Applications of deep learning and time-series analysis in dynamic process optimization and decision making

by

Amirsalar Bagheri

B.S. , Amirkabir University of Technology, 2020

A THESIS

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Chemical Engineering
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2024

Approved by:

Major Professor
Dr.Davood B.Pourkargar

Copyright

Amirsalar Bagheri

2024

Abstract

Deep learning (DL), a subset of Artificial Intelligence (AI), has introduced promising methods for modeling, dynamic optimization, and advanced control of complex chemical processes. More advanced DL approaches, such as Recurrent Neural Networks (RNNs), offered a new perspective in modeling process dynamics by treating the modeling problem as a time-series prediction task, allowing RNNs to learn and find temporal patterns in process data. Several types of RNNs have been proven to be efficient in learning the temporal correlations in chemical process dynamics. For instance, Long Short-Term Memory (LSTM) networks, capable of handling long data sequences, offer a robust time-series modeling of process data.

The application of DL methods is not only limited to data-driven system identification but also are useful in complex multi-scale model reduction. In multi-scale modeling, the thermodynamic and kinetical parameters of the system are computed through non-continuum modeling. Despite the efficiency of this approach, their high computational demands prevent their applicability for real-time decision-making.

In this thesis, we present two distinct case studies to demonstrate the practicality of deep learning in facilitating real-time decision-making. We aim to highlight the effective application of DL methodologies in addressing large-scale challenges characterized by complex nonlinear dynamics and stochastic behaviors. Through these investigations, the case studies illuminate the versatility and efficacy of deep learning techniques in navigating and solving complex problems within dynamic environments.

The first case study is focused on the critical aspects and importance of ammonia and the need to decarbonize its production. This has motivated us to develop a real-time dynamic optimization and control framework based on deep learning for an industrial ammonia synthesis packed-bed reactor aimed at sustainable green ammonia production.

The second case study is focused on the applicability of DL methods for the dynamic modeling of soil water content. Precise dynamic predictions of soil water content are key to managing the harmful effects of global warming and facilitating real-time agricultural decision-making. In this study, we present a time-series data-driven perspective for predictive modeling of soil water content that outperforms existing soil physics-based models and is practical for real-time agricultural decision-making.

Table of Contents

Table of Contents v

List of Figures ix

List of Tables xi

Acknowledgements xi

Dedication xii

1 Introduction 1

1.1 AI-driven research in process systems engineering 1

1.2 Problem statement 4

1.2.1 Case Study 1: Real-time dynamic optimization and control of an industrial ammonia synthesis reactor 5

1.2.2 Case Study 2: Time-series modeling of soil water content for agricultural decision making 7

2 Methods and background 9

2.1 Machine learning workflow 9

2.1.1 Problem definition 10

2.1.2 Data Collection 11

2.1.3 Data Preprocessing 11

2.1.4 Model selection 12

2.1.5 Training . . . 12
2.1.6 Validation and Hyperparameter tuning . . . 14
2.1.7 Evaluations . . . 14
2.2 Artificial Neural Networks (ANNs) . . . 15
2.2.1 Feed Forward Neural Networks (FNNs) . . . 15
2.2.2 Recurrent Neural Networks (RNNs) . . . 17
2.2.3 Encoder-decoder architecture . . . 22
2.2.4 Attention Mechanism . . . 22
2.2.5 Transformers . . . 24
2.2.6 Model evaluations . . . 26
2.3 Model Predictive Control (MPC) . . . 29
2.4 Time-series system identification and real-time decision making . . . 30
2.4.1 FNN-based process system modeling . . . 30
2.4.2 RNN-based process system modeling . . . 31
2.4.3 LSTMs and GRUs in process control . . . 32
2.4.4 Transformers in process control . . . 33

3 LSTM-based system identification on ammonia synthesis process 35

3.1 Ammonia production . . . 36
3.2 Multiscale modeling of ammonia synthesis packed-bed reactor . . . 36
3.3 Problem Statement . . . 37
3.4 Data generation . . . 38
3.5 Irregular time-series and bimodal data distribution . . . 39
3.6 Gappy data regenerator . . . 43
3.6.1 GDR training . . . 43
3.6.2 Regularizing simulation data . . . 43

3.6.3 GDR evaluations . . . 44
3.7 LSTM surrogate model training . . . 46
3.7.1 Network Architecture . . . 46
3.7.2 Single-step and multi-step ahead model configurations . . . 47
3.7.3 Training . . . 48
3.8 Surrogate model performance evaluations . . . 49
3.8.1 Single-step ahead models . . . 49
3.8.2 Multi-step ahead model . . . 50
3.8.3 Parity plots . . . 51
3.8.4 Residual plots . . . 51
3.9 LSTM-MPC framework . . . 53
3.10 Conclusion and future work . . . 56

4 Time-series modeling of soil water content for agricultural decision making 58

4.1 Problem Statement . . . 59
4.1.1 Richards Equation . . . 59
4.1.2 Markov-based soil water content estimation . . . 60
4.1.3 ML-based SWC estimation . . . 62
4.1.4 Proposed solution . . . 62
4.2 Data-driven LSTM-based time-series modeling of SWC . . . 63
4.2.1 Time-series SWC dataset . . . 63
4.2.2 Data Preprocessing and feature engineering . . . 63
4.2.3 Network Architecture . . . 64
4.2.4 Training . . . 65
4.3 Models Evaluations . . . 66

4.3.1 Overall performance metrics 66
4.3.2 Parity plot comparison 67
4.3.3 Residual plot comparison 69
4.3.4 Time-series prediction visualization 69
4.4 Conclusion and future work 69

Bibliography **72**

List of Figures

2.1 Feed Forward neural Network (FNN) 16

2.2 RNN structure 18

2.3 An LSTM Cell 19

2.4 A GRU Cell 21

3.1 Schematics of Haber-Bosch process inside a PBR 37

3.2 Outlet Ammonia concentration values in fixed F = 600 $\frac{L}{min}$ 40

3.3 Maximum Temperature values in fixed F = 600 $\frac{L}{min}$ 40

3.4 Outlet Ammonia concentration values in fixed Q = 20 $\frac{kW}{m^2}$ 41

3.5 Maximum Temperature values in fixed Q = 20 $\frac{kW}{m^2}$ 41

3.6 Bimodal distribution of the control objectives in the ammonia production reactor. 42

3.7 Regenerated distributions of the control objectives in the ammonia reactor. . 44

3.8 Parity plots of GDR model predictions. 45

3.9 Residual plots of GDR model predictions. 46

3.10 Encoder-decoder architecture with LSTM and fully connected layers for multi-step time-series prediction 47

3.11 Time-series sequence with memory and prediction horizon 47

3.12 LSTM models parity plots for $T_{\max} - T_{\max,\,ss}$ 52

3.13 LSTM models parity plots for $C_{NH_3} - C_{NH_3,ss}$. 52

3.14 LSTM models residual plots for $T_{\max} - T_{\max,\,ss}$ 53

3.15 LSTM models residual plots for $C_{NH_3} - C_{NH_3,ss}$ 53

3.16 Multi-platform computational framework developed to control the ammonia reactor in real-time. 54
3.17 Closed-loop simulation with single-step LSTM model 55
3.18 Closed-loop simulation with multi-step LSTM model 55

4.1 Parity plots for each SWC predictive model 68
4.2 Residual plots for each SWC predictive model 68
4.3 Time-seris plot for Butler stations 70
4.4 Time-seris plot for Gypsum stations 70

List of Tables

3.1 GDR Performance Metrics for System Outputs 44

3.2 Model Parameters based on memory length and prediction horizon 48

3.3 Single-step ahead LSTM-based model performance metrics for $T_{\max} - T_{\max,\ ss}$ 49

3.4 Single-step ahead LSTM-based model performance metrics for $C_{NH_3} - C_{NH_3,ss}$ 49

3.5 Multi-step ahead LSTM-based model performance metrics for $T_{\max} - T_{\max,\ ss}$ 50

3.6 Multi-step ahead LSTM model performance metrics for $C_{NH_3} - C_{NH_3,ss}$ . . . 50

4.1 SWC LSTM models' configuration 65

4.2 Comparison of Model Performances 66

Acknowledgments

I would like to express my sincere gratitude to my advisor, Dr. Pourkargar, for his invaluable guidance and unwavering support throughout this research. His insights not only illuminated my academic career but also helped in shaping this thesis.

Special thanks to Dr. Antony, head of the Chemical Engineering Department, and Dr. Amama, the Graduate Program Director, as well as Dr. Caragea from the Computer Science Department, who are also members of my committee. Their expert advice, constructive feedback, and encouragement have been crucial throughout this journey.

I am profoundly grateful to my peers and colleagues in our research team, particularly Thiago O. Cabral, Carlos Velez, and Amir Mohammad Ebrahimi, whose collaborations and encouragement helped me overcome many challenges.

I would like to assert my special acknowledgment towards my colleague, Thiago O. Cabral, for his collaboration and guidance. His expertise in reaction engineering and advanced process system modeling was instrumental in establishing the foundation of this thesis.

I must acknowledge the support provided by Kansas State University through various stages of my studies. Their facilities and resources have been indispensable to my research.

Finally, I would like to thank my family for their love, patience, and understanding throughout my academic journey. Their support has been a cornerstone of my resilience and success.

This thesis would not have been possible without the contributions and commitment of all those mentioned above. I am deeply thankful to each one of them.

Dedication

This thesis is dedicated to my dear friend, Mehdi Miraj, a diligent and bright civil engineering graduate student at Kansas State University, whose presence is deeply missed. Mehdi's hard work, brilliance, and youthful spirit were an inspiration to all who knew him. Though he is no longer with us, his memory and the impact he made continue to light our paths. In honor of his enduring spirit, this work is a tribute to his memory.

Chapter 1

Introduction

1.1 AI-driven research in process systems engineering

Artificial Intelligence (AI) has revolutionized how computers solve problems by enabling them to learn and making them more competent over time. At the heart of AI is machine learning (ML), where computers learn patterns from data to make better decisions[1]. Deep learning (DL) goes even more profound, using layers of calculations to understand complex patterns, much like the human brain[2]. DL's inherent data parallelism makes it agile, fast, and suitable for real-time decision-making problems. Despite the algorithmic innovations, DL methods need high computational powers to find and learn deep patterns in a large dataset. The improvements and innovations in computer hardware, like GPUs and TPUs, have pushed DL forward and made it possible to train big and complex models quickly, opening up new possibilities for real-time decision-making scenarios[3].

These promising innovations have revolutionized numerous industries, especially process systems engineering (PSE). As a discipline deeply rooted in complex processes, reactions, and multifaceted systems, PSE presents a vast array of challenges that can benefit from the learning features of AI, especially DL. These methods are broadly applied for fault diagnosis[4], molecular thermodynamics[5], process design[6], and process control[7]. These benefits

accelerate research and development and pave the way for groundbreaking advancements previously considered unattainable.

The early stages of AI emergence in addressing PSE problems started in the 1980s. The primary focus was on the development and implementation of specialized expert systems. Expert systems, characterized by their knowledge-based or rule-based methodologies, were designed to emulate human problem-solving capabilities within computers. The ability of expert systems in flexible execution of traversing the knowledge-based backward and forward chaining[8] algorithms was used in reaction design and chemical synthesis. Such methodologies led to the development of MYCIN[9], a diagnostic expert system for infectious diseases; CONPHYDE[10], which assesses the thermodynamic properties of intricate fluid systems; and DECADE[11], tailored for catalyst design. Expert systems heavily depended on specialized knowledge, necessitating sophisticated data management solutions. The lack of adequate storage systems management made expert systems impractical for addressing more complex chemical process systems.

DL method's capabilities in learning latent chemical process patterns present a practical approach to modeling complex process systems compared to traditional methods. This powerful strategy emphasizes layer-based algorithms inspired by the structure and function of the human brain, known as artificial neural networks (ANNs)[12,13]. Backpropagation facilitated the training of causal DL models based on ANNs. This novel method presented a new perspective to general dynamic systems identification, allowing researchers to develop casual data-driven models for specific process systems with known boundaries. This data-driven approach is known as black box modeling[14,15].Despite the advantages of early neural networks. They were incapable of learning deep, complex data patterns. Therefore, the early DL methods could not demonstrate reasonable performance for modeling complex process systems.

Furthermore, Deep Neural Networks (DNNs)[16], which are neural networks with more than one hidden layer, emerged as a promising concept for learning deep data patterns.

These could address the limitations of early ANNs in representing complex process systems. However, due to the computational limits of that era, training DNNs was not practically feasible[17]. Another drawback is that this method necessitates a substantial and dependable amount of process data in process systems, which may not always be accessible. This led to a halt in AI-driven research in this field, known as the AI winter[18,19].

The challenges of training DNNs faded away the research's interest in the application of ML methods in process modeling and control, which is known as AI winter[18,20]. During this period, methods such as white box[21] and grey box modeling[22] were investigated to address the complexities of process model derivation. On the other hand, novel numerical methods for solving dynamic process models attracted the attention of PSE researchers[23]. However, traditional transport phenomena modeling systems, such as the Navier-Stokes equations, are hindered by a lack of analytical solutions and the high computational demands of numerical solutions[24]. Therefore, as various aspects of PSE research have evolved, developing robust and fast models for process systems has become increasingly challenging, necessitating accurate methods while requiring lower computational demands.

Introduction of Gradient descent optimization algorithm[25], the emergence of data parallelism[26], and breakthroughs in hardware design technology, such as neural network accelerators, enhanced the computational power of computers, made DNN training possible. This not only marked the end of the AI winter era but also regained the attention of researchers to tackle more advanced problems using DNNs.

Consequently, different types of DNNs, such as Feed-Forward Neural Networks (FNNs) and Recurrent Neural Networks (RNNs), showed promising results in learning complex process system nonlinearities that streamlined framework development for system identification, dynamic optimization, and real-time decision-making using advanced control strategies, such as Model Predictive Control (MPC). This shifted recent PSE research toward developing data-driven dynamic system identification frameworks for different process systems.

In this context, Afram et al.[27] demonstrated the application of a two-layer FNN in MPC

for optimizing HVAC systems, focusing on a residential HVAC system case study. In another work, A dynamic MPC system with an FNN was used to regulate a flotation column's interface level by adjusting tailings flow. The FNN predicted the interface level based on a multi-time step state history. This approach outperformed PID controllers, showing faster convergence and smoother trajectories[28]. While FNNs have successfully integrated with MPC for different applications, the complexity of modeling long-term dynamic processes remained challenging due to the one-way structure of FNNs.

RNNs are a more robust method for capturing complex long-term dynamics due to their capacity to recognize temporal patterns and retain information from previous time steps[29].RNNs have gained attention as a practical data-driven process modeling approach, bridging the gap between the uni-directional nature of FNNs and the demands of long-term process modeling. For instance, Kaiser et al.[30] utilized Long Short-Term Memory (LSTM) networks, a type of RNN, for improved control of nonlinear chemical processes. In another work, Alhajeri et al.[31] Designed an RNN-based and Hybrid-based estimator(FNN), for a nonlinear process system, and the estimators were further integrated into MPC which resulted in optimal control of the process.

1.2 Problem statement

Recent research has highlighted the practical applications of DL-based data-driven modeling techniques for dynamic system identification and real-time process control. These methodologies and their applications are comprehensively discussed in Chapter 2. Despite the effectiveness of DL-based modeling techniques, their application is often specific to particular processes and constrained by the scale of those processes. In other words, the solutions provided by DL-based models are typically limited to the scale of experiments. This limitation arises from two main challenges: the scarcity of process data across multiple scales and that data-driven models do not inherently understand the underlying physical principles of

the processes they model.

In this thesis, we focus on developing and presenting efficient deep learning (DL)-based strategies, enhanced by time-series modeling, to address the challenges of multi-scale and stochastic process systems. We aim to overcome the inherent limitations of data-driven system identification and real-time decision-making. This is achieved through a detailed examination of two case studies, showcasing our contribution to bridging the gap between theoretical modeling and practical application in complex process systems.

1.2.1 Case Study 1: Real-time dynamic optimization and control of an industrial ammonia synthesis reactor

Ammonia, the world's second most synthesized chemical compound, is used for various applications such as energy transportation, fertilizer production, grain drying, and more[32]. Additionally, with the advancement of technology, new applications of ammonia emerge, leading to increased production demands. The importance of green ammonia extends beyond its environmental benefits. According to MacFarlane et al. [33], ammonia's high energy density and the existing global infrastructure for its production and distribution make it a viable candidate for a wide range of energy applications, from power generation to vehicle fuel.

On the other hand, the industrial process of ammonia synthesis consumes a significant amount of fossil fuels[34], leaving a heavy carbon footprint on our planet. Using renewable energies for green ammonia production is a possible solution to reduce greenhouse gas effects while adequately responding to the proliferating demands for ammonia[35,36,37].

Moreover, the green ammonia economy represents a promising pathway toward a more sustainable and environmentally friendly energy system. Its role in facilitating the storage, transport, and utilization of renewable energy underscores its potential to significantly impact the global energy landscape, contributing to reducing carbon emissions and transitioning towards a cleaner, more sustainable energy future.

Efficient green ammonia production at industrial scales hinges on an accurate process

model and a real-time dynamic optimization framework. In industrial settings, relying solely on Computational Fluid Dynamics (CFD) requires significant computational power. Due to the iterative approach of optimization algorithms, the computational bottleneck occurs when the computationally demanding CFD models need to be executed repeatedly. This leads to redundancy and delayed computations, especially for real-time decision-making. This case study presents an efficient real-time dynamic optimization framework based on Sequence-to-Sequence LSTM networks for a multi-scale ammonia synthesis process without considering any simplifying assumptions in the model hierarchy. The LSTM-based surrogate model learns the temporal correlations between the model outputs and the operational conditions with a sequence-to-sequence architecture that treats this problem as a multivariate time-series prediction task. The performance of the LSTM model offers a less redundant and faster approach used in real-time decision-making algorithms, mimicking the CFD model.

Subsequently, the LSTM model is integrated with MPC forming a framework that the interplay between the LSTM surrogate model and the multi-scale CFD model ensures optimal ammonia production with the haber-botch process. It is important to note that the CFD model has not been eliminated in this frame, yet its presence in MPC implementations advocates the robustness of this approach.

The results of this case study show that the DL models are applicable to industrial-scale process systems and effective in providing faster and more reliable solutions for real-time strategies. Finally, using deep learning for sustainable green ammonia production contributes to possible economic opportunities and decision-making in the food and agricultural sectors, offering a cheaper and cleaner method for supplying proper nitrogen-based fertilizers.

1.2.2 Case Study 2: Time-series modeling of soil water content for agricultural decision making

Soil fertilizer absorption rate is a stochastic factor that implicitly affects agricultural yields and crop growth[38]. Soil water content (SWC) is one of the principal measurable soil properties regulating fertilizer absorption rate. Therefore, predicting future soil water content is essential in agricultural decision-making. The precipitation and soil physics play a major role in determining soil water content values and its dynamic modeling formulated using transport phenomena and hydrology concepts in the Richards Equations[39]. Stochastic variables and other soil properties that are not directly measurable hinder the applicability of Richards' model for real-time decision-making. To address this limitation, Markov Decision Processes (MDP) have been used to develop a simplified model for soil water content estimation[40] known as the Markov-based model. In this model, the stochastic and immeasurable parameters are reduced to one periodic variable that accounts for seasonal and stochastic changes of the related variables, such as precipitation.

Despite the efficiency of the Markov-based model, its prediction accuracy drops as the prediction horizon passes. Also, this model shows high prediction errors with extreme weather regimes, indicating that some of the important trends and latent soil water storage patterns are neglected in the Markov-based model.

Time-series modeling using RNNs offers an adequate avenue for soil-water dynamic modeling. Therefore, this case study proposes an RNN-based soil water content modeling framework as a new perspective to overcome the limitations of existing models. RNNs not only excel in learning seasonality and trends but can also learn stochastic patterns in soil water content dynamics, which makes them robust for precise predictions without suffering from the limitations of the Markov-based model.

LSTM networks are particularly capable of considering long-term temporal correlations, meaning that soil water content could be predicted more accurately by considering more historical data. In this case study, the performance of data-driven LSTM models with single-

step and multi-step ahead predicting scenarios is inspected for soil water content values in the state of Kansas.

