Monitoring the Operational Performance of an Anaerobic Wastewater Treatment Plant with Deep Learning Methods

Jose Bolorinos jbolorin@stanford.edu
Chris Pearce cpearce@stanford.edu
Stanford University

Abstract

This paper investigates applying deep learning frameworks to forecast the state of an anaerobic wastewater treatment plant. Various convolutional and recurrent model designs are tested, and model structures capable of making univariate and multivariate predictions are developed. The best performing model designs are shown to outperform state of the art autoregressive and Bayesian forecasting techniques.

1. Introduction:

This paper investigates the use of deep learning to predict operational parameters at an anaerobic membrane-based wastewater treatment facility. Anaerobic wastewater treatment systems (WTSs) are emerging as low-energy alternatives to conventional methods that can harvest energy from wastewater as methane and could make net-energy-positive wastewater treatment possible for the first time[1–4]. However, they require careful operation as anaerobic organisms are quite pH sensitive and generation of methane results from a delicate symbiosis of multiple microbial communities.

Availability of low cost sensors and computing resources means that the pH and biogas production of anaerobic WTFs could be monitored in real time to anticipate and prevent operational issues. Recent advances in convolutional and recurrent deep learning networks have led to breakthroughs in the use of artificial intelligence for image recognition and natural language processing[5]. A growing body of work is studying the use of new deep learning methods for forecasting[6–10] but their application to wastewater treatment is a relatively novel area of investigation.
The purpose of this project is to develop a model that can accurately predict hourly pH in an anaerobic reactor within a time horizon of 24 hours. We also examine the predictive performance two-channel models (i.e. models that take input data from two co-dependent sensors) with the ultimate goal of developing a deep learning framework to validate sensor readings and alert operators of the need to recalibrate faulty sensors.

2. Data:

Data for this project are from the Bill & Cloy Codiga Resource Recovery Center (CR2C), a pilot-scale anaerobic, membrane-based wastewater treatment facility on the Stanford campus. These data include high-resolution flow-rate, pH, conductivity, pressure, and differential pressure measurements sensors located at all critical points in the facility's wastewater treatment process, including its anaerobic reactors. All data are structured and combined through time-stamp and "treatment stage" tags. Two pH sensors from CR2C were selected to train and test our models: The first sensor is in a tank that holds influent wastewater from the Serra St. sewer; the second sensor is in one of the facility's anaerobic reactors. Our two-channel models use data from temperature and conductivity sensors that we believe may be co-dependent pH and are located at the same points in the facility.

Although data are available at ~1m intervals, we aggregated them to hourly data to ease computational load. In practice, predicting pH in anaerobic reactors is important on hourly to daily time scales, as this is the timeframe over which parameters tend to change, and plant managers tend to make operational decisions. In total, 12,768 hours of the treatment plant's operational data were available. We used 10,000 hours to develop our models, segmented into a training set for fitting models (8,000 observations), a validation set for hyperparameter tuning (1,000 observations), and a test set for a final evaluation of predictive performance (1,000 observations).

3. Methods:

3.1. Time Series Benchmarking
The performance of all models we developed was compared to two standard time series methods. The first method was a standard auto-regressive integrated moving average (ARIMA) model that can be fit to a single time series. We examined the autocorrelation matrix in our pH sensors and concluded that the data show autocorrelation on the order of roughly 24 hours. We tested models with a moving average order of 1-5 hours and a differencing order of 0, 1, and 2 obtaining the best performance on the training set with an

autocorrelation order of 24, a moving average order of 1 and a differencing order of 0.

The second time series approach we used to benchmark our predictive performance was the Facebook "Prophet" model proposed recently[11]. Facebook's prophet model combines time series and Bayesian methods to capture autocorrelation, seasonality, changepoints and other irregularities in a time series and make predictions. Since each model fit pertains to the input (training) series, it cannot be used to test more than one out of sample prediction set. We thus randomly sampled time series of length N + 24 where N observations were used to train each prophet model, which was then used to forecast the next 24 hours in the series. We tested training series (N) of length 24, 48, 72, 168 (1 week), 760 (1 month) and 8760 (1 year) and found that models fit to 1 month of data achieved the best performance.

