Data-Driven Prediction of Disinfection By-Product Indicators in Water Distribution Systems

November 27, 2025

- **Topic:** Data-Driven Prediction of Disinfection By-Product Indicators in Water Distribution Systems
- **Name:** Zhou Dafu
- **Number:** A0331761J
- **Supervisor:** Prof. Hu Jiangyong
- **Mentor:** Mr. Sun Yuanpeng

1 Introduction

1.1 Background

Disinfection is a critical step in municipal water treatment, designed to eliminate pathogenic microorganisms and ensure public health. However, chemical disinfectants, most commonly chlorine, react with naturally occurring organic matter (NOM) and other precursors in the water to form a wide array of unintended compounds known as disinfection by-products (DBPs) [1]. Many DBPs are of health concern due to their potential carcinogenic and mutagenic properties [2]. Consequently, water utilities face the challenge of balancing microbial safety with the minimization of DBP formation.

Predicting DBP concentrations within complex water distribution systems (WDS) is a significant challenge due to dynamic hydraulic conditions and complex chemical kinetics. Traditional monitoring relies on infrequent grab sampling and laboratory analysis, which is costly and provides low temporal resolution. This research aims to leverage high-frequency sensor data and supervised machine learning techniques to develop predictive models for DBP indicators, such as Total Residual Chlorine (TRC), Total Organic Carbon (TOC), and pH. The successful development of these models would enable proactive operational control to mitigate DBP formation and improve water quality management. This report outlines the progress made in data preprocessing, model selection, and experimental design for this purpose.

1.2 DBP Formation Chemistry

The formation of DBPs is initiated by the reaction of a primary disinfectant (e.g., chlorine) with organic and inorganic precursors present in the water. In this project, the water treatment process involves the addition of chlorine, lime, and ammonium salts. The primary disinfectant, hypochlorous acid (HOCl), is formed, which then acts as an oxidizing agent. These disinfectants react with NOM to produce halogenated DBPs, while reactions with nitrogen-containing precursors can lead to the formation of nitrogenous DBPs (N-DBPs), which are often more cytotoxic and genotoxic than their regulated counterparts [3]. A simplified representation of the primary reactions is:

$$\begin{aligned} \mathrm{Cl_2 + H_2O} &\rightleftharpoons \mathrm{HOCl + H^+ + Cl^-} \\ \mathrm{HOCl + NOM} &\rightarrow \text{Halogenated DBPs (e.g., THMs, HAAs)} \\ \mathrm{HOCl + NH_3 + NOM} &\rightarrow \text{Nitrogenous DBPs (e.g., HANs)} \end{aligned}$$

The specific types and concentrations of DBPs formed are highly dependent on factors such as precursor type and concentration (TOC, DOC), disinfectant dose (measured by TRC), pH, temperature, and reaction time.

2 Methodology

2.1 System Description and Data Acquisition

The study focuses on a section of a water distribution system comprising five main stages:

1. **Finished Water Tank (FWT):** Post-treatment water storage. Not included in the modeling.
2. **Mixing Tank (DT):** Where disinfection chemicals are introduced and mixed.
3. **Reservoir Tank (RT):** Provides contact time for disinfection.
4. **Pipeline Stage 1 (PPL1):** First section of the distribution pipeline.
5. **Pipeline Stage 2 (PPL2):** Second section of the distribution pipeline.

Water quality data is collected via multi-parameter sensors at different stages. The parameters include Total Residual Chlorine (TRC), pH, Conductivity, Fluorescent Dissolved Organic Matter (fDOM), Dissolved Oxygen (DO), Total Organic Carbon (TOC), and Dissolved Organic Carbon (DOC). Data is recorded at a 5-minute interval. The availability of sensor measurements at each stage is summarized in Table 1.

Table 1: Availability of Sensor Measurements at Each Stage

Parameter	DT	RT	PPL1	PPL2
TRC	✓	✓	✓	✓
pH	✓	✓	✓	✓
Conductivity	✓		✓	✓
fDOM (QSU)		✓	✓	✓
DO		✓	✓	✓
TOC		✓	✓	✓
DOC		✓	✓	✓

2.2 Data Preprocessing

2.2.1 Imputation of Missing Values

A unique challenge in the dataset arises from the measurement setup for PPL1 and PPL2, where a single sensor alternates between the two locations every 30 minutes. This results in blocks of six consecutive missing values for each pipeline stage, alternating with six valid measurements. To create a continuous time series for model training, these missing values must be imputed. Given the temporal correlation in water quality data, advanced imputation techniques such as interpolation, or model-based methods will be employed to fill these gaps realistically. Figure 1 illustrates a conceptual example of this imputation.

Figure 1: Conceptual Example of Missing Value Imputation (Placeholder)

A plot showing a time series with gaps and the imputed values will be placed here.

