Proceedings of the ASME 2019 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2019
August 18-August 21, 2019, Anaheim, CA, USA

IDETC2019-98477

PREDICTION OF NITROGEN CONCENTRATION IN FUEL CELLS USING DATA-DRIVEN MODELING

Tong Lin, Leiming Hu, Shawn Litster, Levent Burak Kara
Department of Mechanical Engineering
Carnegie Mellon University
Pittsburgh, PA 15213

ABSTRACT

This paper presents a set of data-driven methods for predicting nitrogen concentration in proton exchange membrane fuel cells (PEMFCs). The nitrogen that accumulates in the anode channel is a critical factor giving rise to significant inefficiency in fuel cells. While periodically purging the gases in the anode channel is a common strategy to combat nitrogen accumulation, such open-loop strategies also create sub-optimal purging decisions. Instead, an accurate prediction of nitrogen concentration can help devise optimal purging strategies. However, model based approaches such as CFD simulations for nitrogen prediction are often unavailable for long-stack fuel cells due to the complexity of the chemical environment, or are inherently slow preventing them from being used for real-time nitrogen prediction on deployed fuel cells. As one step toward addressing this challenge, we explore a set of data-driven techniques for learning a regression model from the input parameters to the nitrogen build-up using a model-based fuel cell simulator as an offline data generator. This allows the trained machine learning system to make fast decisions about nitrogen concentration during deployment based on other parameters that can be obtained through sensors. We describe the various methods we explore, compare the outcomes, and provide future directions in utilizing machine learning for fuel cell physics modeling in general.

INTRODUCTION

Proton exchange membrane fuel cells (PEMFCs) utilize hydrogen to generate electricity [1]. Due to their low carbon dioxide emissions and high energy density, PEMFCs are becoming increasingly more common in automotive industry as a potential substitute for traditional internal combustion engines. During normal fuel cell operations, nitrogen diffuses through the proton exchange membrane to the anode channel driven by nitrogen concentration gradient as schematically illustrated in Figure 1. The process dilutes the hydrogen in the anode channel which lowers the fuel cell efficiency. Purging out all the gases

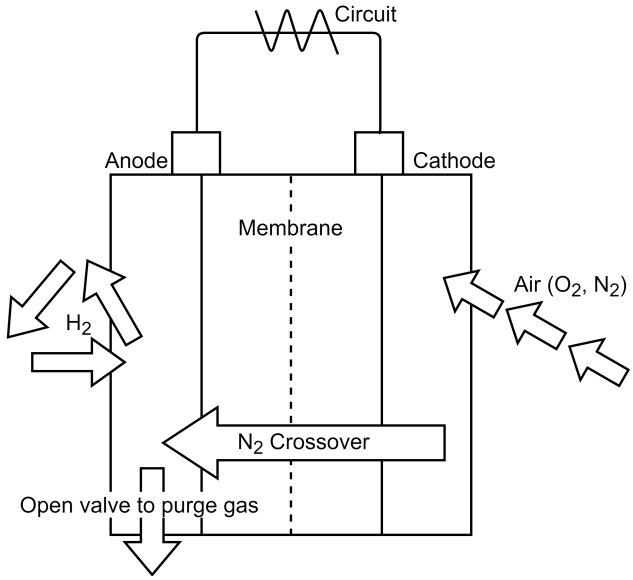


FIGURE 1: Nitrogen crossover in fuel cell.

in the anode channel after nitrogen build-up has been a major approach to combat this problem [2]. Currently, most car manufacturers purge out the anode gases at a constant frequency. The challenge of installing dedicated nitrogen sensors prevents an accurate measurement of nitrogen concentration, which in turn inhibits the development of more effective purging strategies. However, with the increase demand for fuel cell technology in automotive industry, more effective nitrogen purging strategies are needed.

A conventional approach for direct nitrogen prediction involves the use of CFD tools. The prediction at each time step is simulated by a physical model. There are two major drawbacks of using CFD simulations. First, the simulation process is computationally expensive and slow which makes it challenging to use for real-time nitrogen prediction in on-board control systems. Second, CFD-based nitrogen prediction in long-stack fuel cells is very challenging primarily due to the difficulty in creating physically accurate computational models of the highly complex chemical environment. These two disadvantages prevent the direct use of CFD for on-board nitrogen prediction.

In this work, we present a new data-driven approach for nitrogen prediction. The main purpose of this work is to demonstrate that a machine learning (ML) model can serve as an appropriate substitute for the current method of periodic purging by allowing the nitrogen concentrations to be accurately estimated from other data that can be sensed using on-board sensors. While this work uses a simulator to train the proposed ML systems for nitrogen prediction, the long-term vision of this work is to develop ML algorithms that can be trained by simply using the experiments conducted on the real, physical fuel cells. This, in turn, will alleviate the need for complex, model-based CFD simulations of long-stack fuel cells in general.

Specifically, we explore three ML approaches to establish a time-dependent regression model from the input variables to the nitrogen concentration. These include (1) K-Nearest Neighbor model, (2) Input/Output Hidden Markov Model (IOHMM), and (3) Long Short-Term Memory (LSTM) models. Our experiments suggest that after parameter tuning, LSTM outperforms the other two models with mean square value and a coefficient of determination R^2 of prediction curve on test data.

To the best of our knowledge, our work is the first attempt to demonstrate the feasibility of nitrogen concentration prediction using data-driven methods. However, we once again note that the main scope of this work is to demonstrate the feasibility of a data-driven regression model, where the models we utilize have been currently trained and tested on a simulated, short-stack fuel cell model. Future work will utilize experimental data collection, training, and validation on real fuel cells.

RELATED WORK

This work focuses on demonstrating the use of ML for the prediction of anode side nitrogen concentration in fuel cells.