3.2. Model Architectures

Two classes of neural networks were tested, convolutional neural networks (CNNs) and recurrent neural networks (RNNs), and within these a range of architectures were trialled. Initial model development focused on building a univariate model, taking one channel as an input and forecasting that same channel as an output. Later the RNN models were adjusted to accept a multivariate input and produce a multivariate output in a test of the model's ability to undertake a joint learning task.

3.2.1 CNN Architectures

Two separate CNN architectures were tested.

1. A 1D CNN design inspired by Rajgupar et al[12] who use a deep residual network CNN to detect cardiac arrhythmias and studies like Yang et al[13], who use deep 1D CNN's for human activity recognition with inertial sensors. We considered 1-12 layers of stacked modules consisting of convolution, batch normalization, ReLU, dropout, and max-pool shortcut connections. Our search of the hyperparameter space returned MSEs that were more than 5 times higher than the RNN model, so this approach was abandoned.
2. A wide network based on the Inception network for computer vision [14], with a convolutional layer comprising 32 $p \times 1$, $p \times 3$ and $p \times 5$ where p represented the number of 24 hour periods of data inputted into the model. Two smoothing layers were added to the model of size $32 \times 3 \times 1$ and $16 \times 3 \times 1$ to ensure that the model outputs were locally consistent. This model was able to achieve an MSE about 40% higher than the FB Prophet model, so while promising, it was set aside in favour of better performing RNN models.

3.2.2 RNN Architectures

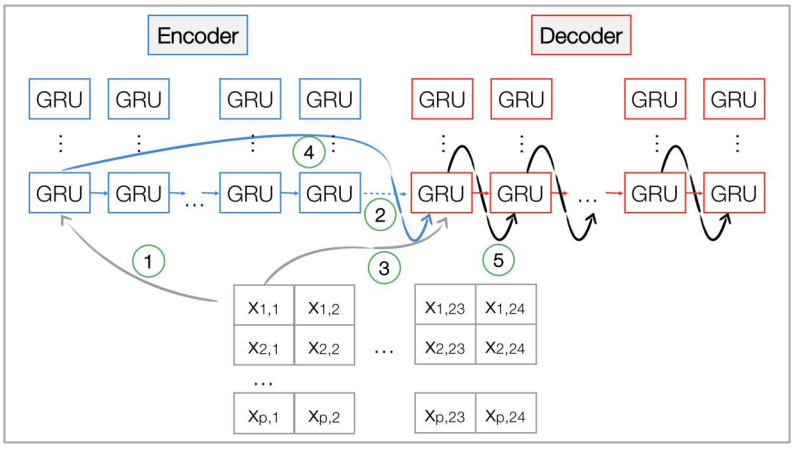
Initial trials with RNN architectures proved to be more successful than the CNN architectures, and as a result a wider variety of RNN models were investigated.

Architectures trialled included

1. RNN encoder with outputs feeding into an affine prediction layer
2. Bi-directional RNN encoder with outputs fed into an affine prediction layer
3. An RNN encoder-decoder pair; the RNN encoder builds a hidden state which is passed as an initialiser to an RNN decoder, and decoder outputs are then fed into an affine layer.
4. An RNN encoder-decoder pair, with time aligned encoder outputs fed as inputs to the respective decoder steps
5. An RNN encoder-decoder pair with an attention mechanism allowing the decoder to attend to the encoder's hidden states
6. An RNN encoder-decoder pair with time aligned encoder outputs and teacher forcing[15], where the output of each decoder step is fed as an input into the next decoder step (see **Figure 1**).

Architectures (1) and (2) were based on classifier model designs, while (3) and (5) were based on neural

Figure 1: Schematic of RNN Architecture 6 - Selected for Detailed Training



Key

1. Feed input timeseries data into encoder
2. Take encoder hidden state and feed into decoder as initial state
3. Feed input timeseries data into decoder
4. Feed encoder output into decoder
5. "Teacher Forcing" - feed output of previous decoder step into next decoder step

machine translation model designs[16], as was the teacher forcing component of model (6). In each case, the models were adapted for use with continuous data instead of categorical datasets. The encoder output feeding approach used in (4) and (6) was an adaptation proposed by the authors, supported by the fact of a strong 24 hour periodic trend in the dataset.

