Estimation of Daily Water Temperature using Fully-Connected Neural Networks

B Steele

November 10, 2023

GH Repo

Scientific motivation and problem statement:

Water temperature is often an indicator of water quality, as it governs much of the biological activity in freshwater systems. While temperature is an important parameter to monitor in freshwater lakes, manual monitoring of waterbodies (by physically visiting a site) and sensor networks to monitor water temperature, are costly endeavors.

In this example, I will use a fully-connected neural network to estimate water surface temperature for reservoirs with long manual monitoring data from Northern Water, the municipal subdistrict that delivers drinking water to approximately 1 million people in northern Colorado and irrigation water for ~600,000 acres of land. The features that I will be using to estimate surface temperature include summary NLDAS meteorological data (air temperature, precipitation, solar radiation, and wind) as well as static values for each of the reservoirs (elevation, surface area, maximum depth, volume, and shoreline distance). The NLDAS data have been summarized for the previous day's weather, 3 days prior, and 5 days prior - meaning, the model does not use *today's* weather for prediction. To capture annual warming and seasonal warm-up/cool-down, which are not always consistent between annual cycles, I've implemented an annual cumulative sum for both temperature and solar radiation and the day of year within the feature set.

In addition to the manual sampling record that is maintained by Northern Water (n = 1125), I will be leveraging surface temperature estimates from the Landsat constellation, Landsat 4-9 (n = 5039). These thermal estimates are well-aligned with the manual monitoring data for the 7 reservoirs and have been bias-corrected for over estimates in the warmest months. 'Surface temperature' in the manual sampling record for this example is any measured temperature at >= 1m depth. I retain only the top-most value for temperature. Static variables are only available for 6 of 7 reservoirs, so Windy Gap reservoir has been dropped from this analysis (loss of 76 and 224 rows in aforementioned datasets).

The comparative baseline for this analysis will be the day-of-year average water temperature across all lakes and years, where there are at least 3 values contributing to the mean. The baseline estimates result in a MAE of 2.15 deg C, MSE of 6.98 deg C, RMSE of 2.64, and MAPE of 23.06%.

Current models to predict water temperature have used recurrent neural networks (specifically LSTMs/process guided models) with a median RMSE of 1.65 deg C across 68 lakes in the midwest (Read et al. 2019) and process guided and metatransfer learning networks with 1.88 deg C across more than 300 lakes (Willard et al. 2021). While these metrics describe error across multiple lake depths (not just surface temperature) and are propagated through more complicated networks, I think these are examples of well-performing model error metrics.

Training/Validation/Testing

Preprocessing

All precipitation data are right skewed heavily biased to low precip values including zero, to account for this and make the distribution more normal, I added 0.0001 to each value and applied a log-10 transformation to this subset. The wind data were left skewed and to transform the distribution, I used a square root transformation. All features and inputs were then scaled using the mean and standard deviation to get the values closer around zero, which are preferable for neural networks.

Train-validation-test split

Eventual implementation of this algorithm will include forecasting of temperature for these lakes as well as lakes that have only Landsat-derived temperature estimates and that are not included in this dataset. Because I want this algorithm to perform well on new lakes, I want to take steps to make sure that the algorithm is not overfit to these specific lakes static characteristics. While this information may be important for alogorithm development, the model may have a propensity to "learn" those key attributes and overfit to the data, not allowing for generalization beyond these lakes.

For training and validation I will use two techniques. First, a leave-one-out method that will result in six NN models where each iteration will use data from a single lake for validation and the other five for training. While the random forest models did not appear to overfit to the static variables, I'm still going to utilize a timeseries split method that will subset the data into ~10 year increments and leave one increment out per training and use it for validation per iteration. Since the intended implementation will be daily forecasts, testing performance will be assessed through hindcasting. The hindcast dataset is a holdout dataset for the years 2021 and 2022 across all lakes including in training/validation.

Model Architecture

The models described in this document are comprised of a fully-connected neural networks with an input layer, a dropout layer, hidden layers (number of layers, number of neurons, and activation functions defined in `settings`), and an output layer with a single neuron with a linear activation function. I used the `tf.keras.optimizers.Adam` optimizer which employs an adaptive learning rate. Loss function was set to MSE (`tf.keras.losses.MeanSquaredError`). For the purposes of this assignment, L1 and L2 regularization were both set at 0 for all models.