Nitrogen Concentration Prediction

In the fuel cell industry, anode gas purging is usually performed at a fixed interval [3]. The determination of the purging strategy relies on a large number of experiments. Siegel et al. [4] experiment on a PEMFC and create a one-dimensional model to predict nitrogen crossover and accumulation. Rabbani and Rokni [5] present a model-based nitrogen crossover of a single-stack fuel cell. Ahluwalia and Wang [6] create a nitrogen buildup model and explore several effects on nitrogen accumulation. Kocha et al. [7] find the membrane permeability by comparing nitrogen crossover and use this value to predict nitrogen accumulation. However, these works focus on short-stack fuel cells and use a physical model, which is not practical for industrially deployed long-stack applications. When the number of stacks are in the hundreds, accurate physics models for simulations are currently unavailable. By contrast, our long-term motivation is to use the real fuel cells to learn an ML-based regression model, which can then be deployed with the fuel cell operationally for real-time nitrogen prediction.

Sequential Data Processing in ML

Nitrogen buildup is a time dependent process. To capture this effect accurately, observed data must be viewed and processed as a time-dependent and sequentially acquired signal. Typical sequential data processing is utilized to model time-dependent phenomena. In air quality prediction, Sun and Sun [8] use principle component analysis (PCA) and support vector machines (SVM) to predict PM2.5 concentration based on the PM2.5 of past days. Guo et al. [9] compare two models, artificial neural networks (ANN) and SVM, in predicting effluent concentration in waste water treatment.

Bayesian networks (BN) have been a prominent modeling framework to process sequential data [10]. Leclerc at al. [11] use a naive bayesian network to predict cyclosporine blood concentration. Dean and Kanazawa [12] use a BN-based temporal model to reason about persistence and causation. Later, the model is largely referred to as dynamic bayesian network (DBN), and has been extensively used in various data processing and dynamic control problems [13–16]. Weigend et al. [17] propose a gated mixture expert model under the DBN framework to allow the algorithm generate complex time series functions.

However, DBN has the drawback of assuming the dynamics within each discretized time slice remains the same. Nodelman et al. [18] create a continuous time BN to mitigate the need for discretized time steps in DBN models. The DBN framework usually assumes a base probabilistic distribution for each random variable. These requirements can limit the ability of DBN mod-

els to model complicated probabilistic distributions.

Sequential Data Processing Using Deep Learning

Historically, the problems in natural language processing have been of primary interest where effective sequential data processing is critical. One variant of DBN, hidden markov model (HMM), has been one prominent technique to formulate and solve such problems. However, with recent advances in neural computing, recurrent neural networks (RNN) and its variants have now mostly replaced HMMs. RNN defines an neural network structure for each time slice, similar to DBNs. Hopfield networks were an early instantiation of RNNs that have demonstrated RNN's ability in deciphering patterns from seemingly corrupt, missing, and noisy inputs [19], and with the advent of backpropagation [20], efficient network optimization became possible.

A major drawback of RNNs is the propensity for memory loss with coarser time steps, as vanishing and exploding gradients cannot properly propagate [21]. Long short-term memory (LSTM) models address this problem by introducing internal states and using gate mechanisms [22]. LSTM has shown its success in natural language processing. Wang et al. [23] use a deep LSTM structure to encode tweets and classify the tweets based on the encoding. Their method outperforms other conventional ML algorithms in terms of classification accuracy. LSTM has also been used for language translation. Sutskever et al. [24] use an LSTM model to encode one language and another LSTM to translate the encoding to another language.

Although ML and more recently deep learning have been successfully utilized to formulate and learn from sequential data, today nitrogen prediction in fuel cells rely solely on physical models. Our approach is thus one step toward allowing fuel cell nitrogen prediction to be formulated as a ML problem.

METHODS

This section first presents the method we use to generate Nitrogen data. Then the machine learning algorithms we have explored are described in details. In this study, the regression model we choose is K-nearest Neighbor (KNN) model, Input/Output Hidden Markov (IOHMM) Model and Long Short-term Memory (LSTM) model. KNN is selected primarily as a baseline algorithm due to its simplicity. IOHMM and LSTM are the algorithm framework we focus on as both of the algorithms are widely used to deal with sequential data.

Data Generation

The training and testing data sets are generated from a two dimensional finite element model of a single-stack fuel cell. In this model, the anode hydrogen recirculation, the purging process, and the nitrogen accumulation have been simulated. The nitrogen diffusion from the cathode to the anode can be divided into four different processes. The nitrogen dissolution at the boundary between the catalyst layer and the proton exchange membrane is governed by Henry's law:

$$\dot{n}_{N_2,dissolved} = k_{N_2}\left(c_{N_2,dissolved} - p_{N_2}/H_{N_2}\right) \tag{1}$$

where $\dot{n}_{N_2,dissolved}$ is the nitrogen mass flux due to dissolution at the catalyst layer and membrane interface, $c_{N_2,dissolved}$ is the nitrogen concentration within the proton exchange membrane, p_{N_2} is the gas partial pressure of nitrogen and H_{N_2} is the Henry's constant of nitrogen gas in the membrane.

The diffusion of nitrogen through the proton exchange membrane is a gas concentration gradient driven process and is governed by Eq(2):

$$\frac{\delta c_{N_2,dissolved}}{\delta t} = \nabla\left(D_{N_2}^{Nafion}\nabla c_{N_2,dissolved}\right) \tag{2}$$

where $D_{N_2}^{Nafion}$ is the diffusivity of nitrogen within the proton exchange membrane.

Nitrogen diffusion in the porous electrode media is described in the model using the following equation:

$$\frac{\delta \varepsilon c_{N_2,dissolved}}{\delta t} = \nabla\left(D_{N_2}^{eff}\nabla c_{N_2}\right) \tag{3}$$

$D_{N_2}^{eff}$ is the effective gas diffusivity of nitrogen in porous materials.

Nitrogen diffusion in the anode gas channel is represented by Eq(4):

$$\frac{\delta c_{N_2}}{\delta t} + \nabla\left(\vec{u}_g c_{N_2}\right) = \nabla\left(D_{N_2}\nabla c_{N_2}\right) \tag{4}$$

where $\vec{u}_g$ is the gas velocity inside the channel.