Kansas Mesonet is a weather system monitoring network that measures different weather and soil properties across several stations in Kansas[41]. Using the provided soil properties data, the proposed models are trained on real data, making them practical for further real-time agricultural decision-making.

Unlike the Markov-based models, the RNN-based models predict soil water storage for several days ahead without relying on only last-day values. This capability of the RNN-based models not only advocates for agricultural real-time decision-making but also is beneficial for longer horizon agricultural planning.

Chapter 2

Methods and background

The recent DL models' effectiveness in capturing the deep behavior of nonlinear systems, as highlighted by the universal approximation theorem[42], has inspired researchers in PSE to adopt DL-based solutions for dynamic system identification and real-time decision-making. This chapter first explains the standard DL methods used for dynamic system identification. Subsequently, the latest contributions in this area are reviewed.

2.1 Machine learning workflow

FNNs, various types of RNNs, and self-attention transformers are primarily used to represent the dynamics of individual process systems within the control algorithm. Choosing an efficient DL method depends solely on the nature of the process. Subsequently, the preferred methods follow a sequential workflow for the robust development of DL models intended to replace traditional numerical or analytical models. This workflow is referred to as the machine learning workflow. In this section, we discuss the machine learning workflow and provide a foundational overview of the machine learning methods frequently employed in process systems.

2.1.1 Problem definition

Generally, the problems or tasks in ML are divided into two categories: regression and classification. Regression analysis, aimed at predicting a continuous outcome, models the relationship between one or more predictors and a numerical response variable, facilitating the prediction of quantities such as soil water content or temperatures[43]. On the other hand, classification tasks predict categorical outcomes, assigning observations to discrete categories or classes based on known data, such as diagnosing diseases or identifying spam emails[44]. The critical distinction lies in the output: regression deals with continuous variables, while classification handles categorical variables, each employing specific evaluation metrics and methodologies tailored to their predictive objectives[43,44].

Different ML algorithms are categorized based on a specific task objective. In Supervised Learning, models are trained using labeled data, where input and desired output data are provided[45]. Unsupervised Learning deals with unlabeled data, focusing on finding patterns or structures within[46]. Semi-supervised learning combines labeled and unlabeled data for improved model training[47]. Lastly, Reinforcement Learning involves an agent interacting with an environment, aiming to maximize a cumulative reward function[48].

For process systems modeling, the supervised learning method is a potential alternative approach for addressing the limitations of traditional modeling. This type of ML model adeptly captures the system's dynamics. It acts as a surrogate model within a decision-making framework. ML models' rapid and low-cost computational features make them an effective system identification strategy for real-time dynamic optimization and process control challenges[49].

The choice of ML algorithm varies based on the system's scale and the complexity of the problem. Usually, for complex process systems with nonlinearities, DL methods are a better choice to learn the complex patterns of a process.

2.1.2 Data Collection

The accurate performance of ML models depends on properly collecting the process dataset. This dataset forms the foundation of the ML models, as the weights of each model are trained based on the information within this data.

Several approaches exist for collecting rich data that contain the physical patterns of a process's dynamics. One approach involves collecting real-time measurements from precise sensors, such as temperature and pH sensors, throughout the process. This real-time data collection method yields robust process data. However, implementing and maintaining a sensor network can be time-consuming and expensive. Also, in some cases, real-time sensing of the desired variables is not feasible. Considering these limitations, another data collection approach is presented through offline simulations of traditional first-principle modeling. The simulation data is then used to train the ML models. This method is proper when no sensory data is available for real-time measurement of process variables, such as concentration values.

Non-continuum space modeling is integrated with first-principle models for multiscale process systems to form multiscale models. These models provide accurate predictions of the process dynamics. However, their high computational demands make them impractical for real-time optimization and control. As a solution, combining offline simulations of multiscale models with ML models is a potential solution to address the computational bottleneck[49].

2.1.3 Data Preprocessing

Data preprocessing is a crucial step in the machine learning workflow that determines how the ML model can see the latent patterns in a dataset. Data is cleaned, adjusted, and prepared for the training phase in preprocessing. This step is not merely about correcting data; operations in this stage, such as normalizing or standardizing, ensure that the ML model understands the patterns in the data equally, leading to enhanced accuracy and reliability[50].

Normalization and standardization are crucial preprocessing steps in the context of neu-

ral network training, each serving to scale input features in slightly different ways. Normalization rescales features to a fixed range between 0 and 1, using the formula $x_{\text{norm}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$, which is particularly beneficial for algorithms that are sensitive to the absolute sizes of feature values. Standardization, on the other hand, transforms data to have a mean of 0 and a standard deviation of 1, according to $z = \frac{(x-\mu)}{\sigma}$, ensuring that features contribute equally to model predictions without being biased by their original scale. The choice between normalization and standardization depends on the model's specific requirements and the nature of the data. Standardization is generally preferred for neural networks as it maintains the data structure without bounding it to a particular range, which is crucial for the algorithm's convergence. However, normalization might be more suitable for data constrained within a specific range or when the algorithm benefits from uniformly scaled input features[1,51].

2.1.4 Model selection

Machine learning offers a wide range of algorithms and architectures for modeling problems. In statistical learning, where the size of the dataset is relatively small and the present patterns are not complex, methods like Support Vector machines (SVMs), Random Forests(RFs), and ensemble models are efficient[43]. However, for larger problems, layer-based neural networks such as FNNs, RNNs, and CNNs are used to learn complex patterns in a large dataset[1].

In process system identification, RNN-based surrogate models address the limitations of traditional modeling approaches, resulting in real-time decision-making frameworks for efficient implementation of dynamic optimization and control algorithms such as MPC[46].

2.1.5 Training

The proper ML model's parameters are found and tuned in the training phase. For Neural networks, the model parameters are the weights assigned to each neuron. Finding the proper

weights that represent the deep data patterns is an optimization problem addressed by a combination of gradient descent algorithm and backpropagation[52,53].

Backpropagation and Gradient-Decent

Backpropagation is a systematic method for calculating the gradients of the loss function for each weight in the network, effectively determining how changes to weights decrease the error between the predicted and actual outputs. It relies on the chain rule from calculus to propagate errors backward through the network, from the output to the input layer. Gradient descent, on the other hand, is an optimization algorithm that iteratively adjusts parameters (weights) by moving in the direction of the negative gradient of the loss function, aiming to find the minimum loss by following the steepest descent. Together, these mechanisms enable neural networks to learn from data, adjusting their weights to improve predictions and decision-making capabilities[1].

Gradient-decent's fundamental principles have paved the way for developing more sophisticated optimizers like RMSprop, SGD, and Adam. These advanced algorithms build upon the basic concept of gradient descent, introducing mechanisms to overcome its limitations, such as slow convergence and susceptibility to local minima[25].

RMSprop

Root Mean Square Propagation (RMSprop) proposed an adaptive learning rate for each model parameter, resolving the problem of a monotonically decreasing learning rate in gradient descent by dividing the gradient by a running average of its recent magnitude[25].

SGD

Stochastic Gradient Descent (SGD) updates the parameters using only a single sample or a small batch of samples at each iteration. This approach significantly reduces the computational burden, making it feasible to train on large datasets. Moreover, the inherent

noise introduced by the stochastic nature of the algorithm helps in escaping local minima, potentially leading to better solutions in complex loss landscapes[54].

ADAM

Adam optimizer combines the advantages of two extensions of stochastic gradient descent: Adaptive Gradient Algorithm (AdaGrad) and RMSprop. Like RMSprop, Adam maintains a per-parameter learning rate that improves performance on problems with noisy or sparse gradients. Additionally, Adam incorporates the concept of momentum by keeping track of an exponentially decaying average of past gradients. This combination allows Adam to navigate the optimization landscape in a way that is both fast and stable, making it a preferred choice for many deep-learning tasks[55].

2.1.6 Validation and Hyperparameter tuning

Validation assesses the model's performance on unseen data, ensuring it doesn't merely memorize the training data but can also generalize to new, unknown data. Common validation methods include cross-validation and hold-out validation[56]. Hyperparameter tuning refines the model's settings to achieve optimal performance. Grid Search, Random Search, and Bayesian optimization are employed to determine the best architecture and training configurations. This ensures the ML model is optimally trained and valid for further use in process optimization and control[57].

2.1.7 Evaluations

The evaluation step determines how well the trained model can predict and control real-world process dynamics. By using metrics like Mean Absolute Error (MAE) or Root Mean Square Error (RMSE), the model's performance is evaluated against test data. A robust evaluation ensures that the model not only captures the historical trends of the process but

is also equipped to handle unforeseen disturbances and variations, guaranteeing stable and optimized process operations[58,59].

2.2 Artificial Neural Networks (ANNs)

Various deep learning algorithms have been developed and refined to address specific objectives, each designed to uncover hidden patterns within datasets through distinct methodologies. This evolution has led to the creation of several pivotal neural network architectures, each contributing uniquely to the field's advancement. Notably, architectures such as Feed Neural Networks (FNNs) for regression tasks[29,1], Recurrent Neural Networks (RNNs) for sequential data analysis[60,1], and more recently, Transformer models for natural language processing tasks[61], are common neural network designs. These architectures underscore the dynamic nature of deep learning research. In this section, the mechanics of these specific neural networks are highlighted.

2.2.1 Feed Forward Neural Networks (FNNs)

FNNs are a class of ANNs where the connection of each neuron layer to the next layers is one-directional. They are the simplest form of neural network architecture, designed to recognize patterns and make decisions without holding any hidden state. The primary components of FNNs are layers: an input layer, one or more hidden layers, and an output layer. Each neuron in a layer is connected to every neuron of the subsequent layer through weights and biases[1].Figure 2.1 shows the architecture of FNNs

Mathematically, the output y of a neuron can be represented as:

$$y = f\left(\sum_i w_i x_i + b\right) \tag{2.1}$$

Where:

- x_i are the input values,
- w_i are the weights associated with the connections,
- b is the bias, and
- f is the activation function.

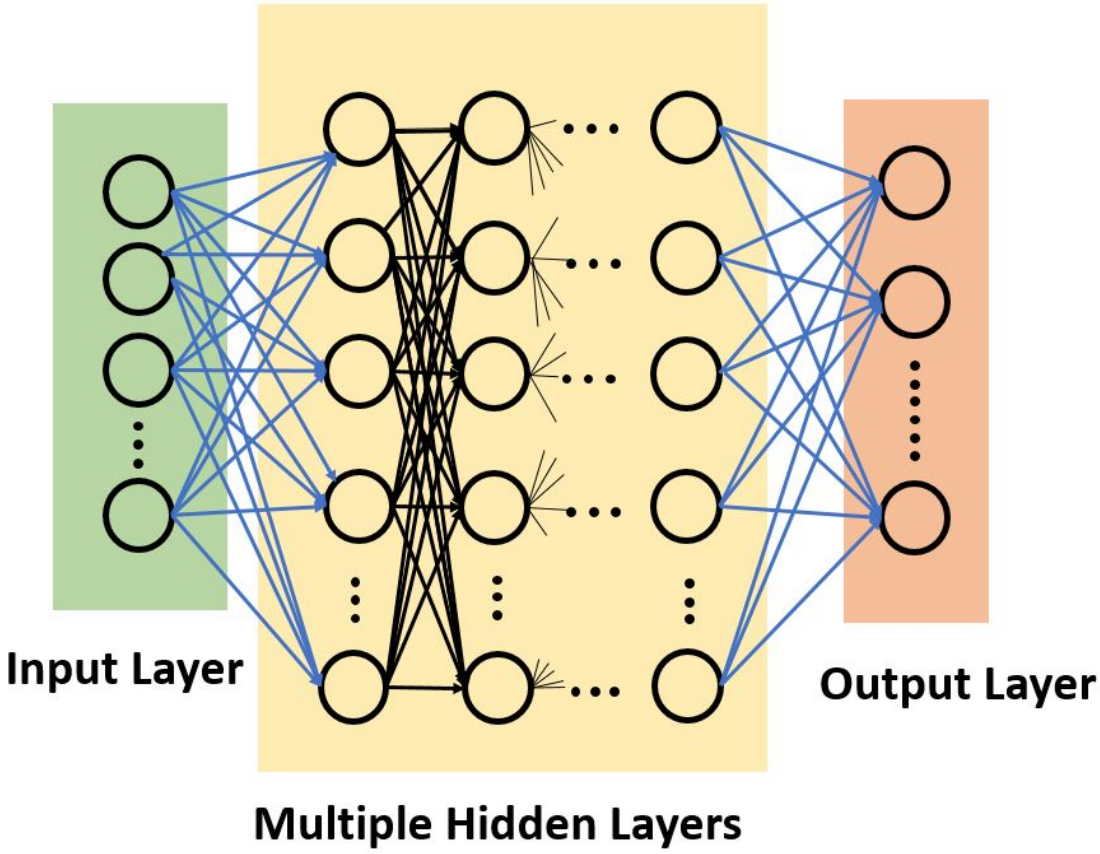


Figure 2.1: Feed Forward neural Network (FNN)

The activation function, such as the sigmoid or ReLU, introduces nonlinearity into the model, enabling it to capture nonlinear relationships in the dataset[29].

Training an FNN involves adjusting the weights and biases to minimize the difference between the predicted output and target values. This is achieved using the backpropagation

algorithm in conjunction with gradient descent. Backpropagation computes the gradient of the loss function concerning each weight by applying the chain rule, which is then used by gradient descent to update the weights and biases, minimizing the loss[25].

2.2.2 Recurrent Neural Networks (RNNs)

Despite the efficiency and the rapidness of FNNs, they are not proficient in recognizing patterns in sequential datasets. Time-series problems and natural language processing are types of sequential tasks where each value(word) is sequentially dependent on the other values(words) in the past.RNNs, as another type of ANNs, address sequential data by creating hidden states through the additional recursive connection of each neuron to itself. The core idea behind RNNs is to maintain the hidden state that captures information about previous steps in the sequence.

Vanilla RNNs

Vanilla Recurrent Neural Networks (RNNs) represent the most basic form of RNN architecture, designed to process sequential data by capturing temporal dependencies through an internal state (memory) that allows previous outputs to be used as inputs while having hidden states. However, they are notably challenged by long-term dependencies due to the vanishing gradient problem, where gradients can either grow or shrink exponentially over long sequences, making it challenging to learn correlations between distant events[29]. Despite this limitation, Vanilla RNNs have been successfully applied in various tasks such as language modeling and time-series forecasting, demonstrating their capability to model sequences effectively for shorter dependencies? . Their simplicity and the intuitive way they mimic temporal sequences make them a fundamental step in developing more complex RNN architectures like LSTMs and GRUs[1]. Figure 2.2 shows RNNs' structure

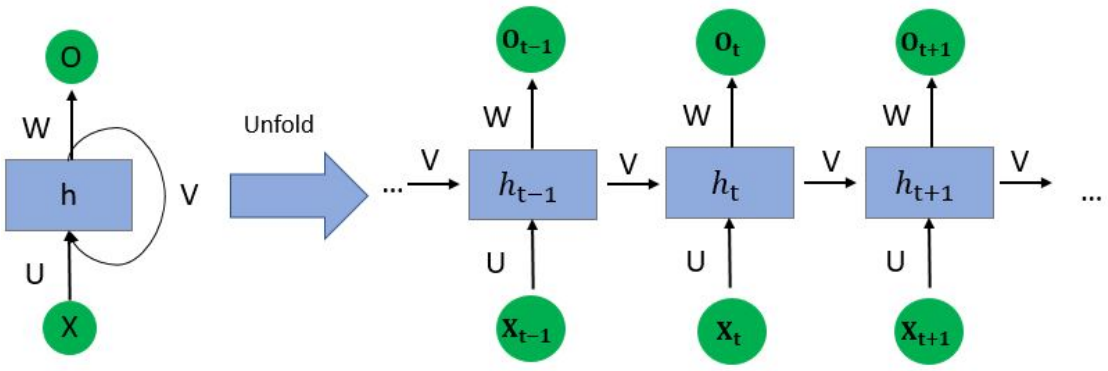


Figure 2.2: RNN structure

Mathematically, the hidden state h_t at time t is computed as:

$$h_t = f(W_{hh}h_{t-1} + W_{xh}x_t + b_h) \tag{2.2}$$

Where:

- x_t is the input at time t,
- W_{hh} and W_{xh} are weight matrices, and
- b_h is the bias.

Long Short-Term Memories (LSTMs)

Vanilla RNN's limitation in learning long sequences inspired the development of LSTMs. LSTMs are a more evolved version of RNN designed to capture long-term dependencies. They introduce memory cells that can maintain information in memory for long periods. The core idea of LSTMs is the cell state, along with three gates (input, forget, and output) that regulate the flow of information.[49,1]. The input gate controls the extent to which new information flows into the memory cell, the forget gate decides what information is discarded, and the output gate determines what data is passed on to the next time step[29].

Figure 2.3 shows the architecture of an LSTM cell.

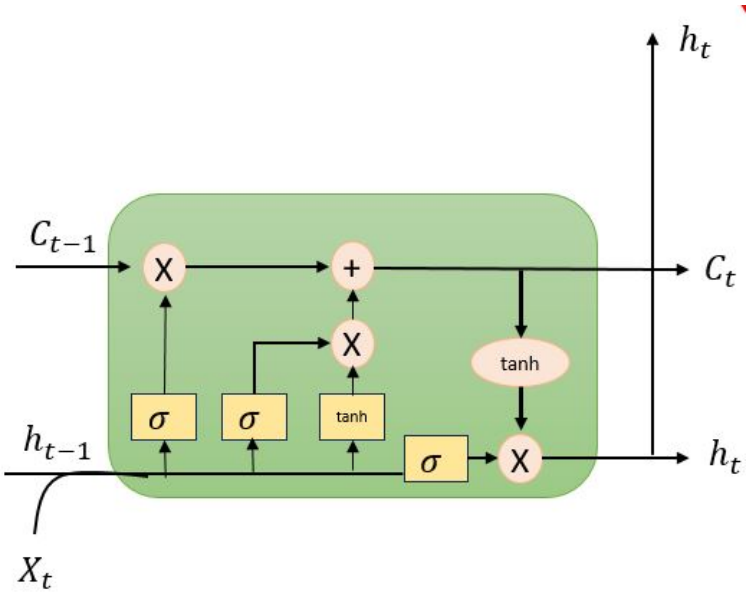


Figure 2.3: An LSTM Cell

The information flow between the gates is mathematically described below:

Given input vector x_t, previous cell state c_{t-1}, and previous hidden state h_{t-1}:

$$\text{Forget gate:} \quad f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.3}$$

$$\text{Input gate:} \quad i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.4}$$

$$\tilde{c}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{2.5}$$

$$\text{Cell state update:} \quad c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{2.6}$$

$$\text{Output gate:} \quad o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.7}$$

$$h_t = o_t \odot \tanh(c_t) \tag{2.8}$$

Where:

- h_{t-1}: Previous hidden state at time step $t-1$.
- σ is the sigmoid function.

- $\odot$ denotes element-wise multiplication.
- W matrices and b vectors are trainable parameters of the LSTM.

.

Despite their strengths, LSTM networks have limitations. They can be computationally intensive and slow to train, especially with large datasets, due to their complex architecture involving multiple gates and recurrent connections. LSTMs also struggle with very long sequences, where their advanced memory management mechanisms may fall short, leading to diminished performance. Furthermore, they require substantial tuning of hyperparameters to optimize performance for specific tasks, which can be time-consuming and expertise-demanding. Lastly, LSTMs might not always effectively handle non-sequential or time-independent data, limiting their applicability across different types of machine learning challenges[1,29].

Gated Recurrent Units (GRUs)

GRUs address the complex nature of LSTMs and their computational challenges in training. GRUs are designed to improve the efficiency and simplicity of LSTM networks. They combine the forget and input gates into a single update gate and merge the cell state and hidden state[1].

GRUs maintain the ability to capture long-term dependencies in sequence data through a simplified gating mechanism consisting of only two gates: the update gate and the reset gate. The update gate determines the degree to which the unit retains old state information, and the reset gate controls the incorporation of new input information, making GRUs adept at sequence modeling tasks with potentially faster training times and fewer parameters than LSTMs. Chung et al.[62] have underscored the efficacy of GRUs in diverse applications, from sequence modeling to speech signal processing, showcasing their versatility and performance.

