ANN Architecture, memorization and overfit

Ryan Ripken
Resource Management Associates
Edward Gross Ph.D., RMA and Eli Ateljevich, DWR

Delta Salinity Management in Drought:
Surrogate Development under Drought,
Landscape Change and Sea Level Rise

CWEMF Annual Meeting
September 24, 2024

Pilot Project Objective

Develop and test a methodology for creating fast surrogate models for use in CalSIM representing the relationship of Delta salinity to hydrology and operations under management alternatives intended to mitigate impacts of extended droughts

Modeling Workflow

Topics

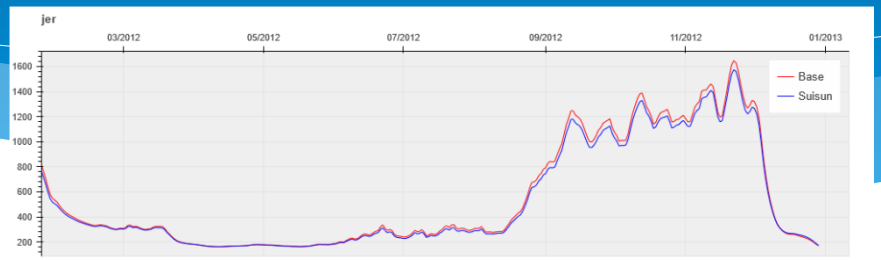
* Goal
* Neural Network Review
* Training
* Overfitting
* Types of Validation
* Techniques to reduce overfitting
* Future

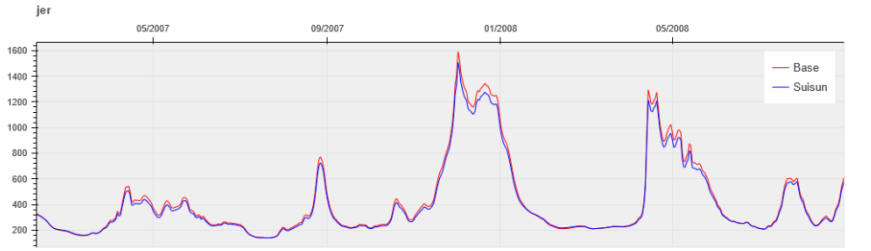
Goal

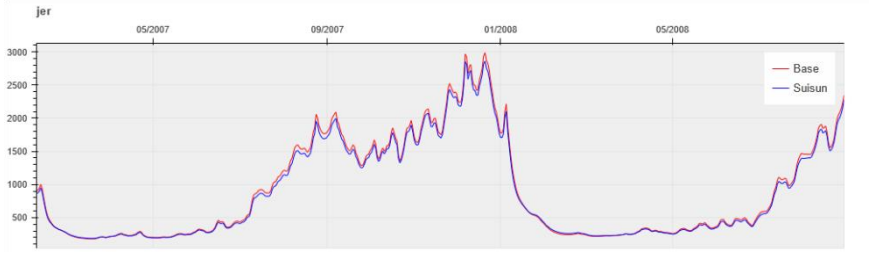
* Create Neural Network surrogates for higher dimensional models (RMA and SCHISM).
* The Neural Networks are trained on model results for the base case and cases that include various restorations.
* Goal: The trained models should predict salinities consistent with the corresponding simulation results.

Goal

* The model results can be different in subtle ways.
* Here are three examples of RMA results for EC values at Jersey Point with and without the Suisun restoration.
* At Jersey Point the EC Values differ by about 3%.
* We'd like our Neural Networks to be able to capture this level of sensitivity