The fuel cell model also takes into account the electrochemical reactions as well as heat generation at the anode and cathode with governing equations used in previous studies [25, 26]. The generated data has 9 features and 1 output. These 9 features (inputs to the ML algorithms) are the cell voltage (V), current density ($\frac{A}{m^2}$), temperature (K), cathode inlet/outlet relative humidity, anode inlet/outlet relative humidity, anode gas velocity ($\frac{m}{s}$) and purge valve opening amount. When the purge valve is fully open, the purge open is 1. When it is fully closed, the value is 0. The output is the anode channel nitrogen concentration in $\frac{mol}{m^3}$, which is the target quantity our approach will try to estimate using the above inputs.

Data Preprocessing

We standardize each feature in the training data through mean subtraction and normalization by its standard deviation, and apply the same standardization coefficients to the test data. This allows the features that can exhibit varying orders of magnitudes to be properly handled by the ML algorithms:

$$x_i \leftarrow \frac{x_i - \mu}{\sigma} \quad \forall i \leq n \tag{5}$$

$$xt_i \leftarrow \frac{xt_i - \mu}{\sigma} \quad \forall i \leq n \tag{6}$$

where n is the number of features, x_i are the standardized values of feature i in the training data, xt_i are the analogous quantities in the test data, and μ and σ are the mean and standard deviation of x_i before standardization respectively.

Test criteria

We use the mean absolute error (MAE), mean square error (MSE) and coefficient of determination (r^2) to test our algorithms' performance.

K-Nearest Neighbor model

K-Nearest Neighbor (KNN) stores the features and outputs of all the data samples in a training set. For a test point, the algorithm finds K nearest samples in the feature space of the training set. The output of the test point is then a weighted sum of the outputs of the K training samples. The weight of a neighbor point is proportional to its proximity (inverse Euclidean distance) to the test point:

$$D_k = ||X^{test} - X^k||_2 \tag{7}$$

$$Y^{test} = \sum_{k=1}^{K} \frac{1}{D_k} Y_k \tag{8}$$

where D_k is the distance of the k^{th} nearest training point X^k to the test point X^{test}. Y_k is the output of the k^{th} point. Y^{test} is the prediction result of the test point.

We studied several K values to obtain the best results. The output of KNN is strongly dependent on the proximity of the training samples in the feature space. Thus, the algorithm does not generalize well for test points not covered in the training set. However, the results of KNN can be used to gain insights into how similar the training set is to the test set.

Data Arrangement for Dynamic ML Models

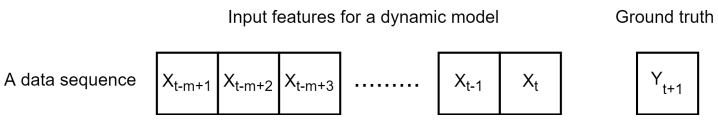


FIGURE 2: Illustration of data arrangement used for dynamic ML models.

For IOHMM: $X_t = [\ x_1\ x_2\ \ldots\ x_{n-1}\ x_n\ y\ ]_t$

For LSTM: $X_t = [\ x_1\ x_2\ \ldots\ x_{n-1}\ x_n\ ]_t$

FIGURE 3: Explanation of input features.

Both the dynamic Bayesian network (DBN) model and the Long Short-Term Memory (LSTM) model are dynamic models, which require data to be modeled as a sequential set. Both of the models need past data to predict a future quantity. For current time t, the arrangement of one data sequence at time t is shown in Figure 2. X_t is the input features X for the dynamic models at time t. m is the time step of the model. It represents how much past information we want to feed in our dynamic model and is a hyper-parameter that we need to tune. Y_{t+1} is the quantity that we want to predict at time $t+1$. It corresponds to nitrogen concentration at time $t+1$.

The quantity X_t varies depending on the model we choose. In this paper, we have studied Input/Output Hidden Markov Model (IOHMM), which is a variant of DBN, and LSTM. The two models require different X at time t. For IOHMM, X at time t is a 10-D vector consisting of the 9 features at time t from our generated data and the nitrogen concentration at time t. The model requires nitrogen concentration at each time step for its output node. The concept of the output node will be explained in IOHMM Model section. For LSTM, the nitrogen concentration at time t is omitted. The two settings of X_t are illustrated in Figure 3, where lower case x_n is the n^{th} feature in our generated data. Thus, the maximum value of n is 9 in our case. For a data set of N points, we have N-m sets of such data sequence and N-m target values. These sets of data sequence are used for training our dynamic models.

IOHMM Model

Bayesian network uses a graph structure to represent the conditional probability distribution of each random variable appearing in a node. The bayesian network model we study is

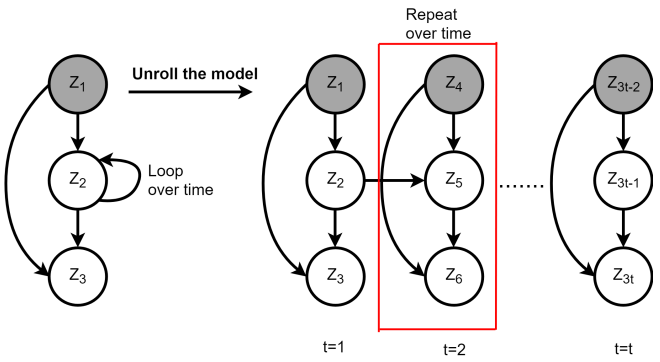


FIGURE 4: Illustration of IOHMM model.

called an Input/Output Hidden Markov Model (IOHMM) (Figure 4 left). The shaded node (input node) is deterministic. The two white nodes are probabilistic. Node Z_3 represents the output of the model and is called output node. The conditional distributions of the two white nodes are:

$$Z_2 \sim P(Z_2|Z_1) \tag{9}$$

$$Z_3 \sim P(Z_3|Z_1) \tag{10}$$

IOHMM models the relationship between the inputs and an output at time t, and is called a one time-slice model. This model repeats at each time step to represent a time-dependent dynamic process. Since the one time-slice model repeats after $t = 1$, the complete IOHMM can be unrolled (Figure 4 right). Once a time step m is set, a data sequence of length m is generated. The model is unrolled m times until it covers the time step. Moreover, since the model structure at $t = 2$ repeats (red rectangle in Figure 4), the entire conditional dependence of the IOHMM model can be fully defined by the model structure consisting of the model at $t = 1$ and $t = 2$ only.