The model classes were subsequently adapted to accept both single and multi-channel inputs, and models (3) – (6) were adapted to incorporate multi-layer RNN models. In early testing, GRU, LSTM and basic RNN memory units were trialled. GRU and LSTM models performed similarly, and both were significantly better than the basic RNN. The GRU memory unit trained significantly faster than the LSTM however, so it was used in all model classes.

3.2.3 RNN Data Inputs

Two different formats were trialled to feed data into the model. Flat form data was structured as $x_i \epsilon \{p \times 12{,}1\}$ and allowed for arbitrary length inputs to be run through the model. Deep form data was structured as $x_i \epsilon \{24, p\}$, where the input data was a vector 24 observations wide and p periods deep. This reflected the fact that there was clear 24 hour periodicity in the data, but forced a decision to be made at training time as to the value of p. For multichannel inputs the data was structured as $x_i \epsilon \{p \times 12{,}1, n\}$ and $x_i \epsilon \{24, p, n\}$ respectively, where n is the number of channels. Deep form data structures performed better in preliminary model development, and so were adopted for the final model design.

3.3. Loss Function and Model Evaluation

The model uses an MSE loss function combined with L2 regularisation. All models were evaluated on the basis of MSE.

3.4. Selected Architecture Development

During the model design work, architecture (6) was found to be able to consistently outperform the benchmarks based on CPU training. **Figure 1** provides an illustration of this model. This was selected for detailed testing of hyperparameter settings. Following this testing the preferred hyperparameter settings were transferred to other models in the RNN category for final testing and evaluation.

3.5. Hyperparameter Search

A detailed search of hyperparameter settings was performed on architecture (6). Our hyperparameter search space consisted of the following 5 dimensions, each tested for both single-channel models (i.e. time series inputs from just one sensor) and two-channel (i.e. time series inputs from 2 sensors):

1. The number of stacked encoder and decoder layers shown in **Figure 1** (1-3 layers were tested)

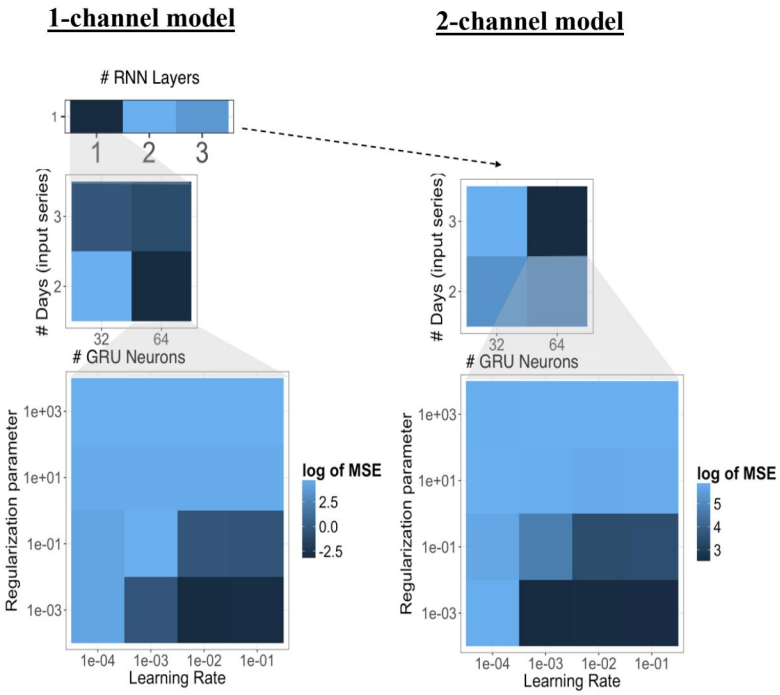


Figure 2: *Hyperparameter search for models with 1 sensor (LEFT) and 2 sensors (RIGHT)*

2. The number of neurons in the GRU's comprising the encoder and decoder layers (16, 32, 64, and 128 neurons were tested)
3. The number of days of previous hourly pH readings (i.e. the dimensionality of the 24-h time series input into the model; 1,2,3, and 4 days were tested)
4. The size of the L2 regularization parameter applied to the model's mean squared error loss function (values between 1E-7 and 1E+3 were tested)
5. The model's learning rate (values between 1E-4 and 1E-1 were tested).