Hyper-parameter tuning

I first attempted to overfit the neural network to assure that my workflow was operating correctly with zero dropout and five hidden layers of 30 neurons each, all with leakyReLu activation. Learning rate was set to 0.001 with a patience of 500 and max epochs of 1000. Batch size was set at 128.

For the purposes of this analysis, I define an *overfit* model as that has minimizing loss over epochs for the training set and exhibits increasing loss with increasing epochs for the validation. This behavior indicates that the model has been customized to the training data and is no longer transferable to other datasets with any viable accuracy. Because I used the same settings for the LOO and the timeseries splits, there were different degrees of overfit between the two training-validation split types. The LOO was substantially overfit with these settings (Supplemental Figure A), but the timeseries split did not seem to be as overfit as the LOO scenario (Supplemental Figure B).

From here, I manually tuned the batch size, number of hidden layers, and number of neurons per layer until I felt like the model was performing well and where overfitting was minimized. I reduced the number of hidden layers and the number of neurons per layer, reduced the batch size, and added a dropout of 0.2.

```
"leaky_basic" = {
    "hiddens": [20, 20, 20],
    "activations": ["leaky_relu", "leaky_relu", "leaky_relu"],
    "learning_rate": 0.001,
    "random_seed": 57,
    "max_epochs": 1000,
    "batch_size": 64,
    "patience": 200,
    "dropout_rate": 0.2
}
```

I looked for validation losses that were stable over epochs to determine that overfitting was minimized (Supplemental Figure C, D).

Ensemble Model

Given the performance of the ensemble model in (Willard et al. 2021), which decreased error by nearly half a degree in their efforts, I'm going to implement that strategy for both the LOO and timeseries models.

Results

Hindcasting application

Because the LOO models do not perform consistently across lakes, I employed an ensemble method with the 6 models for the hindcasting (test) application. The predicted temperature will be the mean value from all 6 models. The hindcasting test set performed similarly with the LOO ensemble and the timeseries ensemble. MAE was 1.79 and 1.79 deg C, MSE was 4.84 and 4.91 deg C, RMSE was 2.2 and 2.21 deg C, and MAPE was 20.71 and 20.25 percent respectively. Both models consistently underpredicted temperature in Horsetooth reservoir, particularly in the Fall of 2022 (Figure 1, 2). Granby, an outlier during model development looked more accurate than Willow Creek when predicted using either ensemble model.

Discussion

Model Performance

The training-validation splits for LOO did not perform consistently across lakes. The datasets with Granby Reservoir (LOO #4) and Willow Creek Reservoir (LOO #6) performed more poorly than the other training-validation splits. I don't think this is because the network 'learned' the individual lakes and was therefore not generalizable, as much as the fact that these two waterbodies are both outliers within this dataset. Granby Reservoir is the largest (by surface area) reservoir in Colorado and therefore has physical properties and patterns that can not be mimicked in smaller waterbodies. Willow Creek is a very small reservoir that is not always used for storage, meaning sometimes it hold little water - I believe this results in water temperatures that mimic streams, rather than waterbodies (where the water mass can retain heat). This result is more of a preamble of caution when applying to waterbodies that are outside of this dataset.