In Eq(10), Z_2 is not included because Z_2 is a hidden variable. However, Eq(10) can be expanded as:

$$P(Z_3|Z_1) = \sum_{i=1}^{n_z} P(Z_3|Z_1, Z_2 = i)P(Z_2 = i|Z_1) \tag{11}$$

where $P(Z_2|Z_1)$ is a hidden distribution that depends on the input features in the data set. $P(Z_3|Z_1, Z_2)$ is the base distribution for the output and its value depends on both the hidden variables and the inputs. From Eq(11) it can be seen that the model is able to represent a combination of different distributions that involve the input features and the output. This allows a more accurate discovery of the underlying distribution in the data set.

To represent the time-dependent buildup of nitrogen, we assume that the hidden state from the previous step influences the current hidden state. Thus, the distribution of Z_5 needs to be dependent of Node 2:

$$Z_5 \sim P(Z_5|Z_2, Z_4) \tag{12}$$

The model inside the red rectangle in Figure 4 repeats over time. Thus, Eq(9), Eq(10), and Eq(12) fully define the correlation of each node in the IOHMM model.

In Eq(9-12), the nature of the underlying distributions are user-defined. Since the output is continuous, we use the widely used continuous linear Gaussian to associate the input features and nitrogen concentration in $P(Z_3|Z_1, Z_2 = i)$. Z_3 and Z_5 are discrete hidden variables. Thus, we use the softmax function to model the probability.

LSTM based model

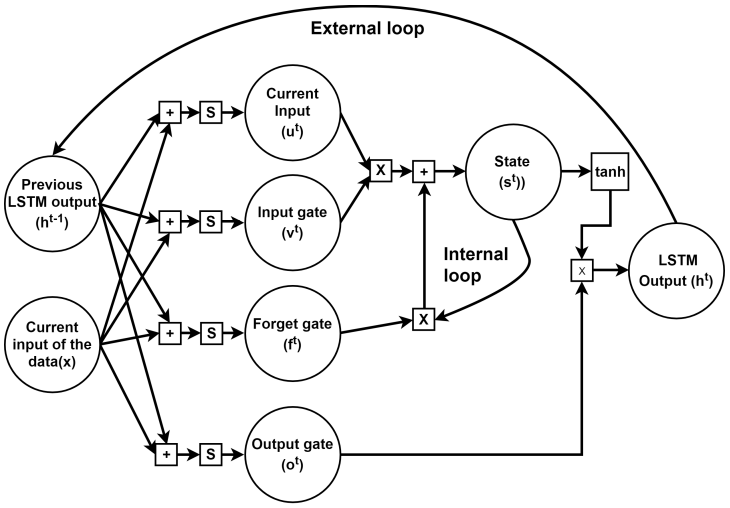


FIGURE 5: An LSTM block.

LSTM is an RNN based model designed for sequential data. Its structure is shown in Figure 5, governed by the following relationships:

$$u_i^t = S(b_i^u + \sum_{j=1}^{n'} T_{i,j}^u x_j^t + \sum_{j=1}^{m} W_{i,j}^u h_j^{t-1}) \tag{13}$$

$$v_i^t = S(b_i^v + \sum_{j=1}^{n'} T_{i,j}^v x_j^t + \sum_{j=1}^{m} W_{i,j}^v h_j^{t-1}) \quad (14)$$

$$f_i^t = S(b_i^f + \sum_{j=1}^{n'} T_{i,j}^f x_j^t + \sum_{j=1}^{m} W_{i,j}^f h_j^{t-1}) \quad (15)$$

$$s_i^t = f_i s_i^{t-1} + v_i u_i \quad (16)$$

$$o_i^t = S(b_i^o + \sum_{j=1}^{n'} T_{i,j}^o x_j^t + \sum_{j=1}^{m} W_{i,j}^o h_j^{t-1}) \quad (17)$$

$$h_i^t = tanh(s_i^t) o_i^t \quad (18)$$

In all the above equations, b, T, W are the trainable coefficients of the LSTM model. b represents bias. T and W represent the weights of the current input and weights of the previous LSTM output.

The LSTM model repeats itself at each time step. The current data sequence x^t and the previous model results h_{t-1} are fed into the model at current time instance, as represented in Eq(13). A sigmoid function S is used as a non-linear activation function. The repetition of the model is denoted with the external loop in Figure 5.

The original RNN model suffers from vanishing or exploding gradients if the model repeats for a long time duration during training. LSTM adopts two strategies to address this issue. First, it implements a gate mechanism, which is a sigmoid function that is able to learn how much gradient to backpropagate through an analysis of the current inputs. For example, Eq(14) is the equation for input gate v_i^t. It is a sigmoid function whose inputs are the previous LSTM results and the current data sequence. The information attenuation is represented by the term "$v_i u_i$" in Eq(16).

Second, LSTM creates a state variable s_i that also repeats itself over time inside the whole model. This is shown as the internal loop in Figure 5. The state variable receives information from the current input and the previous state. The amount of information to accept depends on the input gate and the forget gate. This is shown in Eq(16). This ability to integrate knowledge from previous states serves as a memory. The internal loop produces a path for gradient flow over longer time durations. A forget gate f_i controls how much information from the previous state to output to the next time instance. This is shown in Eq(15). The final output of an LSTM block at time t is calculated via Eq(18). The output gate o_i in Eq(18) is calculated via Eq(17).

The basic structure of LSTM can be used as a block. Figure 5 is a single LSTM block. Other models can be stacked above the core LSTM block to increase model complexity.