2.2.2 Time Alignment

As a parcel of water flows through the WDS, there is a travel time (hydraulic delay) between consecutive stages. To model the chemical transformations accurately, the input data (from DT and RT) must be time-aligned with the output data (from PPL1 and PPL2) to ensure the model learns the relationship between the same parcel of water at different points in time. This involves calculating the time lag between stages and shifting the data series accordingly before creating input-output pairs for the models.

3 Model Selection and Training

3.1 Model Architectures

To capture the complex, non-linear dynamics of DBP formation, several neural network architectures are considered for this regression task:

- **Multilayer Perceptron (MLP):** A foundational feedforward neural network that can model complex non-linear relationships. It consists of an input layer, one or more hidden layers, and an output layer. Each layer contains neurons that are fully connected to the

neurons in the subsequent layer. MLPs are powerful function approximators but do not have inherent memory, treating each input independently.

Figure 2: Structure of a Multilayer Perceptron (Placeholder)

A diagram illustrating the layered structure of an MLP will be placed here.

- **Recurrent Neural Network (RNN):** Designed to handle sequential data by maintaining a hidden state that captures past information. The output at a given time step is a function of the current input and the hidden state from the previous time step. This recurrent connection allows RNNs to model temporal dependencies, but they can struggle with long-term dependencies due to the vanishing gradient problem.

Figure 3: Structure of a Recurrent Neural Network (Placeholder)

A diagram illustrating the recurrent loop in an RNN will be placed here.

- **Long Short-Term Memory (LSTM):** A specialized RNN architecture that uses gating mechanisms (input, forget, and output gates) to overcome the vanishing gradient problem and effectively learn long-term dependencies [4]. The gates control the flow of information, allowing the network to selectively remember or forget information over long sequences, making it well-suited for time-series forecasting.

Figure 4: Structure of a Long Short-Term Memory Cell (Placeholder)

A diagram illustrating the gates within an LSTM cell will be placed here.

- **Gated Recurrent Unit (GRU):** A simplified version of the LSTM with fewer parameters, which often performs comparably on many tasks. It combines the input and forget gates into a single "update gate" and merges the cell state and hidden state. This makes the GRU more computationally efficient than the LSTM while still retaining its ability to capture long-term dependencies.

Figure 5: Structure of a Gated Recurrent Unit Cell (Placeholder)

A diagram illustrating the gates within a GRU cell will be placed here.

3.2 Model Training

The models will be trained using standard deep learning principles. The process involves feeding the model with input-output pairs and optimizing its internal weights to minimize a loss function, which quantifies the error between the model's predictions and the true values.

Figure 6: Model Training and Evaluation Workflow (Placeholder)

A flowchart illustrating the data splitting, training, validation, and testing process will be placed here.

- **Loss Function:** Mean Squared Error (MSE) is chosen as the loss function, suitable for regression tasks.

- **Optimization:** The Adam optimizer [5], an adaptive learning rate optimization algorithm, is used to update the model weights via backpropagation.
- **Dataset Splitting:** The data is split into training, validation, and test sets. The model learns from the training set, its hyperparameters are tuned based on performance on the validation set, and its final performance is evaluated on the unseen test set.
- **Regularization:** To prevent overfitting, techniques like Dropout [6] are used. Additionally, an early stopping mechanism is implemented: training is halted if the validation loss does not improve for a specified number of epochs, and the model with the best validation performance is saved.

3.3 Hyperparameter Optimization

The performance of neural networks is highly sensitive to the choice of hyperparameters. Key hyperparameters include learning rate, batch size, dropout rate, number of layers, number of units per layer, and the history length for sequential models. While methods like grid search and random search [7] exist, this project employs **Bayesian Optimization** [8]. Bayesian optimization builds a probabilistic model of the objective function (e.g., validation loss) and uses it to intelligently select the most promising hyperparameters to evaluate next. This approach is more efficient than exhaustive or random searches, often finding better hyperparameter configurations in fewer iterations.

4 Experiment Design and Improvements

4.1 Initial Multi-Output Approach

The initial experimental design involved a single model with multiple output heads to simultaneously predict all target variables for PPL1 and PPL2, using sensor data from DT and RT as input. While straightforward, this approach can be suboptimal if the target variables have different scales or underlying complexities.

4.2 Normalized Rate of Change Regression

An alternative approach, termed "rate" regression, was explored to improve model sensitivity to small variations. Instead of predicting the absolute downstream concentration, the model is trained to predict a normalized rate of change. This "rate" is defined as the relative change from the upstream concentration:

$$\text{Rate} = \frac{Y_{\text{downstream}} - Y_{\text{upstream}}}{Y_{\text{upstream}}}$$

This can be intuitively understood as the percentage by which the concentration has increased or decreased. The primary motivation for this transformation is that the absolute concentrations between upstream and downstream points often change very little. When training on absolute

values, the large, relatively stable upstream value can dominate the input features, saturating the neural network's weights and making it difficult for the model to learn the small but significant variations that occur downstream. By transforming the target variable into a normalized rate, the model can focus specifically on capturing these subtle changes, leading to a more effective representation of the system's dynamics.

