Convolutional Neural Network for Data Imputation

1st Shadduck, Dylan
dept. of Electrical Engineering (University of California, Davis)
Davis CA, United States
dmshadduck@ucdavis.edu

***Abstract*—The Sacramento-San Joaquin River Delta (the Delta) is home to many native species of plants and wildlife. The Delta supplies, "fresh water to two-thirds of the [California's] population and millions of acres of farmland" (California Department of Water Resources). Naturally, such an important water system is closely monitored for salinity to ensure that crops and drinking water supplies aren't spoiled. The Department of Water Resources (DWR) monitors 12 different stations in the Delta for salinity in addition to monitoring several inputs to the water system including tidal forces and river flow rates. This information can be used in conjunction with Artificial Neural Networks (ANNs) to help predict future water salinity and prevent disaster. In order to better train these networks, missing salinity data can be replaced to improve prediction accuracy. Methods such as median replacement or K Nearest Neighbors (KNN) can be simple for data imputation, but don't make predictions with the accuracy desired. In this paper I propose a convolutional neural network to provide a more complex and more accurate data imputation technique. This type of network is designed to learn the connection between stations and return accurate imputation results.**

***Index Terms*—CNN, Convolution, Neural Network, Mask, Data Imputation, Image Inpainting**

I. Introduction

The Sacramento-San Joaquin River Delta is a versatile water estuary located in the western part of the Central Valley in Northern California. This meandering water system is fed by the Sacramento and San Joaquin rivers while supplying nearby cities and fertile farmland with water. This estuary also feeds into the northern part of the San Francisco bay, which contributes salt water to the delta, thus making the water flowing there a mixture of freshwater and saltwater known as brackish water. The concentration of salt (salinity) in the brackish water varies dramatically depending on a multitude of factors, but most notably, the incoming flow from the rivers, the export flow, and the tide in the bay. Since farmers depend on this water system for crop irrigation and cities depend on it for drinking, there is a maximum salinity threshold that delta water can have before the crops begin to die or it becomes undrinkable.

To better control the salinity of the water in the delta, water can be released at specific flow rates from reservoirs up river from both the Sacramento and San Joaquin rivers. Unfortunately, it can take up to several days for the effect of water released from the reservoirs to impact the salinity of the delta. Knowing when to release the water upstream is impossible without close monitoring of the Delta salinity in real time and accurate models that can predict what future salinity values will be. In the past, DWR has relied on salinity predictions based on very complex modeling of the water movement throughout the entire Delta. These types of models are very computationally intensive and can take several hours to run on the machines that are available to them. Recent work done in [3] has shown that they are able to replicate the results of these complex water simulations using a quite simple ANN structure. The ANN they have designed returns results of similar accuracy to water simulations in a fraction of the time.

To further improve the ANN designed in [3], large missing sections of the dataset used can be filled in and replaced. To improve ANN training, it is essential to replace these missing values as accurately as possible. The convolutional neural network (CNN) proposed in this paper for data imputation will attempt to provide accurate data imputation for this dataset. The CNN model will be explained at length in section three. The datasets used in this paper are both provided by DWR and will be explained in further detail in section two. The metric for evaluating the CNN will be discussed in section four, conclusions will be provided in section five, and future work will be discussed in section six.

II. Datsets

A. Introduction

The datasets used in this paper were both provided by the California Department of Water Resources. Under ideal conditions, this data would all be historical and recorded in the Delta area. In reality, both datasets are provided from different complex Delta water simulation models. One model is able to predict water salinity at a frequency of one sample per month. This model was originally developed to follow the movements of groundwater (water that resides below earth's surface) and thus has a very low time-series granularity. The second model provides data at a frequency of one sample every 15 minutes, but has a problem of producing large missing sections of data.

The dataset provided with a granularity of one sample every month has simulated values from January 1940 through August 2019 across 12 different Delta stations. In addition, this monthly dataset has been further interpolated such that the frequency of the dataset is one sample per day. The method of interpolation used by DWR is linear interpolation. The dataset providing samples every fifteen minutes has simulated values from January 2000 through September 2019 across 26 different Delta stations.

Both datasets measure salinity in micro Siemens per centimeter $\mu S/cm$ This measurement is really a measurement of the conductivity of the body of water under consideration. As the salinity of water increases, the conductivity increases linearly, and therefore is a valid measurement for water salinity. For context, most salt water has a salinity on the order of 54,000 $\mu S/cm$ while fresh water can have salinity as low as 100 $\mu S/cm$. Typically brackish water, like that found in the Delta, has a salinity of 27000 $\mu S/cm$ or less.