All models were run on GPU cores on Stanford's Sherlock supercomputing cluster, using Keras "CuDNNGRU" GRU class to optimize computational performance. Each model's error was computed on our dev set of 1,000 samples. **Figure 2** displays the results of our hyperparameter search. Our initial search found that the 2- and 3-layer RNN's showed considerably worse performance than 1-layer RNNs, so we restricted further search accordingly. Within the subset of 1-layer RNNs, we found best performance with 64-neuron GRUs taking a 2-day input signal, although 3-day input signals were better for the 2-channel models. Within the subset of these model architectures, the best values for the learning rate and regularization parameter were 1E-2 and 1E-3, respectively, although visual inspection of **Figure 2**

suggests that we could explore this parameter space more thoroughly.

4. Results:

4.1. Predictive Performance:

Figure 3 plots the root mean squared error (RMSE) of the best 1-channel RNN computed on the test set, for each forecast horizon (1-24 hours). As shown in the figure, our RNN achieves a predictive performance that is on-par with the Facebook prophet model (and somewhat more stable), and far better than the ARIMA model. Also notable is the fact that pH is much easier to predict in the reactor (AT305) since it is mediated by the anaerobic microorganisms. Influent pH (AT203) is highly variable and dependent on exogenous shocks to Stanford's Escondido Village sewershed that no time series model can capture effectively.

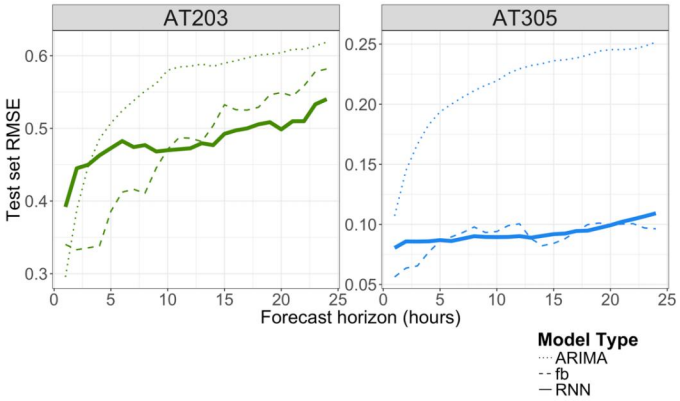


Figure 3: *Predictive performance of 1-channel models on the test set. Note: AT203 refers to pH of influent wastewater, AT305 refers to pH in reactor.*

4.2. Co-dependent sensors:

Figure 4 shows the predictive performance of the model fit to input series from 2 different sensors (performance is measured relative to the same two pH sensors shown in **Figure 3**). The figure confirms the notion that multivariate forecasts tend to be less accurate than univariate forecasts. Indeed, when we trained a model on the reactor's pH sensor (AT305) alongside the reactor's influent flow, predictive performance declined considerably as inflow into the reactor is operationally determined and only adds noise to the model. Nonetheless, the results from the 2-channel model can still be used to assess which of the other types of sensors (temperature, conductivity, or nearby pH sensors) might provide more information about a pH sensor whose measurements we want to validate. For the wastewater influent (measured by AT203) conductivity seems to be most closely related to pH (related to exogenous changes in the quality of incoming wastewater), whereas in the reactor, temperature and the pH of the influent (which flows into the reactor) are better co-predictors.