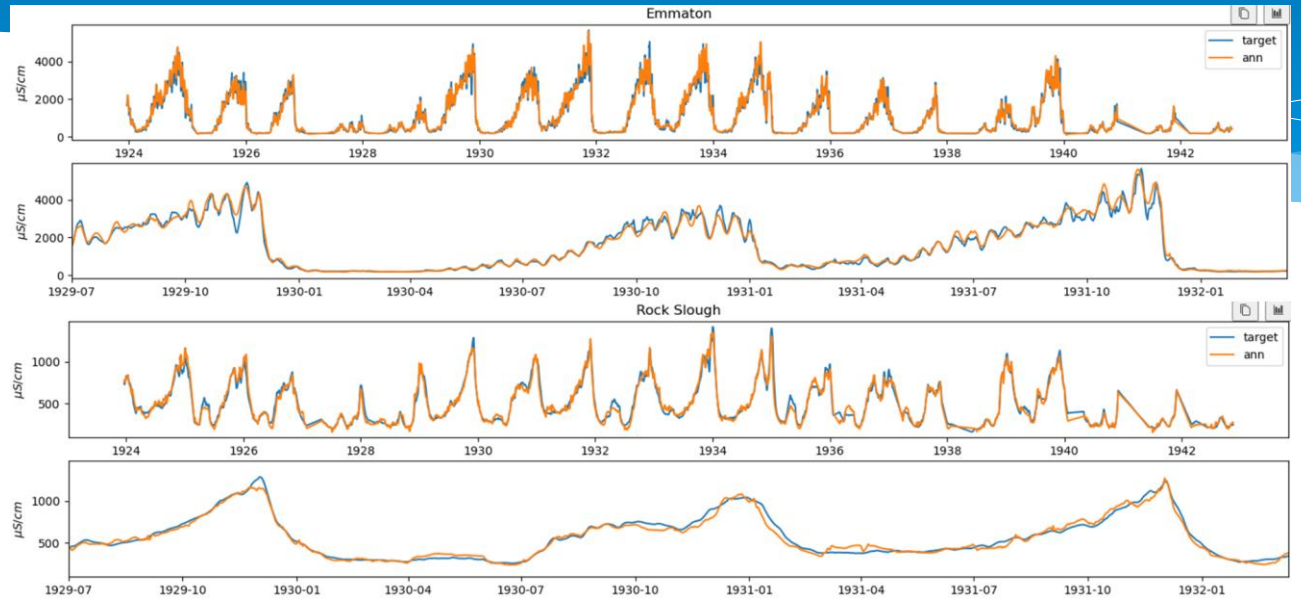




Neural Network Review

* Calsim includes a Multi-Layer-Perceptron Neural Network for predicting EC and X2.
* Timeseries prediction is accomplished by adding time-lagged inputs and treating the problem as a Supervised Learning-Regression problem.
* Use a feedforward neural network with fully connected layers where each neuron receives input from all neurons in the previous layer and outputs to all neurons in the next.

Existing Calsim ANN



* The existing Calsim DSM-2 trained Neural Network has outstanding performance.
* What performance can we get on a much smaller training dataset?

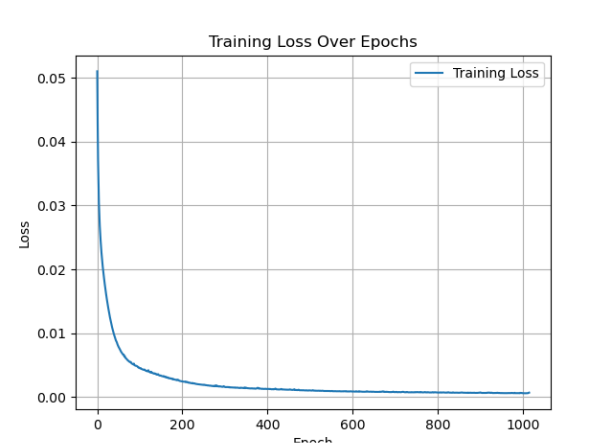
Neural Network Review

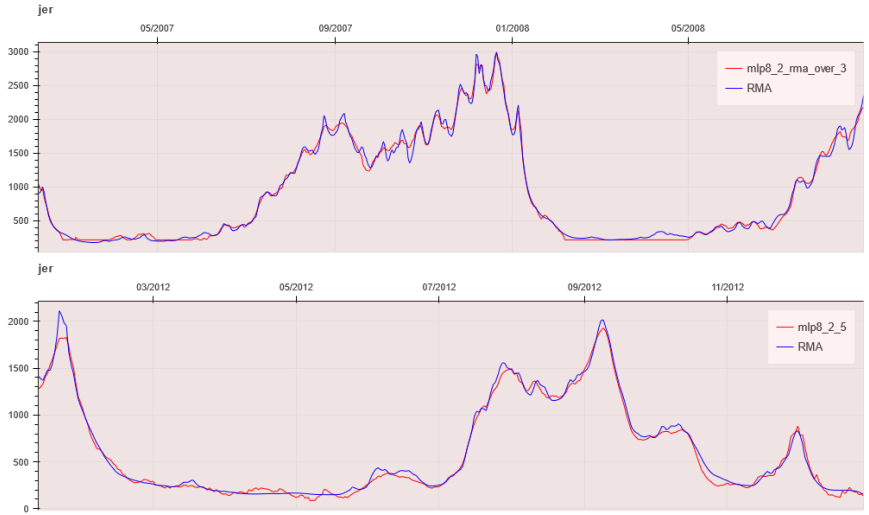
* In Python using Keras:

```
model = Sequential([
    Dense(8, input_shape=(total_features,),  activation='sigmoid' ),
    Dense(2, activation='sigmoid' ),
    Dense(1)
])
model.compile(optimizer=(Adam(learning_rate=0.001)), loss=mean_squared_error,
metrics=[mean_absolute_error])
history = model.fit(scaled_X, scaled_Y, epochs=1500, verbose=1)
```