B. Masking

In order to evaluate the effectiveness of the proposed CNN, we need to perform the data imputation task on a known portion of data. This way, we can compare the values predicted by our imputation techniques with the ground truth to expose the error that results from our chosen method. The idea here is that the imputation error that we calculate from known data should be similar to the error produced from truly unknown values, provided that our sample size is large enough and we can accurately model the missingness of the data.

Since our daily dataset does not have any missing values, it is impossible to model what missing values for this dataset would look like. The chosen model for the missing data will result in randomly selected day station pairs within a given window. The number of missing values within a window of data is constant throughout the entire dataset and can be varied to test the performance of our imputation methods at many different missing rates. The window chosen for this dataset is a 10 day span across all 12 salinity measurement stations. The dataset can be grouped into a large sample size of these 10 by 12 windows with a corresponding mask.

Once the data has been reorganized and masked, the true values that are to be artificially removed by the mask are filled in with a placeholder value predicted by linear interpolation. This placeholder value will give the training network a good starting point for determining the predicted salinity.

C. Scaling

Before our data is input to our training network, we will scale the data. The reason for doing so is to mitigate any boundary issues for neural network activation functions. Some activation functions like "Rectified linear unit" have a domain of $[0, \infty)$, while others such as the"Sigmoid" or "Tanh" function have a domain of $(-\infty, \infty)$. Even though the sigmoid and tanh functions have a domain of negative and positive infinity, the gradient diminishes greatly beyond a domain of [-1, 1]. Because of this, very large values either positive or negative produce very small gradients. To account for any potential activation function, the input salinity data is bounded [0,1] following (1)

$$s = \frac{x - m}{r} \tag{1}$$

where s is the scaled salinity value, x is the original salinity value, m is the minimum of the column/station, and r is the range of the column/station. The column range is defined as the difference between the column maximum and column minimum.

D. Train/Test Splitting

Finally, after the salinity data has been masked and appropriately scaled, the data must be split into training and test sets. This ensures that the data used to evaluate the network has not been used to train the network and prevents the evaluation from being biased. A split of 70 percent for training and 30 percent for testing/evaluation was chosen. This split is done using random selection without replacement and is performed by "train test split": a function built into the python package scikit learn.

III. Neural Network

The method explored here for the problem of data imputation is a Convolutional Neural Network. The reason this method was chosen is because of the similarity of this problem to the problem of image inpainting. Image inpainting is a method of filling in a masked image with an accurate or realistic prediction. This problem is explored in [1] and the results of their CNN based approach are incredibly realistic. Due to the success in [1], this CNN model will draw inspiration from their convolutional autoencoder "Completion Network".

A. Design

The design of this neural network follows an autoencoder structure similar to [1]. The reason behind this has to do with the randomness of the mask applied to our salinity data window. Since the CNN does not always know which values to predict, we have to train the neural network to predict all of the salinity values from the window with the input given. When the input shape to a neural network matches the output shape of a neural network, the network can be considered an autoencoder.

The input to the neural network is our 10 by 12 window of salinity samples. The next several layers perform 1D convolution along the temporal axis. The reason for choosing this axis is because it is assumed that the similarity of salinity values within the same column are more likely and therefore can be more easily compressed using convolution. Once the dimensionality of the network is sufficiently reduced, a fully connected dense layer is used. Finally, we perform the inverse process of a convolution, a convolution transpose. The convolution transpose is a method of performing a convolution, but instead with fractional strides. This allows the output of the convolution to be a larger shape than the input. A full breakdown of each layer used in this network is shown in table I.

B. Training

The training of this neural network is quite straightforward using the python package tensorflow. After the network layers have been designed, they are compiled into a tensorflow model. This step defines the input tensor(s) to the model as well as the output(s). Next the model is compiled and the

TABLE I
NEURAL NETWORK LAYERS

Layer Type	***Filters***	***Kernel Size***	***Strides***	**Out Shape**
Input				10x12
Conv1D	12	2	1	9x12
Conv1D	12	2	1	8x12
Conv1D	12	2	1	7x12
Flatten				84
Dense				84
Reshape				7x12
Conv1D Transpose	12	2	1	8x12
Conv1D Transpose	12	2	1	9x12
Conv1D Transpose	12	2	1	10x12
Output				10x12