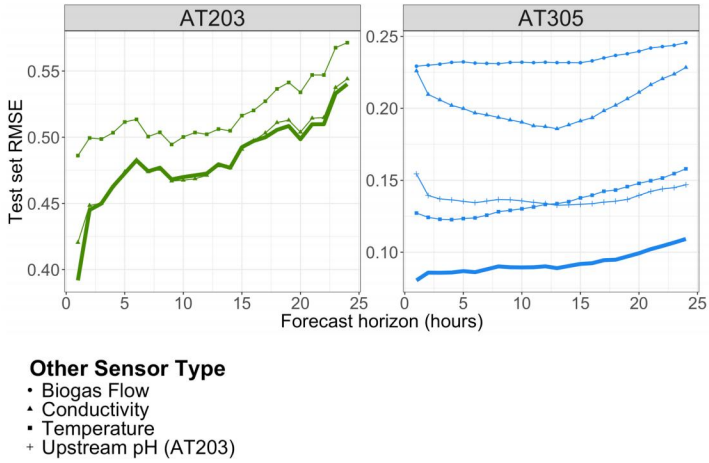


Figure 4: *Predictive performance of 2-channel models on the test set of the sensors shown. Note: AT203 refers to pH of influent wastewater, AT305 refers to pH in reactor. The solid lines correspond to the results of the 1-channel models displayed in* ***Figure 3***

4.3. Final Test Accuracy:

Having established the general range of preferred parameters, the final stage of development involved rerunning the full range of RNN architectures within the preferred hyperparameter range.

Table 1: *Performance statistics for all RNN models following hyperparameter tuning*

Model	Train MSE	Test MSE
(1) RNN → Affine	0.0172	0.0115
(2) Bi-RNN → Affine	0.0033	0.0030
(3) Seq → Seq	0.0144	0.0094
(4) Seq → Seq w. Encoder Feeding	0.0678	0.0621
(5) Seq → Seq w. Attention	0.0448	0.0290
(6a) Seq → Seq w. Teacher Forcing		0.0999
(6a) Seq → Seq w. Teacher Forcing - Multichannel		0.2209

A range of the other architectures were found to perform significantly better following tuning, with the bidirectional RNN encoder with affine decoder performing an entire order of magnitude better than

the training benchmark. Some of the more complicated architectures did not perform as well, prompting us to investigate some of the causes.

For the attention-based model, we extracted the attention weights at various points through training to see what it was doing. Early on in training, the model did use the attention mechanism to access data from other time periods, but later the model was found to have started assigning equal weights across all time periods, effectively switching the attention mechanism off. We speculate that this is because the deep structure of the data input already provides the most important historic information to the model, and that the RNN structure further allows for data to be passed across time periods. Therefore the model may have found the attention mechanism to be redundant, with regularisation then resulting in the parameters being minimised.

Similarly, many of the additional components added to the more complex architectures trialled (i.e. models (4) to (6)) effectively provided the model with additional ways to access various aspects of the historic data from the model. For this application however, it appears that a well-tuned bi-directional RNN architecture is ultimately able to pass information forwards and backwards sufficiently to be able to deliver state of the art performance for this forecasting exercise.

5. Conclusions:

We have shown that a RNNs can be used to predict the pH inside a reactor more accurately than state-of-the-art time series forecasting methods. Our model can predict pH in a reactor 24-hours ahead with a RMSE of +/- 0.11, sufficiently accurate to anticipate endogenous pH shifts a day in advance and take it offline to prevent damage to its microorganisms.

Given the high performance achieved with some of the simpler architectures, there may be an opportunity to revert and build a multi-channel RNN model using one of these architectures. These modelling approaches may be further adapted to address additional prediction challenges that the plant faces. One application could be to use deep learning to predict "drift" in pH sensors that are in need of re-calibration.

Our next step in this work is to use these 1- and 2-channel models to train a network to recognize such sensor "drift" with validated laboratory pH measurements. We hope to demonstrate the efficient deep learning models can use increasingly available sensor data to make substantial improvements to wastewater treatment process efficiency.

6. Contributions:

Both authors contributed equally to this research paper. Bolorinos and Pearce collaborated to determine the project direction, prepare the final presentation and draft the project documents.