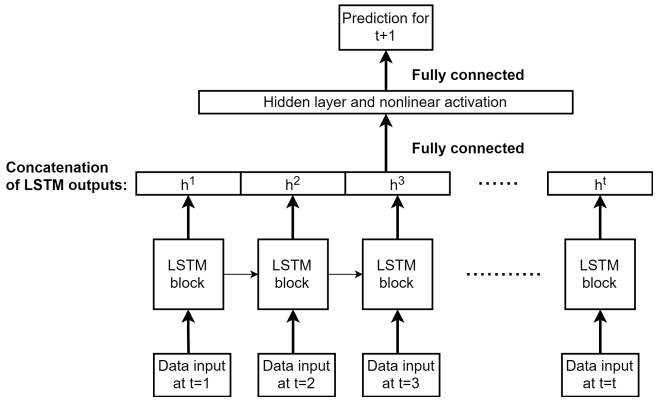


FIGURE 6: The structure of LSTM based model.

We hypothesize that the output at every time step of the model has an effect on the final nitrogen concentration. Thus, in our model, we concatenate the output of each LSTM block at the individual time slices. Then, we augment a multiple layer perceptron (MLP) to the fully unrolled LSTM model. The MLP is used for predicting the nitrogen concentration at $t+1$. The complete structure is shown in Figure 6. The outputs of the LSTM module is used as the input for MLP. The mean square error (MSE) is used as the loss function. Thus, our goal is to optimize:

$$argmin_{W_{MLP}, W_{LSTM}} (\sum_{j=m+1}^{N} (y_j - F(H_j)))^2 \quad (19)$$

where N denotes the total number of pieces of training data, m denotes the time step, y_j is the j^{th} N_2 concentration in the training set. $F(.)$ is our MLP model. W_{MLP} is the trainable weights in MLP. W_{LSTM} is b,T and W in Eq(13-17). H_j is the concatenation of the outputs of the LSTM block at each time step.

Several layer structures and activation functions have been experimented and we report the optimal structure in the next section.

EXPERIMENTS AND RESULTS

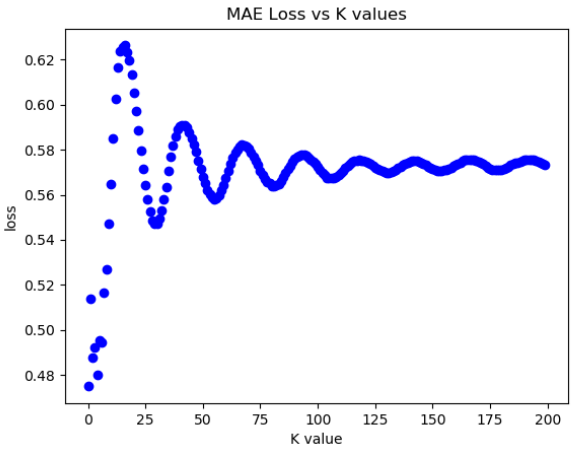
Data set

The data is generated uniformly at a sample rate of 2Hz using our single-stack fuel cell simulator. The training set contains

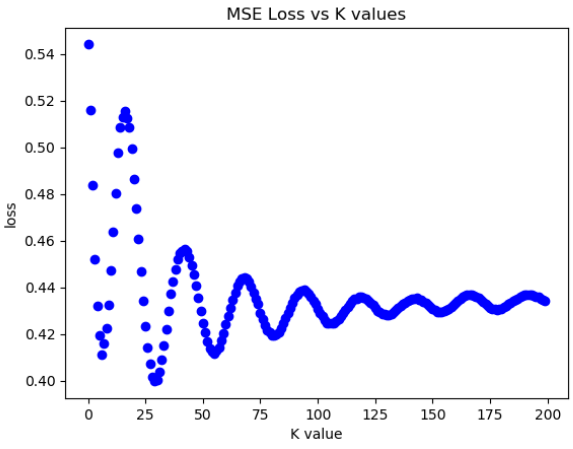
2701 samples, whereas the test set contains 2240 samples. For test data set, the maximum value of the nitrogen concentration is 2.52 $\frac{mol}{m^3}$. The minimum value is 0 $\frac{mol}{m^3}$. The range is thus 2.52 $\frac{mol}{m^3}$. Due to high linear correlation between cathode inlet relative humidity and cathode outlet relative humidity, we discarded the latter feature in our experiment.

KNN

The KNN algorithm serves as a baseline algorithm for this study. The algorithm is able to reveal the similarity between the training and test data.



(a) MAE Loss.



(b) MSE Loss.

FIGURE 7: MAE and MSE losses with different K values.

We have tested K values from 1 to 200. The MAE and MSE results are shown in Figure 7. The lowest MAE result is 0.475 (at K=1). The lowest MSE result is 0.400 (at K=30). Both of the losses tend to converge. This is due to the fact that as K increases, the algorithm averages more points around the target point. This can be seen in Figure 8 and Figure 10, where the increased number of neighbors amplifies the averaging effect of KNN.

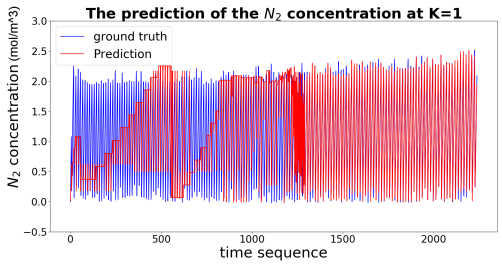


FIGURE 8: KNN prediction $K = 1$.

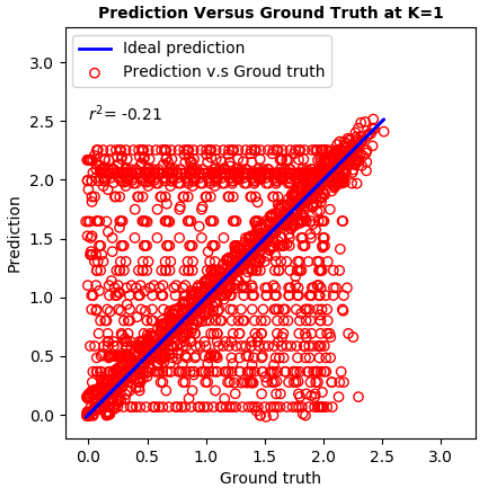


FIGURE 9: R^2 plot using KNN at K=1.

