Process Data Analytics Using Deep Learning Techniques

Majid Moradi Aliabadi
Wayne State University

Follow this and additional works at: https://digitalcommons.wayne.edu/oa_theses

Part of the Chemical Engineering Commons, and the Computer Sciences Commons

Recommended Citation
https://digitalcommons.wayne.edu/oa_theses/761

This Open Access Thesis is brought to you for free and open access by DigitalCommons@WayneState. It has been accepted for inclusion in Wayne State University Theses by an authorized administrator of DigitalCommons@WayneState.
PROCESS DATA ANALYTICS USING DEEP LEARNING TECHNIQUES

by

MAJID MORADI ALIABADI

THESIS

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

2020

MAJOR: COMPUTER SCIENCE

Approved By:

Co-advisor Date

Co-advisor Date
DEDICATION

To my lovely wife, Hajar Emami Gohari for her endless love, encouragement and support.
ACKNOWLEDGMENTS

I would like to express my deep and sincere gratitude to my advisors, Prof. Ming Dong and Prof. Yinlun Huang for their continuous supports and academic advises during my research.
# TABLE OF CONTENTS

DEDICATION ................................................................................................................................. ii

ACKNOWLEDGMENTS ................................................................................................................. iii

LIST OF TABLES .......................................................................................................................... vi

LIST OF FIGURES ......................................................................................................................... vii

CHAPTER 1 INTRODUCTION ......................................................................................................... 1

1.1 Process Data Analytics Using Machine Learning Techniques .................................................. 3

1.2 Motivations and Objectives ..................................................................................................... 5

1.3 Thesis Outline ......................................................................................................................... 6

CHAPTER 2 DEEP LEARNING: FOCUS ON RECURRENT NEURAL NETWORKS .................................. 8

2.1 Recurrent Neural Networks (RNNs) .......................................................................................... 11

2.2.1 Gated Recurrent Neural Networks ..................................................................................... 13

2.3 Deep learning applications in manufacturing processes .......................................................... 17

CHAPTER 3 ATTENTION-BASED RNN FOR MULTI-STEP-AHEAD PREDICTION ......................... 19

3.1 Problem Statement .................................................................................................................. 19

3.2 Attention-Based Encoder-Decoder RNN Model ....................................................................... 20

3.2.1 Encoder RNN ..................................................................................................................... 20

3.2.2 Decoder RNN ..................................................................................................................... 22

3.2.3 Temporal Attention Model ................................................................................................. 22

3.2.4 Computational procedure .................................................................................................. 23

3.3 Loss function and model training .............................................................................................. 25

3.4 Summary .................................................................................................................................. 27

CHAPTER 4 CASE STUDY: CATALYST ACTIVITY PREDICTION ................................................... 28
LIST OF TABLES

Table 1. List of key process parameters................................................................. 33
Table 2. Computed average MAE for ablation of our model for MS prediction............. 39
Table 3. Comparative results between our model and M-SVR. ................................... 40
Table 4. LPMEOH operating program campaigns ..................................................... 46
LIST OF FIGURES

Figure 1. Process data hierarchy. ........................................................................................................... 1
Figure 2. Process data types: (a) time series (b) thermography, and (c) spectrum ......................... 2
Figure 3. A framework for process data analytics using machine learning techniques .............. 5
Figure 4. Relation between classic machine learning and representation learning. Shaded boxes show the components that can learn from data. ................................................................. 8
Figure 5. Feedforward neural network ................................................................................................. 9
Figure 6. MLPs with different number of layers for function approximation (adopted from [9]) 10
Figure 7. Recurrent Neural Network: (a) connections in hidden layer, and (b) equivalent unfolded network through the time for four-time steps ................................................................. 12
Figure 8. (a) SRN unit, and (b) LSTM block .................................................................................... 15
Figure 9. Gated recurrent unit (GRU) ............................................................................................... 16
Figure 10. Attention-based RNN model for MS prediction ............................................................... 21
Figure 11. Simplified process flow diagram of the liquid phase methanol process .................... 29
Figure 12. A simple schematic of LPMEOH slurry reactor .............................................................. 30
Figure 13. Process data of three parameters during the first campaign ........................................ 35
Figure 14. A sliding window approach for data sampling ............................................................... 36
Figure 15. The effect of hidden size on model performance ............................................................ 37
Figure 16. The effect of lookback window length on model performance ........................................ 38
Figure 17. Catalyst performance during the first campaign ............................................................ 47
Figure 18. Catalyst inventory during the first campaign ................................................................. 47
Figure 19. Catalyst performance during the second campaign ....................................................... 48
Figure 20. Catalyst inventory during the second campaign ............................................................ 49
Figure 21. Catalyst performance during the third campaign. .......................................................... 50

Figure 22. Reactor temperature during the third campaign. .......................................................... 50

Figure 23. Catalyst performance during the fourth campaign. ....................................................... 51
CHAPTER 1 INTRODUCTION

Because of the advances in smart manufacturing, Industrial Internet of Things (IIOT), and data storage, large amount of data is created and collected at different levels in modern chemical plants. They could be equipment data, process operational data (historical or real-time), process design and product quality data, economic and environmental (including process safety, waste emission and health impact) data. The process data hierarchy that starts from equipment sensor measurements at the bottom level to the customer data at the top level is shown in Figure 1 (Adopted from [1]). Also, different types of data including time series, image, and spectral data (e.g. gas chromatography) are generated in a chemical manufacturing plant. For example, Figure 2 shows the three types of data (time series, thermography, and spectrum) that are usually generated in any chemical plants. Beside size and variety characteristics of the data, process operational data might be (i) noisy, (ii) Dynamic, (iii) different in terms of sampling time (e.g. online measurements and laboratory analysis), (iv)
Figure 2. Process data types: (a) time series (b) thermography, and (c) spectrum

incomplete or unlabeled, and (v) highly correlated [2]. Because of these characteristics, effective knowledge extraction from process data is a very challenging task.

Machine learning is one of the fastest growing fields in computer science. Based on the Wikipedia definition, machine learning is a subset of artificial intelligence in the field of computer science that evolved from the study of pattern recognition and computational learning theory. Machine learning methods are used as data mining tools to extract useful information from large database. They are also intelligent as they have the ability to learn and adopt to changes in a changing environment [3]. Machine learning techniques are classified in four categories as follows [3], [4]:

- **Supervised learning**: these techniques are used when training data that comprises examples of the input vectors along with their corresponding targets are available. This is called classification when the target is categorical (e.g. fault classification). If the desired output consists of one or more continuous variables, then the task is called regression.

- **Unsupervised learning**: these techniques are used for knowledge discovery when the data doesn’t have label. Approaches to unsupervised learning include *clustering* when the goal is to discover groups of similar examples within the data, *density estimation*
to determine the distribution of data within the input space, *dimensionality reduction* for feature extraction and data visualization. A study by Thomas *et al.* [5] shows some of the applications of these techniques in chemical processes.

- **Semi-supervised learning (SSL):** these techniques are used when a part of training data doesn’t have label. They are useful when unlabeled data contains a lot of information or cost of the labeling is expensive or time consuming.

- **Reinforcement learning:** this technique is concerned with the problem of finding best policy (a sequence of actions that maximize the total reward) by a process of trial and error. Here the output of the system is a sequence of actions that are unknown for the learner, in contrast to supervised learning where the outputs are known.

### 1.1 Process Data Analytics Using Machine Learning Techniques

During the past decades, machine learning and data mining techniques have been applied in process industry for different types of data analytics including descriptive analytics (e.g. process monitoring, data visualization, outlier detection, etc.), diagnostic analytics (e.g. process fault diagnosis), predictive analytics (e.g. fault classification, soft sensor modeling, and key performance index prediction), and perspective analytics (e.g. control and decision problems). For example, Figure 1 shows how data-driven modeling methods such as principal component analysis (PCA) and projection to latent structures (PLS) can be used for abnormality detection in the process.

