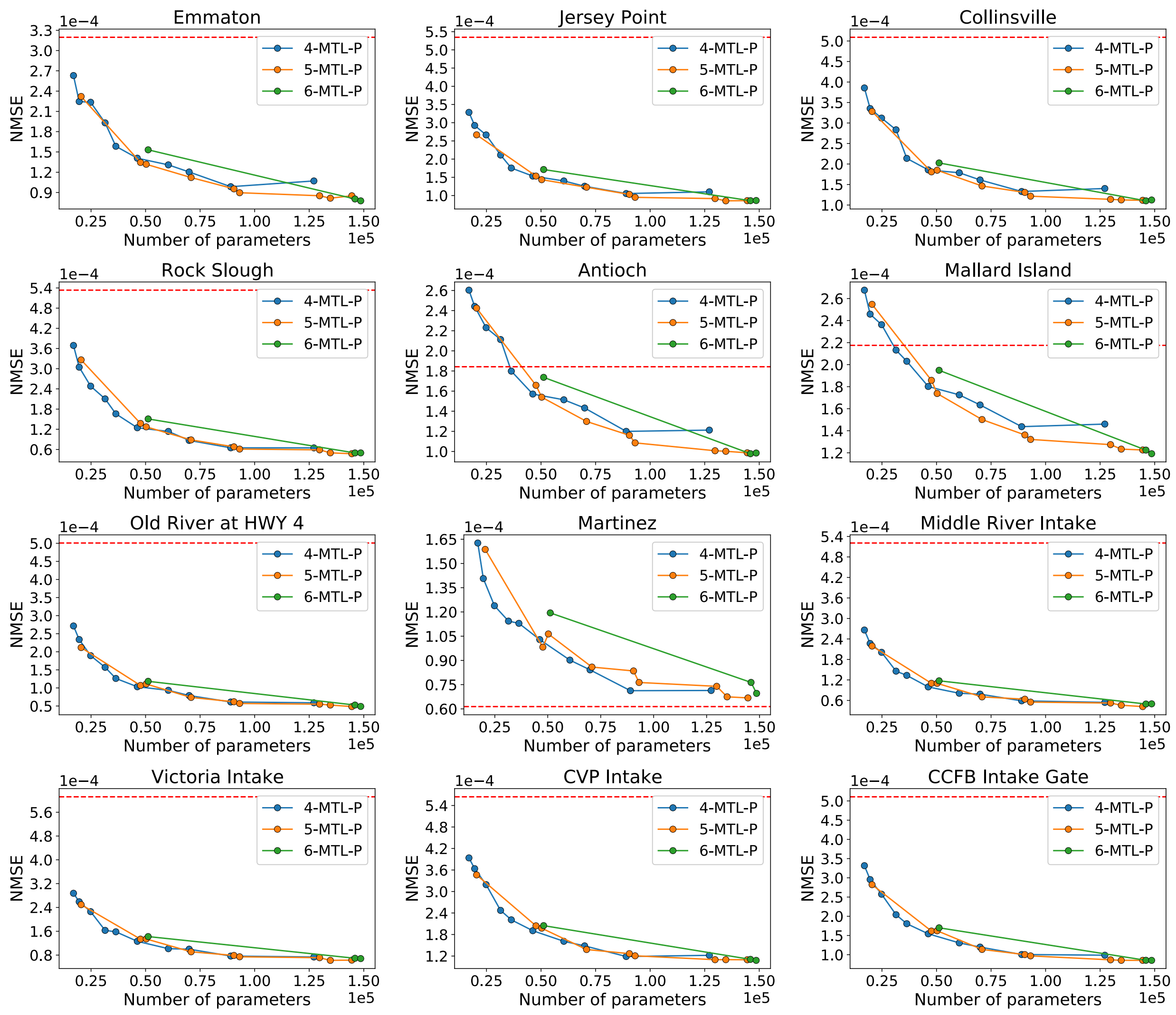


Fig. 11. NMSE values for 12 monitoring stations versus number of parameters in three MTL ANNs with different layers. Red dashed lines mark NMSE obtained by the STL-LM baseline in Table 1.