Figure 2.4 shows the architecture of a GRU cell.

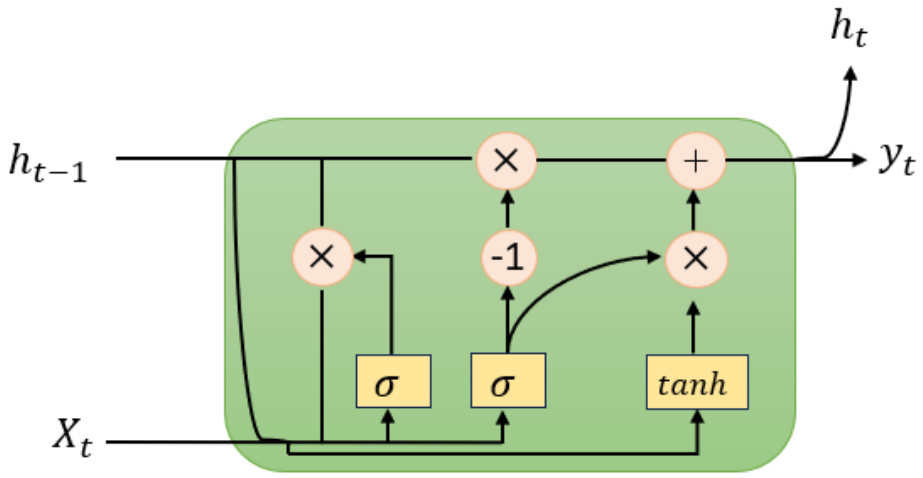


Figure 2.4: A GRU Cell

The following equations can represent the GRU cell computations:

$$\text{Update Gate:} \quad z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \tag{2.9}$$

$$\text{Reset Gate:} \quad r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \tag{2.10}$$

$$\text{Candidate Hidden State:} \quad \tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t]) \tag{2.11}$$

$$\text{Final Hidden State:} \quad h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \tag{2.12}$$

Where the variables represent:

- h_{t-1}: Previous hidden state at time step $t - 1$.
- z_t: Update gate vector at time step t.
- r_t: Reset gate vector at time step t.
- $\tilde{h}_t$: Candidate hidden state vector at time step t.
- h_t: New hidden state vector at time step t.

- W matrices and b vectors are trainable parameters of the LSTM.

RNNs, especially LSTMs and GRUs, have shown great promise in time-series prediction, a critical component in MPC. By predicting future process values, MPC can make informed decisions to optimize and control processes, especially in scenarios where traditional control strategies might fall short[49].

2.2.3 Encoder-decoder architecture

The encoder-decoder architecture is robust for sequence-to-sequence learning tasks. The encoder processes the input sequence into a fixed-length vector representation, capturing the essence of the input data, while the decoder takes this representation and generates the output sequence. This architecture effectively handles sequences of variable lengths, making it highly versatile and powerful for various applications. Cho et al.[63] were among the pioneers to introduce and elaborate on this concept, demonstrating its efficacy in machine translation tasks. Sutskever et al.[64] further cemented the architecture's significance by showcasing its application in translating sequences with deep LSTM networks, achieving remarkable performance improvements. The encoder-decoder framework's adaptability has since been enhanced with attention mechanisms, as illustrated by Bahdanau et al.[65], which enable the model to focus on specific parts of the input sequence when generating each word of the output sequence, thereby resolving long-range dependency issues.

2.2.4 Attention Mechanism

RNNs are promising for time-series modeling. As explained in detail, each RNN architecture demonstrates potential applications in system identification for real-time decision-making. Despite the advantages of RNNs, their training is computationally demanding due to the recurrence and complex structure. Moreover, the full advantage of captured information for time-series problems in RNNs is often not realized. This is the main reason for RNNs' capa-

bility to process long sequences. This problem was first noted in NLP, where long-sequence learning was necessary for natural language translation. In the time-series context, developing RNN-based models for multi-step prediction requires a strategy for taking advantage of the hidden state information with a new approach. Therefore, Bahdanau et al.[66] introduced the attention mechanism to leverage the captured data in RNNs for enhanced sequence-to-sequence modeling. The attention mechanism allows models to focus on different parts of the input sequence when predicting each word of the output sequence, akin to how human attention shifts to other parts of a scene when processing visual information. It addresses the limitations of sequence-to-sequence models by enabling the model to dynamically assign weights to different inputs at each step of the output generation, which is particularly beneficial for processing long input sequences.

The correlation in the attention mechanism of a sequence-to-sequence model is quantified by attention weights, which are computed as follows:

$$c_j = \sum_{i=1}^{T_x} \alpha_{ij} h_i \tag{2.13}$$

Where:

- h_i is the hidden state of the input at position i.
- $\bar{h}_j$ is the hidden state of the output at position j.
- score$(h_i, \bar{h}_j)$ is a function that computes the alignment score between h_i and $\bar{h}_j$.
- α_{ij} is the attention weight for the input at position i and the output at position j.
- T_x is the length of the input sequence.
- c_j is the context vector for the output at position j.

The general form of the attention weights can be represented as follows:

$$\mathcal{A}(q, K, V) = \sum_i p(a(k_i, q)) \times v_i$$

where

- $\mathcal{A}(q, K, V)$ is the output of the attention mechanism, a weighted sum of the values.
- q is the query, compared against each key in the set K.
- K is the set of keys, with each k_i corresponding to an element in the input sequence.
- V is the set of values, with each v_i associated with a key k_i and representing the information to be weighted.
- $a(k_i, q)$ is the compatibility function that measures how well the query q matches each key k_i.
- $p(a(k_i, q))$ represents the probability distribution over the keys obtained by applying a softmax to the compatibility scores.

Several compatibility functions are proposed from different perspectives. Choudhari et al.[67] have discussed each compatibility function in detail.

Finally, the attention mechanism enhanced the capability of RNNs for more extended sequence modeling; however, the computational demands for training such models are still high, thus making them computationally intensive for large datasets.

2.2.5 Transformers

Using attention-based RNNs for training sequential models on a large data corpus suffers from heavy computational demands. To address this problem, Vaswani et al., in the renowned paper titled "Attention is all you need" introduced encoder-decoder-based transformers[61].

Transformers mark a paradigm shift in sequence-to-sequence models by eliminating the need for recurrent layers in favor of the self-attention mechanism. This mechanism enables the model to consider the entire input sequence simultaneously, thereby improving computational efficiency and model performance on tasks requiring an understanding of long-range dependencies.

Structurally, the transformer consists of two core components: an encoder that maps an input sequence to a sequence of continuous representations and a decoder that generates an output sequence from these representations. Both the encoder and decoder are composed of a series of identical layers, with each layer in the encoder containing a multi-head self-attention mechanism and a position-wise fully connected feed-forward network, and each layer in the decoder adding a third sub-layer that performs multi-head attention over the encoder's output. The inclusion of residual connections[68] and layer normalization[69] around each sub-layer facilitates deeper model architectures by mitigating the vanishing gradient problem

The introduction of the Transformer model has facilitated advancements across a range of natural language processing tasks, evidenced by its foundational role in the development of models such as BERT[70] and GPT[71], which have set new benchmarks in language understanding and generation, respectively. On the other hand, the advantages of transformers are crucial for time-series modeling.

In process systems, the potential of transformers lies in their ability to model complex temporal sequences for large-scale sequences with reduced training time, which is crucial for predicting system behaviors and implementing MPC. Their ability to capture complex patterns in sequential data can be harnessed to enhance the precision and reliability of MPC systems[72].

2.2.6 Model evaluations

Understanding several important evaluatory metrics is essential for evaluating the general performance of DL-based models. In regression tasks, common evaluation metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE), and Mean Absolute Scaled Error (MASE) indicate the average performance of the trained model. Depending on the problem's configuration, each metric can show the effectiveness of the DL model. On the other hand, visual evaluations such as parity and residual plots show the model's performance for each data point and also provide a critical understanding of its specific strengths and weaknesses.

Mean Squared Error

Mean Squared Error calculates the average of the squares of the differences between actual and predicted values, offering a clear measure of model accuracy. The squaring of the errors ensures that larger errors are penalized more heavily than smaller ones, making MSE sensitive to outliers. The general formula for MSE is represented as

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{2.14}$$

where n is the number of observations, y_i is the actual value, and $\hat{y}_i$ is the predicted value by the model[73]. This metric is beneficial in regression analysis and has been extensively utilized for model evaluation and comparison[74].

Root Mean Squared Error

Root Mean Squared Error is the square root of the MSE, providing a scale-sensitive accuracy measure that quantifies the standard deviation of prediction errors. In this context, prediction errors are the differences between actual values and the predictions made by the model. The RMSE is especially valuable because it gives the errors in the units of the variable of interest, making the interpretation straightforward. The formula for calculating

RMSE is expressed as:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2.15}$$

This metric is advantageous for its sensitivity to outliers and its ability to provide a clear indication of model accuracy in the same units as the observed data, facilitating easier model comparisons and assessments[75,74,73].

Mean Absolute Error

Mean Absolute Error (MAE) calculates the average magnitude of the errors between predicted and actual values, disregarding their direction. MAE is defined as:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.16}$$

One of the main advantages of MAE is its interpretability; since it is expressed in the same units as the data, it is straightforward for practitioners to understand the error magnitude. However, MAE does not penalize significant errors as heavily as other metrics like Mean Squared Error (MSE), which can be both an advantage and a limitation depending on the application context. However, MAE remains popular due to its simplicity and clarity in conveying the average prediction error[75,74,73].

Mean Absolute Scaled Error

Mean Absolute Scaled Error (MASE) is a highly regarded metric in time series forecasting, renowned for its utility in evaluating the accuracy of predictive models. Unlike other metrics that may not account for the data's scale or the forecast context, MASE provides a normalized error measure that is easily interpretable and comparable across different time series datasets. The brilliance of MASE lies in its ability to measure the prediction error relative

to a naive benchmark forecast, typically the naive method of using the last observed value for one-step forecasts. This scaling ensures that the MASE is dimensionless and allows for meaningful comparisons between models and across time series with different units or scales of measurement.

The formula for MASE is given by:

$$\text{MASE} = \frac{\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|}{\frac{1}{n-1}\sum_{i=2}^{n}|y_i - y_{i-1}|}$$

The denominator is the mean absolute error of the one-step naive forecast method, serving as the benchmark for scaling the forecast error[75,74,73].

Parity plots

Parity plots evaluate the performance of predictive models by comparing predicted values against observed or actual values. These plots graphically depict the degree of agreement between the model outputs and the corresponding observed data, with each point representing an observation-prediction pair. The diagonal line shows that the predicted values match the actual values perfectly. When there is zero error and 100% prediction accuracy, all the predicted points will align with this line.[76,77].

Residual plots

Residual plots assess predictive models by illustrating the differences, or residuals, between observed values and those predicted by the model. These plots offer a granular view of each data point's prediction error, facilitating the identification of patterns in the residuals that might indicate potential problems with the model's assumptions, such as nonlinearity or the presence of outliers[78,79].

2.3 Model Predictive Control (MPC)

MPC is an advanced control strategy that computes control actions by solving an optimization problem at each time step. The optimization aims to minimize a cost function over a finite prediction horizon, subject to system dynamics and constraints. In nonlinear systems, this optimization often involves a nonlinear quadratic cost function. The objective in MPC for such systems is expressed as:

$$J(U) = \int_t^{t+N} \left(x(\tau)^T Q x(\tau) + u(\tau)^T R u(\tau) \right) d\tau \tag{2.17}$$

where $x(\tau)$ is the state of the system at time τ, $u(\tau)$ is the control input, Q and R are positive definite weighting matrices, and N is the prediction horizon. This integral form accumulates the cost over the horizon, with quadratic terms penalizing state deviations and excessive control efforts. The optimization problem in MPC becomes:

$$\min_U \int_t^{t+N} \left(x(\tau)^T Q x(\tau) + u(\tau)^T R u(\tau) \right) d\tau \tag{2.18}$$

subject to the nonlinear system dynamics:

$$\dot{x}(\tau) = f(x(\tau), u(\tau)), \quad \tau \in [t, t+N] \tag{2.19}$$

and any constraints on states and controls. The solution provides a sequence of control actions over the horizon N, of which only the first is typically implemented. The procedure is then repeated at the next time step, using updated state measurements or estimates. This approach allows MPC to handle multivariable systems, constraints, and nonlinearities, making it particularly suitable for complex process control applications[45,80].

2.4 Time-series system identification and real-time decision making

In this section, we highlight recent studies on data-driven deep learning approaches for time-series modeling of process systems and their integration with Model Predictive Control (MPC). To better understand each deep learning architecture's application, we have organized the highlighted studies based on the specific architecture each employs.

2.4.1 FNN-based process system modeling

The FNN's capability of capturing and predicting nonlinear functions advocated by the universal approximation theorem[81] has attracted process systems researchers to use FNNs for complex processes. For instance, Çıtmacı et al.[82] have leveraged this capability to propose an MPC framework for the electrochemical reduction of CO2. They gathered relevant process data from an experimental setup of an electrochemical reactor. Instead of traditional modeling methods for this setup, an FNN with two hidden layers is trained based on real process data. Furthermore, the trained FNN is integrated with MPC, successfully optimizing the electrochemical reduction of CO2 and showcasing the potential applications of FNNs in black-box system identification of natural processes for control purposes.

In another work, Dodhia et al.[83] demonstrated the application of FNNs, combined with the advantages of reduced-order models, to enhance MPC for diffusion-reaction processes. First, they employed the Karhunen-Loève expansion[84] to derive PDE systems from the process data. Then, using theGalerkin's model reduction framework[85], they reduced the Partial Differential Equation (PDE) models into several Ordinary Differential Equations (ODEs). However, using the reduced-order model was still computationally intensive for MPC. Therefore, they employed FNNs to approximate the reduced model, which was subsequently integrated into MPC, underscoring the robustness of FNNs. Furthermore, Alhajeri et al.[31] integrated FNNs into MPC by designing an FNN-based hybrid model for an

extended Luenberger observer[86]. To demonstrate the efficacy of the hybrid model, they applied the hybrid model to a nonlinear process, which was simulated using a first-principle model. The process data was obtained from the simulation and further used in the training of the hybrid model. This hybrid model was encapsulated in the observer, successfully implementing Lyapunov-based MPC[87] for the nonlinear process.

2.4.2 RNN-based process system modeling

Despite their deep pattern-capturing capabilities, FNNs fall short of understanding the sequential patterns inherent in data. FNNs are commonly used in process systems and control to map manipulated inputs to the system's states. Yet, their limitations prevent them from observing the temporal effects between different states.

RNNs, by creating a recurrence in each neuron, manage to capture the temporal relation between the steps. Xiao et al.[88] leveraged this ability to propose an RNN-based MPC framework for plasma etch processes on 3D materials. They combined a multiscale model with kinetic Monte Carlo's detailed substrate etching. Galerkin's reduction method was also applied to reduce the multiscale model[85]. Separate RNN models were subsequently trained on both the multiscale and the reduced model simulations to understand the dynamics of the plasma etch system. The RNN models were then incorporated into an MPC strategy, and their performance was benchmarked against a conventional PI controller. The results showed that the proposed RNN-based MPC outperformed the PI controller.

RNNs have also been helpful in Economic MPC (EMPC). Hu et al.[89] employed an RNN model to integrate EMPC for switched nonlinear systems. This work was extended to create another RNN model for real-time hybrid nonlinear systems. Continuing the previous work, Hu and Wu,[90] adopted an online learning approach where the RNN network updates its weights based on real-time data received from the process rather than relying exclusively on offline simulated data. This method resulted in enhanced MPC performance when compared to offline training approaches.

2.4.3 LSTMs and GRUs in process control

Long Short-Term Memory (LSTM) is a type of recurrent neuron designed to alleviate the vanishing gradient problem in simple RNN neurons. From another perspective, LSTMs resemble human short-term memory. Because of the sequential learning capability of the LSTM network, it has become one of the frequently used deep neural network architectures for developing MPC for highly nonlinear process systems[49]. Alongside the proposal of an FNN-based hybrid model for MPC, Alhajeri et al.[31] further developed an extended Luenberger observer based on an LSTM network. The proposed network consisted of two stacked LSTM layers with a hyperbolic tangent activation function, trained on a numerically simulated first-principle model for a nonlinear reaction process in a CSTR. The results showed that integrating the LSTM-based observer with LMPC outperforms the FNN-based hybrid model. Furthermore, Çıtmacı et al.[82] demonstrated the application of LSTM networks for nonlinear modeling of electrochemical reduction of CO2. Consequently, the successful integration of the LSTM model with MPC emphasizes the effectiveness of sequential ML models for carbon dioxide reduction.

Also, LSTM networks have proven promising for acting as a surrogate model for kinetic Monte Carlo (kMC) multiscale simulations. Shah et al.[91] introduced an LSTM-based model predictive control system to optimize the Kraft pulping process to meet the growing demand for diverse paper types. By merging multiscale models with a kinetic Monte Carlo algorithm and training an LSTM- network, the proposed controller efficiently achieves target values, demonstrating high accuracy and computational efficiency.

While the applications above are categorized as black-box modeling, Alhajeri et al.[92] showcased the applications of RNN models in grey-box modeling. They developed a partially connected LSTM network for implementing MPC on an Ethylbenzene (EB) production process. This hybrid model incorporates process structural knowledge with its learning capabilities, leading to a less oscillatory and more energy-efficient MPC than the fully connected model.

The computational burden of training LSTMs has prompted the investigation of GRUs for process modeling and optimal control applications. Krzysztof and Ławryńczuk,[93] employed GRUs to develop faster process models for a polymerization reactor and a neutralization (pH) process. They also trained LSTMs for the same processes and integrated GRU and LSTM models into MPC. Their research showcased the efficiency of GRU-based models compared to LSTM-based ones. They concluded that GRUs outperform LSTM models in learning the process dynamics and MPC implementation while being faster and requiring fewer model parameters to train. Similarly, in another work, Bonassi et al.[94] used GRUs to model an unknown plant with a nonlinear MPC framework. They have further tested this control system with a neutralization (pH) process benchmark, resulting in an offset-free control strategy.

Alongside the challenges of process modeling, MPC requires robust ML-based models and is susceptible to local minimum problems, which often lead to sub-optimal control actions and deviation from control objectives. On the other hand, traditional PI/PID controllers necessitate adaptive parameter tuning that varies according to the setpoints. Sitapure and Kwon,[95] addressed these challenges by creating an LSTM-based controller framework for a batch crystallization process. The proposed controller, LSTMc, is an LSTM network trained on control data. This data is derived from executing multiple MPC scenarios on the process. To underscore the significance of their contribution, they also introduced an LSTM-based surrogate model for the same process and integrated it with MPC. Ultimately, their results indicate that LSTMc outperformed the LSTM-MPC method and exhibited fewer set point deviations in the presence of noisy sensor data.

2.4.4 Transformers in process control

The self-attention mechanism resulting in the transformer's architecture revolutionized Natural Language Processing (NLP). This allowed models like BERT and GPT to outperform RNNs in recognizing sequential data patterns, leading to the creation of ChatGPT.To ad-

dress this breakthrough in process control, Sitapure and Kwon,[72] developed time-series transformers (TST) by integrating multiple attention blocks with a neural network. Using batch crystallization data, they trained TSTs and found they predicted nonlinear dynamics more accurately than RNNs. By merging TSTs with RNNs, they improved nonlinear predictions and demonstrated optimized control with MPC. In later research, they addressed the system-to-system (S2S) transferability using TSTs. Their "Crystal GPT" framework, based on TSTs, tracks the temporal progression of crystals in the crystallization process, resulting in superior MPC performance compared to the RNN-MPC approach[96].

Park et al.[97] highlighted the use of transformers in MPC, presenting a transformer-based framework for multi-step ahead predictions. They evaluated the performance of single-step and multi-step ahead transformer models against LSTM-based models in MPC. Their findings indicated that, as process surrogate models, transformers perform more efficiently than LSTMs in accuracy and optimization speed.