A framework for process data analytics that includes all the necessary steps to build an effective data-driven model is shown in Figure 3. These include data collection, data preparation and preprocessing, and choosing an appropriate machine learning technique. In the first step, data that could be historical and online data are collected from different
resources. In data preparation and data pre-processing step, most appropriate data samples and variables are selected for modeling and different pre-processing techniques including data normalization, removing outliers, and missing value estimation are carried out to improve the quality of the data. In the next step, a machine learning algorithm is selected for data-driven modeling based on the characteristics of the process data and the modeling task. These two steps are highly related to each other for the modeling as the performance of the model is dependent on selected features and the quality of the data. Note that although there are some feature selection and extraction techniques, they all carried out separately and are not embedded in the step of model training.

A list of supervised and unsupervised machine learning techniques is provided in the framework that can be used for different applications. Applications of unsupervised learning algorithms in process industries include process monitoring, data visualization, outlier detection, and dimensionality reduction. Applications of supervised learning algorithms include process monitoring, fault classification, soft sensor, and quality prediction. A very comprehensive review on machine learning techniques and their applications in process industry was provided by Ge et al. [6]. Semi-supervised learning that is a combination of supervised and unsupervised learning can be used when appropriate.

The ability of conventional machine learning techniques to process raw data is limited. They usually need careful engineering and considerable domain expertise to design a feature extractor that transformed the raw data into a suitable internal representation for model construction. This process becomes more complicated with considering the characteristics of the data mentioned above. To overcome these challenges, deep learning techniques have been developed.
Deep learning is a class of machine learning techniques that has been developed based on the representation learning that allows a machine to be fed with raw data and to automatically discover the representations needed for detection or classification [7]. It has gained more attention recently because of the availability of more powerful computers, larger datasets and techniques to train deeper neural networks. In spite of the conventional machine learning methods that use hand-designed features for training the model, deep learning-based methods use a representation learning algorithm to extract the features automatically from the data that result in much better performance.

Figure 3. A framework for process data analytics using machine learning techniques.

1.2 Motivations and Objectives

The goal of this research is two-fold. Firstly, we conduct a review on the recent applications of different deep learning techniques in manufacturing systems. We focus on
recurrent neural networks (RNNs) as they are more appropriate for modeling because of the characteristics of process data generated in the chemical plants. These types of neural networks are very powerful dynamic systems for sequential data and time series modeling. Secondly, we introduce an attention-based RNN for multi-step-ahead prediction that can have many applications in chemical plants such as model predictive control system, fault diagnosis system, process performance prediction, etc. This model consists of an RNN that encodes a sequence of input time series data into a new representation (called context vector) and another RNN that decodes the representation into output target sequence. An attention model integrated to the encoder-decoder RNN model allows the network to focus on parts of the input sequence that are relevant to predicting the target sequence. The attention model is jointly trained with all other components of the model. By having a deep architecture, the model can learn a very complex dynamic system. In order to show the effectiveness of the proposed approach, an application on process performance prediction is presented. A comparative study is performed to show the superior performance of the proposed approach compared to conventional machine learning techniques such as support vector regression (SVR).

1.3 Thesis Outline

In the next chapter, we give a brief introduction about deep learning techniques with more focus on recurrent neural networks. We also review some applications of most common deep learning methods in manufacturing processes. In chapter three, we propose an attention-based recurrent neural networks model for multi-step-ahead prediction that has many potential applications in chemical industry. The model consists of an encoder RNN, attention model, and a decoder RNN that learns a complex mapping between input and output sequences. The
temporal attention mechanism allows the network to select relevant encoder hidden states across all time steps for predicting the target sequence. To demonstrate the effectiveness of the modeling approach, a comparative study on the problem of catalyst activity prediction is illustrated.
CHAPTER 2 DEEP LEARNING: FOCUS ON RECURRENT NEURAL NETWORKS

Deep learning is a class of machine learning techniques that has gained much attention in recent years. In spite of conventional machine learning techniques that require careful engineering and considerable domain expertise to design a feature extractor, deep learning techniques use representation learning methods to extract the relevant features from the raw data and discover the representations needed for model building [7]. In this way, the machine learning algorithm learns not only the mapping from the representation to output but also the representation itself that result in much better performance [8]. Figure 4 shows that how classic machine learning and deep learning that uses representation learning work. Deep learning methods are representation learning methods with multiple levels of representation obtained by transforming the less abstract features into a more abstract representation.

Figure 4. Relation between classic machine learning and representation learning. Shaded boxes show the components that can learn from data.
A deep learning model consists of multiple layers of simple modules that many of which perform a nonlinear mapping. For example, deep feedforward neural network models (also called MLPs) with multiple layers are the most common deep learning models used for many applications. This type of models has several layers including the first layer (also called input layer), hidden layers, and output layer shown in Figure 5. Each hidden layer consists of a set of computational units that are called neurons. Each neuron computes the weighted sum of its inputs from the previous layer and pass it to the next layer after applying a nonlinear function.

![Feedforward neural network](image)

**Figure 5.** Feedforward neural network

Deep feedforward neural networks have more power in function approximation and have higher generalization capacity than the shallow ones (networks with one hidden layer). They can learn a very high nonlinear function with fewer number of parameters and samples to achieve a desired approximation. Mhaskar et al. [9] show that deep networks can approximate the class of compositional functions (e.g. \( f(x_1, x_2, \cdots, x_d) = h_1(h_2 \cdots (h_j(h_{i_1}(x_1, x_2), h_{i_2}(x_3, x_4)), \cdots)) \)) as well as shallow networks but with exponentially lower number of training parameters and sample complexity. A simple example from this study (see Figure 6) shows that MLPs can learn a very
high nonlinear function with much fewer numbers of neurons and parameters. In other study, Liang and Sirkant [10] show that the number of neurons needed by a shallow network to approximate a large class of pricewise smooth functions is exponentially larger than the corresponding number of neurons needed by a deep network for a given degree of function approximation. They show that a multilayer neural network that uses $\Theta \left( \log \left( \frac{1}{\varepsilon} \right) \right)$ layers needs $\mathcal{O} \left( \text{poly} \log \left( \frac{1}{\varepsilon} \right) \right)$ neurons, while $\Omega \left( \text{poly} \left( \frac{1}{\varepsilon} \right) \right)$ neurons are required by neural networks with $o \left( \text{log} \left( \frac{1}{\varepsilon} \right) \right)$ layers for $\varepsilon$-approximation of functions with enough piecewise smoothness.

![Figure 6. MLPs with different number of layers for function approximation (adopted from [9])](image)

Convolutional Neural Networks (CNNs) are another type of deep feedforward network that have gained many applications in different domains especially in computer vision. This type of networks is designed to process data in the array form such as signals (1D arrays), images or audio spectrograms (2D arrays), and videos (3D arrays). They use stacked of
convolution and pooling layers to build high level features from low-level features followed by fully connected layers. They have received many successful applications in image processing, Natural Language Processing (NLP), speech recognition, and drug discovery. Although CNNs have many applications in manufacturing industry, we don’t provide the details of this method as the main focus of this thesis is on Recurrent Neural Networks (RNNs). In the next section, we discuss RNNs in detail and review some of its application in manufacturing industry.

2.1 Recurrent Neural Networks (RNNs)

RNNs are a class of artificial neural network models for processing time series data or data that are generated sequentially. Inspired from cyclical connectivity of neurons in the brain, they could have different architecture including recurrent connections in hidden units or recurrent connections from output to the hidden units. For example, Figure 7(a) shows an RNN with one hidden layer with connections between units in the hidden layer. This model is parametrized with three matrices:

- Weight matrix $W$ for connections between input and the hidden layer with bias vector $b_0$.
- Weight matrix $R$ for recurrent connections in the hidden units.
- Weight matrix $V$ for hidden to output connections with bias vector $b_1$.