Training

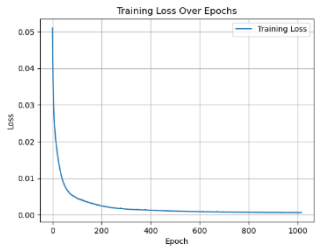
* Trained for a large number of epochs
* Monitoring the training loss (mse)
* Amazing looking performance
* Neural Network overfitting.

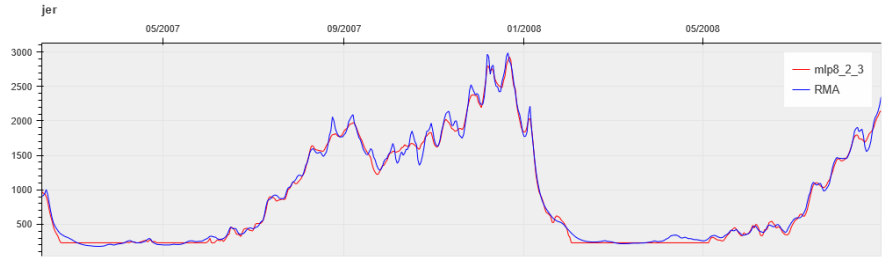


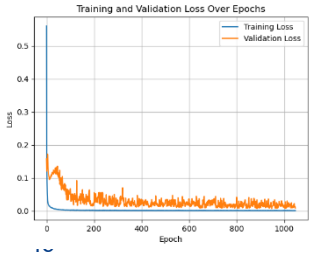


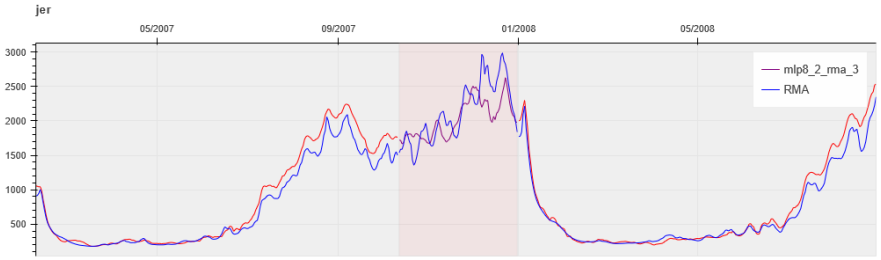
Training

* Hold back a Validation Set.
* Monitor Validation Loss
* Performance in Validation set is worse
* Performance on the Training set is also worse!
* What happened?









Training Validation Training

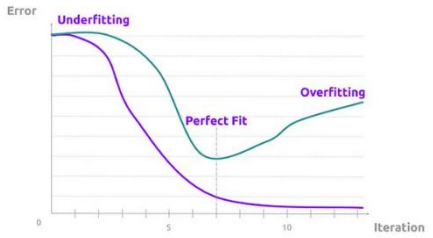
Dataset

* Dataset is small.
* 7 scenarios **2800** data points.
* Many of the points are less interesting high-flow low-salinity periods
* The Neural Net is also small, two hidden layers with 8 neurons and 2 neurons.
* Time lags add large number of input features => 1397 tunable parameters.
* Holding ‘interesting’ data back for validation resulted in less training data in the ‘interesting’ category.