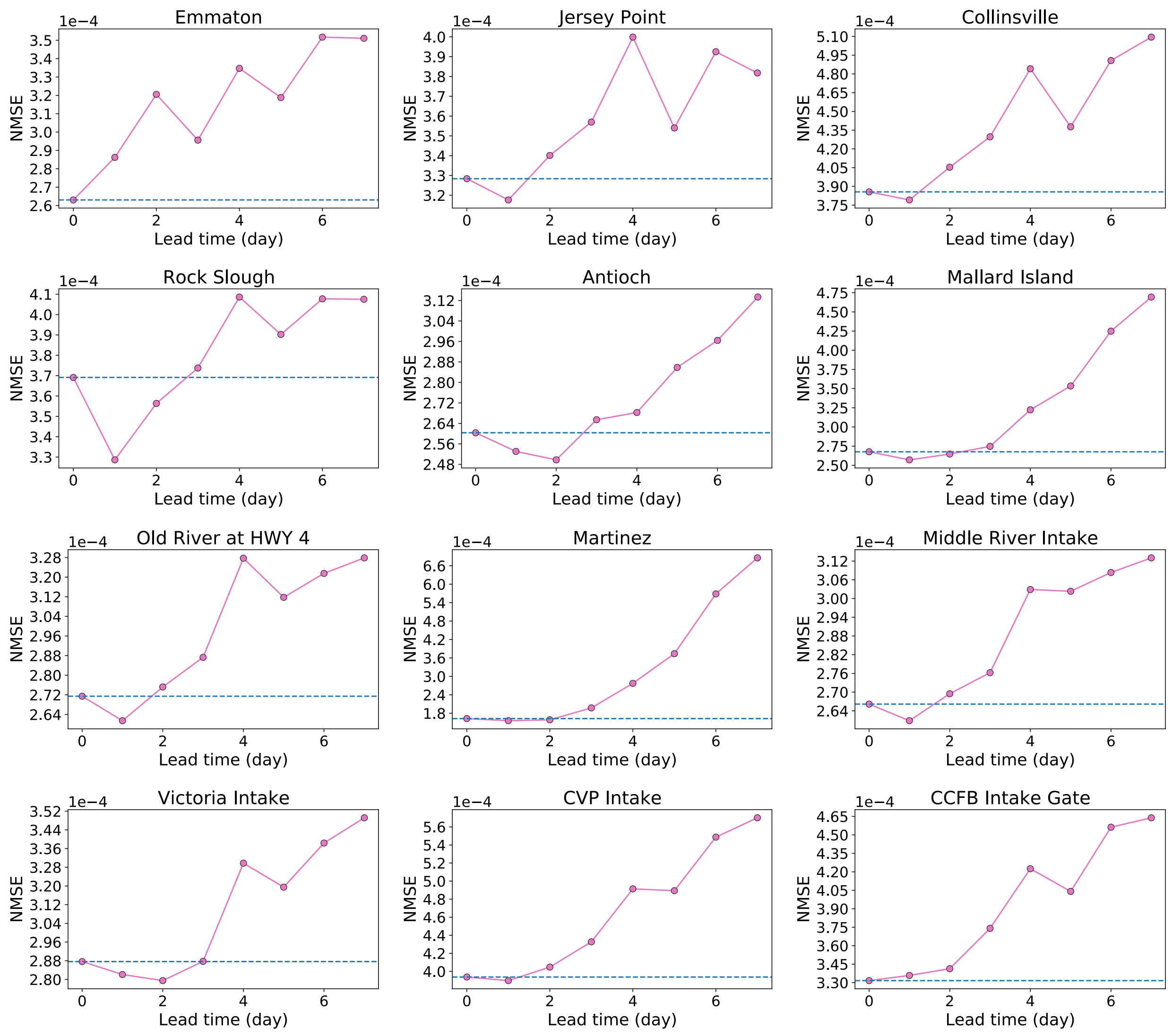


Fig. 12. NMSE values for 12 monitoring stations versus lead time (in days) with a 4-MTL-P ANN. Blue dashed lines mark NMSE obtained by the 4-MTL-P case in Table 1.