These parameters are shared across the time as shown in Figure 7(b) for four-time steps. At time step $t$, the hidden units receive the sum of weighted input $x_t$ and the weighted hidden state $h_{t-1}$, and then pass it to the output layer after a nonlinear mapping $f$. Therefore, an RNN model can be described as a nonlinear dynamical system as follows:

$$h_t = f(h_{t-1}, x_t, \theta) \quad (2.1)$$
\( \hat{y}_t = g(h_t, \theta) \)  \hspace{1cm} (2.2)

where \( \theta \) are the parameters that need to be learned through the training process; \( f \) is a nonlinear function (e.g. hyperbolic tangent activation function); \( g \) is a linear function for regression problems or a sigmoid function following by a SoftMax operation for classification problems. Note that we only show one output in the figure. However, the formulation is general, and the output can be a vector.

Figure 7. Recurrent Neural Network: (a) connections in hidden layer, and (b) equivalent unfolded network through the time for four-time steps.
The parameters $\theta$ are obtained through optimizing an objective function. Different loss functions are used for different applications. For example, the mean squared error (MSE) and mean absolute error (MAE) objective functions are usually used for regression problems.

One solution for training the RNN model is to unroll the recurrent model to its equivalent unrolled graph and the using any gradient-based optimization approach to find optimal values of parameters. However, there are some difficulties such as vanishing and the exploding gradient problems [11,12] with computing gradient in the backpropagation through time (BPTT) algorithm that is very similar to the generalized back-propagation (BP) algorithm. Hopefully, there are some approaches that reduce these difficulties and allow the network to learn long-term dependencies. We discuss some of them in the following.

2.2.1 Gated Recurrent Neural Networks

Gated RNNs are the most effective recurrent models used in practical applications. These networks use special units in their architecture that allows them to create paths through the time that neither vanish nor explode. These units have parameters that allow the network to decide when to update the cell state based on the new information. We discuss two types of gated RNNs, long short-term memory (LSTM) network and networks based on the gated recurrent unit (GRU), in this section.

The LSTM model was proposed by Hochreiter and Schmidhuber [13] as a novel recurrent neural network in 1997. It has received many successful applications in speech recognition, language modeling and translation, image captioning, and time series forecasting. As we mentioned above, the LSTM networks use a special block in their hidden layers shown in Figure 8. The simple recurrent network (SRN) unit is also depicted for showing the differences between computational units in two models. As we can see, the SRN unit has only a nonlinear activation
function with recurrent connections from the output (i.e., hidden state) of the unit to the input of the unit. However, the LSTM block has a cell state that its state is controlled using different parameters. It consists of three gates (input, forget, and output), block input, a single cell, and recurrent connections from output to all gates and block input. The LSTM block is formulated by equations 2.3-2.8 as follows:

\[
\begin{align*}
    z_t & = g(W_z x_t + R_z h_{t-1} + b_z) \quad (2.3) \\
    i_t & = \sigma(W_i x_t + R_i h_{t-1} + b_i) \quad (2.4) \\
    f_t & = \sigma(W_f x_t + R_f h_{t-1} + b_f) \quad (2.5) \\
    c_t & = z_t \odot i_t + c_{t-1} \odot f_t \quad (2.6) \\
    o_t & = \sigma(W_o x_t + R_o y_{t-1} + b_o) \quad (2.7) \\
    h_t & = g(c_t) \odot o_t \quad (2.8)
\end{align*}
\]

where \( \sigma \) is logistic sigmoid activation function; it is used as gate activation function in equations 2.4, 2.5, and 2.7 that formulate the input gate, forget gate, and output gate respectively; hyperbolic tangent \( g(x) = \tanh(x) \) is used as block input and output activation function; equation 2.6 shows the state of the cell at time \( t \); operator \( \odot \) is used for point-wise multiplication.
Gated RNNs that use gated recurrent unit (GRU) in their hidden layers are called GRU RNNs. The GRU unit proposed by Cho et al. [14] is another type of unit that can remember and forget adaptively and update its memory content using reset and update gates. A graphical representation of a GRU is shown in Figure 9. This unit has only two gates, reset gate $r_t$ and update gate $z_t$ to update its memory. The update gate is computed based on the previous hidden state $h_{t-1}$ and the input $x_t$:

$$z_t = \sigma(W_zx_t + U_zh_{t-1})$$  \hspace{1cm} (2.9)

where $\sigma$ is logistic sigmoid function. The forget gate $r_t$ is computed by equation 2.10:

$$r_t = \sigma(W_rx_t + U_ry_{t-1})$$  \hspace{1cm} (2.10)

The new content of the memory $\bar{h}_t$ is computed by equation 2.11:

$$\bar{h}_t = \tanh(Wx_t + r_t \odot Uh_{t-1})$$  \hspace{1cm} (2.11)
where $\odot$ is a point-wise multiplication. Finally, the cell state is updated based on the new content of the cell $\tilde{h}_t$ and cell state at time $t - 1$:

$$ h_t = z_t \tilde{h}_t + (1 - z_t) h_{t-1} $$  \hfill (2.12)

All the parameters including weights and reset gate and update gate are learned through the training process.

![Gated recurrent unit (GRU)](image)

Figure 9. Gated recurrent unit (GRU)

There are some other strategies to reduce the difficulties of learning long-term dependencies such as adding skip connections through time, leaky units, and removing connections that allow the network to learn multiple time scales [8]. There are also other recurrent neural networks such as Nonlinear AutoRegressive models with exogenous (NARX) and Echo-State Network (ESN) that we do not discuss them here. The readers are referred to [15], [16] for more information about these models. In the next section we review the applications of CNNs and RNNs in manufacturing processes.
2.3 Deep learning applications in manufacturing processes

It is known that deep learning techniques have received many successful applications in different domains including computer vision (e.g. object recognition and detection), speech recognition (e.g. speech to word), natural language processing (NLP), drug discovery, etc. In this section, we review the applications of deep learning techniques in manufacturing processes. These include descriptive analytics, diagnostic analytics, and predictive analytics as we mentioned in the first chapter.

One of the applications of deep learning in manufacturing processes is fault detection and diagnosis that is a very important problem in process systems engineering. The models are built based on the process historical data and then used as a classifier to classify different faults. Yu et al. [17] proposed a nonlinear Gaussian belief network fault diagnosis technique for industrial processes. They showed that the proposed three-layer model outperforms the classical techniques such as kernel PCA, statistical pattern analysis (SPA), and Moving Window KPCA. In other study, Wu and Zhao [18] used a deep convolutional neural network for fault diagnosis in chemical processes. Wang et al. [19] introduced a local feature-based gated recurrent unit (LFGRU) networks for machine health monitoring tasks including tool wear prediction, gear box fault diagnosis, and incipient fault diagnosis of rolling element bearings. Haidong et al. [20] proposed a deep autoencoder feature learning model for rotating machinery fault diagnosis. For the same task, Janssens et al. [21] proposed a CNN based approach for automatic bearing fault detection; the proposed model achieves better classification accuracy compared to support vector machine and random forest.

Another application of deep learning technique is predictive analytics in manufacturing systems. Wang et al. [22] proposed a data-driven predictive model based on
deep belief network for material removal rate prediction during chemical mechanical polishing. Wu et al. [23] introduced an approach based on long short-term memory recurrent neural network for remaining useful life estimation of engineered systems.

Deep learning techniques are also used for product quality control and surface inspections. For example, Weimer et al. [24] introduce a CNN based approach for detecting defects on the surface. For the product quality purpose, Wang et al. [25] use a CNN for defect detection on the product surface. In other study, Ren et al. [26] proposed a generic approach based on the CNN for automated surface inspection.