When K=1, only the nearest neighbor is used for prediction, which helps reveal how well the test set is covered by the training set. From Figure 8, we see that KNN has a better prediction on the latter part of the test set. This implies that the output of the latter part of our test data set is similar to the training set. By contrast, the first half of the test data is only weakly covered by the training set. This is strongly indicated by R^2 plot shown in Figure 9. The r^2 value indicates that the prediction is poor. However, we observe that large number of prediction

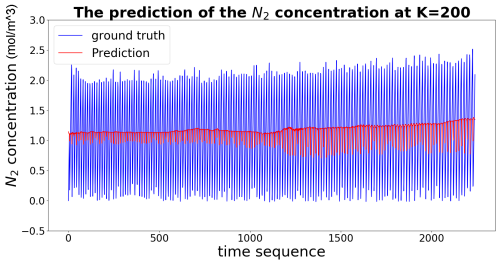


FIGURE 10: KNN prediction $K = 200$.

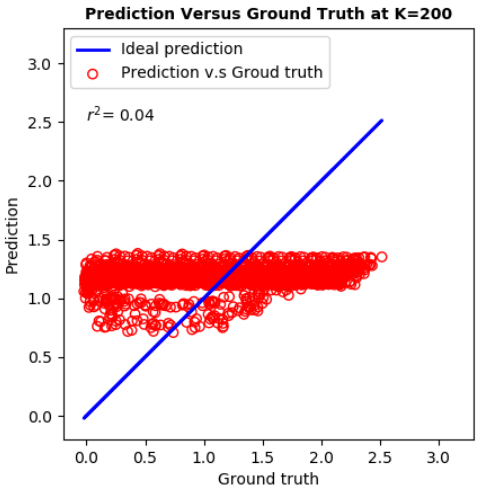


FIGURE 11: R^2 plot using KNN at $K = 200$.

points cluster around the ideal prediction curve while the rest points scatter.

As mentioned, KNN performs simply as an overall means estimator for high K values. We thus treat the loss at K=200 as a baseline *reference* loss that must be outperformed by all methods. The MAE and MSE loss are 0.573 and 0.434, respectively. Hence, for IOHMM and LSTM, we expect them to perform (ideally much) better than this quantified baseline performance. The outcome of KNN at K=200 is shown in Figure 10 and we can see that the predicted values are close to the average value of the ground truth. The R^2 plot is shown in Figure 11. That r^2 value is close to zero also reveals the average effect for K at 200.

IOHMM

For IOHMM, we perform a parametric grid search at time step (T) from 1 to 10 and on discrete hidden variable (Z) from 1 to 6. The best MSE of 0.431 is obtained at $T = 2$ and $size(Z) = 2$. The best MAE of 0.563 happens at $T = 7$ and $size(Z) = 6$. By comparing the losses with the KNN baseline average loss, it can be concluded that IOHMM does not perform any better than the baseline average prediction of the nitrogen concentration. This is shown in Figure 12.

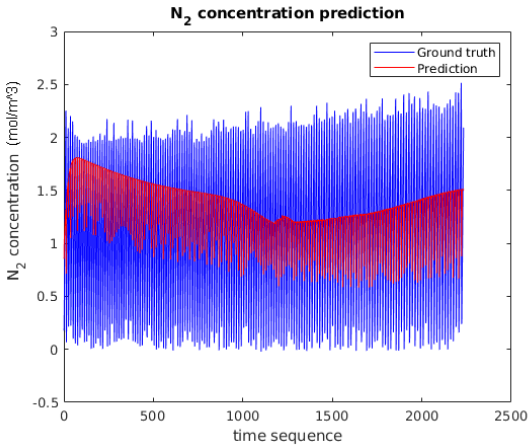


FIGURE 12: IOHMM prediction.

Figure 13 shows the R^2 plot for IOHMM, which also reveals the poor prediction ability of IOHMM.

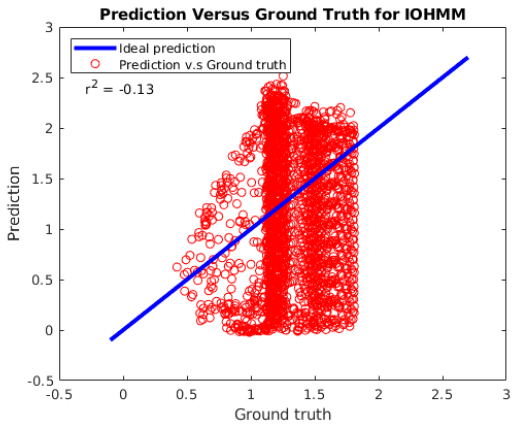


FIGURE 13: R^2 plot for IOHMM.

LSTM

For the LSTM model, we varied the number of hidden nodes. For the emprically optimized LSTM, we use a single LSTM block. The number of outputs (h) of LSTM is set to 256. One fully connected MLP layer is appended to the LSTM block. The number of hidden units for the fully connected layer is 256. We use 100 time steps. Our algorithm is trained for 500 epochs. The best test MSE error is 0.00884 at 113 epoch, shown in green dot in Figure 14.

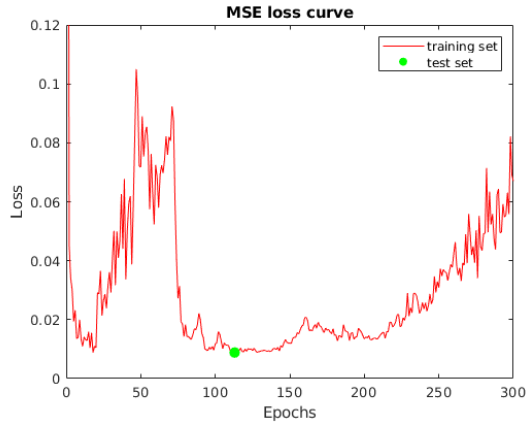


FIGURE 14: LSTM MSE of test set over epochs.