4.3 Decoupled Model for TRC

Preliminary experiments indicated that TRC is particularly challenging to predict accurately, yet it is a crucial driver of DBP formation. To address this, a decoupled modeling strategy was designed. One model is trained exclusively to predict TRC, allowing for a more focused architecture and hyperparameter search. A second, multi-output model is then trained to predict the remaining, less-critical parameters. This specialization is hypothesized to yield better overall predictive accuracy.

5 Results and Analysis

This section presents the quantitative results from the different modeling approaches. The performance of each model is evaluated using Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE). The results for predicting TRC, pH, and TOC at the PPL2 stage are summarized in Tables 2, 3, and 4, respectively. In each table, the best performing model for each metric is highlighted in red.

Table 2: Performance Metrics for TRC Prediction at PPL2

Model	MSE	RMSE	MAE
LSTM-Rate	**0.004105**	**0.064071**	0.064056
GRU-Rate	0.004119	0.064181	0.064169
MLP-Rate	0.004132	0.064282	0.064276
MLPHIS-Rate	0.004225	0.064999	0.064992
RNN-Rate	0.004274	0.065373	0.065365
GRU-Value	0.005223	0.072270	**0.061207**
LSTM-Value	0.006218	0.078852	0.070913
MLPHIS-Value	0.006462	0.080386	0.072379
MLP-Value	0.007443	0.086271	0.072452
RNN-Value	0.007742	0.087986	0.076849

6 Conclusion

(This section is a placeholder for future results.)

This section will summarize the key findings of the study, highlighting the most effective modeling approaches and the implications for water quality management.

Table 3: Performance Metrics for pH Prediction at PPL2

Model	MSE	RMSE	MAE
GRU-Value	**0.000943**	**0.030707**	**0.023314**
RNN-Value	0.001014	0.031850	0.027056
MLPHIS-Value	0.001075	0.032787	0.026606
MLP-Value	0.001234	0.035129	0.029048
LSTM-Value	0.001317	0.036284	0.030846
LSTM-Rate	0.006097	0.078086	0.061832
MLP-Rate	0.006363	0.079769	0.062610
RNN-Rate	0.006593	0.081196	0.062938
MLPHIS-Rate	0.006601	0.081245	0.063193
GRU-Rate	0.006605	0.081271	0.063178

Table 4: Performance Metrics for TOC Prediction at PPL2

Model	MSE	RMSE	MAE
GRU-Value	**0.000121**	**0.011009**	**0.008832**
RNN-Value	0.000126	0.011223	0.009155
LSTM-Value	0.000215	0.014674	0.012685
MLP-Value	0.000281	0.016771	0.014317
LSTM-Rate	0.000282	0.016790	0.013792
MLPHIS-Rate	0.000332	0.018233	0.015136
GRU-Rate	0.000339	0.018420	0.015334
RNN-Rate	0.000348	0.018663	0.015582
MLPHIS-Value	0.000343	0.018517	0.015614
MLP-Rate	0.000391	0.019768	0.016724

7 Future Work

Based on the progress to date, several future directions have been identified:

- **Expand Dataset:** Incorporate new data as it becomes available to increase the size and diversity of the training set, which is crucial for model generalization.
- **Enhance Optimization:** Increase the number of trials in the Bayesian optimization runs to allow for a more thorough exploration of the hyperparameter space, potentially yielding more optimal models.
- **Explore Advanced Models:** Investigate state-of-the-art time series models, such as the Transformer [9] and its recent variants (e.g., iTransformer, PatchTST), which have shown strong performance on long-sequence forecasting tasks.
- **Develop a Second-Stage DBP Model:** Create a subsequent model to establish a mapping from the predicted water quality parameters (TRC, TOC, pH) to actual DBP concentrations measured by offline laboratory instruments. This would achieve the ultimate goal of predicting specific DBP levels from sensor data.

References

[1] Hervé Galliard and Urs von Gunten. Organic matter and disinfection by-products. *Water Research*, 38(1):45–56, 2004.

[2] Mst Asma Siddique, Abdul Khaleque, and Md Aminul Islam. A review of disinfection by-product (dbp) formation. *International Journal of Environmental Sciences*, 2(4):2356–2366, 2012.

[3] Guanghui Hua and David A Reckhow. Formation and control of nitrogenous disinfection by-products. *Current Opinion in Environmental Science & Health*, -1:28–34, 2015.

[4] Sepp Hochreiter and Júrgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[5] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[6] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[7] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(Feb):281–305, 2012.

[8] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, and Ł Kaiser. Attention is all you need.