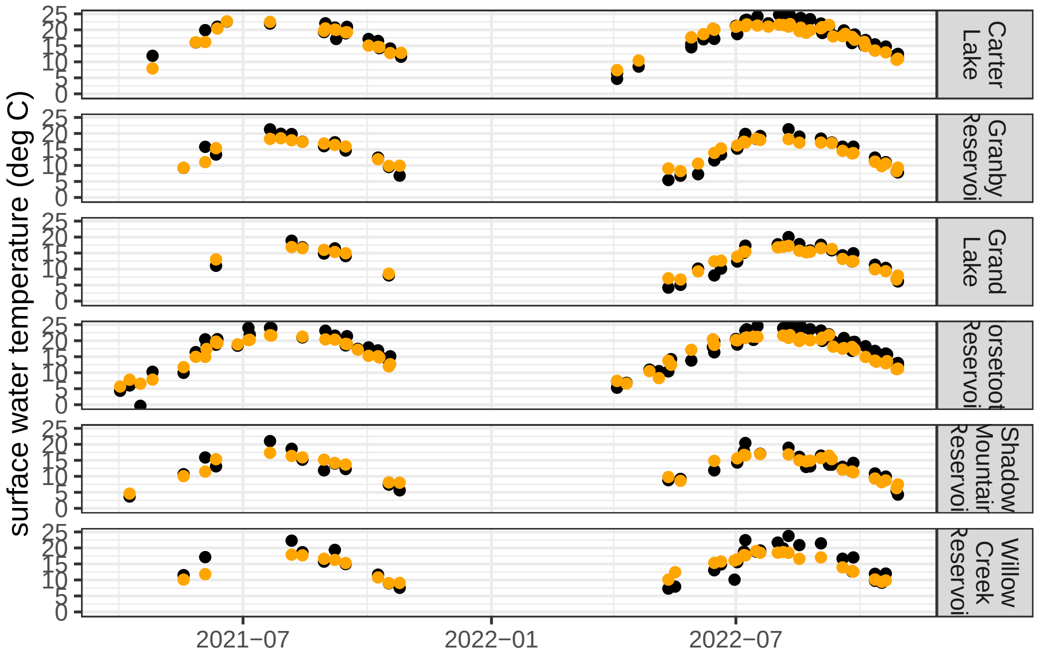


Figure 1: Predicted surface water temperature during the hindcast period of 2021 and 2022 using the leave-one-out ensemble model. Black dots are actual values, gold dots are predicted.

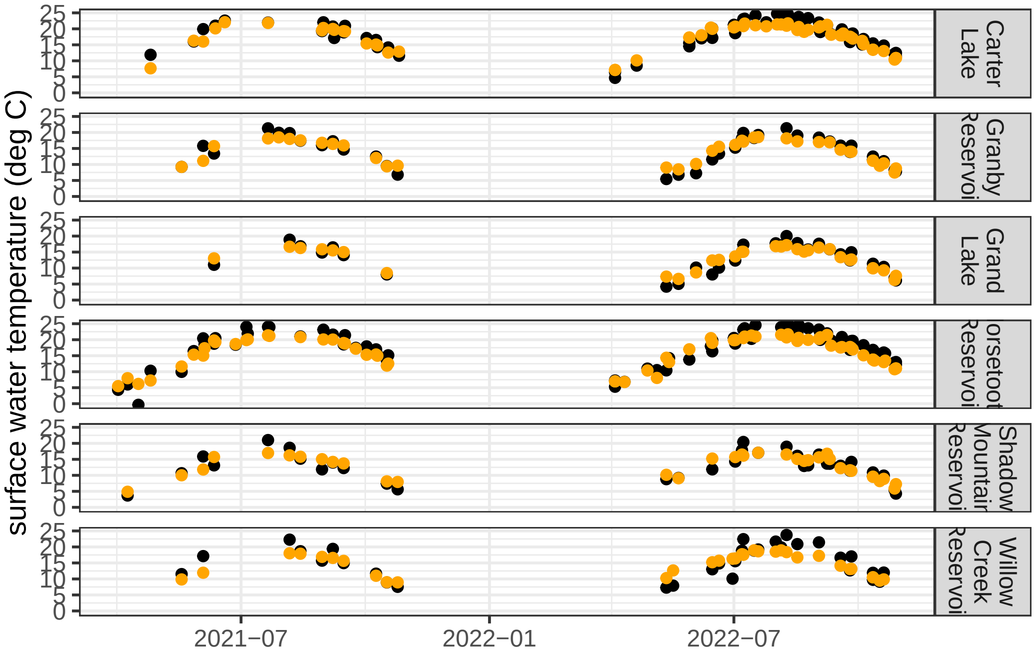


Figure 2: Predicted surface water temperature during the hindcast period of 2021 and 2022 using the timeseries ensemble model. Black dots are actual values, gold dots are predicted.

When the ensemble models were applied to the hindcast dataset, It was striking how poorly the estimates for Horsetooth Reservoir were, where the model consistently underpredicted water temperature. 2020 forward was a particularly dry time in the climatological past - most of Colorado experience severe drought and many reservoirs were at historically low levels until 2023. The ensemble model performed reasonably well at Granby Reservoir, but quite poorly at Willow Creek, the two outliers from model development.

It is possible that some of the predicted-observed outliers are more due to errors in the Landsat-derived dataset than the model. It would be useful to see if the error metrics are consistent between the manual sampling record and the Landsat record. You can clearly see a value at Horsetooth Reservoir early in 2021 (Figure 1, 2) that I would argue is physically impossible, considering the neighboring values. While the MSE for the Landsat-derived data is around 1.5 degrees, the members of the tail could be adversely affecting model development and performance.