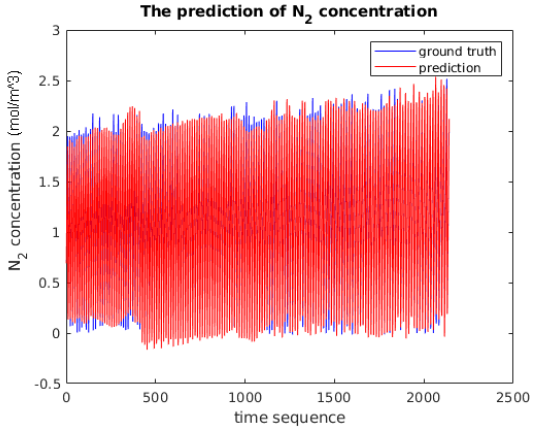


FIGURE 15: LSTM prediction.

Step Size	MSE	MAE
10	0.0209	0.117
50	0.00893	0.0773
100	0.00884	0.0757

TABLE 1: Effect of different step sizes on LSTM.

LSTM prediction results are shown in Figure 15. Compared to Figure 8, Figure 10 and Figure 12, we can see that the LSTM prediction has a better coverage of the ground truth than KNN and IOHMM. A zoomed in picture of a specific region is shown in Figure 17 to reveal the prediction accuracy. The R^2 plot is shown in Figure 16. All the prediction points cluster around the ideal prediction curve. The r^2 value is close to one. The plot reveals both low variance and low bias of the prediction result.

For time step determination, we use the LSTM time steps of 10, 50 and 100 as shown in Table 1. The time step of 100 results

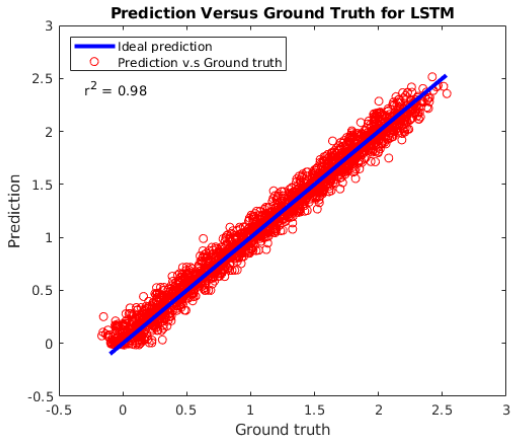


FIGURE 16: R^2 plot using LSTM.

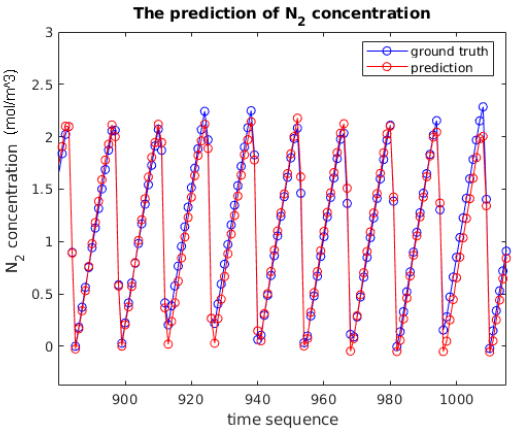


FIGURE 17: A zoom in of Figure 15.

Performance of Different Models on Test Set		
Model	MSE	MAE
KNN (Average)	0.434	0.573
KNN	0.400	0.475
IOHMM	0.431	0.563
LSTM Based Model	**0.00884**	**0.0757**

TABLE 2: Comparison of different models.

in the best prediction as revealed by the MSE and MAE losses.

Comparison of Different Algorithms

Table 2 shows the results of different algorithms. LSTM performs the best by far. With respect to MSE, it outperforms KNN averaging by 98.0%, the best KNN result by 97.8%, and

Input Effect Test		
Input variable	Fixed at Min (MSE)	Fixed at Max (MSE)
Voltage	598%	4903%
Current Density	25%	521%
Temperature	407%	433%
Relative Humidity of Cathode Inlet	300%	389%
Relative Humidity of Anode Inlet	24.7%	108%
Relative Humidity of Anode Outlet	41.5%	510%
Anode Gas Velocity	194%	209%
Purge Open Amount	15350%	72486%

TABLE 3: Effect of different features on LSTM.

IOHMM by 98.0%. We also note that IOHMM has a similar weak performance to KNN averaging ($K = 200$) method.

Input Features' Effect on LSTM-based Algorithm

Based on LSTM's performance, we further study each input feature's influence on this model. The analysis is important as (1) it is able to tell us if the LSTM based model captures the most important information correctly, and (2) if any input feature is deemed to have an insignificant influence on nitrogen concentration, this can signal an opportunity to avoid instrumenting sensors for that particular feature on the physical fuel cell.

For each test sample, we freeze one of the input features at its minimum value in the training set and analyze how the nitrogen prediction performance and MSE loss change. The same experiment is repeated by freezing the feature at its maximum value in the training set. The relative MSE change is shown in Table 3. The results show that every input has an impact on the prediction performance. The minimum change of MSE is 24.7% when relative humidity of anode inlet is fixed at its minimum. We note that even a 500% worsening of MSE may not be considered significant, noting that the output range is 2.52 and our best MSE is 0.00884. However, the relative degradation of MSE with each feature provides insights into the qualitative significance of the features. For instance, the largest degradation happens when the purge open amount is fixed. This indicates that the LSTM based algorithm suggests the purge open amount to be the most critical input feature that affects nitrogen concentration. This aligns well with the simulation model, where all gas release in the anode is directly impacted by the purge valve.

CONCLUSION

Due to the complex electrochemistry and external interventions (such as valve purging), real-time nitrogen prediction in fuel cells presents a major challenge. This study utilizes and compares three different ML algorithms that model the highly non-linear dynamics of nitrogen concentration buildup as a data-driven, time-dependent regression problem. Our LSTM model produces by far the best performance, and is able to predict nitrogen concentration with significant accuracy. To the best of our knowledge, this work is the first study of that uses a data-driven approach for predicting nitrogen concentration in PEMFCs.

Our current data set is based on data obtained from a simulation model of a single-stack fuel cell. In later work, we intend to collect experimental data from long-stack fuel cells to validate our LSTM-based nitrogen concentration prediction method in more realistic settings.