Overfitting/Underfitting

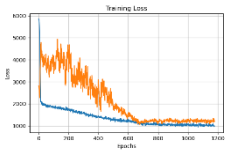
* Underfitting/Overfitting

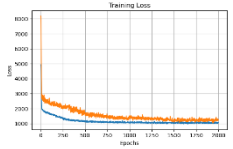
Text-book example



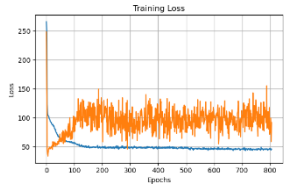
Real-world Examples

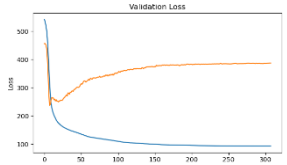
Okay

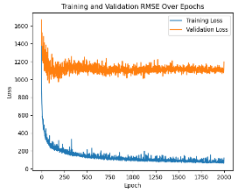




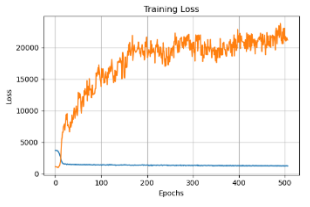
Not good

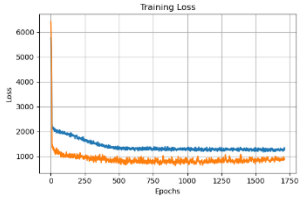






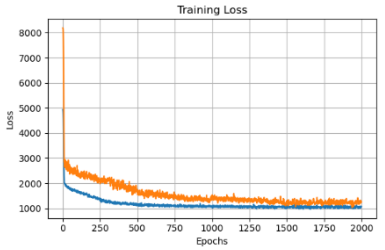
Pathological



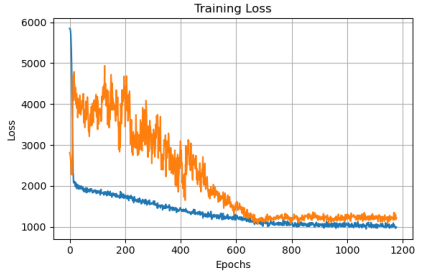


https://towardsdatascience.com/overfitting-and-underfitting-visually-explained-like-youre-five-8a389b511751

Overfitting/Underfitting

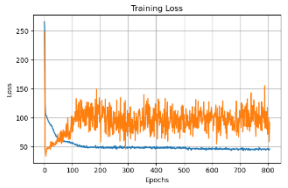


Almost ideal. Performance improved on training set and validation set throughout the training.

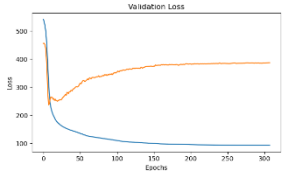


Noisy start but performance improvement.

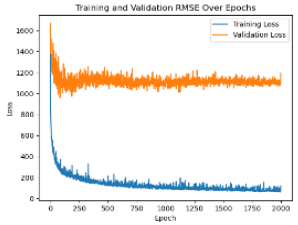
Overfitting/Underfitting



Somehow the first couple training steps scored amazingly well in the validation data. This is problematic when combined with restore_best_weights

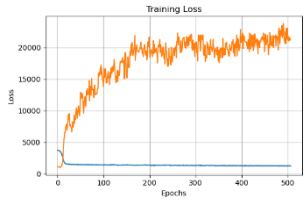


Once again best validation scores happened in first couple Epochs. Use lower learning rate.

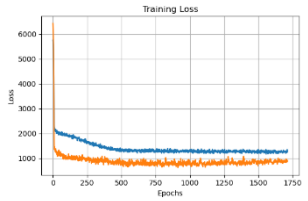


Overfitting and large gap between Training and Validation performance.

Overfitting/Underfitting



Validation loss grows to tremendous levels



The neural network consistently does better on data its never seen and isn't training towards.

Overfitting/Underfitting

* What can we do?
  * L1/L2 Regularization
  * BatchNormalization
  * DropOut
  * EarlyStopping
  * Fewer Features
  * Smaller Network
  * Data Augmentation
  * More Data
  * Different architecture

Overfitting/Underfitting

* L1/L2 Regularization
  * L1 penalty moves weights towards zero.
  * L2 penalty on squared weights.