Chapter 3

LSTM-based system identification on ammonia synthesis process

Ammonia, represented as NH_3, is a compound made of nitrogen and hydrogen that plays a crucial role in agriculture and energy sectors. It is widely used as a fertilizer in agriculture to enhance soil fertility and increase crop yields. In the energy sector, ammonia's potential as a sustainable energy carrier is gaining attention due to its high energy density and the fact that it does not emit carbon when used as a fuel. The infrastructure for ammonia production and distribution, which has developed over a century, along with advancements in ammonia synthesis and decomposition technologies, supports the transition towards renewable ammonia production. This transition is driven by the need to reduce carbon emissions and promote a sustainable energy landscape[35,33].

The role of ammonia in achieving a sustainable future is underscored by its versatility as an energy carrier and its potential to decarbonize various sectors, including transportation and power generation. The technological roadmap to the ammonia energy economy highlights ammonia's capacity to store and transport renewable energy, addressing the intermittency of renewable sources such as wind and solar[98]. However, realizing an ammonia-based energy economy faces challenges, including developing efficient, low-cost, and green

ammonia production methods and advancing technologies for the safe and effective use of ammonia as a fuel. Overcoming these challenges is essential for leveraging ammonia's full potential as a critical component of a sustainable energy and agricultural system, leading to a future where energy storage and supply are both sustainable and resilient.

Deep learning-based frameworks have the potential to address some of these challenges. This case study presents an LSTM-based real-time decision-making framework that resolves the computational difficulties of an ammonia synthesis reactor digital twin and presents an efficient way of using deep learning methods for smooth and low-cost integration of renewable energies in ammonia production.

3.1 Ammonia production

Ammonia is produced through a reversible reaction between nitrogen (N_2) and hydrogen (H_2) gases under high pressure and temperature, using an iron catalyst[99,34] known as the Haber-Bosch process. this process has significantly impacted agricultural productivity and global food supply. This method has been fundamental in producing nitrogen fertilizers, directly influencing agricultural yields and sustainability. The Haber-Bosch process is renowned for its efficiency and scale, enabling the production of ammonia on an industrial scale necessary to meet the global demand for fertilizers. This emphasizes the necessity of real-time decision-making frameworks to meet global demands further while integrating renewable energies and addressing environmental concerns.

3.2 Multiscale modeling of ammonia synthesis packed-bed reactor

Dynamic optimization and optimal control hinge on an accurate process model that describes the system's nonlinear behavior. We defined a packed-bed reactor (PBR) as our ammonia

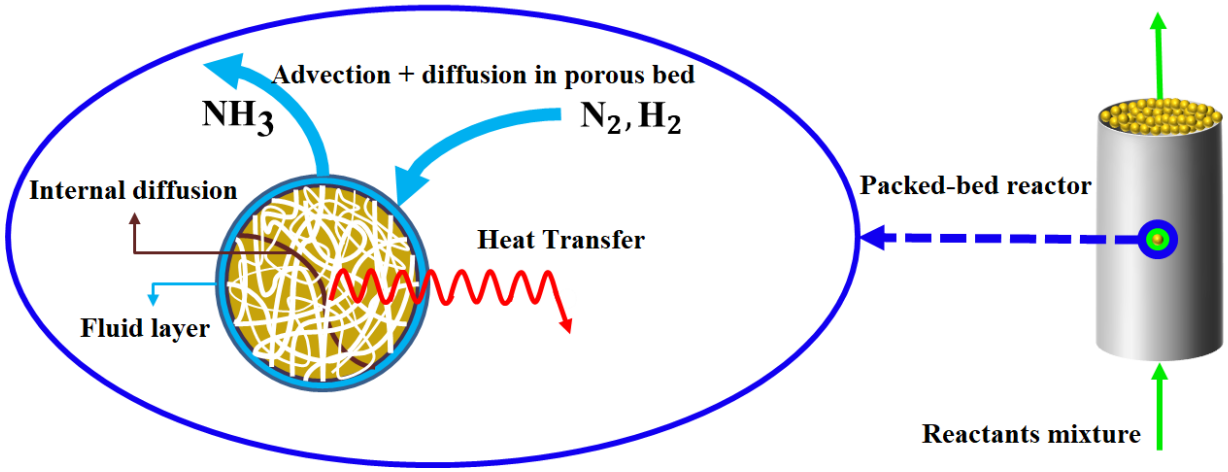


Figure 3.1: Schematics of Haber-Bosch process inside a PBR

synthesis system in this case study (Figure 3.1). We use heat, momentum, and mass balances to develop a detailed multi-scale model for this system. Combined with computational fluid dynamics (CFD) tools, we created a digital twin of ammonia synthesis that replicates the real process.

The implementation of digital twins facilitates a deeper understanding of systems, prediction of failures before they occur, and testing solutions in a virtual environment, thereby reducing costs and downtime while improving performance[100].

Remark 1 : The development of the multiscale model is another topic of our research group. We abstained from explaining the details of the multiscale model since this thesis focuses on time-series modeling of the ammonia synthesis PBR using LSTM. However, the details can be found in our recent paper

3.3 Problem Statement

The developed multiscale model is solved using the finite-element method incorporated in COMSOL multiphysics[101]. Despite the robustness and accuracy of this method, it is slow and cannot contend with the required response time of a real-time dynamic optimization algorithm. These limitations primarily stem from the high computational cost of solving

large-scale problems, which require significant memory and processing power, especially for three-dimensional models and nonlinear material behavior[102]. Additionally, the iterative nature of optimization processes and the detailed simulations required at each step exacerbate these computational challenges. As a result, the multiscale models are limited to real-time decision-making[103].

Deep learning's unique learning capabilities offer a new perspective to bridge the computational gap between multiscale modeling and real-time control and optimization. In this case study, we propose an efficient real-time optimization and control framework empowered by LSTMs. We employ time-series modeling using LSTMs to address the computational limitations of the FEM methods, resulting in a fast and accurate surrogate model of the system. Furthermore, we utilize both the surrogate and multiscale models in an MPC framework to control maximum temperature (T_{max}) and outlet ammonia concentration (C_{NH_3}) by manipulating removed heat (Q) and inlet flow (F). Integrating the surrogate model for MPC optimization and the multiscale model for MPC implementation will result in successful real-time control of the ammonia synthesis PBR.

3.4 Data generation

Gathering sufficient process data is crucial for training the surrogate model. Capturing all possible nonlinear behaviors within the process data is key to ensuring the surrogate model's accurate optimization performance. We generated this dataset using offline CFD simulations of the ammonia synthesis reactor executed in COMSOL Multiphysics across a comprehensive array of input parameters. We discretized the removed heat Q in an array varying from 0 to 100 kW/m^2 in 5 kW/m^2 increments, and the flow rate F from 50 to 1000 L/min, with each increment being 50 L/min. To ensure steady-state conditions, each simulation was executed for 1000 minutes. This procedure generates a dataset containing 420 distinct dynamic simulations representing the complex dynamics of the ammonia synthesis

process in the packed bed reactor.

Figure 3.2 shows the outlet ammonia averaged concentration under fixed inlet flow (600 $\frac{L}{min}$). This figure emphasizes the thermodynamic and kinetic effects of the reaction on ammonia production. The smaller removed heat values result in lower steady-state concentration conditions.

For maximum temperature, similar dynamics are observed across different values of removed heat; however, each decrease in removed heat is associated with higher maximum temperature dynamics (Figure 3.3). This trend is explained by the reduced efficiency in heat dissipation. As less heat is extracted from the system, the residual energy accumulates, resulting in elevated maximum temperatures.

On the other hand, Figures 3.4 and 3.5 show the behavior of maximum temperature and outlet concentration under different inlet flows and fixed removed heat. In concentration, higher flow rates cause a faster steady-state with more steep concentration peaks. The significant peaks observed in the outlet ammonia concentration result from the accumulation of ammonia within the reactor before the bulk fluid arrives at the outlet. This is anticipated due to the continuous production of ammonia and the fact that the outlet is not reached instantaneously. In maximum temperature, higher values are observed with smaller flow rates; this is due to the slower flow velocity that results in increased residence time.

3.5 Irregular time-series and bimodal data distribution

Statistical analysis of the offline simulation data is the first step in the data processing. The offline simulation data is expected to have 4.2×10^5 data points; however, early analysis shows that the generated dataset only contains 1.2×10^5 data points. We realized that the temporal profiles of the generated dataset are not in constant time steps. This reveals that the computational limitations of the FEM model not only make multiscale models

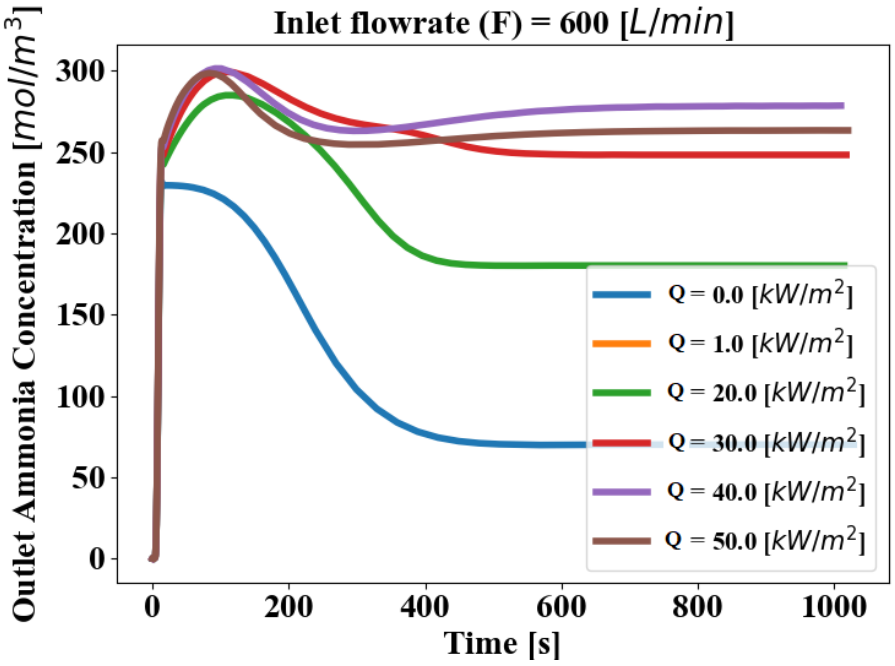


Figure 3.2: Outlet Ammonia concentration values in fixed F = 600 $\frac{L}{min}$

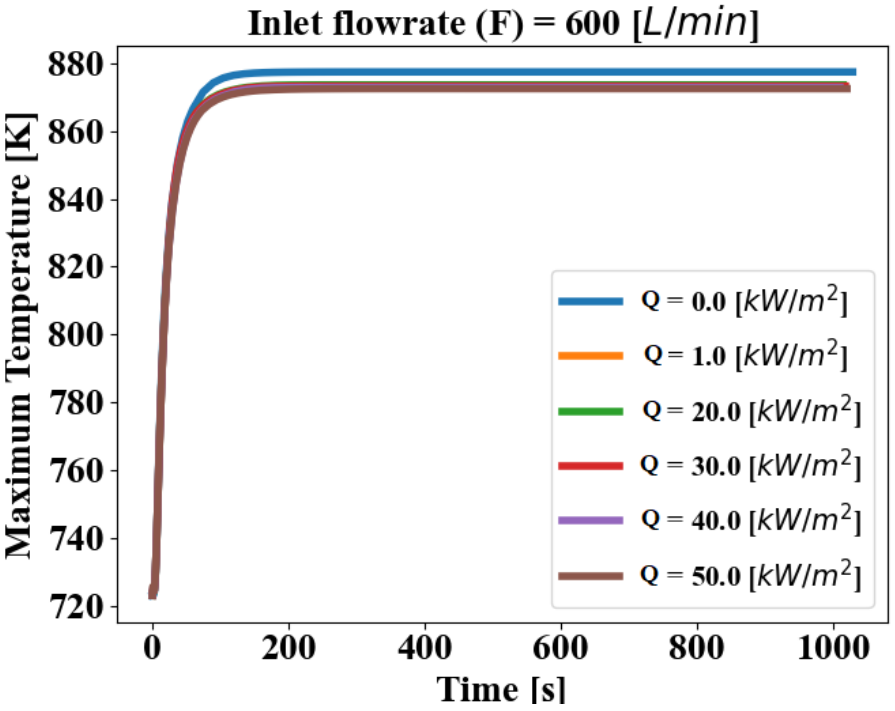


Figure 3.3: Maximum Temperature values in fixed F = 600 $\frac{L}{min}$

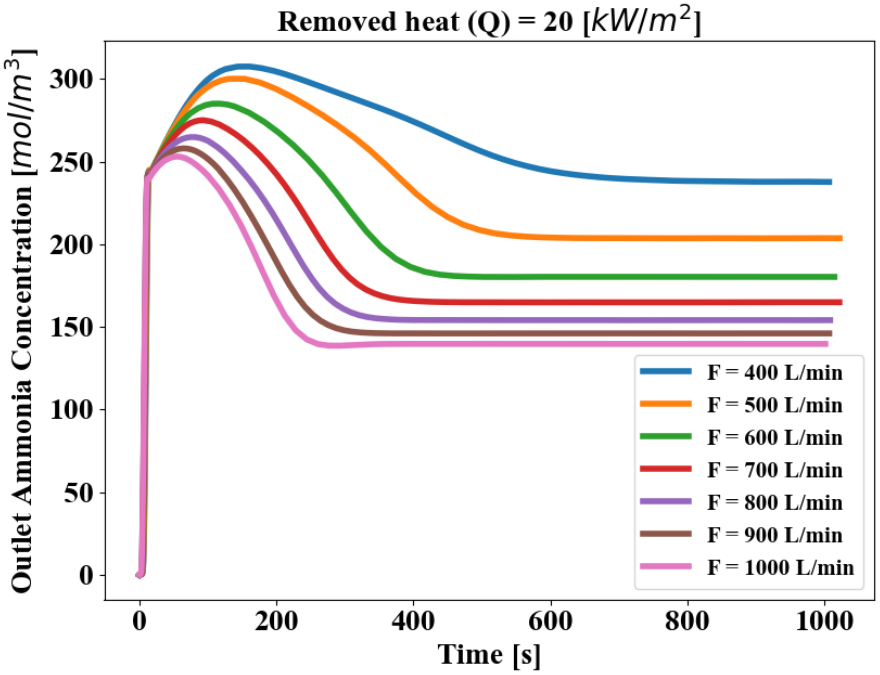


Figure 3.4: Outlet Ammonia concentration values in fixed Q = 20 $\frac{kW}{m^2}$

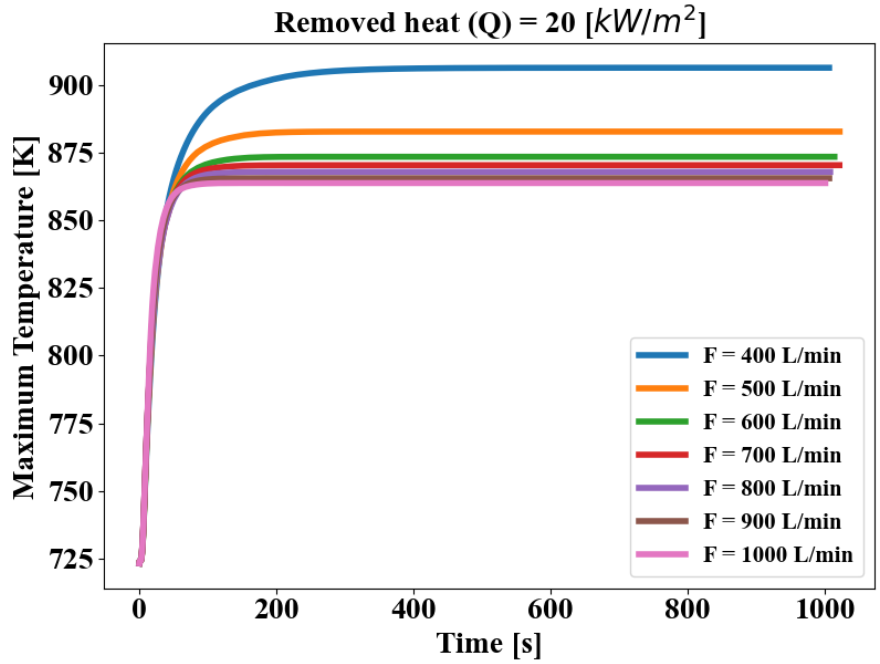


Figure 3.5: Maximum Temperature values in fixed Q = 20 $\frac{kW}{m^2}$

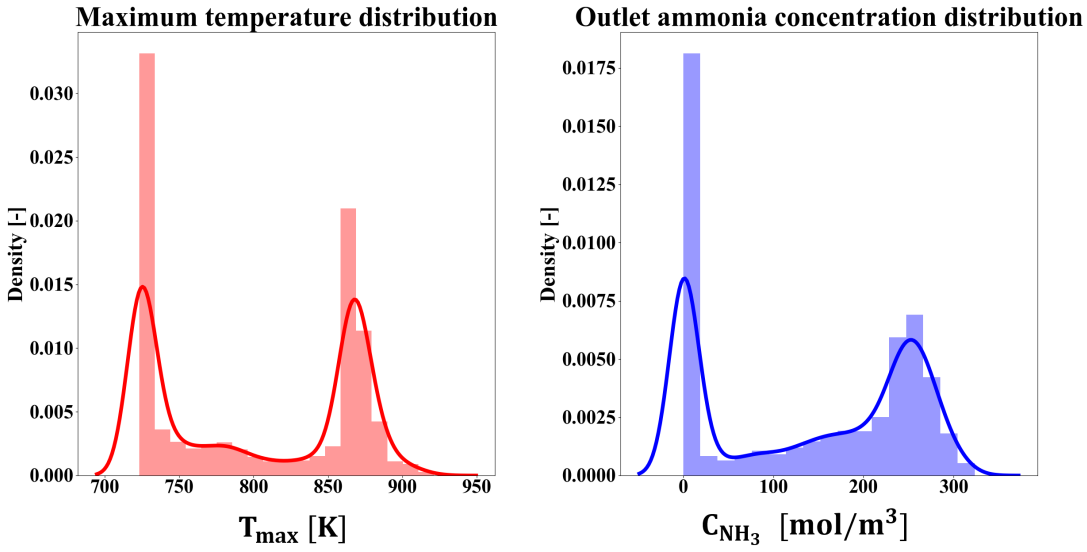


Figure 3.6: Bimodal distribution of the control objectives in the ammonia production reactor.

unsuitable for real-time decision-making but also introduce another limitation for proper data generation for surrogate model training.

From a time-series perspective, the nonuniform timestep distribution is considered an irregular time-series problem. Training LSTMs on irregular datasets would result in poor and biased performance, depending on the statistical distribution of the data. As a result, the simulation data display a dual-peaked distribution, known as bimodal distribution[104], depicted in the distribution plots for each controlled output in Figure 3.6. The initial peak reflects the early dynamics of the reaction process, whereas the second peak represents the phase where the system stabilizes into a steady state. These two patterns introduce a notable bias in the training of LSTMs, potentially compromising the accuracy of the surrogate model. This complication stems from the LSTM model's attempt to concurrently learn the sequential patterns evident in both peaks, which can obscure the distinctness of each pattern. To facilitate effective training, validation, and evaluation of the LSTM-based surrogate model, it is critical to transform this irregular dataset into a regular time-series dataset where the time steps are consistent[103,105].

3.6 Gappy data regenerator

To create a uniform time-series dataset, we devised a data regeneration strategy known as Gappy Data Regenerator (GDR), essentially an FNN trained on the simulated data. The (GDR) model is structured with seven fully connected layers. To avoid overfitting, the architecture includes two dropout layers and uses "ReLU" and "leaky ReLU" activation functions[106,107]. The Mean Squared Error (MSE) is the loss function, with the ADAM optimizer[108,55] applied for model optimization.

3.6.1 GDR training

To reshape the time-series problem into a supervised learning challenge, the time variable (t) is integrated alongside other parameters as an independent variable in the input vector. Consequently, $\mathbf{I} = [t, \mathrm{Q}, \mathrm{F}]$ denotes the GDR's input features, and $\mathbf{O} = [T_{\max}, C_{\mathrm{NH_3,outlet}}]$ represents the output vector. The dataset is split into 80% for training and 20% for validation, with all data standardized. For optimal training, we implemented k-fold cross-validation ($k = 5$) for enhanced validation. The GDR is trained over 36 epochs, incorporating an early stopping criterion to prevent overtraining.