REFERENCES

[1] Balakrishnan, J., 2007. "Fuel cell technology". *2007 IEEE Canada Electrical Power Conference*, October, pp. 159–160.

[2] Chen, Y.-S., 2014. "Implementation and evaluation for anode purging of a fuel cell based on nitrogen concentration". *Applied Energy,* ***113***, 01, p. 1519.

[3] Müller, E. A., Kolb, F. M., Guzzella, L., Stefanopoulou, A. G., and McKay, D. A., 2010. "Correlating nitrogen accumulation with temporal fuel cell performance".

[4] Siegel, J., Bohac, S., Stefanopoulou, A., and Yesilyurt, S., 2010. "Nitrogen front evolution in purged polymer electrolyte membrane fuel cell with dead-ended anode". *Journal of The Electrochemical Society - J ELECTROCHEM SOC,* ***157***, 01.

[5] Rabbani, R., and Rokni, M., 2013. "Effect of nitrogen crossover on purging strategy in pem fuel cell systems". *Applied Energy,* ***111***, pp. 1061–1070.

[6] Ahluwalia, R., and Wang, X., 2007. "Buildup of nitrogen in direct hydrogen polymer-electrolyte fuel cell stacks". *Journal of Power Sources,* ***171***(1), pp. 63 – 71. Scientific Advances in Fuel Cell Systems, Turin, Italy, 13-14 September 2006.

[7] Kocha, S. S., Deliang Yang, J., and Yi, J. S., 2006. "Characterization of gas crossover and its implications in pem fuel cells". *AIChE Journal,* ***52***(5), pp. 1916–1925.

[8] Sun, W., and Sun, J., 2017. "Daily pm2.5 concentration prediction based on principal component analysis and lssvm optimized by cuckoo search algorithm". *Journal of Environmental Management,* ***188***, pp. 144 – 152.

[9] Guo, H., Jeong, K., Lim, J., Jo, J., Kim, Y. M., pyo Park,

J., Kim, J. H., and Cho, K. H., 2015. "Prediction of effluent concentration in a wastewater treatment plant using machine learning models". *Journal of Environmental Sciences,* ***32***, pp. 90 – 101.

[10] Krause, P., 1998. "Learning probabilistic networks". *The Knowledge Engineering Review,* ***13***, 12, pp. 321–351.

[11] Leclerc, V., Ducher, M., and Bleyzac, N., 2018. "Bayesian networks: A new approach to predict therapeutic range achievement of initial cyclosporine blood concentration after pediatric hematopoietic stem cell transplantation". *Drugs in R&D,* ***18***, 02.

[12] Dean, T., and Kanazawa, K., 1989. "A model for reasoning about persistence and causation". *Comput. Intell.,* ***5***(3), Dec., pp. 142–150.

[13] Onisko, A., Druzdzel, M. J., and Austin, R. M., 2009. "Application of dynamic bayesian networks to cervical cancer screening".

[14] Deventer, R., Denzler, J., and Niemann, H., 2000. "Control of dynamic systems using bayesian networks". In Proceedings of the IBERAMIA/SBIA 2000 Workshops (Atibaia, Sdo Paulo.

[15] Marini, S., Trifoglio, E., Barbarini, N., Sambo, F., Di Camillo, B., Malovini, A., Manfrini, M., Cobelli, C., and Bellazzi, R., 2015. "A dynamic bayesian network model for long-term simulation of clinical complications in type 1 diabetes". *J. of Biomedical Informatics,* ***57***(C), Oct., pp. 369–376.

[16] Kim, S., Imoto, S., and Miyano, S., 2003. "Dynamic bayesian network and nonparametric regression for nonlinear modeling of gene networks from time series gene expression data". *Bio Systems,* ***75 1-3***, pp. 57–65.

[17] Weigend, A. S., Mangeas, M., and Srivastava, A. N., 1995. "Nonlinear gated experts for time series: discovering regimes and avoiding overfitting". *International journal of neural systems,* ***6 4***, pp. 373–99.

[18] Nodelman, U., Shelton, C. R., and Koller, D., 2002. "Continuous time bayesian networks". In Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence, UAI'02, Morgan Kaufmann Publishers Inc., pp. 378–387.

[19] Hopfield, J. J., 1988. "Neurocomputing: Foundations of research". MIT Press, Cambridge, MA, USA, ch. Neural Networks and Physical Systems with Emergent Collective Computational Abilities, pp. 457–464.

[20] Rumelhart, D. E., Hinton, G. E., and Williams, R. J., 1988. "Neurocomputing: Foundations of research". MIT Press, Cambridge, MA, USA, ch. Learning Representations by Back-propagating Errors, pp. 696–699.

[21] Pascanu, R., Mikolov, T., and Bengio, Y., 2013. "On the difficulty of training recurrent neural networks". In Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28, ICML'13, JMLR.org, pp. III–1310–III–1318.

[22] Hochreiter, S., and Schmidhuber, J., 1997. "Long short-term memory". *Neural Comput.,* ***9***(8), Nov., pp. 1735–1780.

[23] Wang, X., Liu, Y., Sun, C., Wang, B., and Wang, X., 2015. "Predicting polarities of tweets by composing word embeddings with long short-term memory". In ACL.

[24] Sutskever, I., Vinyals, O., and Le, Q. V., 2014. "Sequence to sequence learning with neural networks". In Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, NIPS'14, MIT Press, pp. 3104–3112.

[25] Babu, S. K., Chung, H. T., Zelenay, P., and Litster, S., 2017. "Modeling electrochemical performance of the hierarchical morphology of precious group metal-free cathode for polymer electrolyte fuel cell". *Journal of The Electrochemical Society,* ***164***(9), pp. F1037–F1049.

[26] Ogawa, S., Babu, S. K., Chung, H. T., Zelenay, P., Padgett, E., Muller, D. A., Kongkanand, A., and Litster, S., 2017. "Microstructural modeling of pefc catalyst layer performance and durability". In Meeting Abstracts, no. 32, The Electrochemical Society, pp. 1374–1374.