Jose Bolorinos: Collected, cleaned and prepped the sensor data for the project, developed the ResNet CNN approach tested early in the project, performed the benchmarking analyses using the facebook prophet and the ARIMA time series models, and performed the hyperparameter search on the final RNN model architecture.

Chris Pearce: Developed the inception covnet and the RNN architectures, developed the multichannel model structures, conducted final model training on the preferred hyperparameter sets and prepared the code repository.

7. Code:

The code for this project is available on the project Github repository;

github.com/rktby/cs230

Notebooks for running each of the models are available in the Production Notebooks folder;

github.com/rktby/cs230/tree/master/prodn_notebooks

References

1. McCarty, P. L., Bae, J. & Kim, J. Domestic wastewater treatment as a net energy producer - can this be achieved? *Environ. Sci. Technol.* **45,** 7100–6 (2011).
2. Smith, A. L. *et al.* Navigating wastewater energy recovery strategies: A life cycle comparison of anaerobic membrane bioreactor and conventional treatment systems with anaerobic digestion. *Environ. Sci. Technol.* **48,** 5972–5981 (2014).
3. Shin, C. & Bae, J. Current status of the pilot-scale anaerobic membrane bioreactor treatments of domestic wastewaters: A critical review. *Bioresour. Technol.* 0–1 (2017). doi:10.1016/j.biortech.2017.09.002
4. McCarty, P. L., Bae, J. & Kim, J. Domestic wastewater treatment as a net energy producer-can this be achieved? *Environ. Sci. Technol.* **45,** 7100–7106 (2011).
5. Lecun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521,** 436–444 (2015).
6. Gensler, A., Henze, J., Sick, B. & Raabe, N. Deep Learning for solar power forecasting - An approach using AutoEncoder and LSTM Neural Networks. *2016 IEEE Int. Conf. Syst. Man, Cybern. SMC 2016 - Conf. Proc.* 2858–2865 (2017). doi:10.1109/SMC.2016.7844673
7. Kong, W., Dong, Z. Y., Hill, D. J., Luo, F. & Xu, Y. Short-Term Residential Load Forecasting based on Resident Behaviour Learning. *IEEE Trans. Power Syst.* **33,** 1–1 (2017).
8. Hsu, D. MULTI-PERIOD TIME SERIES MODELING WITH SPARSITY VIA BAYESIAN VARIATIONAL INFERENCE. *arXiv:1707.00666v3* (2018).
9. Fischer, T. & Krauss, C. Deep learning with long short-term memory networks for financial market predictions. *Eur. J. Oper. Res.* **270,** 654–669 (2018).
10. Borovykh, A. & Bohte, S. Conditional time series forecasting with convolutional neural networks arXiv : 1703 . 04691v5 [stat . ML] 17 Sep 2018. 1–22 (2018).
11. Taylor, S. J. *et al.* Forecasting at scale. *10.7287/peerj.preprints.3190v2* 1–25 doi:10.7287/peerj.preprints.3190v2
12. Rajpurkar, P., Hannun, A. Y., Haghpanahi, M., Bourn, C. & Ng, A. Y. Cardiologist-Level Arrhythmia Detection with Convolutional Neural Networks. *http://arxiv.org/abs/1707.01836* (2017). doi:1707.01836
13. Yang, J. B., Nguyen, M. N., San, P. P., Li, X. L. & Krishnaswamy, S. Deep convolutional neural networks on multichannel time series for human activity recognition. *IJCAI Int. Jt. Conf. Artif. Intell.* **2015–Janua,** 3995–4001 (2015).
14. Szegedy, C. *et al.* Going Deeper with Convolutions. *arXiv.org* **1409,** (2014).
15. Williams, R. J. & Zipser, D. A learning algorithm for continually running fully recurrent neural networks. *Neural Comput.* **1,** 270–280 (1989).
16. Vaswani, A. *et al.* Attention Is All You Need. *http://arxiv.org/abs/1706.03762* (2017). doi:10.1017/S0952523813000308