* Neither is effective for this problem. L1 can be used to make an overly large network more sparse but our neural net is already small
* L2 reduces the effect of outliers but in our application we care the most about the outliers. Tends to trim of the peaks of predictions

Overfitting/Underfitting

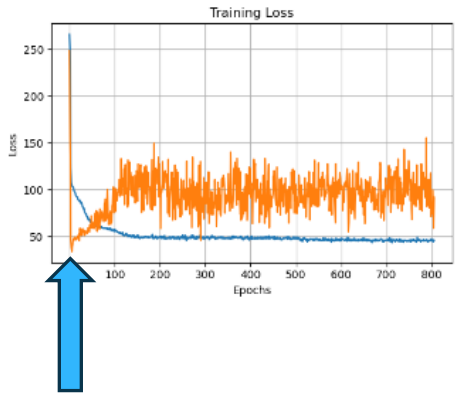
* BatchNormalization
  * Adjusts the values to mean 0 and standard dev 1.
  * Each Epoch takes slightly longer to compute
  * Training converges faster overall
  * Better performance

Overfitting/Underfitting

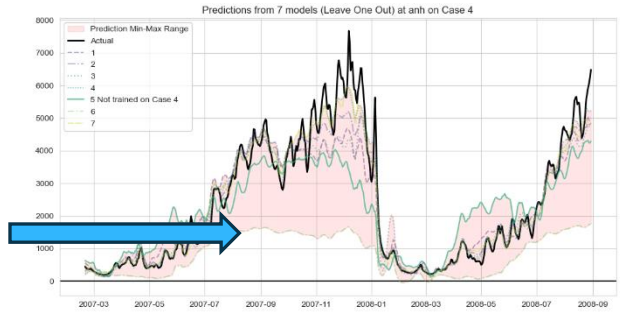
* DropOut
  * Neurons are randomly removed during training.
  * Neural Network is already small
  * Removing 25% or even 10% of the network is significant.

Overfitting/Underfitting

* EarlyStopping
  * Usually helpful
  * Can be combined with restore_best_weights
  * Occasionally catastrophic.



Best validation score happened in first couple Epochs.
Restore_best_weights restored to that point in the training and the resulting model performs poorly, even on data in its training set.



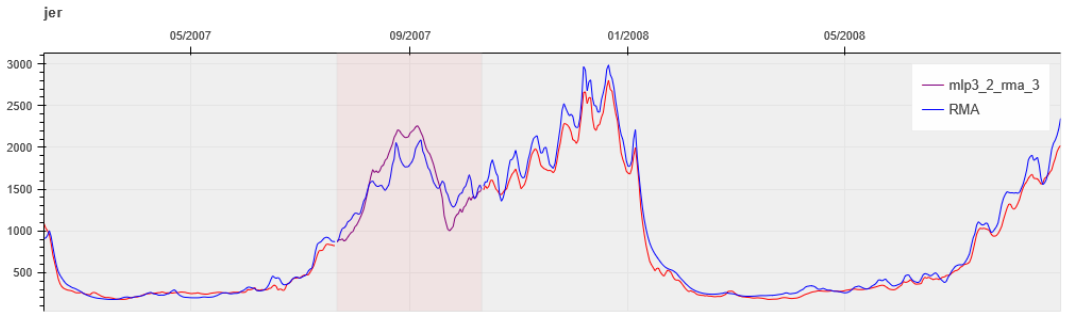
Overfitting/Underfitting

* Fewer Features
  * Possibly remove one of [Sac, Sjr, CU, Exports, NDO]
  * Possibly reduce individual day lags
  * Reduce number of time of time windows
  * Increase the size for the windows
  * Need to optimize for each Location
  * Can use SHAP or permutation importance to help determine impact of individual features.

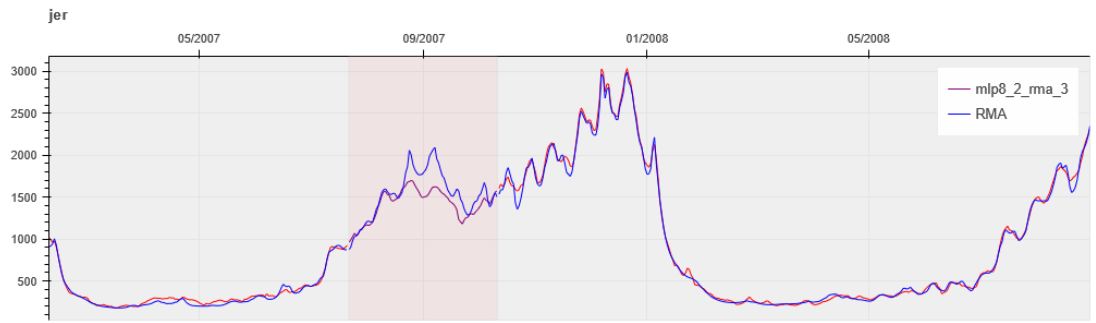
Overfitting/Underfitting

* Smaller Network
  * Can do a grid search for optimal shape.