Additionally, inferences are drawn using constant time steps ($dt = 1$) with the trained GDR, generating a dataset of high temporal resolution, which aligns with the input parameters used in the COMSOL simulations. Despite being based on offline simulations, the regenerated dataset presents uniform multivariate time series process data. The entire data preprocessing pipeline, alongside the network's configuration, training, and data regeneration inferences, is executed using the TensorFlow library in Python.

3.6.2 Regularizing simulation data

Interpolating simulation data using the GDR method regularizes the time-series process data. As depicted in Figure 3.7, the bimodal distribution of $T_{\max}$ is normalized, whereas for

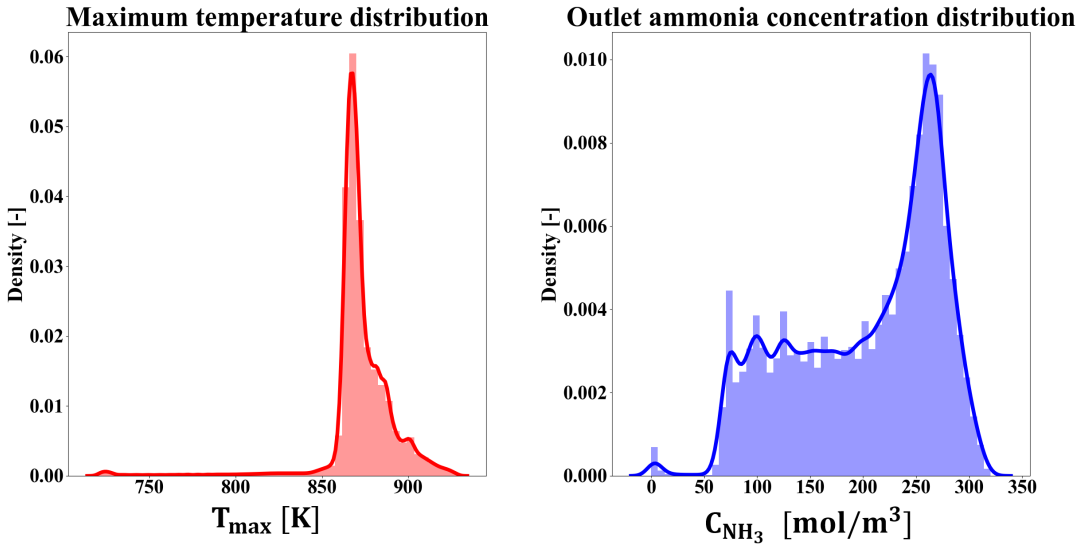


Figure 3.7: Regenerated distributions of the control objectives in the ammonia reactor.

$C_{\mathrm{NH_3,outlet}}$, the principal peak illustrates the dominant dynamics of $C_{\mathrm{NH_3,outlet}}$ throughout the synthesis process. Moreover, the regenerated dataset guarantees a normal data distribution, representing the physical dynamics of the reactor. It adeptly balances the normalization of data with the preservation of critical features. The regenerated dataset contains 4.2×10^6 time-series data points, thoroughly encompassing the full dynamics of the ammonia synthesis reactor.

3.6.3 GDR evaluations

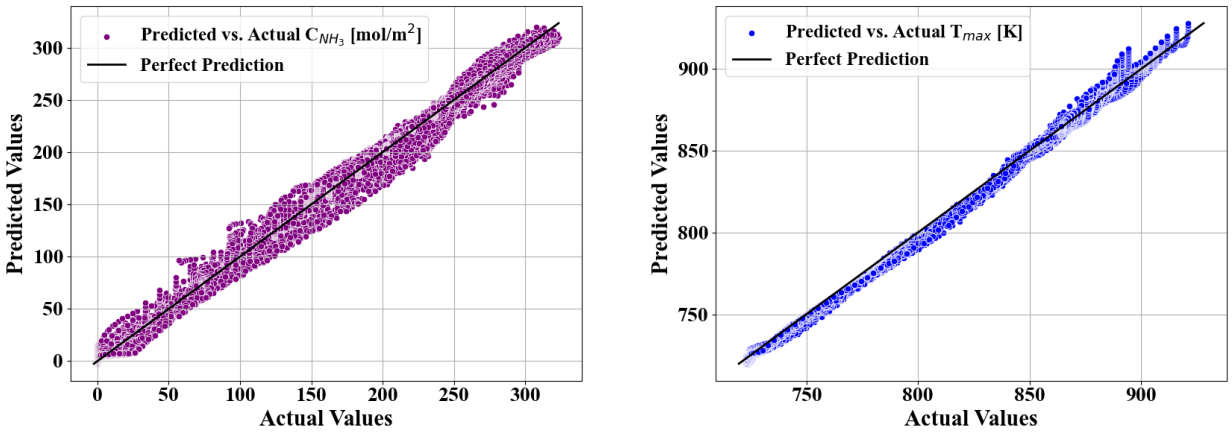
The performance of the GDR model is critical for further development of the real-time control framework since the associated errors would be propagated through the surrogate model training. Table 3.1 shows the error metrics of the developed GDR model.

Table 3.1: GDR Performance Metrics for System Outputs

System Outputs	RMSE	MASE	MAE
$C_{\mathrm{NH_3,outlet}}$	7.62	1.15	5.26
T_{max}	2.85	0.85	2.12

The error metrics show that the GDR model interpolates maximum temperature with better performance compared to outlet concentration. As observed in Figure 3.2,3.4, outlet concentration exhibits more nonlinear behavior than maximum temperature, making it more challenging for the GDR model to interpolate. Overall, the scores show minimal propagated error to surrogate model training.

For further analysis, Figure 3.8 shows the parity plots for the GDR model to visually evaluate its performance. The parity plots show that the predictions are aligned with the perfect prediction line.



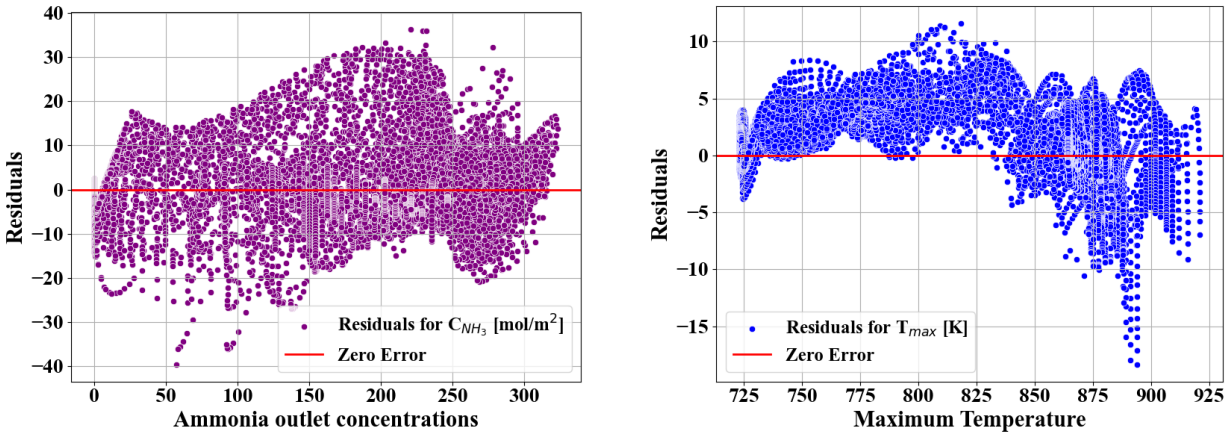
(a) Parity plot for concentration

(b) Parity plot for temperature

Figure 3.8: Parity plots of GDR model predictions.

Despite the parity plots exhibiting consistent model performance, pinpointing the precise conditions under which the GDR model manifests suboptimal performance remains challenging. To address this issue, we employ residual analysis (Figure 3.9) as a strategic approach to identify the instances of the GDR model's poorest performance. This methodology enables us to quantitatively assess the maximum error propagation affecting the surrogate model's training process, thereby enhancing the robustness and reliability of the model's predictive capacity.

Despite employing k-fold cross-validation, the residual analysis reveals outlier predictions by the GDR model concerning maximum temperature and outlet ammonia concentra-



(a) Residual plot for concentration

(b) Residual plot for temperature

Figure 3.9: Residual plots of GDR model predictions.

tion. Further error calculations disclose the maximum percentage error propagated to the surrogate model training under the worst-case scenarios of GDR predictions. Specifically, this value for maximum temperature is 9.27%, and for outlet ammonia concentration, it is 12.27%. It is important to note that these values account for only a single prediction, representing the worst prediction of the GDR model.

3.7 LSTM surrogate model training

3.7.1 Network Architecture

By obtaining a regularized process dataset using GDR, we can train an LSTM-based surrogate model that reduces the computational complexity of the multiscale models. We use the encoder-decoder architecture (Figure 3.10), where stacked LSTM layers are used for the encoder, and a fully connected layer is used for decoding the time-series predictions. The LSTM layers in the encoder part include three stacked LSTM hidden layers activated by the *tanh* function.

The encoder-decoder architecture learns the temporal patterns by processing previous data $(t-1, t-2, \ldots, t-k)$ to predict future data $(t+1, t+2, \ldots, t+m)$. The number of

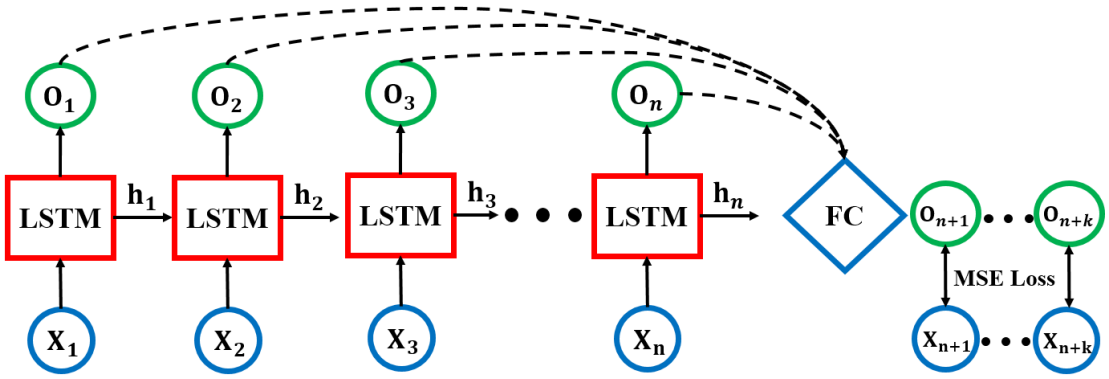


Figure 3.10: Encoder-decoder architecture with LSTM and fully connected layers for multi-step time-series prediction

the previous data points is often referred to as memory length (k), while the number of the future predicted steps is called the prediction horizon (m) (Figure 3.11). Usually, models with a prediction horizon of one ($m = 1$) are regarded as single-step ahead models, while models with a prediction horizon exceeding one ($m > 1$) are known as multi-step ahead models.

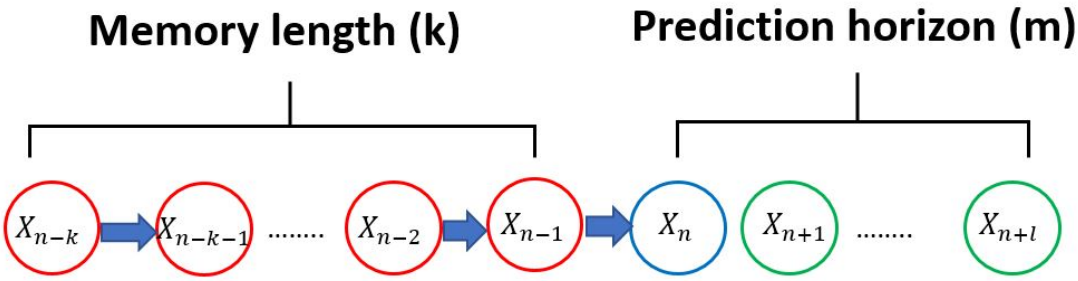


Figure 3.11: Time-series sequence with memory and prediction horizon

3.7.2 Single-step and multi-step ahead model configurations

we explored and analyzed the performance of single-step and multi-step ahead surrogate modeling with varying memory lengths to achieve the most accurate settings for training the LSTM-based surrogate model. we inspected the impact of varying memory lengths, specifically $k = [5, 10, 20]$, on the performance of both single-step ($m = 1$) and multi-step

($m = [5, 10, 15]$) predictive models. While single-step models are noted for their precision, their integration with MPC necessitates dependence on values predicted in prior steps for determining MPC prediction horizons. This dependency often results in unwanted error propagation within the MPC framework. Conversely, multi-step models offer a solution to this challenge by aligning the count of forecasted steps directly with the MPC prediction horizon. For further analysis of these surrogate models, we trained all combinations of memory lengths and prediction horizons under the constraint that the prediction horizons would not exceed the memory lengths. Table 3.2 presents different model parameters.

Table 3.2: Model Parameters based on memory length and prediction horizon

Index	**Memory length (k)**	**Prediction horizon (m)**	**Trainable Parameters**
1	5	1	638,702
2	5	5	644,710
3	10	1	640,202
4	10	5	652,210
5	10	10	667,220
6	20	1	643,202
7	20	5	667,210
8	20	10	697,220
9	20	15	727,230

3.7.3 Training

For smooth integration of the surrogate model and MPC, we transform the process data into deviations from the steady-state conditions, and the data is normalized using MinMax scaling[109]. Additionally, the LSTM model's inputs are defined as $I_{\text{LSTM}} = [U_1', U_2', X'_{1(t-k)}, X'_{2(t-k)}]$, with outputs $O_{\text{LSTM}} = [X'_{1(t+1)}, X'_{2(t+1)}]$, where $U_1' = Q - Q_{\text{SS}}$, $U_2' = F - F_{\text{SS}}$, $X_1' = T_{\text{max}} - T_{\text{max, SS}}$, and $X_2' = C_{NH_3} - C_{NH_3,\text{SS}}$. the system states (X_1', X_2') are dependent variables, indicating that their future values $(t+1)$ have a temporal correlation with past sequence values $(t-n)$.The LSTM-based surrogate model interprets this multi-input multi-output (MIMO) system as a task of multivariate parallel time-series regression task.

We employed the ADAM Optimizer[55] alongside the mean squared error loss function with 30 epochs and a batch size of 32 for training each LSTM-based surrogate model. Furthermore, we used Hyperband optimization for hyperparameter tuning[110]. TensorFlow/Keras with Optuna was used for training and inference implementations[111,112].

3.8 Surrogate model performance evaluations

Precise dynamic optimization and real-time control of the ammonia reactor hinges on the performance of the developed surrogate models. To find the best model, we first analyzed each model's general performance to identify the optimal memory length and prediction horizon. We employed the standard metrics to find the best single-step and multi-step ahead models.

3.8.1 Single-step ahead models

Table 3.3: Single-step ahead LSTM-based model performance metrics for $T_{\max} - T_{\max,\,ss}$

Memory Length	RMSE	MASE	MAE
5	18.41	9.34	13.13
10	28.90	7.88	20.93
20	16.77	2.66	11.59

Table 3.4: Single-step ahead LSTM-based model performance metrics for $C_{NH_3} - C_{NH_3,ss}$

Memory Length	RMSE	MASE	MAE
5	34.86	9.32	23.78
10	28.18	4.38	19.71
20	55.70	6.34	40.09

Tables 3.3 and 3.4 present the error metrics for the prediction of maximum temperature and ammonia outlet concentration steady-state deviations, respectively. MASE facilitates the assessment of model performance across both targets on a unified scale. The analysis reveals that the single-step model with a memory length of twenty ($k = 20$) exhibits the

minimum average MASE value, indicating superior predictive accuracy. However, when considering the RMSE, this superiority is not uniformly observed across both targets; it registers the maximum RMSE value for the outlet concentration predictions. Considering both the MASE for consistency in performance evaluation and the RMSE for predictive accuracy, the model with a memory length of ten ($k = 10$) demonstrates overall better performance.

3.8.2 Multi-step ahead model

Memory Length	Prediction Horizon	RMSE	MASE	MAE
5	5	8.07	5.06	4.95
10	5	13.64	6.41	11.06
10	10	9.31	3.53	6.09
20	5	4.55	1.24	3.44
20	10	11.36	3.02	8.38
20	15	8.86	2.16	5.99

Table 3.5: Multi-step ahead LSTM-based model performance metrics for $T_{\text{max}} - T_{\text{max, ss}}$

Memory Length	Prediction Horizon	RMSE	MASE	MAE
5	5	27.02	20.76	20.71
10	5	21.47	8.41	15.76
10	10	35.34	13.74	25.74
20	5	43.22	8.89	31.62
20	10	31.54	6.26	22.27
20	15	43.69	9.00	32.02

Table 3.6: Multi-step ahead LSTM model performance metrics for $C_{NH_3} - C_{NH_3,\text{ss}}$

Tables 3.5 and 3.6 present error metrics for multi-step ahead LSTM surrogate models. The MASE analysis indicates that the LSTM model with a memory length of twenty and a prediction horizon of ten ($k = 20, m = 10$) outperforms the other multi-step ahead models. However, when considering the RMSE, it becomes apparent that this model has a relatively higher error for outlet ammonia concentration. Considering both MASE and RMSE metrics,

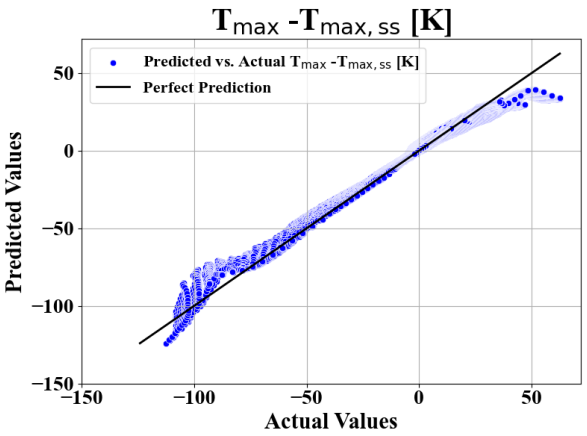
the model with a memory length of ten and a prediction horizon of five ($k = 10, m = 5$) demonstrates more effective performance for both outputs.

3.8.3 Parity plots

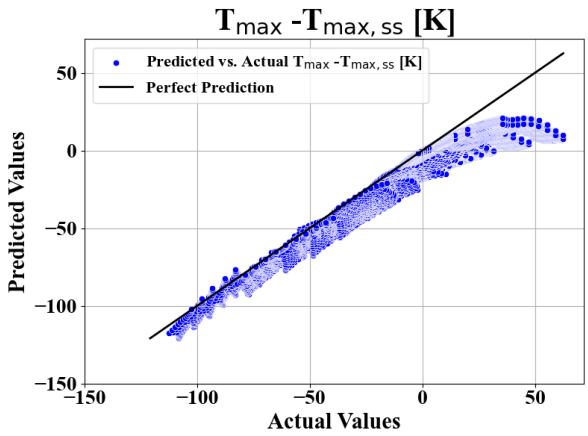
The parity plots demonstrate the overall accuracy of the LSTM models across all parallel time series. Figure 3.12 displays the parity plots for single-step and multi-step ahead models concerning the steady-state deviations of maximum temperature. The predictions from the single-step ahead model closely align with the ideal prediction line, in contrast to the multi-step ahead models, where the predictions diverge more significantly from the perfect prediction. This divergence results from the multi-objective training phase inherent to the multi-step ahead models, highlighting a trade-off between accuracy and the complexity of forecasting over extended steps. The LSTM models demonstrate different behaviors in predicting concentrations. Figure 3.13 showcases the parity plots for the steady-state deviations of outlet ammonia concentration. The multi-step ahead model outperforms its counterparts. In this context, the multi-objective training proves advantageous, effectively capturing the complex dynamics of ammonia concentration. It is crucial to emphasize that, unlike maximum temperature, the transient states of outlet ammonia concentration are inherently complex, making predictive data-driven time-series modeling more challenging.