Attention-based neural networks is another type of deep networks that recently received many successful applications in image processing [27], machine translation [28], etc. However, their applications for time series modeling has not been investigated. In the next chapter, we propose an attention-based RNN for multi-step-ahead prediction.
CHAPTER 3 ATTENTION-BASED RNN FOR MULTI-STEP-AHEAD PREDICTION

Building a data-driven model for Multi-step-ahead (MS) prediction of a dynamic system is a challenging problem as the output target need to be predicted many time-steps into future without having the measurements in the horizon of interest. Most of models that proposed for this task use a single-step-ahead (SS) predictor recursively for MS prediction. Using a SS predictor for MS prediction usually leads to a poor prediction accuracy as a small prediction error at the beginning of the horizon is accumulated and propagated into future.

In this chapter, motivated by Cho et al. [14] and Bahdanau et al. [28] works, we propose an attention-based recurrent neural network for multi-step-ahead prediction that can have applications in model predictive controllers, fault diagnosis systems, process performance prediction, etc. This model consists of one RNN encoder that encodes a sequence of input time series into a new representation (called context vector) and one RNN decoder that decodes the representation into output target sequence. The attention model integrated to encoder-decoder RNN model allows the network to focus on parts of the input sequence that are relevant to predicting the target sequence. Because of having a deep architecture, the model can learn a very complex dynamics system and it is robust to noise. First, we give the problem statement and the explain the methodology in detail.

3.1 Problem Statement

For an MS prediction problem, an RNN-based multi-step-ahead predictive model is to be built to predict the target variable, $y$, for $T'$ steps into future, based on the last $T$ observations of the inputs (also called driving series or exogenous inputs) and output observations. The model can be described by equation (3.1):

$$Y = F(X)$$ (3.1)
Where \( Y = (y_{T+1}, y_{T+2}, \ldots, y_{T+T'}) \in \mathbb{R}^{T'} \) is a vector represents a sequence of target variable \( y \in \mathbb{R} \) for the future \( T' \) steps, and \( X = (x^1, x^2, \ldots, x^q, y) \top = (x_1, x_2, \ldots, x_T) \in \mathbb{R}^{(q+1) \times T} \) is a window (i.e., lookback window) contains past \( T \) observations of \( q \) exogenous inputs and one desired output; \( x^k = (x^k_1, x^k_2, \ldots, x^k_T) \in \mathbb{R}^T \) and \( y = (y_1, y_2, \ldots, y_T) \in \mathbb{R}^T \) indicate a driving series and a target series respectively. We employ \( x_j = (x_j^1, x_j^2, \ldots, x_j^q, y_j) \in \mathbb{R}^{q+1} \) to denote a vector of \( q \) exogenous inputs and one desired output \( y \) at time step \( j \) \((1 \leq j \leq T)\). The model uses the last \( T \) observations of the inputs and the desired output to predict the target output \( y \) for next \( T' \) steps.

### 3.2 Attention-Based Encoder-Decoder RNN Model

The architecture of the model is shown in Figure 10. The model consists of an RNN that encodes the input sequence into a new representation (called context vector) and another RNN that decodes the representation into an output target sequence. An attention model integrated to the encoder-decoder RNN model allows the network to focus on parts of the input sequence that are relevant to predicting the target sequence. The attention model is jointly trained with all other components of the model. Each component of the model is described below.

#### 3.2.1 Encoder RNN

The encoder is an RNN that reads input \( x_j \) at time step \( j \) until it reaches to the end of the sequence. At each time step \( t \), the hidden state of the encoder is updated using the equation below:

\[
    h_t = f(h_{t-1}, x_t)
\]

(3.2)

where \( f \) is a nonlinear activation function that its parameters are shared across the time; it can be a simple logistic sigmoid function, LSTM, or GRU explained in the previous chapter. In
this study, we use the LSTM network because of its superior performance in accurately modeling both short- and long-term dependencies in time series data. Hidden state $h_t$ shows the current state of the hidden layer that is a function of the previous hidden state $h_{t-1}$ and input $x_t$. For example, if $t = j$, we have $h_j = f(h_{j-1}, x_j)$ as shown in Figure 10. Note that hidden state $h_t$ is a vector that its size is equal to the size of the hidden layer.

At each time step that $h_t$ is updated, the hidden state contains a summary of the previous time steps. When it reaches to end of the input sequence $X$, the final hidden state $h_T$ is a summary of the whole sequence. The final hidden state $h_T$ that is also called context vector that can be used in two ways in the decoder RNN as we discussed in the previous section; it can be used as the initial state of the decoder or it can be provided as an input to the hidden units of the decoder at each time step.

![Figure 10. Attention-based RNN model for MS prediction.](image-url)
One of the drawbacks of this encoding strategy is that some of the information are lost during the encoding process and only the last hidden state is used in the decoder RNN. An attention mechanism introduced by Bahdanau et al. [28] to address this issue. Here we discuss the RNN decoder and then the attention model is explained.

3.2.2 Decoder RNN

The decoder RNN is another RNN that generate the target sequence $(\hat{y}_{T+1}, \hat{y}_{T+2}, \cdots, \hat{y}_{T+T'})$ sequentially as shown in Figure 10. The hidden state of the decoder is updated as follows:

$$s_i = g(s_{i-1}, \hat{y}_{i-1}, c_i)$$

(3.3)

where $g$ is a nonlinear activation function (e.g. LSTM, or GRU) that its parameters are shared across the time like function $f$. The hidden state of the decoder is updated based on the current state of context vector $c_i$ that comes from the attention model, previous hidden state $s_{i-1}$, and estimated target variable $\hat{y}_{i-1}$ at the last time step $i - 1$. The target variable is estimated using equation (3.4) at each time step:

$$\hat{y}_i = o(s_i)$$

(3.4)

where $o$ is the linear activation function ($o(x) = x$). At each time step $i$ the target is estimated based on the current hidden state $s_i$, previous output $\hat{y}_{T+i-1}$, and current context vector $c_i$. We explain how attention model is used to generate $c_i$ in the next section.

3.2.3 Temporal Attention Model

As we mentioned before, the bottleneck of improving the performance of an RNN sequence to sequence model is using the fixed-length vector $c$. That is, it uses the fixed-length
vector $c$ that is set to the last hidden state of the encoder after processing the input sequence. To address this issue, the attention model derives a context vector $c_i$ at each time step $i$ in the decoding phase to capture relevant input information for predicting $y_i$. As shown in Fig. 10, the attention model receives encoder hidden states $(h_1, h_2, \cdots, h_T)$ and the decoder’s hidden state $s_{i-1}$ to calculate the context vector $c_i$ at time step $i$. The context vector $c_i$ is computed as follows:

$$c_i = \sum_{j=1}^{T} \alpha_{ij} h_j$$

where $\alpha_{ij}$ is the weight of hidden state $h_j$ at time step $i$ during the decoding process. It actually shows the importance of hidden state $h_j$ respect to the previous hidden state $s_{i-1}$ in deciding the next state $s_i$ and predicting $y_{T+i}$. It is computed by equation (3.6).

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T} \exp(e_{ik})}$$

where $e_{ij}$ is computed by the alignment model $e_{ij} = align(s_{i-1}, h_j)$. This model is parametrized as a feedforward neural network that is trained jointly with other components of the model.

At each time step $i$, the scores $e_{ij}$s that show the importance of the hidden state $j$ in predicting the target $y_{T+i}$ are computed using the alignment model and then normalized using the SoftMax function (see equation (3.6)).