Future steps

I'd like to add in reservoir height to this model to try to offset the inter-annual and -seasonal variability that may be driving the differences between performance at these reservoirs. Additionally, I think it would be appropriate to bring in the other Northern Reservoirs where we have remotely-sensed water temperature only as a way to build out the model, but only after we've constrained the Landsat-derived temperature error a bit more.

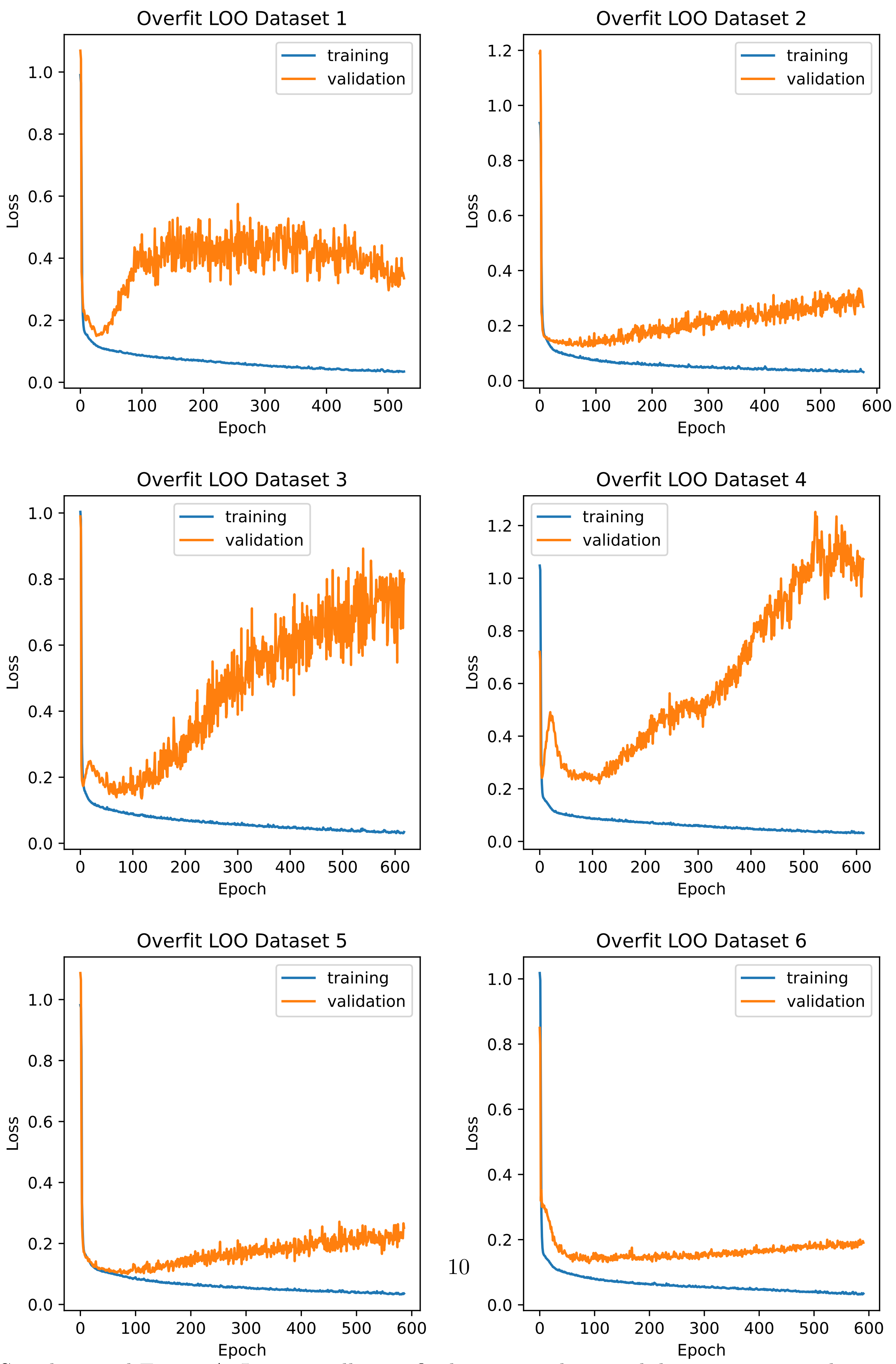
I do think this dataset and scientific question would be a good candidate for LSTM or other autoregressive NN techniques. While the ensemble model performs reasonably well, it does not outperform the models presented in the literature.