3.8.4 Residual plots

Figure 3.14 presents the residual plot for the steady-state deviations of maximum temperature, corroborating earlier observations from the parity plots (Figure 3.12) that suggest single-step models exhibit lower error margins for maximum temperature. Furthermore, Figure 3.15 displays the residual plots for steady-state deviations of ammonia outlet concentrations. The residual analysis indicates that the multi-step model exhibits lower error margins. This difference is attributed to the complex transient behavior of the ammonia synthesis reactor, which aligns well with the multi-objective nature of the multi-step models,

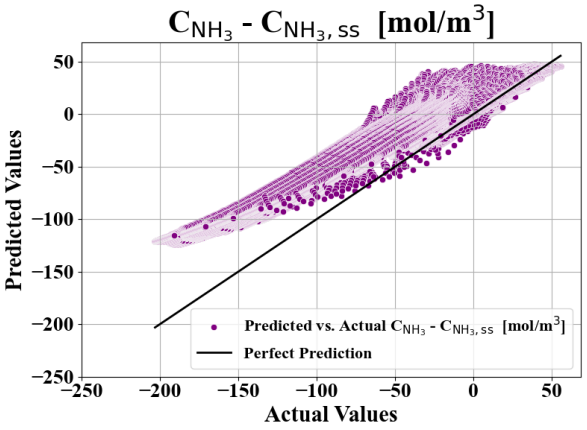


(a) Single-step ahead ($k = 10$).

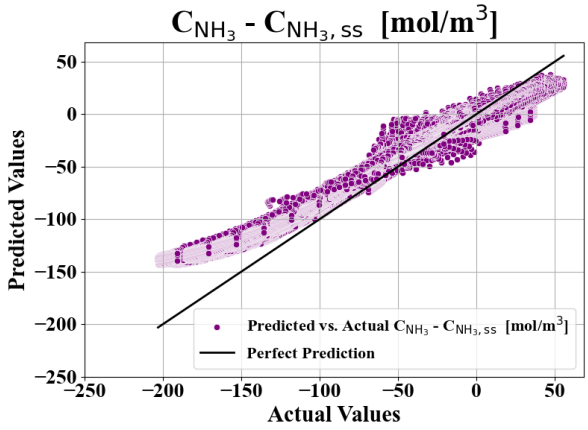


(b) Multi-step ahead ($k = 10$,$m = 5$).

Figure 3.12: LSTM models parity plots for $T_{\max} - T_{\max,\ ss}$



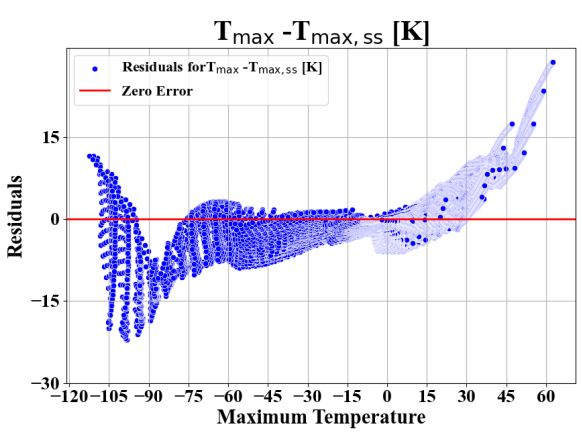
(a) Single-step ahead ($k = 10$).



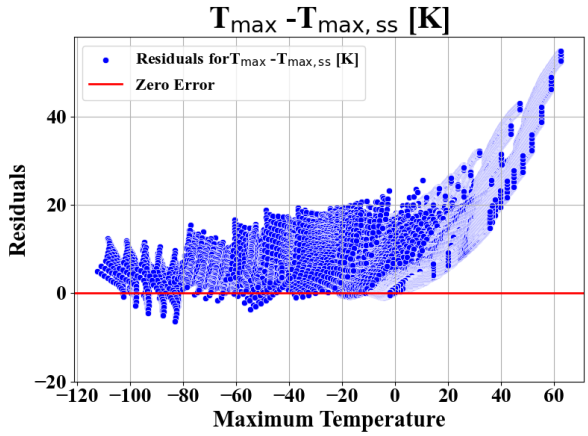
(b) Multi-step ahead ($k = 10$,$m = 5$).

Figure 3.13: LSTM models parity plots for $C_{NH_3} - C_{NH_3,\text{ss}}$.

facilitating more accurate predictions. Additionally, the regenerated distributions (Figure 3.7) for outlet ammonia concentrations are not entirely Gaussian form, leading to increased error propagation during LSTM model training for outlet ammonia concentrations.

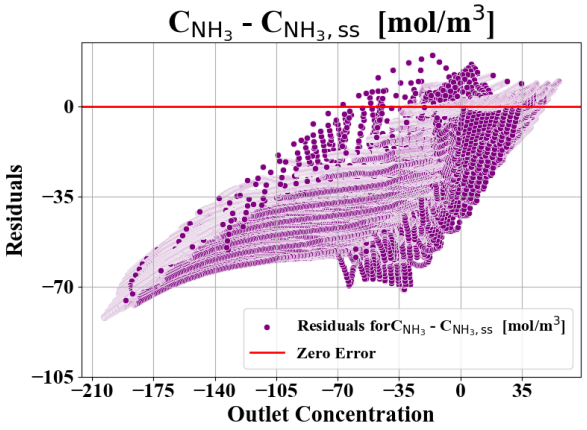


(a) Single-step ahead ($k = 10$).

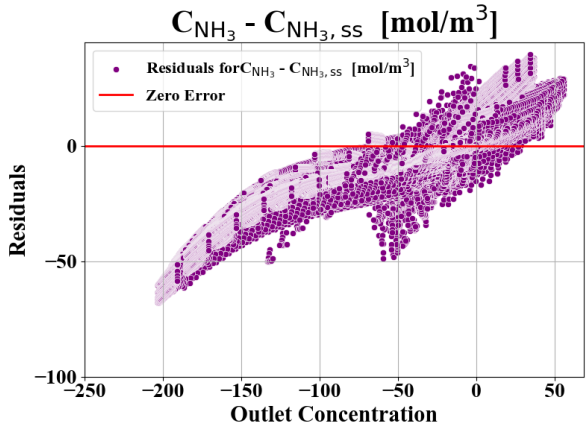


(b) Multi-step ahead ($k = 10$,$m = 5$).

Figure 3.14: LSTM models residual plots for $T_{\max} - T_{\max,\ ss}$.



(a) Single-step ahead ($k = 10$).



(b) Multi-step ahead ($k = 10$,$m = 5$).

Figure 3.15: LSTM models residual plots for $C_{NH_3} - C_{NH_3,\text{ss}}$

3.9 LSTM-MPC framework

The final step of the proposed real-time dynamic optimization framework for this case study is the integration of surrogate models with Model Predictive Control. It is critical to mention that relying solely on the surrogate model for MPC might raise safety concerns. As indicated in the evaluation section, there is an error margin in the surrogate model's predictions.

To address this, we proposed an LSTM-based MPC method, termed LSTM-MPC, for ammonia synthesis reactor control that also leverages the multiscale model. The schematic

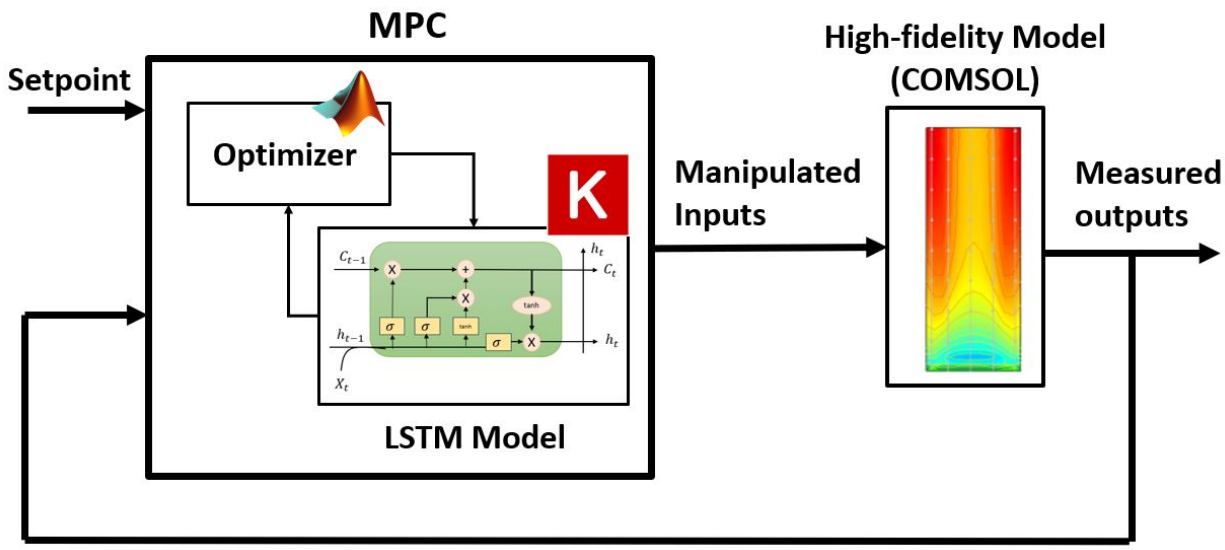


Figure 3.16: Multi-platform computational framework developed to control the ammonia reactor in real-time.

of this framework is illustrated in Figure 3.16.

In MPC, the computational bottleneck occurs during optimization; thus, we utilized the LSTM surrogate model as a rapid system replica for optimization purposes. This step identifies the optimal manipulated inputs in the system space, used as the system input in the implementation phase. This phase involves a singular computation of the system; hence, it is not computationally demanding. This allows for using the multiscale model to compute the outputs that determine the current system states. These states are further assessed to conduct optimizations in the future. Maintaining the multiscale model alongside the surrogate model corrects unintended errors by the surrogate model and ensures the safety of the LSTM-MPC framework.

We conducted closed-loop simulations using LSTM-MPC with a sampling rate of one, utilizing both single-step and multi-step models. Figure 3.17 illustrates the closed-loop simulations of the single-step models with an MPC prediction horizon of 10. The system outputs smoothly and moves toward a steady state. However, using single-step models with an MPC prediction horizon of more than one causes more oscillations in the manipulated

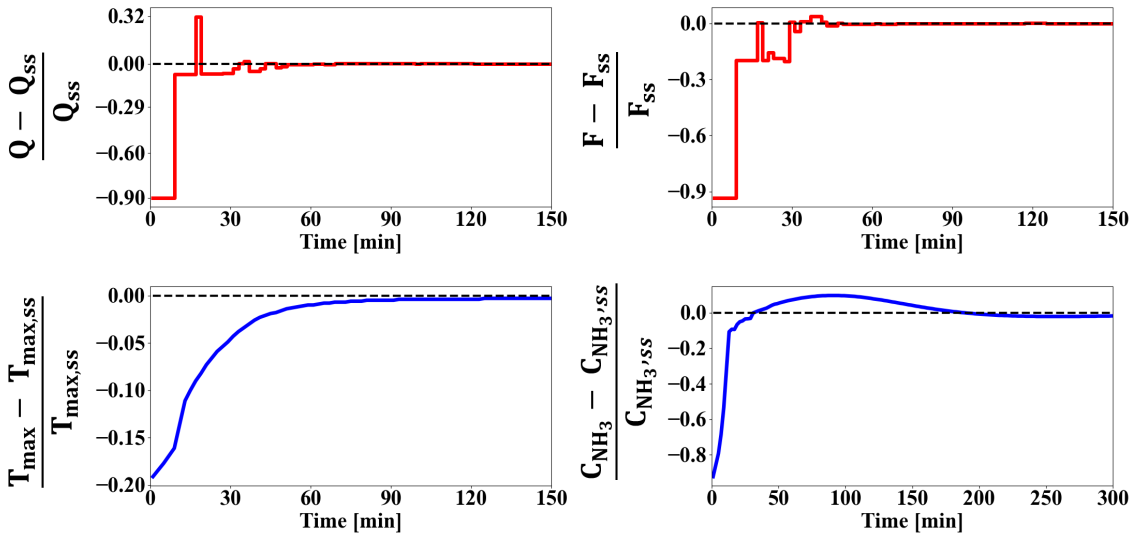


Figure 3.17: Closed-loop simulation with single-step LSTM model

inputs. Furthermore, employing the same MPC configurations with a multi-step model resulted in less oscillatory manipulated inputs, as shown in Figure 3.18.

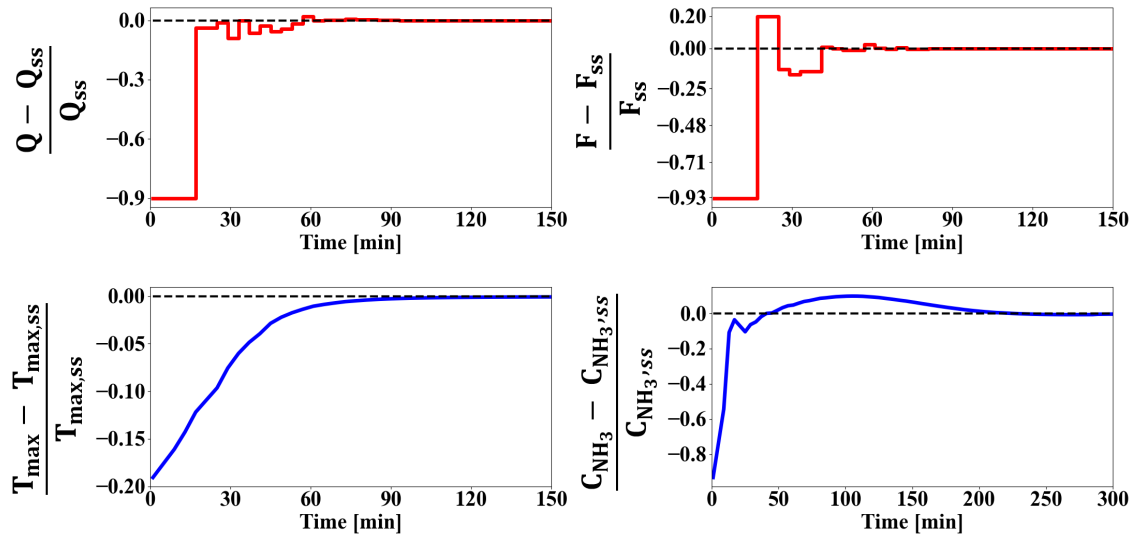


Figure 3.18: Closed-loop simulation with multi-step LSTM model

3.10 Conclusion and future work

Sustainable ammonia synthesis using renewable energies depends on efficient real-time dynamics and optimal control frameworks to manage renewable energy sources while ensuring continuous production. Developing these frameworks necessitates a detailed understanding and analysis of the industrial-scale ammonia synthesis process. This advanced analysis is achieved through CFD simulations of the multiscale model of the Haber-Bosch process for a packed-bed reactor, known as the CFD digital twin of the ammonia synthesis process. The multiscale model yields an accurate and robust analysis of the intricate behavior of the system; however, its computational demands limit its real-time application.

In this chapter, we demonstrated how deep learning time-series modeling offers a promising approach to reducing the computational complexities of digital twins, further enabling their integration for optimization and control purposes. For training single-step and multi-step ahead LSTM surrogate models, we initially utilized offline CFD simulations from a high-fidelity model across 420 different operational conditions. However, subsequent data analysis revealed that computational limitations forced the FEM solver to simulate the reactor with inconsistent time steps, resulting in an irregular time series dataset and a bimodal data distribution. The statistical distribution indicated that the computational bottleneck occurs when computing the transient states of the reactor, leading to data being more concentrated at the beginning of the reaction and at steady-state conditions. This concentration could hinder the ability of LSTM models to observe and learn the transient dynamics of the reactor. To address this issue, we introduced a gappy data generator to regularize the offline CFD simulations. Furthermore, we trained the LSTM models for a multivariate parallel time series task to predict the maximum temperature and outlet ammonia concentration steady-state deviations.

The performance evaluations yielded intriguing results. The overall performance metrics suggest that models with a memory length of ten ($k = 10$) provide better accuracy and consistency. We also observed that the single-step ahead model outperforms the multi-step

ahead model for maximum temperature steady-state deviations; however, the multi-step ahead models demonstrate lower error margins for ammonia concentrations. Subsequently, we integrated the surrogate models into MPC optimization, with the digital twin in MPC implementation acting as the real process. The closed-loop simulations indicate smooth convergence to the control trajectories.It is crucial to note that the presence of the multiscale model in MPC implementation does not pose computational challenges and also corrects any potential prediction errors, enabling safer implementations of the proposed real-time dynamic optimization framework.

Adopting data-driven surrogate modeling techniques facilitates real-time control of ammonia synthesis. However, there's room for improvement in these models' performance. One potential avenue for exploration is hybrid modeling[113,114,115]. In this approach, instead of training LSTM models to represent the entire process, we train them to mimic the nonlinear operators of the multiscale model. The existing data parallelization capabilities of neural networks lead to a faster resolution of the nonlinear operators, thereby reducing the computational demands of the CFD model.

Physics-informed neural networks (PINNs) present another robust approach to surrogate modeling[116,117,118,119]. In this methodology, LSTM networks are trained not solely on process data but also incorporate the physical principles governing the reactor. This is achieved by including the model's equations within the training phase and utilizing a cumulative loss function that accounts for both the process data and the equations of the physical model. This approach is anticipated to yield superior accuracies compared to purely data-driven methods, owing to its integration of the system's underlying principles.

Chapter 4

Time-series modeling of soil water content for agricultural decision making

The impact of global warming has been imposing severe climatic patterns, causing extensive droughts and water shortages, and exerting detrimental effects on ecological processes[120]. Soil water content (SWC) is a crucial parameter for understanding the environmental and hydrological processes within terrestrial ecosystems that are susceptible to the impacts of global warming. SWC directly affects the soil's physical and chemical properties, plant growth, and microbial activity, thereby influencing agricultural productivity and environmental quality[121,122]. Therefore, the unprecedented climatic patterns cause fluctuations in SWC, which implicitly impact irrigation, crop growth, and agricultural productivity. The accurate measurement and modeling of SWC are thus fundamental for water resource management, agricultural planning, and climate change adaptation strategies. In recent years, advanced remote sensing techniques have significantly improved the spatial and temporal resolution of SWC measurements, offering new insights into SWC dynamics at global scales[123].

4.1 Problem Statement

On-site SWC measurements and dynamic modeling are essential needs for real-time agricultural decision-making. Several sensing methods have been proposed for SWC measurements[124]. On the other hand, the combination of hydrological concepts with transport phenomena has pioneered a modeling framework for SWC.

4.1.1 Richards Equation

The Richards equation, a fundamental partial differential equation in soil physics, describes the movement of water in unsaturated soils. Developed by L.A. Richards in 1931, this equation revolutionized the understanding of unsaturated zone hydrology by providing a mathematical framework to describe the complex dynamics of water flow through soil pores under varying water content conditions[125]. The Richards equation is based on Darcy's law and the conservation of mass, and it accounts for the non-linear relationship between the soil's hydraulic conductivity and the water content. This non-linearity poses significant challenges for analytical and numerical solutions, making the equation a focus of research for hydrologists and soil scientists aiming to model soil water dynamics accurately. The importance of the Richards equation extends beyond theoretical hydrology; it is crucial for predicting water movement in agricultural fields, managing irrigation efficiently, and understanding the ecological impacts of water flow.[126,127].

The general form of the Richards equation is given by:

$$\frac{\partial \theta}{\partial t} = \nabla \cdot (K(\theta) \nabla h) + S \tag{4.1}$$

where

- θ is the volumetric water content,
- t is time,

- $K(\theta)$ is the hydraulic conductivity as a function of water content,
- h is the pressure head,
- S represents sources or sinks of water (e.g., root water uptake, evaporation).

While the Richards equation provides a comprehensive mathematical framework for modeling water flow in unsaturated soils, its practical application in real-time agricultural decision-making faces significant limitations. The equation's complexity, stemming from its non-linear partial differential nature, requires sophisticated numerical methods for its solution, making real-time analysis challenging[127]. Additionally, the accurate parameterization of soil hydraulic properties, such as hydraulic conductivity and soil water retention characteristics, which are highly variable in space and time, poses a substantial hurdle[128]. These properties are often difficult to measure directly in the field with the precision required for the Richards equation, leading to uncertainties in model predictions. As a result, simpler models or empirical approaches are often preferred for operational use, where speed and ease of use are important[129]. Despite its theoretical importance for understanding soil water dynamics, these significant challenges limit the practical application of the Richards equation in real-time agricultural decision-making.