Conv only parameter

optimizer, loss function, and CNN metrics are defined. Finally, the model is trained by calling the fit method on the network model and passing in the training dataset along with the true values of that set. The parameters of the training step are the batchsize, the number of epochs, and the validation split. The batchsize defines how many salinity windows the model will train on before applying gradient descent to the network following the optimization specified earlier. The epochs are the number of times the neural network is trained on all of the data. Even though the network has already been trained on this data, it is good to feed the full dataset through multiple times. The validation split used reserves a percentage of the data passed in for training to validate the accuracy of the model. This simple step can prevent the neural network from over training/over-fitting on the dataset. All of the training parameters are as follows:

- Batchsize = 16
- Validation Split = 0.2
- Epochs = 500
- Optimizer = ADAM
- Loss = MSE
- Learning Rate = variable [4.8e-3, 1e-6]

IV. NEURAL NETWORK EVALUATION

A. Error Deffinitions

To evaluate the predicted values of the CNN, we will be using two different metrics: Mean Squared Error (MSE) and Mean Absolute Percent Error (MAPE). Mean squared error is a metric that is more negatively affected by larger erroneous results.

$$MSE = \frac{1}{N}\sum_{n=1}^{N}(x_n - \hat{x}_n)^2 \tag{2}$$

Where $\hat{x}_n$ is a predicted value, x_n is the true value, and N is the total number of data points. This means that as predicted values stray further from the truth, the error received is the square of the difference between predicted and truth. MAPE, on the other hand, does not punish larger erroneous values any more than small erroneous values.

$$MAPE = \frac{1}{N}\sum_{n=1}^{N}\left|\frac{x_n - \hat{x}_n}{x_n}\right| \tag{3}$$

The error received increases linearly with the difference between truth and predicted values.

B. Baseline Comparison

To compare the CNN with other imputation methods, a baseline is required. The baseline chosen for this project is linear interpolation, the same method as is used to replace the missing values during data masking. The reasons for this choice are that linear interpolation is a relatively simple method and thus our CNN should outperform it, and that this is the method currently used by DWR to interpolate any missing values they encounter.

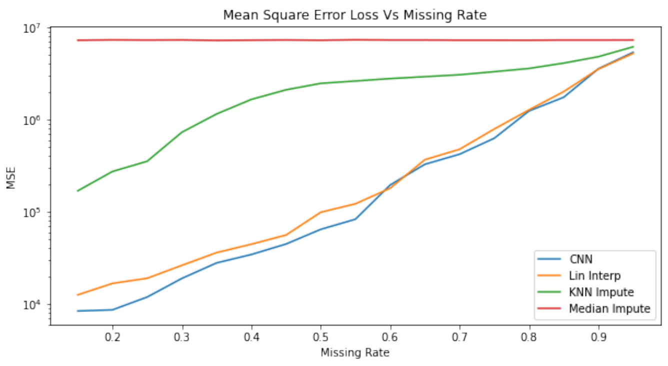
Initially comparing these two methods, data is masked with a chosen missing rate of 20 percent. This means that in a 10 by 12 window with 120 salinity values, 24 of these values will be masked at random without replacement. Training the neural network with the training parameters specified above and running linear interpolation on the data resulted in table II.

TABLE II
CNN VS LINEAR INTERPOLATION

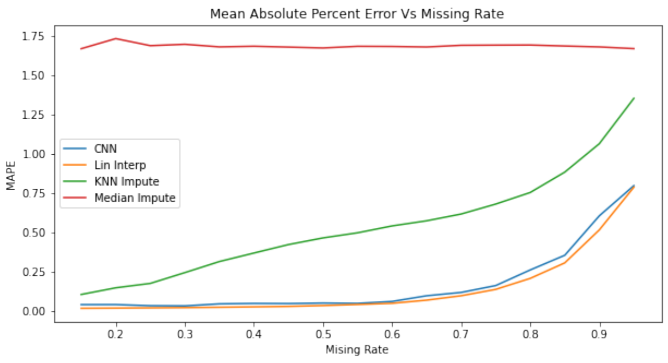
	CNN		**Lin Interp**	
Station	**MSE**	**MAPE**	**MSE**	**MAPE**
Emmaton	4.84e3	0.048	4.28e3	0.022
Jersey Point	8.51e2	0.028	3.14e3	0.014
Collinsville	1.04e4	0.063	2.75e4	0.034
Rock Slough	4.40e1	0.0092	2.24e1	0.0039
Antioch	3.28e3	0.055	1.45e4	0.024
Mallard	2.54e4	0.073	6.18e4	0.034
Los Vaqueros	3.66e1	0.009	5.86e1	0.0066
Martinez	1.36e5	0.039	1.26e5	0.024
MiddleRiver	3.96e1	0.009	1.93e1	0.0044
Vict Intake	4.30e1	0.01	4.28e1	0.0099
CVP Intake	8.74e1	0.012	8.74e1	0.0099
CCFB OldR	1.07e2	0.014	9.43e1	0.0095
Average	1.50e4	0.031	1.98e4	0.016