### 3.2.4 Computational procedure

The following steps are executed at each time step $i$ during the decoding process after encoding the input sequence to a set of internal states $(h_1, h_2, \cdots, h_T)$:
Step 1. The alignment model that is a feedforward NN computes the scores $e_{ij}$ based on the hidden state of the decoder ($s_{i-1}$), and hidden states ($h_1, h_2, \ldots, h_T$). For example, at time $i = 1$, the scores $(e_{11}, e_{12}, \ldots, e_{1T})$ are evaluated based on the encoder’s hidden states and the initial hidden state $s_0$ of the decoder.

Step 2. The scores are normalized using Eq. (3.6) to give the attention weights $\alpha_{ij}$. For example, we have attention weights vector $\alpha = (\alpha_{11}, \alpha_{12}, \ldots, \alpha_{1T})$ at the first-time step that its elements show the importance of hidden states $h_1, h_2, \ldots, h_T$ in predicting the $y_1$ respectively.

Step 3. Context vector $c_i$ is calculated using equation (3.5). Then, it is used for predicting the target at time step $i$.

Step 4. The context vector, $c_i$, is concatenated with the output from the previous time step.

Step 5. The concatenated vector $(c_i, \hat{y}_{i-1})$ is used as the input to the decoder to predict the output at time step $i$. Note that $y_0$ is used as the start point at the first state, as there is no state before that. The decoder hidden state is also updated based on equation (3.3).

These five steps are executed at each time step sequentially until the end of the target sequence to be reached. For example, the decoder hidden state $s_1$ is used in the attention model to compute new context vector $c_2$ based on the new set of scores and attention weights. The context vector $c_2$ and predicted output $\hat{y}_1$ are concatenated and used as input to the decoder to predict $\hat{y}_2$. This process is repeated until the end of the target sequence.

The application of the model is not limited to MS prediction of single output. It can be used for MS prediction of the system's performance with multiple outputs. In this case, the design of the predictor model is same as that of the proposed model, except that the decoder RNN predicts multiple outputs.
3.3 Loss function and model training

Mean Absolute Error (MAE) is used as a loss function in this study. Since the parameters of the model are updated using a mini-batch of training samples, the objective function is defined below:

\[
J(\theta) = \frac{1}{m} \sum_{k=1}^{m} (MAE)_k
\]

where \(m\) is the batch size and \(\theta\) are the model’s parameters; \(MAE\) is the mean absolute error between the predicted sequence and the target sequence:

\[
MAE = \frac{1}{T'} \sum_{i=1}^{T'} |y_{T+i} - \hat{y}_{T+i}|
\]

where \(y\) and \(\hat{y}\) are the actual (ground truth) and estimated target respectively.

Many optimization algorithms have been proposed for training deep neural network models. All these methods are classified as gradient-based optimization approaches as they use the gradient of the loss function to update the model’s parameters at each step. For example, the well-known stochastic gradient descent (SGD) optimization approach used the following steps to find the optimal value of the model’s parameters:

**Step 1.** A batch of training samples \((x, y)\) are drawn randomly from the dataset. The stochastic term refers to random selection of the samples.

**Step 2.** The network is run on input \(x\) to give predictions \(\hat{y}\)

**Step 3.** The loss of the network on the batch is computed. This step is also called the forward pass.

**Step 4.** The gradient of the loss function respect to model’s parameters is computed using backpropagation algorithm (backward pass)
Step 5. The model’s parameters are updated using equation below:

\[ W_i = W_{i-1} - \eta \nabla F(W) \]  
(3.9)

where \( \eta \) is the learning rate that should not be too small or too large.

There are several variants of SGD (e.g. SGD with momentum, Adagrad, RMSProb, Adam, etc.) that differ in updating the model’s parameter (weights and biases). These optimizers use a concept called momentum that controls the convergence speed of the algorithm and local minima. In this study, we use Adam optimizer.

The detailed procedure of network optimization is presented in Algorithm 1. The model is trained using Adam optimizer Kingma and Ba (2015) that is a mini-batch stochastic gradient descent optimization algorithm. It uses an adaptive approach to compute the learning rates from the estimates of the first and second moments of the gradients (\( m_t \), and \( v_t \) respectively). Hyper-parameters \( \beta_1, \beta_2 \in [0, 1) \) control the exponential decay rates of these moments. Hyper-parameter \( \alpha \) is the step size that is set to 0.001.

Two training procedure, non-teacher forcing (NTF) and teacher forcing (TF), can be used to train the model. In NTF procedure, at each time step of decoding, the decoder RNN uses the previous output prediction \( \hat{y}_{T+i-1} \) to generate the next output prediction \( \hat{y}_{T+i} \) during the train time. However, in TF procedure, the actual output \( y_{T+i-1} \) is fed to the decoder to generate the next output prediction \( \hat{y}_{T+i} \) during the train time. Since the actual output is not available at the test time, the predicted output value is fed back to the network.
**Algorithm 1:** Model training procedure: the model’s parameters are updated until they converge. $g_t^2$ indicates the elementwise square $g_t \odot g_t$. All operations on vectors are elementwise. $\beta_1^t$ and $\beta_2^t$ are $\beta_1$ and $\beta_2$ to the power $t$. Good default settings for the tested machine learning problems are $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \text{and } \epsilon = 10^{-8}$.

Randomly initialize model’s parameters $\theta$

for number of training iterations do

- $t \leftarrow t + 1$
- Sample minibatch of $m$ examples $\{(X_k, y_k)\}_{k=1}^m$
- Update $\theta$ at iteration $t$:
  1. $g_t \leftarrow \nabla_{\theta} \left( J(\theta_{t-1}) \right)$
  2. $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$ (Update biased first moment estimate)
  3. $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ (Update biased second raw moment estimate)
  4. $\hat{m}_t \leftarrow \frac{m_t}{(1-\beta_1^t)}$ (Compute bias-corrected first moment estimate)
  5. $\hat{v}_t \leftarrow \frac{v_t}{(1-\beta_2^t)}$ (Compute bias-corrected second raw moment estimate)
  6. $\theta_t \leftarrow \theta_{t-1} - \frac{\alpha \hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}}$ (Update parameters)

end for

3.4 Summary

In this chapter, we proposed attention-based RNN for multi-step-ahead (MS) prediction. The model uses an attention mechanism to focus on parts of the inputs that most relevant for predicting the output at each time step. In the next chapter, we will discuss the application of the proposed method for the MS prediction of catalyst activation in liquid phase methanol synthesis process. We will discuss the process, dataset in detail. We will also conduct a comparative analysis with other methods for MS prediction.
CHAPTER 4 CASE STUDY: CATALYST ACTIVITY PREDICTION

In this chapter, an application of the proposed methodology for MS prediction of catalyst activity in the liquid phase methanol process is presented. First, the whole process including main process units, flow streams, and measurements is explained briefly. Second, the dataset that was obtained through different experiments on process parameters is discussed. Then, the results of the proposed model for MS prediction of the catalyst activity is discussed. Finally, we have a comparative analysis on the predictive performance of the proposed approach and with Support Vector Regression (SVR).

4.1 Liquid Phase Methanol Process

The simplified process flow diagram (PFD) of the liquid phase methanol synthesis process is shown in Figure 12. The process information and data were obtained from [30]. The process consists of five main sections:

- The feed gas purification unit that includes feed gas supply tie-ins (streams 10, 20, and 30) and a carbonyl guard bed. The stream 20 (H₂ Gas) is used to adjust the H₂/CO ratio.
- Compression section that includes the recycle gas compressor (C-1) and associated systems.
- The reactor and catalyst reduction unit that includes a slurry reactor and associated equipment for catalyst preparation and handling.
- The distillation unit that comprises of two distillation columns to reach the refined-grade methanol.
- The storage and miscellaneous section that includes storage tanks with other components.
Unlike conventional gas-phase reactors that use fixed beds of catalyst pellets, the liquid phase methanol reactor is a slurry reactor with powder-size catalyst particles suspended in mineral oil. The syngas (reactants) is entered the reactor through a gas sparger and then bubbles up through the slurry where the reactants (CO and H₂) dissolve in the oil and diffuse to the catalyst surface. Three highly exothermic reactions occur on the catalyst surface:

\[ CO + 2H_2 \rightarrow CH_3OH \] (4.1)
\[ CO_2 + 3H_2 \rightarrow CH_3OH + H_2O \]  \hspace{1cm} (4.2)

\[ CO + H_2O \rightarrow CO_2 + H_2 \]  \hspace{1cm} (4.3)

An internal heat exchanger is designed in the reactor to remove the heat of reaction and control the reactor temperature that is very important to achieve optimum catalyst life and reaction rate. Note that excessive temperatures reduce the catalyst life seriously. The recovered head is used for steam generation. A simple schematic of the slurry reactor is shown in Figure 12.