References

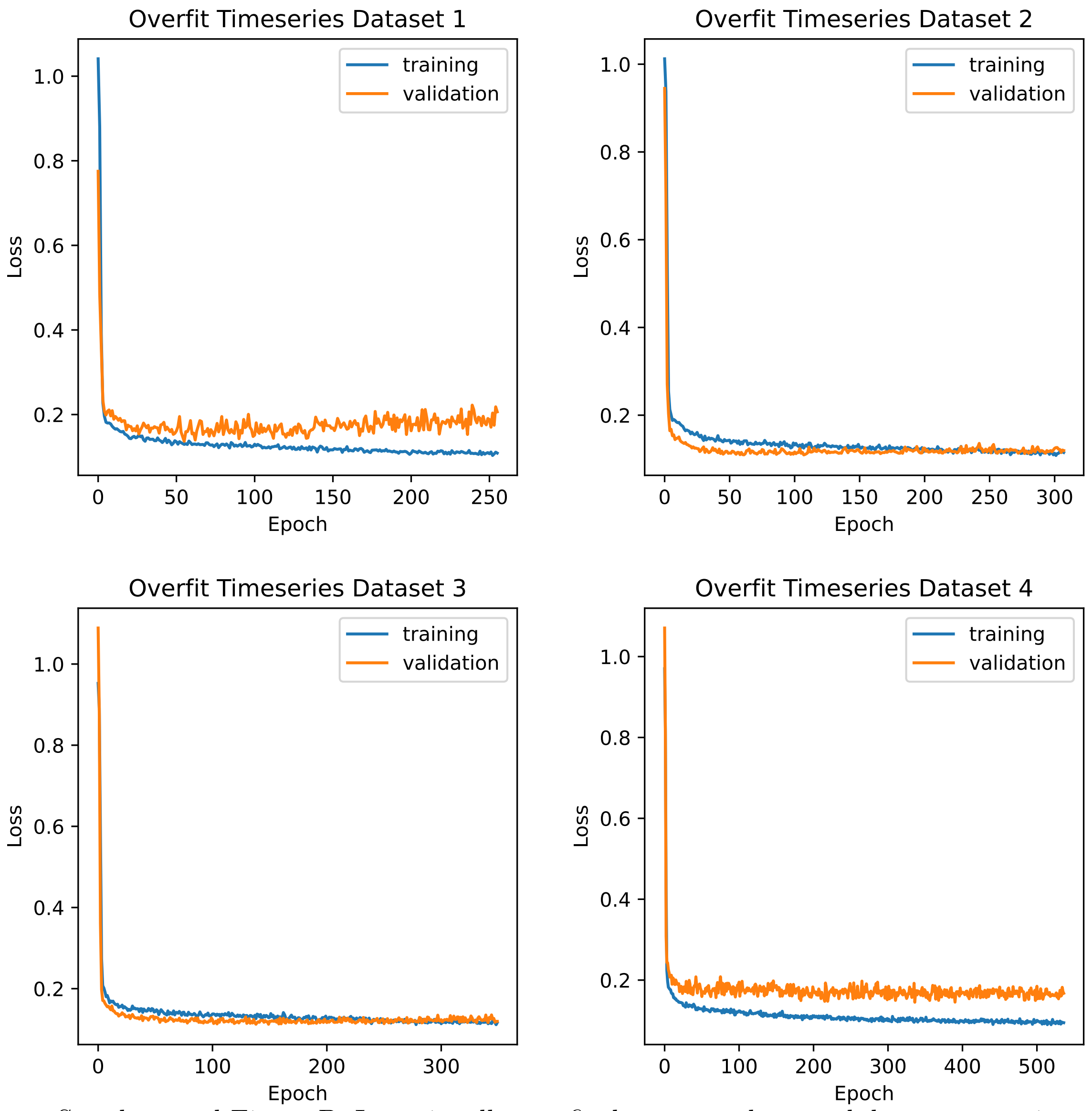
Read, Jordan S., Xiaowei Jia, Jared Willard, Alison P. Appling, Jacob A. Zwart, Samantha K. Oliver, Anuj Karpatne, et al. 2019. "Process-Guided Deep Learning Predictions of Lake Water Temperature." *Water Resources Research* 55 (11): 9173–90. https://doi.org/10.1029/2019WR024922.