To overcome this challenge, Markov Decision Processes (MDP) have been employed to create a simplified model for estimating soil water storage[40].

4.1.2 Markov-based soil water content estimation

The Markov-based model significantly simplifies the Richards equation, offering a more practical approach to SWC predictions. By leveraging the principles of Markov chains to model the stochastic nature of SWC dynamics, these models reduce the computational complexity associated with the non-linear partial differential equations of the Richards model. Markov-based models treat SWC as a random process, where the future state of SWC depends only on its current state and not on the sequence of events that preceded it[130,131,132].

This simplification allows for faster computations and easier parameterization, making it well-suited for real-time agricultural decision-making. Unlike the Richards equation, which requires detailed information on soil hydraulic properties and extensive computational resources, Markov-based models can provide quick and reasonably accurate SWC estimates using readily available weather data and basic soil characteristics. As a result, Markov-based models have become a valuable tool for farmers and agricultural managers, enabling them to make informed decisions based on the predicted SWC levels with a greater emphasis on operational practicality and less on computational intensity.

Equation (4.2) represents the conventional formulation of the Markov chain-based model accounting for precipitation, seasonal water fluctuations, and climate impact on soil water storage[133].

$$S_t = min\big(S_{min} + \alpha(S_{t-1} - S_{min}) + P_t,\ S_{max}\big) \tag{4.2}$$

where S_t describes water storage during time t, α is the loss coefficient, and P_t represents total precipitation as a function of time, and S_{min} and S_{max} are the lower and upper bounds of the water storage, respectively. The loss coefficient denotes all the water losses caused by runoff and drainage, which can be estimated throughout the year

$$\alpha = c + (1 - c)\ sin\big(2\pi\frac{DOY - \phi}{365} + \frac{\pi}{2}\big) \tag{4.3}$$

where c is an empirical constant, ϕ represents a phase constant, and DOY is day of the year.

Markov-based models face limitations due to their simplifying assumptions. These models assume SWC's future state depends only on its current state, potentially overlooking the soil's water content memory effect, which can lead to inaccuracies, especially in water-retentive soils or under unusual climatic scenarios. Furthermore, their simplified representation of soil-water-atmosphere interactions may not accurately capture extreme conditions or spatial variability, limiting their precision for detailed agricultural applications[130].

4.1.3 ML-based SWC estimation

Addressing the shortcomings of Markov-based models, Researchers in the area have proposed several ML-based techniques to enhance the SWC estimation. For Instance, Prakash et al.[134] utilized the TAMU and OzNet SWC databases to create data-driven models for predicting SWC. They implemented various algorithms, such as support vector machines (SVMs), and multiple linear regression (MLR), training them on these databases to forecast SWC levels up to a week in advance. In a similar work, Singh et al.[135] developed predictive models for SWC using techniques like gradient boosting regression trees (GBRT), random forest regression (RFR), and elastic net regression (ENR), aiming to enhance intelligent irrigation practices. Furthermore, Wyatt et al.[136] employed Rosetta3[137] to refine soil moisture predictions within the Oklahoma Mesonet network. In another work, Karandish and Simunek[138] explored the efficacy of machine learning (ML) versus physical-based models in predicting soil water content. Additionally, Nguyen et al.[139] introduced data-driven soil moisture estimators for precision farming, leveraging relative vector machines (RVMs) and SVMs with data from micaZ and VH400 wireless sensor networks. Despite the robustness of the proposed ML technical, they don't consider the temporal correlations within SWC.

4.1.4 Proposed solution

In this case study, we utilize the time-series modeling perspective to propose a new approach for the dynamic modeling of SWC. By employing an encoder-decoder architecture for the time-series modeling of SWC, we addressed the limitations of the Markov-based and ML-based models. The proposed data-driven model considers the temporal correlations of historical SWC data for more accurate SWC predictions, offering a more advanced DL-based method that facilitates robust real-time agricultural decision-making.

4.2 Data-driven LSTM-based time-series modeling of SWC

For time-series modeling of SWC, we chose LSTMs as the encoder part and a fully connected layer as the decoder section of the model. Training this model requires a dataset that contains the full dynamics of soil water content. The dataset must include the full seasonality of the SWC values.

4.2.1 Time-series SWC dataset

The Kansas Mesonet is a crucial network of environmental monitoring stations spread across Kansas, providing real-time data on various atmospheric and soil parameters. Established to enhance the state's ecological monitoring capabilities, the Kansas Mesonet offers invaluable information for weather forecasting, agricultural planning, and research. This study used a Kansas Mesonet dataset for about 75 stations across Kansas. The dataset provides real-time meteorological data, including temperature, humidity, wind speed, precipitation, and root zone soil water content observations[41].

This dataset encompasses average daily data of recorded soil properties. In this study, we utilized the recorded data from January 2021 to February 2024, containing daily records for each station. Preliminary analysis of the data revealed that SWC values were not recorded for all stations. Therefore, we selected the stations where SWC data were available for the entire period to avoid further prediction bias. With this consideration, the dataset has a size of $5.2 * 10^3$ data points with 47 stations.

4.2.2 Data Preprocessing and feature engineering

The dataset analysis revealed missing SWC values in the daily sequence, resulting in irregular time series. Removing the irregular time series would result in losing valuable information,

which could benefit time-series modeling. To address this issue, we adopted a method of imputation that leverages available historical data. Specifically, we filled the gaps by calculating the weekly average of SWC values and used these averages to impute the missing data points. This approach ensures continuity and consistency within the dataset. Using historical averages as a basis for imputation, we mitigate the potential bias that missing values could introduce, ensuring a more robust model training.

The dataset comprises water content values at depths of 5 cm, 10 cm, and 20 cm. We implemented an aggregation method to predict soil water content within a soil profile. Equation 4.4 illustrates the correlation used to estimate soil water content at a depth of 20 cm:

$$S_{20} = 50\, V_{W5} + 50 \left(\frac{V_{W5} + V_{W10}}{2} \right) + 100 \left(\frac{V_{W10} + V_{W20}}{2} \right) \tag{4.4}$$

Where S_{20} signifies the soil water content at 20 cm depth, while V_{W5}, V_{W10}, and V_{W20} represent the soil volumetric water content at depths of 5 cm, 10 cm, and 20 cm, respectively.

The present features in this dataset are time, station name, precipitation, and SWC. To include the time seasonality, we compute the day of the year (DOY) and include it in the training features. Finally, the described models in the following section are trained to predict SWC values over time for each station.

4.2.3 Network Architecture

The LSTM-based model follows the encoder-decoder architecture, where two stacked LSTM layers with tanh activation functions comprise the encoder part, and the decoder part incorporates a fully connected layer. This case study aims to assess the efficiency of single-step and multi-step ahead models for SWC estimation. Thus, we aim to predict future week values by considering a week ($k = 7$) as the memory length. The models are trained over a prediction horizon vector that ranges from 1 to 7, with one-step increments. Each model's configurations and sizes are detailed in Table 4.1. The encoder-decoder architecture is adept

at learning the seasonality, trends, and hidden temporal patterns in SWC across all stations, resulting in a model that is empowered by real measured data and incorporates historical correlations.

#	Prediction Horizon (days)	Trainable parameters
1	1	201601
2	2	202498
3	3	203395
4	4	204292
5	5	205189
6	6	206086
7	7	206983

Table 4.1: SWC LSTM models' configuration

We chose LSTMs because they can process long sequences, outperforming vanilla RNNs. Moreover, within the scope of this case study, training the proposed LSTM model does not present computational challenges[140].

4.2.4 Training

The problem specifics indicate that the LSTM model should be trained for a multivariate parallel time series prediction. Therefore, we considered encoded station codes, precipitation, DOY, and previous SWC data as system inputs, and the model is trained for future SWC outputs. The encoded station codes are independent variables determining each time series, whereas the other features are dependent variables that contain meaningful patterns.

We split each time series present in the dataset into 70% training data, 10% for validation, and 20% for testing purposes. Furthermore, we employed the ADAM optimizer[55] with the MSE loss function and trained each model for a minimum of 15 epochs and a batch size of 32. Also, we utilized early stopping[141] and dropout techniques[142] to prevent overfitting. To ensure optimal training of the models, Hyperband optimization for hyperparameter tuning with Optuna[112,110] was used. Finally, we used PyTorch for training and inference implementations.

We employed the ADAM Optimizer[55] alongside the mean squared error loss function with 30 epochs and a batch size of 32 for training each LSTM-based surrogate model. Furthermore, we used Hyperband optimization for hyperparameter tuning[110]. TensorFlow/Keras with Optuna was used for training and inference implementations[111,112].

4.3 Models Evaluations

The Markov-based model, empowered by soil physics for predicting SWC, has shown acceptable performance despite not incorporating historical data into its predictions. In this section, we assess the performance of the Markov-based model by comparing it to real observed SWC data from the Kansas Mesonet. Additionally, we compare the performance of data-driven models with the Markov-based model, offering a comprehensive analysis of their relative effectiveness in SWC prediction.

4.3.1 Overall performance metrics

The overall analysis of each model is conducted on the test dataset for all the stations. We computed RMSE, MASE, and MAE to gain a general insight into the performance of each model (Table 4.2).

Model Name	**RMSE (mm)**	**MASE**	**MAE**
Single-Step (1 Day ahead)	5.84	0.50	4.46
Multi-Step (2 Day ahead)	6.72	0.53	4.71
Multi-Step (3 Day ahead)	7.86	0.59	5.23
Multi-Step (4 Day ahead)	9.52	0.81	7.12
Multi-Step (5 Day ahead)	9.34	0.64	5.63
Multi-Step (6 Day ahead)	11.32	0.89	7.87
Multi-Step (7 Day ahead)	10.04	0.79	6.99
Markov-based Model	17.54	1.50	13.27

Table 4.2: Comparison of Model Performances

Based on Table 4.2, The Single-Step (1 Day ahead) model exhibits the lowest overall error

values, indicating it offers the most accurate predictions for a one-day horizon with the least error. As the prediction horizon extends from one to seven days, the MSE and RMSE values progressively increase, highlighting a trend where longer-term forecasts become less precise. Notably, the Multi-Step models show a gradual degradation in accuracy as the forecast horizon extends, with a significant spike in error rates observed for the 6-day and 7-day ahead predictions. However, while distinct in its approach by considering soil physics, the Markov-based Model shows considerably higher error metrics across all evaluated categories, suggesting its performance is substantially inferior to the data-driven models for the time-series SWC forecasting. This analysis underscores the challenges in multi-step forecasting and highlights the trade-off between forecast horizon and accuracy.

4.3.2 Parity plot comparison

For a more comprehensive analysis, the parity plots illustrate the dispersion of the predictions versus actual values. This visual comparison underscores the integrity and robustness of each model. Figure 4.1 presents the parity plots for each model, emphasizing the superiority of the data-driven approach. Specifically, Subfigure H depicts the parity plot for the Markov-based model, revealing a higher incidence of outliers in its predictions. It can further be inferred that incorporating historical data into data-driven models (Subfigures A-G) yields more precise predictions with fewer outliers, highlighting the significance of historical context in enhancing model accuracy and reliability.

Focusing on the parity plots of the data-driven models (Subfigures A-G), the trade-off between prediction horizon and accuracy becomes more evident. The training phase evolves into a multi-objective optimization problem as the prediction horizon extends. With the increased dimensions, model accuracy is reduced, as expected.

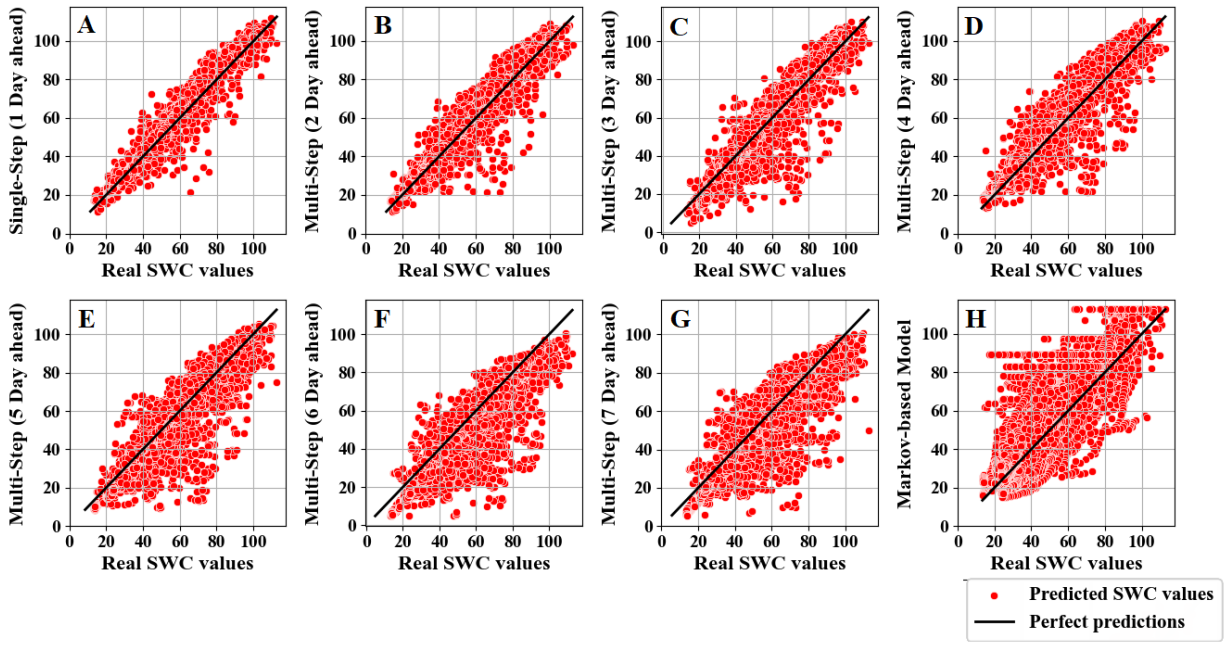


Figure 4.1: Parity plots for each SWC predictive model

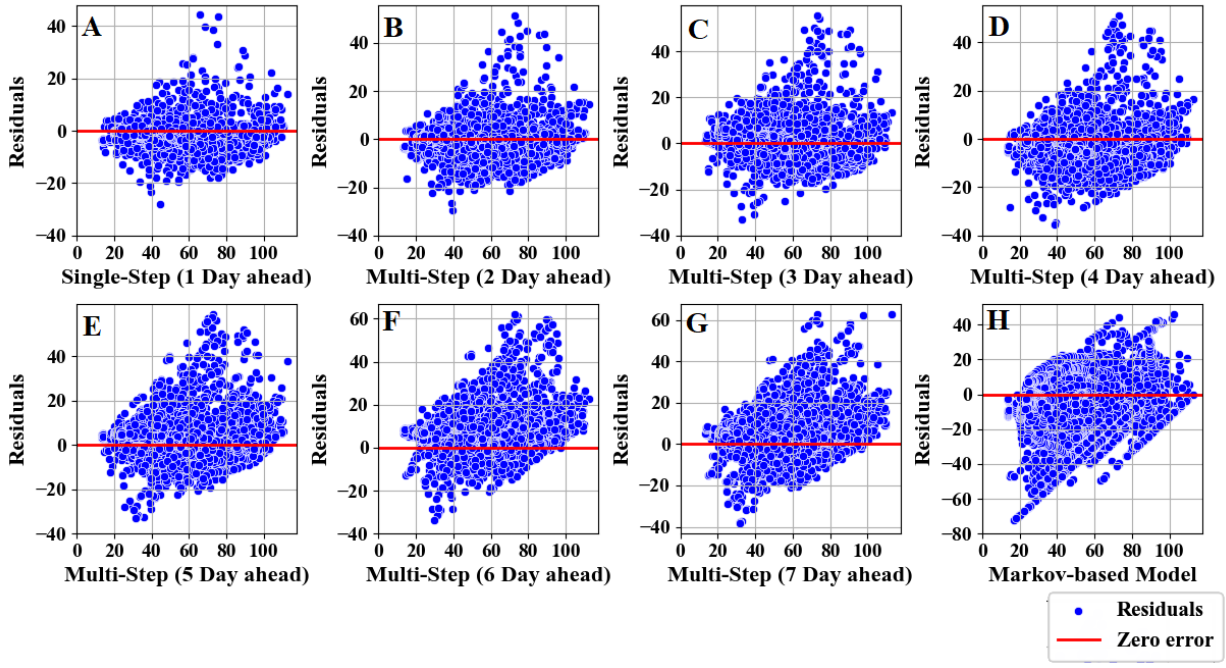


Figure 4.2: Residual plots for each SWC predictive model

4.3.3 Residual plot comparison

Residual analysis for each model assesses the discrepancies between observed values and predictions, quantifying under-predictions and over-predictions. Figure 4.2 presents the residual plots for each model. Notably, the residuals of the Markov-based model (Subfigure H) demonstrate a tendency towards more significant under-predictions compared to the data-driven models. Conversely, within the data-driven models, it is observed that the magnitude of prediction errors grows as the prediction horizon increases.

4.3.4 Time-series prediction visualization

Time-series prediction plots illustrate the Soil Water Content (SWC) values forecasted by the models compared to the actual SWC values over time. These plots reveal the dynamics of SWC and evaluate the performance of each SWC predictive model over time. Real-time agricultural decision-making relies on the dynamic predictions of each model; thus, we can determine which model is more effective and robust using time-series plots. Figure 4.3 , 4.4 display the time-series plot for two stations: Butler and Gypsum. The dynamics presented in the figures underscore the superiority of the data-driven models and highlight the trade-off between accuracy and prediction horizons.