3-2

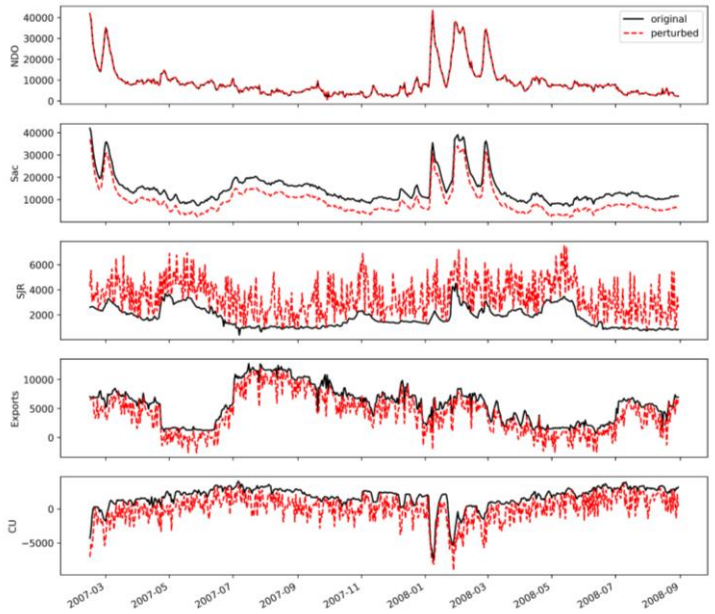


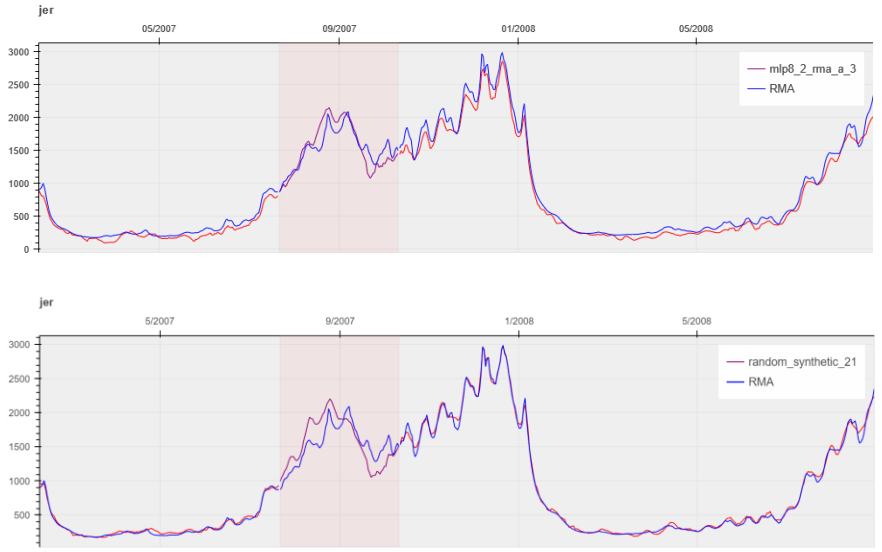
8-2



Overfitting/Underfitting

* Data Augmentation
  * Add noise to the existing data
  * Synthetic inputs that reallocate portions of Sacramento flow while maintaining NDO





Overfitting/Underfitting

* Add More Data
  * Many of the challenges are simplified if we add data.
  * Expect Dataset size to increase by approximately 25%
  * Possibly add 2 additional years for test/validation.