The product gas leaving from top of the reactor (Stream 120) is cooled in the feed/product heat exchanger, and any condensed oil is collected in the high-pressure oil separator and returned to the reactor. The product gas is further cooled to condense the methanol product so it can be separated in the methanol separator. Part of the unreacted syngas is recycled to the reactor through stream 149 and the rest is sent to boiler.

The condensed methanol (Stream 204) is then sent to the distillation unit for removal of higher alcohols, water, and other impurities. The recovered refined grade methanol from this unit is sent to the storage tanks.

![Schematic of LPMEOH slurry reactor](image.png)

Figure 12. A simple schematic of LPMEOH slurry reactor.
4.3 Data Acquisition

The process parameters (temperature, pressure, flowrate, etc.) are measured using the control and instrumentation systems located in the plant. The real-time plant data are accessible by a distributed control system (DCS) that is used for automatic control, monitoring, etc. In order to calculate some of the key parameters of the process, gas chromatographs were used to analyze the syngas feed streams, the streams entering and exiting the reactor, purge streams leaving the reactor loop and distillation columns, and also the methanol streams exiting the distillation unit. To study the catalyst activation level, seventeen key process parameters are considered in the commercial-scale demonstration of the liquid phase methanol process. Some of the parameters are measured directly from the sensor located in the plant (e.g. reactor temperature and pressure) while some of them are calculated indirectly from directly measured variables.

4.4 Dataset

According to Heydorn et al. (2003), in order to study catalyst performance in process operation, four catalyst campaigns were made. In each campaign, a number of experiments were conducted, and real-time plant data of 17 process parameters were collected (see Table 1). In the report by Heydorn et al. (2003), a daily average of each parameter for the plant operation of 69 months is included. The definition of each parameter is provided below. The details of each campaign is given in Appendix A.

1. Reactor temperature (°C): this parameter shows the reactor temperature during the operation.
2. Reactor pressure (psig): this parameter shows the reactor pressure during the operation.
3. Fresh Feed (KSCFH): this parameter indicates the flow rate of the fresh feed that enters the reactor. It is sum of the Balanced Feed Gas, \( H_2 \) Feed Gas, and \( CO \) Feed Gas flow rates.
4. *Recycle Gas* (KSCFH): this parameter shows the flow rate of the recycle stream.

5. *Reactor Feed H₂/CO ratio*: this parameter specifies the ratio of the $H_2$ over $CO$ at the inlet (stream 109) of the reactor. It is adjusted using the Balanced Feed Gas, $H_2$ Feed Gas, and $CO$ Feed Gas streams.

6. *Purge Gas* (KSCFH): this parameter shows the flow rate of the purge gas.

7. *Inlet Superficial Velocity* (ft/s): The ratio of the actual cubic feet of gas at the reactor inlet (calculated at the reactor temperature and pressure) to the reactor cross-sectional area (excluding the area contribution by the internal heat exchanger):

$$Inlet\ Superficial\ Velocity = \frac{Reactor\ Feed\ Flow\ (Ibmole/\text{hr}) \times v \times \left(\frac{ft^3}{Ibmole}\right)}{3600 \times \left(\frac{s}{hr}\right) \times A \times (ft^2)}$$ \hspace{1cm} (4.4)

where $v$ is the molar volume of the reactor feed that is calculated at the reactor temperature and pressure operating condition.

8. *Space Velocity* (l/hr-kg): this parameter is defined as the ratio of the volumetric flow rate of the reactants to the catalyst weight.

9. *Slurry Concentration* (wt%): this parameter is the percentage of weight of slurry (solid plus liquid) which is catalyst.

10. *Gassed Slurry Height* (ft): this parameter shows the height of gassed slurry in the reactor.

11. *Gas Holdup*: The percentage of reactor volume up to the *Gassed Slurry Height* which is gas.

12. *Catalyst Inventory* (Ib): this parameter indicates the amount of catalyst in the reactor.

13. *CO Conversion to Methanol* (%): This parameter shows the percentage of $CO$ consumed across the reactor.
14. **Overall Conversion (%)**: Percentage of energy (on a lower heating value (LHV) basis) in the reactor feed converted to methanol. It is calculated using equation below:

\[
Reactor\ O - T - M\ Conversion\ (%) = \frac{LHV\ of\ raw\ methanol}{HHV\ of\ reactor\ feed} \times 100
\]  

(4.5)

15. **Syngas Utilization** (SCF/lb Methanol): this parameter is defined as the number of standard cubic feet of Balanced Feed Gas plus CO Feed Gas to the reactor required to produce one pound of raw methanol

16. **Raw Methanol Production** (TPD): this parameter is sum of the refined grade methanol and crude grade methanol.

---

**Table 1. List of key process parameters.**

<table>
<thead>
<tr>
<th>No.</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Reactor temperature (°C)</td>
</tr>
<tr>
<td>2</td>
<td>Reactor pressure (psig)</td>
</tr>
<tr>
<td>3</td>
<td>Fresh feed flow rate (KSCFH)</td>
</tr>
<tr>
<td>4</td>
<td>Recycle stream flow rate (KSCFH)</td>
</tr>
<tr>
<td>5</td>
<td>Ratio of H2 over CO at the reactor inlet</td>
</tr>
<tr>
<td>6</td>
<td>Purge gas flow rate (KSCFH)</td>
</tr>
<tr>
<td>7</td>
<td>Superficial velocity at the reactor inlet (ft/s)</td>
</tr>
<tr>
<td>8</td>
<td>Space velocity (1/hr-kg)</td>
</tr>
<tr>
<td>9</td>
<td>Slurry concentration (mass fraction)</td>
</tr>
<tr>
<td>10</td>
<td>Gassed slurry height (ft)</td>
</tr>
<tr>
<td>11</td>
<td>Gas holdup (%)</td>
</tr>
<tr>
<td>12</td>
<td>Catalyst inventory (lb)</td>
</tr>
<tr>
<td>13</td>
<td>CO conversion to methanol (%)</td>
</tr>
<tr>
<td>14</td>
<td>Overall conversion (%)</td>
</tr>
<tr>
<td>15</td>
<td>Syngas utilization (SCF/lb methanol)</td>
</tr>
<tr>
<td>16</td>
<td>Raw methanol production (tons/day)</td>
</tr>
<tr>
<td>17</td>
<td>Catalyst activity</td>
</tr>
</tbody>
</table>

*Catalyst activity* is a desired target that is to be predicted. It is defined as the ratio of the rate constant at any point in time to the rate constant of a freshly reduced catalyst, i.e.,

\[
\eta = \frac{k_0(t)}{k_0(t=0)}
\]  

(4.6)
The available datasets need to be preprocessed as some parameters contain missing values, outliers, etc., such as the time-series data of three parameters during the first campaign plotted in Figure 13, where the data for days 26 to 67 are missing, and some outliers show in Fig. 13(c).