Willard, Jared D., Jordan S. Read, Alison P. Appling, Samantha K. Oliver, Xiaowei Jia, and Vipin Kumar. 2021. "Predicting Water Temperature Dynamics of Unmonitored Lakes with Meta-Transfer Learning." *Water Resources Research* 57 (7): e2021WR029579. https://doi.org/https://doi.org/10.1029/2021WR029579.

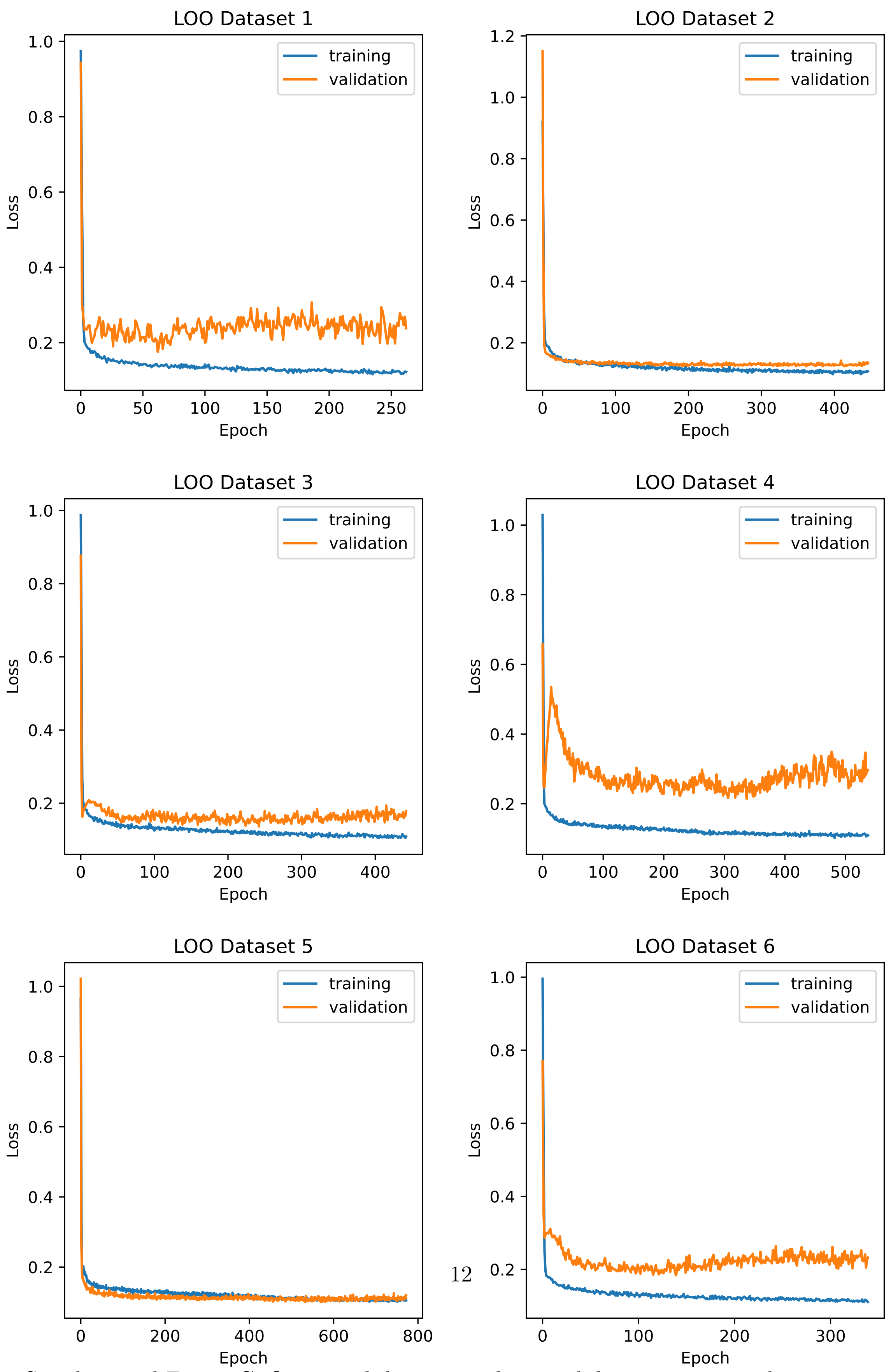
Supporting Figures



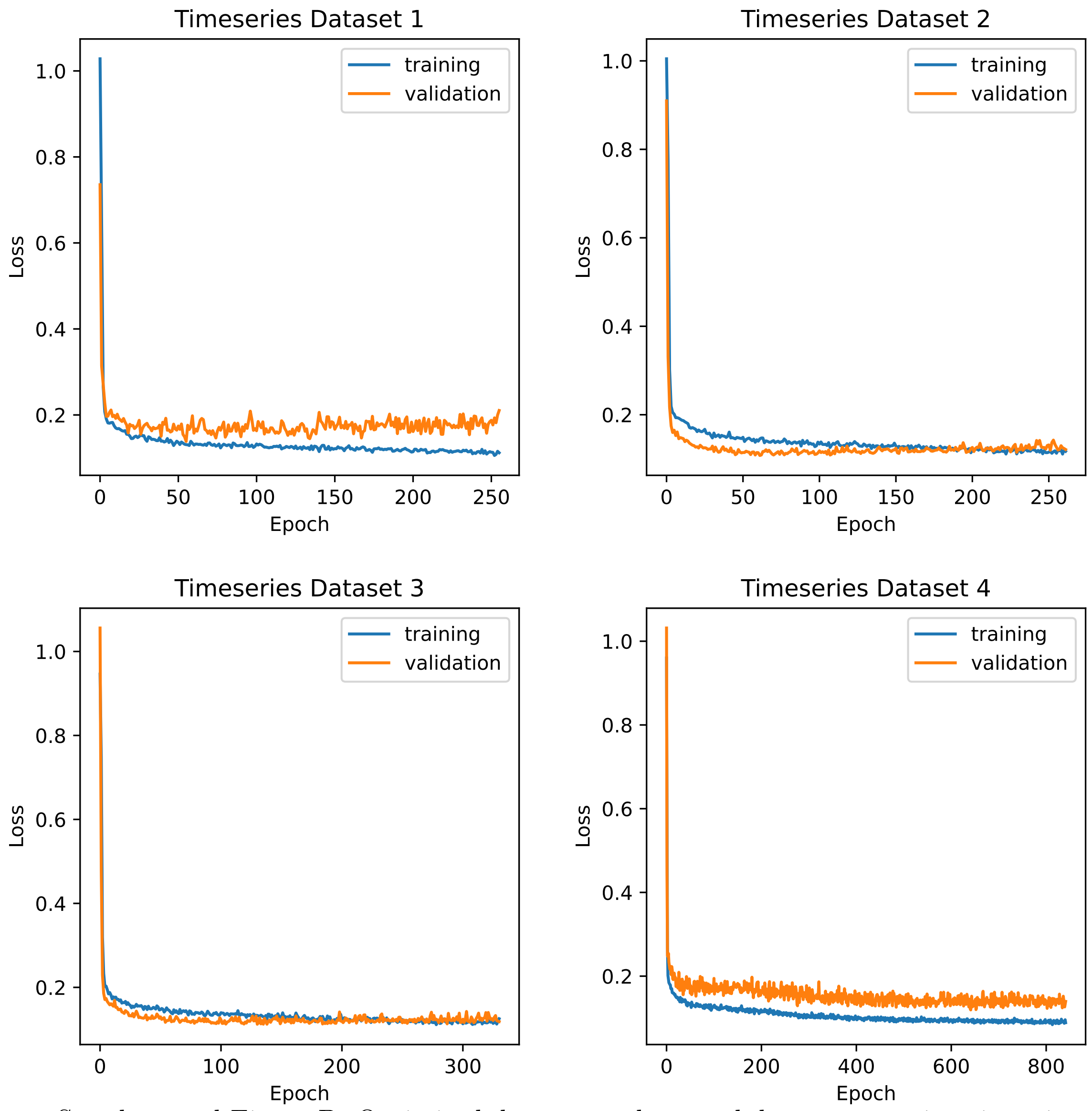
Supplemental Figure A. Intentionally overfit dense neural network loss curves using leave one out technique.



Supplemental Figure B. Intentionally overfit dense neural network loss curves using timeseries technique.



Supplemental Figure C. Optimized dense neural network loss curves using leave one out technique.



Supplemental Figure D. Optimized dense neural network loss curves using timseries technique.