Future Directions

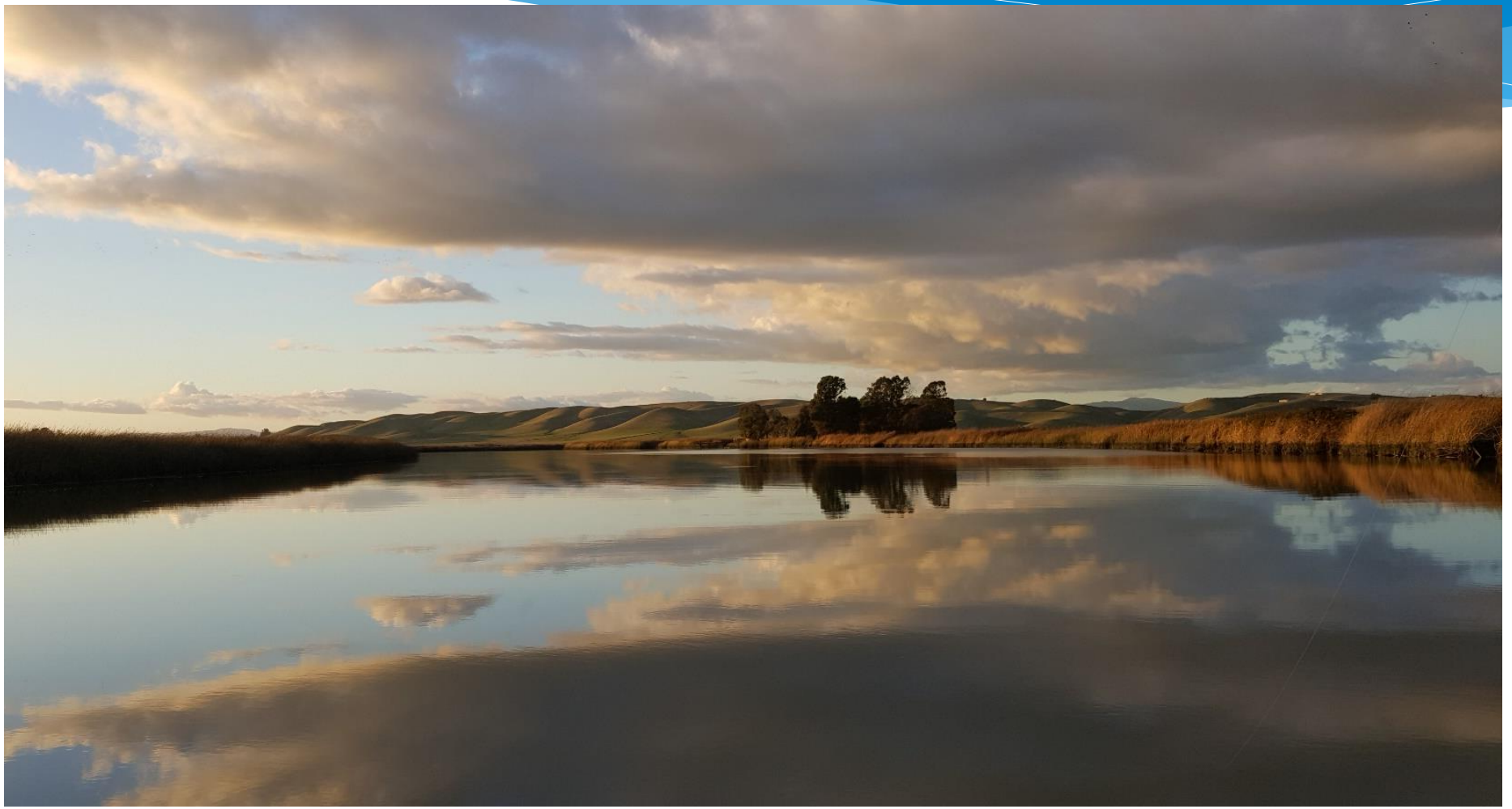
* Different Architectures
  * LSTM/GRU
    * Experiments ongoing, preliminary results promising.
  * Multi-variate Neural Networks can predict several locations at once.

* Predict a Residual
  * Transfer Learning
  * Residual Network

Summary

* Trying to predict a very subtle signal.
* Training dataset was designed to be small
* Additional runs are being done to supply more data.
* Overfitting and Memorization are a problem.
* Used several strategies to mitigate.

Questions?



Contact Information

John F. DeGeorge, Ph.D., P.E.
Resource Management Associates, Inc.
4171 Suisun Valley Road, Suite J
Fairfield, CA 94534
(707) 864-2950
jfdegeorge@rmanet.com
www.rmanet.com

Eli Ateljevich, Ph.D., P.E.
California Department of Water Resources
1516 Ninth St, 2-207
Sacramento, CA 95814
(916) 902-6984
Eli.Ateljevich@water.ca.gov