(a) Catalyst/Activity

(b) Raw methanol production (tons/day)
4.4 Data preprocessing and preparation

Data preprocessing is one of the most important steps in data mining. It involves handling missing values, data normalization, etc. As shown in Fig. 4, there are many missing values in the accessible dataset that need to be filled out with some appropriate technique, such as the linear interpolation, spline, or polynomial technique. In this study, the linear interpolation technique is used because of the non-seasonal characteristic of the process time series data. Note that a non-seasonal time series consists of a trend component and an irregular component (random fluctuations). Data normalization is performed when the features in the input data have values in different ranges. Each feature is scaled independently into a small range. In this study, the min-max normalization is adopted to scale the range of each feature in [0, 1].

After data preprocessing, training and test samples should be selected from the dataset. As stated before, the encoder processes an input sequence of length $T$ and the decoder generates the desired output sequence of length $T'$ (prediction horizon). In this study, the training and test samples are generated using a sliding window approach, by which the window slides right by one
time step. Figure 14 shows two consecutive samples generated using this approach with different colors. For each sample, the input is a matrix of size \((q + 1) \times T\) (lookback window) and the desired output \(y\) (blue curve) is a one dimensional vector with size of \(T'\). Note that \(q\) is the number of the driving series.

In general, the input of an RNN is a 3D array of shape (batch size, length of the input, input dimension). The first, second, and third dimension respectively are number of samples in each batch, input length, and the number of features. In this study, the prediction horizon is set to 30 days (i.e., \(T' = 30\)) in this application. Among the four campaigns, the data from three campaigns are used for model training and validation. The model is trained using the Adam algorithm. The most common evaluation metric, mean absolute error (MAE) is used to evaluate the performance of the model. The validated model will be used to predict the catalyst activity, and the prediction result will be compared with the data from the other campaign.

![Figure 14. A sliding window approach for data sampling.](image-url)
4.5 Results and discussion

In model development, the following four parameters should be determined: the length of the lookback window \((T)\), the number of hidden layers of encoder and decoder, the size of each hidden layer, and the size of the attention layer. The experiments on the number of hidden layers show that the model with two hidden layers in the encoder (denoted as \(n_1\) and \(n_2\)) and one hidden layer in the decoder (denoted by \(n_3\)) has the best performance. Assuming all hidden layers have the same size \((n_1 = n_2 = n_3)\), it is found that in the search for size among 5, 10, 15, 20, and 25, size of 10 shows the best performance on the validation set as shown in Fig. (15). A similar search for the size of the attention layer also gives 10 as the best. Note that the model is trained using 85\% of the data from three campaigns, and the rest 15\% of the data are used to validate the model, during which the model parameters are fine tuned.

Figure 15. The effect of hidden size on model performance.
To study the impact of the lookback window length \(T\), a grid search is conducted among five options for \(T\) (i.e., 10, 20, 30, 40 and 50). The averages of MAE performance of the model on the validations set are shown in Fig. (16). Obviously, the length of 30 is the best in terms of MAE.

![Graph showing the effect of lookback window length on model performance.](image)

Figure 16. The effect of lookback window length on model performance.

### 4.5.1 Ablation study

An ablation study is conducted to investigate the effectiveness of the incorporation of attention mechanism in the encoder-decoder RNN in multi-step-ahead prediction. The effect of two different training procedures, teacher forcing and non-teacher forcing, is also studied. In this study, the attention layer is removed from the attention-based RNN model. This reduces our model (Attention-based RNN) to an Encoder-Decoder RNN model that composed of an encoder RNN and a decoder RNN only. The encoder with two hidden layers (size of 10) reads the input sequence of length \(T\), and the decoder with one hidden layer (size of 10) uses that last hidden state to generate the target sequence with length \(T'\). The model is trained using either teacher forcing (TF) or non-teacher forcing (NTF) procedure discussed in section 2.3.
The results of the ablation are summarized in Table 2. The results show that attention-based RNN outperforms Encoder-Decoder RNN in terms of MAE indicating that incorporation of attention model into the Encoder-Decoder RNN model improves the prediction performance significantly either using TF or NTF training procedures. This is because of that the attention-based RNN model is capable to select the relevant hidden states across all time steps in the encoder while Encoder-Decoder RNN model uses only the last hidden state. The network training time results are also summarized in the table showing that the training time of attention-based RNN is comparable to the training time of the network without attention. Furthermore, it is shown that the teacher forcing training procedure is more effective in both prediction accuracy and network training time.

<table>
<thead>
<tr>
<th>Model</th>
<th>MAE</th>
<th>Training Time (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoder-Decoder RNN-NTF</td>
<td>0.071</td>
<td>8.6</td>
</tr>
<tr>
<td>Encoder-Decoder RNN-TF</td>
<td>0.056</td>
<td>4.8</td>
</tr>
<tr>
<td>Attention-based RNN-NTF</td>
<td>0.043</td>
<td>9.6</td>
</tr>
<tr>
<td>Attention-based RNN-TF</td>
<td><strong>0.039</strong></td>
<td><strong>5.7</strong></td>
</tr>
</tbody>
</table>

### 4.5.2 Comparative Analysis

In this section, a comparative analysis between our approach and support vector regression (SVR) is performed. SVR is a machine learning technique developed by Vapnik [31] based on statistical learning theory. This method has been used for time series modeling for many years and received many successful applications in different domains such as financial market prediction, electricity utility forecasting, etc. [32]. The details of the methodology are not discussed here. The readers are directed to the excellent tutorial on SVR authored by Smola and Schölkopf (2004).
In this study, a direct strategy is used to build an SVR model for MS prediction. Direct strategy learns $T'$ different forecasting models between the inputs $X$ and the $T'$ outputs:

$$y_{t+k} = \varphi_k(X, \theta_k) + e_k$$

where $k = 1, \ldots, T'$. Each $\varphi_k$ is a support vector regression model that predicts the $k$-th output in the horizon. We call this as multi-output SVR (M-SVR). The comparative results between our method and M-SVR are shown in Table 3. The results show that the performance of our model for MS prediction is significantly better than M-SVR.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Attention-based RNN-TF</th>
<th>M-SVR</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAE</td>
<td>0.039</td>
<td>0.117</td>
</tr>
</tbody>
</table>

4.5.3 Model’s prediction performance on test data

The first campaign dataset that includes 205 days of plant’s operating data is selected as the test dataset. The plots of three parameters including catalyst activity during this campaign was shown in Fig.4. The test samples are generated using the sliding window approach discussed in section 3.3. The trained model uses 30 days of the plant’s history to predict the catalyst activity for 30 days in future. Two challenging test samples in terms of prediction, $75^{th}$ sample that include the plant’s history between days 77 and 106 and $95^{th}$ sample that include the plant’s history between days 97 and 126, are selected to test the model. The model’s predictions for these two test samples are shown in Fig. 8 (a, b), in which the black, red, and green curves show the history, actual values, and model’s prediction of the catalyst activity respectively. It is clearly seen that attention-based RNN predicts the catalyst activity for the next 30 days with a high accuracy.
In application mode, the trained model can be used for prediction of catalyst activity in future after the last measurements. To show this, the last 30 days (i.e., days 176 to 205) is fed to the model to predict the next 30 days that no data is available for catalyst activity. The model’s prediction of catalyst activity for 30-day in future (days 206 to 235) is shown in Fig. (9). It is seen that the model can extrapolate the learned pattern into the future very well.
Figure 9. Model’s application for MS prediction of catalyst activity.

The prediction performance of the model is highly dependent on the length of the prediction horizon. To study this, five different length of the prediction horizon (i.e., $T' = 1, 5, 10, 20, 30$) are considered in our experiment. The prediction performance of the trained model in terms of MAE for each horizon is shown in Fig. (10). The results show that the model has smaller prediction error when the prediction horizon is shorter. For example, the MAE for single step prediction is 0.005 that is significantly better than MAE for 30-step-ahead prediction.