C. Comprehensive Comparison

To further evaluate the CNN, a comparison will be made with two more methods that are used frequently to impute missing data: KNN and Median Value. KNN imputation, like that used in [2], is a method of imputing data based on other similar conditions. An example of this would be if a single station is missing a salinity value for a given day, we can look at the other 11 stations to see what their values are. To predict the missing value we can select a number, K, other data samples with the most similar salinity values for the other 11 stations and then average the value for our missing station across all K similar samples. Median Value replacement is a much more simple method than KNN. The basis of this method is to determine the median value for each station and then replace any missing values with the corresponding median value for that station.



(a) Mean Square Error



(b) Mean Absolute Percent Error

Fig. 1. Full Imputation Method Comparison (a) MSE (b) MAPE

The full comparison of all four imputation methods (CNN, Linear Interpolation, KNN, and Median Value Replacement) is performed across a wide range of missing value rates and then MSE and MAPE are computed. Performing this test over a missing value range of 15 - 95 percent results in Fig. 1.

V. Conclusions

The results shown in the section above are promising, but do not meet the initial goals of this work. The CNN easily outperforms simple methods like Median Value Replacement and even more complex methods like K Nearest Neighbors Imputation in both metrics we explored. Unfortunately, the CNN does not truly outperform the baseline method of linear interpolation. Across all missing rates tested, the CNN does outperform linear interpolation in terms of MSE. On the other hand, linear interpolation is consistently outperforming the CNN in terms of MAPE. This could be for a multitude of reasons, but the most likely is that the loss function used to train the CNN is MSE loss. That being said, if our CNN could truly learn the patterns between the data, it should be able to outperform a simple method such as linear interpolation across both metrics.

Other shortcomings of this work include the dataset selection. The dataset used to train the CNN was originally linearly interpolated by DWR prior to data handling. This could be another big reason that linear interpolation is performing so well during testing. If only one sample out of every month is true and the rest are filled in with linear interpolation, then the likelihood of a randomly masked value being an interpolated value is quite high (around 96 percent for 30 day month). Then when masked data is filled in with linear interpolation, the performance is great (MAPE of 0.016 with missing rate of 20 percent).

To truly test the capabilities of the CNN, a new dataset with more realistic salinity values should be chosen to prevent bias towards linear interpolation. Some preliminary works on such a dataset are elaborated on in the next section.

VI. Future Work

Continuing the work detailed in this paper, the CNN described here can be applied to the 15 minute dataset from DWR. Despite large sections of missing data present in this dataset, there is enough recorded data to perform adequate CNN training and testing.

A. Data Masking

Rather than apply masking to random data points, masking is performed in sections, similar to the nature of the missing data in this set. Also, rather than looking at 12 different stations, the initial focus will be on a single station. To provide the CNN with enough data to make accurate predictions, the number of data samples provided to the network is increased. The new input to the neural network is a four day section of time series salinity data that has been scaled and masked. Given that samples are taken every fifteen minutes, 384 salinity samples are provided to the network in addition to a 384 sample corresponding mask.

B. Model Refresh

TABLE III
Neural Network Layers: Revisited

Layer Type	***Filters***	***Padding***	***Kernel Size***	***Strides***	**Out Shape**
Salinity (Input)					384
Mask (Input)					384
Concatenate					2x384
Conv1D	4	none	5	2	4x192
Batch Norm					4x192
Conv1D	4	none	4	2	4x96
Batch Norm					4x96
Conv1D	5	same	4	2	5x48
Batch Norm					5x48
Conv1D	5	same	4	2	5x48
Batch Norm					5x48
Flatten					240
Dense					240
Reshape					5x48
Conv1D	5	same	4	2	5x48
Batch Norm					5x48
Conv1D	5	same	4	2	5x48
Batch Norm					5x48
Conv1D Transpose	4	none	2	2	4x96
Batch Norm					4x96
Flatten					384
Output					384

Conv only parameter

The shape of the CNN designed earlier will need to be changed to accommodate the change in input shape. Rather than a 10 by 12 matrix, there are now two inputs of 384 samples each. The same 1D convolution layers are used to reduce the dimensionality of the data and a dense layer is included at the center of our autoencoder structure. 1D transpose convolutional layers are used to expand the dimensions of the data until the final shape of 384 samples is reached. "Same" padding is used in some convolutional layers to preserve the same shape at the input and output of the layer. This allows for additional training parameters for the model without reducing the dimensionality of the data any further.