Figure 9. Effect of prediction horizon length on model’s prediction performance.
4.6 Summary

In this chapter, we showed the application of the proposed approach for multistep-ahead prediction of catalyst activity in liquid phase methanol synthesis process. The hyper parameters of the model were tuned using a grid search approach. The ablation study showed that incorporation of temporal attention mechanism to basic encoder-decoder RNN can improve the MS prediction significantly. The results also showed that teacher forcing training approach is more effective than non-teacher forcing training approach.
CHAPTER 5 CONCLUSION AND FUTURE STUDIES

5.1 Conclusion

In today’s manufacturing plants, huge amount of process operational data (historical or real-time) are accessible that can be used for optimizing the operations. In the era of big data, development of data-driven predictive modeling for MS prediction of process performance is desirable for many applications in process industry. Inspired by attention-based recurrent neural networks originally developed for neural machine translation, we proposed a novel MS predictive modeling approach with many potential applications in process industries including predictive maintenance, fault diagnosis, process performance prediction, etc. The proposed model consists of an RNN that encodes a sequence of input time series data into a new representation (called context vector) and another RNN that decodes the representation into output target sequence. The attention mechanism incorporated in the model allows the network to attend on parts of the input sequence that are relevant to predicting the target sequence that leads to significantly better MS prediction performance. By having a deep architecture, the model can learn a very complex dynamic system, and it is robust to noise. The proposed model is applied for multi-step-ahead prediction of the catalyst activity in liquid phase methanol process. The comparative analysis showed its superior performance over conventional machine learning techniques.

5.2 Future studies

In this research we propose an attention-based RNN for multistep-ahead prediction. In spite of our contribution in introducing this type of model for MS prediction with many potential applications in chemical processes, there exist some opportunities for future studies:

1. We showed the application of the model on process performance prediction. Other applications of the proposed model in different fields can be investigated.
2. Although the relationship between input driving series is considered implicitly in model building and training, but it is not known which one is more important at each time step. The model can be extended to include a spatial attention mechanism to address this issue.
APPENDIX

Catalyst Campaigns

Four catalyst campaigns were considered to study the catalyst performance during the plant operation. The details of each campaign is given in Table 4. In the first campaign, a batch-wise activation method including addition of fresh slurry to the reactor during the operation was conducted to maintain catalyst productivity in the reactor. In the second campaign, the activation method is same as the catalyst campaign 1 along with withdrawal of spent catalyst slurry from the reactor. An in-situ activation method was used in the third and fourth catalyst campaigns. To maintain the catalyst productivity, temperature programming was used during the third catalyst campaign.

Table 4. LPMEOH operating program campaigns.

<table>
<thead>
<tr>
<th>Campaign number</th>
<th>Run start date</th>
<th>Run end date</th>
<th>Days on stream</th>
<th>Activation method</th>
<th>Productivity control method</th>
<th>Operating temperature (°C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4/6/97</td>
<td>11/3/97</td>
<td>210</td>
<td>Batch</td>
<td>Catalyst addition only</td>
<td>250 to 260</td>
</tr>
<tr>
<td>2</td>
<td>12/20/97</td>
<td>08/06/01</td>
<td>1325</td>
<td>Batch</td>
<td>Catalyst withdrawal/addition</td>
<td>235</td>
</tr>
<tr>
<td>3</td>
<td>8/24/01</td>
<td>6/4/02</td>
<td>284</td>
<td>In-situ</td>
<td>Temperature programming</td>
<td>216 to 242</td>
</tr>
<tr>
<td>4</td>
<td>6/28/02</td>
<td>12/31/02</td>
<td>187</td>
<td>In-situ</td>
<td>-</td>
<td>215</td>
</tr>
</tbody>
</table>

The performance of the catalyst during the first campaign is shown in Figure 17. The reported data for this catalyst campaign includes some missing values as we can see in the figure (e.g. from 5/6/1997 to 6/18/1997). Six batches of 2300 pounds of fresh catalyst was activated and added to the reactor on 4 July, 11 July, 23 July, 12 September, 17 September, and 03 October.
1997. The changes in catalyst inventory of the reactor is shown in Figure 18. We can see fluctuations in catalyst activity/age in Figure 17 because of addition of fresh catalyst to the reactor.

Figure 17. Catalyst performance during the first campaign.

Figure 18. Catalyst inventory during the first campaign.
The second catalyst campaign started in December of 1997 with only a partial charge of 19500 pounds of activated fresh catalyst to the reactor. The catalyst performance during the second operating program is shown in Figure 19. Fresh catalyst was activated and added to the reactor during the first period of the campaign (from 12/20/97 to 01/29/99) as shown in Figure 20. The performance of the catalyst was also examined by changing the reactor temperature. More information about these experiments can be found in the report. In the second portion of the campaign, the catalyst withdrawal and addition productivity control method was used. The fluctuations in the catalyst activity performance because of these experiments can be seen in Figure 19.

![Catalyst performance during the second campaign](image)

Figure 19. Catalyst performance during the second campaign
In spite of batch activation method used in the first and second campaign, in-situ activation method was used in the third and fourth campaigns. A full charge of fresh catalyst (41580 pounds) was added to the reactor and activated only one time in the third campaign. The change in catalyst performance during this campaign is shown in Figure 21. Temperature programming was used as productivity control method during this campaign. The reactor temperature was increased during the operating period as shown in Figure 22; the reactor temperature ranged from 216°C to 242°C during this campaign.

The same activation approach (in-situ) was undertaken in the fourth campaign started on June of 2002. The performance results are presented in Figure 23. The reactor temperature and pressure were set on 215°C and 450 psig respectively during this campaign. The dataset contains many missing values as shown in the figure.
Figure 21. Catalyst performance during the third campaign.

Figure 22. Reactor temperature during the third campaign.
Figure 23. Catalyst performance during the fourth campaign.
REFERENCES


ABSTRACT

PROCESS DATA ANALYTICS USING DEEP LEARNING TECHNIQUES

by

MAJID MORADI ALIABADI

May 2020

Advisor: Dr. Ming Dong

Co-advisor: Dr. Yinlun Huang

Degree: Master of Science

In chemical manufacturing plants, numerous types of data are accessible, which could be process operational data (historical or real-time), process design and product quality data, economic and environmental (including process safety, waste emission and health impact) data. Effective knowledge extraction from raw data has always been a very challenging task, especially the data needed for a type of study is huge. Other characteristics of process data such as noise, dynamics, and highly correlated process parameters make this more challenging.

In this study, we introduce an attention-based RNN for multi-step-ahead prediction that can have applications in model predictive control, fault diagnosis, etc. This model consists of an RNN that encodes a sequence of input time series data into a new representation (called context vector) and another RNN that decodes the representation into output target sequence. An attention model integrated to the encoder-decoder RNN model allows the network to focus on parts of the input sequence that are relevant to predicting the target sequence. The attention model is jointly trained with all other components of the model. By having a deep architecture, the model can learn a very complex dynamic system, and it is robust to noise. In order to show the effectiveness of the
proposed approach, we perform a comparative study on the problem of catalyst activity prediction, by using conventional machine learning techniques such as Support Vector Regression (SVR).
AUTOBIOGRAPHICAL STATEMENT

Majid Moradi Aliabadi is a Ph.D candidate in Chemical Engineering and Material Science Department at Wayne State (WSU) University, Detroit, Michigan. He received his master’s degree in Chemical Engineering from Sharif University of technology, Tehran, Iran in 2011. He started his second master’s program in Computer Science at WSU in 2017. He has a bachelors’ degree in Chemical Engineering from AmirKabir University of Technology, Tehran, Iran in 2008. His major research interests include Sustainable manufacturing, Smart Manufacturing, and Machine Learning. His papers have been published in top conferences and journals and received different prestigious awards.