The biggest addition to this network is a batch normalization layer after each convolutional layer (including transpose). This additional step ensures that the output data from the convolution or transpose convolution has a mean of zero and a standard deviation of approximately one. This step ensures that the gradient computed at each layer does not diminish over time. This new model is described in table III

VII. Preliminary Results

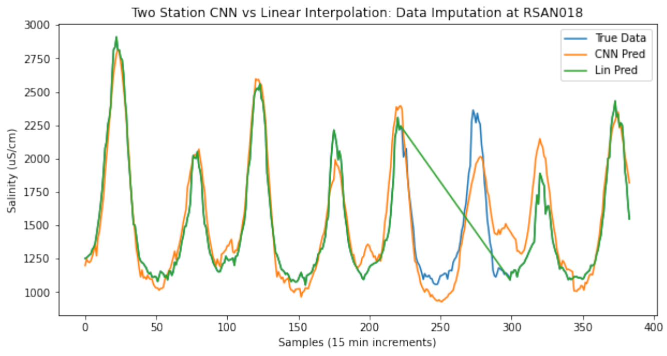


Fig. 2. CNN vs Linear Interpolation: Large Range

This neural network was trained on the 15 minute dataset from DWR with a missing rate of 20 percent on only one station, RSAN018. The predictions made by this CNN are able to accurately fill in missing data that exhibits a high degree of variability in the missing region. The linear interpolation method is not able to predict missing data in the same manner and often falls short in this preliminary test. The predicted values from both methods as well as the ground truth are presented in Fig. 2.

Evaluating a few more four day sections of data on the trained CNN reveals an unfortunate consequence of learning to predict the oscillating salinity. The CNN tends to perform best on four day sections that have a large range (> 1000 $\mu S/cm$). Sections that have a small range (< 200 $\mu S/cm$) are difficult for the CNN to predict and the result is often over-fitting. This problem can be seen clearly in Fig. 3.

Overall, the average MSE for the CNN was 2.84e4 compared to 3.42e4 with linear interpolation. The average MAPE for the CNN was 0.178 compared to 0.033 with linear interpolation.

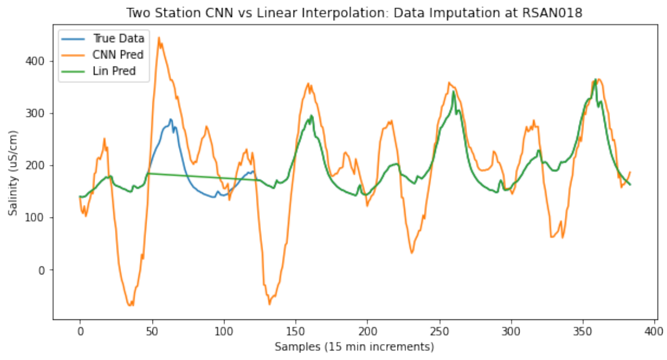


Fig. 3. CNN vs Linear Interpolation: Small Range

A. Further Improvements

Further improvements could be made to this CNN by making a few minor adjustments. As the data is scaled prior to being input to the CNN, sample windows with a very small range will likely present data input with a variation on the order of 1e-4. These values can be handled by the CNN, but the gradient computed will be very small. This can lead to the CNN favoring sample windows that have a large range as seen above. To correct this, each window can be scaled to the desired range after being input to the CNN. This should prevent the network from favoring large variability and overfitting on certain windows.

Another adjustment that can be made is to include additional discriminator neural networks like those discussed in [1]. This could prevent wildly unlikely values from being predicted by the network and also produce smoother transitions within the predicted regions.

References

[1] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Globally and locally consistent image completion. *ACM Trans. Graph.*, 36(4), jul 2017.

[2] Phayung Meesad and Kairung Hengpraprohm. Combination of knn-based feature selection and knnbased missing-value imputation of microarray data. In *2008 3rd International Conference on Innovative Computing Information and Control*, pages 341–341, 2008.

[3] Siyu Qi, Zhaojun Bai, Zhi Ding, Nimal Jayasundara, Minxue He, Prabhjot Sandhu, Sanjaya Seneviratne, and Tariq Kadir. Enhanced artificial neural networks for salinity estimation and forecasting in the sacramento-san joaquin delta of california. *Journal of Water Resources Planning and Management*, 147(10):04021069, 2021.