4.4 Conclusion and future work

In this case study, we developed time-series deep learning models to predict Soil Water Content (SWC) values across 47 stations in the Kansas Mesonet. Traditionally, the Richards model offers a dynamic approach to SWC prediction. However, its computational demands and variable limitations restrict this model's practical applications. Simplifying assumptions and Markov chain processes have reduced the Richards model into the Markov-based SWC predictive model. While the Markov-based model does not pose limitations for practical applications, their neglect of historical SWC data often leads to inaccurate predictions, which

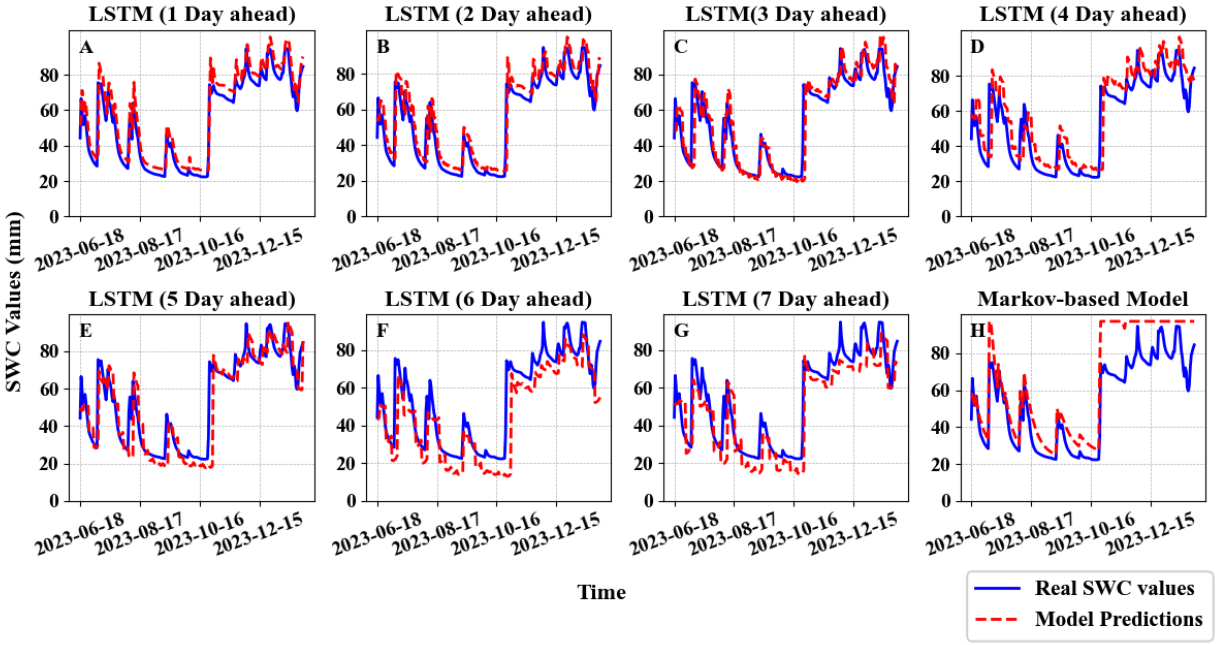


Figure 4.3: Time-seris plot for Butler stations

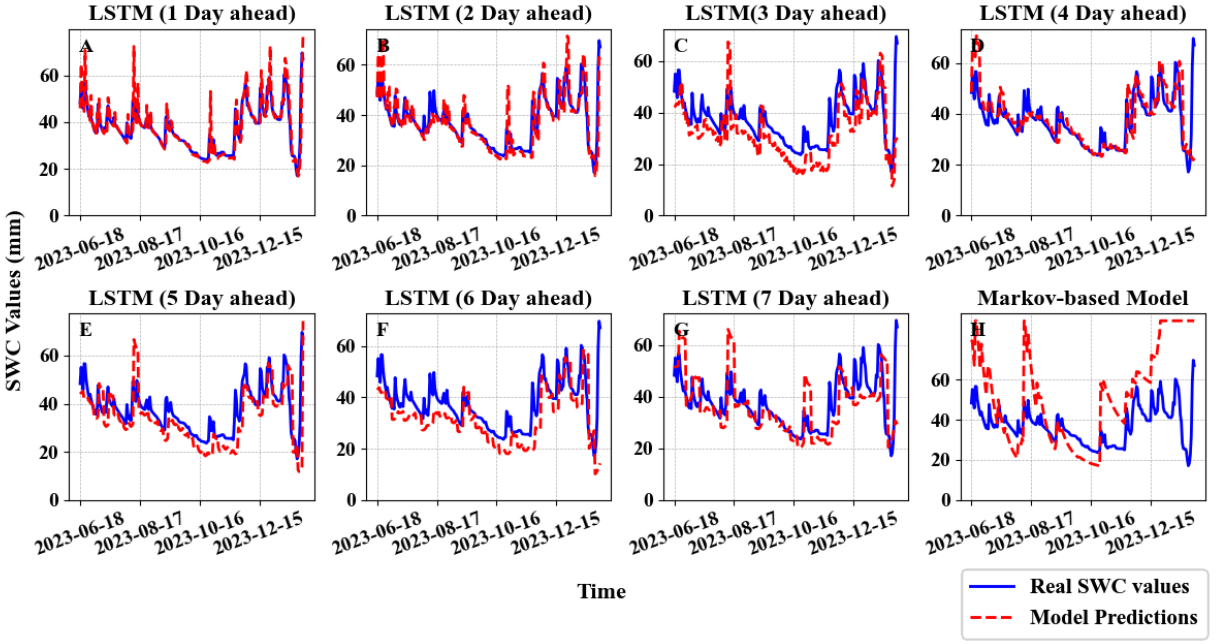


Figure 4.4: Time-seris plot for Gypsum stations

could adversely impact real-time decision-making. To address this issue, we introduced data-driven LSTM time-series models as a new perspective for the dynamic modeling of SWC. These proposed models are designed to consider average daily SWC values for a week to predict future values. Model evaluations revealed the overall superiority of data-driven models compared to the Markov-based model, benefiting further real-time decision-making applications. This is attributed to the temporal learning capabilities of the data-driven models, which enable them to uncover hidden sequential SWC patterns overlooked by the Markov model.

Despite the robustness of the data-driven model, as observed in residual plots (Figure 4.2), they include some outlier predictions. This arises from the fact that the data-driven models are now aware of soil physics. Conversely, the Markov model is derived from soil physics principles. Therefore, a more robust and accurate model could be developed using physics-informed neural networks, where the models learn from data and soil physics principles. Additionally, in creating the data-driven models, we did not consider the spatial features of each station. The spatial features offer more insight into the SWC values across states and can be beneficial for developing a more accurate data-driven model.

Bibliography

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[2] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521 (7553):436–444, 2015. doi: 10.1038/nature14539. URL https://www.nature.com/articles/nature14539.

[3] Norman P Jouppi et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, pages 1–12. ACM, 2017. doi: 10.1145/3079856.3080246. URL https://dl.acm.org/doi/10.1145/3079856.3080246.

[4] Venkat Venkatasubramanian, Raghunathan Rengaswamy, Surya N Kavuri, and Kewen Yin. A review of process fault detection and diagnosis: Part iii: Process history based methods. *Computers & chemical engineering*, 27(3):327–346, 2003.

[5] Weishun Zhong, Jacob M Gold, Sarah Marzen, Jeremy L England, and Nicole Yunger Halpern. Machine learning outperforms thermodynamics in measuring how well a many-body system learns a drive. *Scientific Reports*, 11(1):9333, 2021.

[6] Artur M Schweidtmann, Erik Esche, Asja Fischer, Marius Kloft, Jens-Uwe Repke, Sebastian Sager, and Alexander Mitsos. Machine learning in chemical engineering: A perspective. *Chemie Ingenieur Technik*, 93(12):2029–2039, 2021.

[7] Zhe Wu, Anh Tran, David Rincon, and Panagiotis D Christofides. Machine learning-based predictive control of nonlinear processes. part i: theory. *AIChE Journal*, 65(11): e16729, 2019.

[8] Ajlan Al-Ajlan. The comparison between forward and backward chaining. *International Journal of Machine Learning and Computing*, 5(2):106, 2015.

[9] Edward Hance Shortliffe. *MYCIN: A rule-based computer program for advising physicians regarding antimicrobial therapy selection*. PhD thesis, Stanford University Ph. D. dissertation, 1974.

[10] Rene Banares-Alcantara, Arthur W Westerberg, and Michael D Rychener. Development of an expert system for physical property predictions. *Computers & chemical engineering*, 9(2):127–142, 1985.

[11] R Banares-Alcantara, EI Ko, AW Westerberg, and MD Rychener. Decade—a hybrid expert system for catalyst selection—ii. final architecture and results. *Computers & Chemical Engineering*, 12(9-10):923–938, 1988.

[12] Nicolas Papernot, P. Mcdaniel, S. Jha, Matt Fredrikson, Z. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. 2016.

[13] Bradley C. Boehmke and Brandon M. Greenwell. Interpretable machine learning. 2019.

[14] Junghui Chen and Tien-Chih Huang. Applying neural networks to on-line updated pid controllers for nonlinear process control. *Journal of process control*, 14(2):211–230, 2004.

[15] Timo Sorsa, Heikki N Koivo, and Hannu Koivisto. Neural networks in process fault diagnosis. *IEEE Transactions on systems, man, and cybernetics*, 21(4):815–825, 1991.

[16] Philipp Rentzsch, M. Schubach, J. Shendure, and Martin Kircher. Cadd-splice—improving genome-wide variant effect prediction using deep learning-derived splice scores. 2021.

[17] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12): 2295–2329, 2017.

[18] Venkat Venkatasubramanian. The promise of artificial intelligence in chemical engineering: Is it here, finally? *AIChE Journal*, 65(2):466–478, 2019.

[19] Amirhosein Toosi, Andrea G Bottino, Babak Saboury, Eliot Siegel, and Arman Rahmim. A brief history of ai: how to prevent another winter (a critical review). *PET clinics*, 16(4):449–469, 2021.

[20] J. Hendler. Avoiding another ai winter. *IEEE Intelligent Systems*, 23(2):2–4, 2008. doi: 10.1109/MIS.2008.20. URL https://ieeexplore.ieee.org/document/4470337.

[21] Donal O'Callaghan and Peter Cunningham. Modern process control techniques in the production of dried milk products–a review. *Le Lait*, 85(4-5):335–342, 2005.

[22] Reinout Romijn, Leyla Özkan, Siep Weiland, Jobert Ludlage, and Wolfgang Marquardt. A grey-box modeling approach for the reduction of nonlinear systems. *Journal of Process Control*, 18(9):906–914, 2008.

[23] Martin Wörner. Numerical modeling of multiphase flows in microfluidics and micro process engineering: a review of methods and applications. *Microfluidics and nanofluidics*, 12(6):841–886, 2012.

[24] Maarten R Dobbelaere, Pieter P Plehiers, Ruben Van de Vijver, Christian V Stevens, and Kevin M Van Geem. Machine learning in chemical engineering: strengths, weaknesses, opportunities, and threats. *Engineering*, 7(9):1201–1211, 2021.

[25] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[26] Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, et al. Pytorch distributed: Experiences on accelerating data parallel training. *arXiv preprint arXiv:2006.15704*, 2020.

[27] Abdul Afram, Farrokh Janabi-Sharifi, Alan S. Fung, and Kaamran Raahemifar. Artificial neural network (ann) based model predictive control (mpc) and optimization of hvac systems: A state of the art review and case study of a residential hvac system. *Energy and Buildings*, 141:96–113, 2017. ISSN 0378-7788. doi: https://doi.org/10.1016/j.enbuild.2017.02.012.

[28] Swati Mohanty. Artificial neural network based system identification and model predictive control of a flotation column. *Journal of Process Control*, 19(6):991–999, 2009. ISSN 0959-1524. doi: https://doi.org/10.1016/j.jprocont.2009.01.001.

[29] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[30] Lukasz Kaiser, Cameron Nowzari, Tyler Summers, and Vivienne Sze. Nonlinear model predictive control using long short-term memory neural networks. In *Conference on Neural Information Processing Systems*, 2018.

[31] Mohammed S Alhajeri, Zhe Wu, David Rincon, Fahad Albalawi, and Panagiotis D Christofides. Machine-learning-based state estimation and predictive control of nonlinear processes. *Chemical Engineering Research and Design*, 167:268–280, 2021.

[32] A. Klerke. Ammonia for hydrogen storage: challenges and opportunities.

[33] Douglas R. MacFarlane, Naoki Tachikawa, Maria Forsyth, Jennifer M. Pringle, Patrick C. Howlett, Gloria D. Elliott, James H. Davis, Masayoshi Watanabe, Patrice Simon, and C. Austen Angell. Energy applications of ionic liquids. *Energy & Environmental Science*, 7(1):232–250, 2014.

[34] V. Pattabathula and J. Richardson. Introduction to ammonia production. *Chem. Eng. Prog*, 112(9):69–75, 2016.

[35] Matthew J Palys, Hanchu Wang, Qi Zhang, and Prodromos Daoutidis. Renewable ammonia for sustainable energy and agriculture: vision and systems engineering opportunities. *Current Opinion in Chemical Engineering*, 31:100667, 2021.

[36] C.H. Christensen, T. Johannessen, Rasmus Zink Sørensen, and Jens K Nørskov. Towards an ammonia-mediated hydrogen economy? *Catalysis Today*, 111(1-2):140–144, 2006.

[37] Rong Lan, John T.S. Irvine, and Shanwen Tao. Renewable ammonia synthesis. *Journal of the Royal Society Interface*, 9(71):1703–1717, 2012.

[38] Cédric Philibert. Producing ammonia and fertilizers: new opportunities from renewables. *IEA Rep*, pages 1–6, 2017.

[39] Ven Te Chow, David R Maidment, and Larry W Mays. *Applied hydrology*. 1988.

[40] Moonhyuk Kwon, Hyun-Han Kwon, and Dawei Han. A spatial downscaling of soil moisture from rainfall, temperature, and amsr2 using a gaussian-mixture nonstationary hidden markov model. *Journal of Hydrology*, 564:1194–1207, 2018.

[41] Andres Patrignani, Mary Knapp, Christopher Redmond, and Eduardo Santos. Technical overview of the kansas mesonet. *Journal of Atmospheric and Oceanic Technology*, 37(12):2167–2183, 2020.

[42] Yulong Lu and Jianfeng Lu. A universal approximation theorem of deep neural networks for expressing probability distributions. *Advances in neural information processing systems*, 33:3094–3105, 2020.

[43] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning*. Springer, 2013.

[44] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2009.

[45] Zhe Wu, Anh Tran, David Rincón, and P. Christofides. Machine learning-based predictive control of nonlinear processes. part i: Theory. *AIChE Journal*, 2019. doi: 10.1002/AIC.16729.

[46] Zhe Wu, David Rincón, Quanquan Gu, and P. Christofides. Statistical machine learning in model predictive control of nonlinear processes. *Mathematics*, 2021. doi: 10.3390/math9161912.

[47] Tim Salzmann, Elia Kaufmann, Jon Arrizabalaga, M. Pavone, D. Scaramuzza, and Markus Ryll. Real-time neural mpc: Deep learning model predictive control for quadrotors and agile robotic platforms. *IEEE Robotics and Automation Letters*, 2022. doi: 10.1109/LRA.2023.3246839.

[48] Songda Wang, T. Dragičević, G. Gontijo, S. Chaudhary, and R. Teodorescu. Machine learning emulation of model predictive control for modular multilevel converters. *IEEE Transactions on Industrial Electronics*, 2021. doi: 10.1109/tie.2020.3038064.

[49] Yi Ming Ren, Mohammed S Alhajeri, Junwei Luo, Scarlett Chen, Fahim Abdullah, Zhe Wu, and Panagiotis D Christofides. A tutorial review of neural network modeling approaches for model predictive control. *Computers & Chemical Engineering*, page 107956, 2022.

[50] Jiawei Han, Jian Pei, and Hanghang Tong. *Data mining: concepts and techniques*. Morgan kaufmann, 2022.

[51] Michal S Gal and Daniel L Rubinfeld. Data standardization. *NYUL Rev.*, 94:737, 2019.

[52] Serdar Ekinci Davut Izci Erdal Eker, M. Kayri. Comparison of swarm-based metaheuristic and gradient descent-based algorithms in artificial neural network training. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, 2023. doi: 10.14201/adcaij.29969.

[53] Xiangwen Wang Xianghong Lin, Tiandou Hu. One-pass online learning based on gradient descent for multilayer spiking neural networks. *IEEE Transactions on Cognitive and Developmental Systems*, 2023. doi: 10.1109/TCDS.2021.3140115.

[54] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

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

[56] C. Nalini F. Amenta S. Tayebati G. Battineni, G. Sagaro. Comparative machine-learning approach: A follow-up study on type 2 diabetes predictions by cross-validation methods. *Machines*, 2019. doi: 10.3390/machines7040074.

[57] Ryan Turner, David Eriksson, M. McCourt, J. Kiili, Eero Laaksonen, Zhen Xu, and Isabelle M Guyon. Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. *NeurIPS*, 2021.

[58] A. Mesbah, K. P. Wabersich, Angela P. Schoellig, M. Zeilinger, S. Lucia, T. A. Badgwell, and J. Paulson. Fusion of machine learning and mpc under uncertainty: What advances are on the horizon? *ACC*, 2022. doi: 10.23919/ACC53348.2022.9867643.

[59] Tim Salzmann, Elia Kaufmann, Jon Arrizabalaga, M. Pavone, D. Scaramuzza, and Markus Ryll. Real-time neural mpc: Deep learning model predictive control for quadrotors and agile robotic platforms. *IEEE Robotics and Automation Letters*, 2023. doi: 10.1109/LRA.2023.3246839.

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

[61] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017. URL https://arxiv.org/abs/1706.03762.

[62] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[63] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[64] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27:3104–3112, 2014.

[65] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2014.

[66] Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations*, 2015. URL https://arxiv.org/pdf/1409.0473.pdf.

[67] Sneha Chaudhari, Varun Mithal, Gungor Polatkan, and Rohan Ramanath. An attentive survey of attention models. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 12(5), Oct 2021. doi: 10.1145/3465055.

[68] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[69] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[70] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[71] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018. URL https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf.

[72] Niranjan Sitapure and Joseph Sang-Il Kwon. Exploring the potential of time-series transformers for process modeling and control in chemical systems: an inevitable paradigm shift? *Chemical Engineering Research and Design*, 194:461–477, 2023.

[73] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The Elements of Statistical Learning*. Springer series in statistics New York, 2001.

[74] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Science & Business Media, 2009.

[75] Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2014.

[76] Cort J Willmott and Kenji Matsuura. Advantages of the mean absolute error (mae)

over the root mean square error (rmse) in assessing average model performance. *Climate Research*, 30(1):79–82, 2005.

[77] Ian T Jolliffe and David B Stephenson. *Forecast Verification: A Practitioner's Guide in Atmospheric Science*. John Wiley & Sons, 2012.

[78] Alexander Smith and Emily Johnson. Enhancing model accuracy through residual analysis. *Journal of Advanced Modeling Techniques*, 15(4):245–260, 2021.

[79] Ming Zhao and et al. The role of residual plots in machine learning pipelines. *Machine Learning Research*, 22(1):101–115, 2022.

[80] J.B. Rawlings and D.Q. Mayne. *Model Predictive Control: Theory and Design*. SIAM, 2017.

[81] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[82] Berkay Çıtmacı, Junwei Luo, Joon Baek Jang, Carlos G Morales-Guio, and Panagiotis D Christofides. Machine learning-based ethylene and carbon monoxide estimation, real-time optimization, and multivariable feedback control of an experimental electrochemical reactor. *Chemical Engineering Research and Design*, 191:658–681, 2023.

[83] Aarsh Dodhia, Zhe Wu, and Panagiotis D Christofides. Machine learning-based model predictive control of diffusion-reaction processes. *Chemical Engineering Research and Design*, 173:129–139, 2021.

[84] Limin Wang. *Karhunen-Loeve expansions and their applications*. London School of Economics and Political Science (United Kingdom), 2008.

[85] Clarence W Rowley, Tim Colonius, and Richard M Murray. Model reduction for compressible flows using pod and galerkin projection. *Physica D: Nonlinear Phenomena*, 189(1-2):115–129, 2004.

[86] Michael Zeitz. The extended luenberger observer for nonlinear systems. *Systems & Control Letters*, 9(2):149–156, 1987.

[87] Maaz Mahmood and Prashant Mhaskar. Lyapunov-based model predictive control of stochastic nonlinear systems. *Automatica*, 48(9):2271–2276, 2012.

[88] Tianqi Xiao, Zhe Wu, Panagiotis D Christofides, Antonios Armaou, and Dong Ni. Recurrent neural-network-based model predictive control of a plasma etch process. *Industrial & Engineering Chemistry Research*, 61(1):638–652, 2021.

[89] Cheng Hu, Scarlett Chen, and Zhe Wu. Economic model predictive control of nonlinear systems using online learning of neural networks. *Processes*, 11(2):342, 2023.

[90] Cheng Hu and Zhe Wu. Online learning-based predictive control of switched nonlinear systems with disturbances. In *2023 American Control Conference (ACC)*, pages 92–99, 2023. doi: 10.23919/ACC55779.2023.10156082.

[91] Parth Shah, Hyun-Kyu Choi, and Joseph Sang-Il Kwon. Lstm-based control of cellulose degree of polymerization in a batch pulp digester. In *2023 American Control Conference (ACC)*, pages 4659–4664. IEEE, 2023.

[92] Mohammed S Alhajeri, Junwei Luo, Zhe Wu, Fahad Albalawi, and Panagiotis D Christofides. Process structure-based recurrent neural network modeling for predictive control: A comparative study. *Chemical Engineering Research and Design*, 179:77–89, 2022.

[93] Krzysztof Zarzycki and Maciej Ławryńczuk. Lstm and gru neural networks as models of dynamical processes used in predictive control: A comparison of models developed for two chemical reactors. *Sensors*, 21(16):5625, 2021.

[94] Fabio Bonassi, Caio Fabio Oliveira da Silva, and Riccardo Scattolini. Nonlinear mpc

for offset-free tracking of systems learned by gru neural networks. *IFAC-PapersOnLine*, 54(14):54–59, 2021.

[95] Niranjan Sitapure and Joseph S Kwon. Require process control?\textbf {LSTMc} is all you need! *arXiv preprint arXiv:2306.07510*, 2023.

[96] Niranjan Sitapure and Joseph Sang-Il Kwon. Crystalgpt: Enhancing system-to-system transferability in crystallization prediction and control using time-series-transformers. *Computers & Chemical Engineering*, 177:108339, 2023.

[97] Junho Park, Mohammad Reza Babaei, Samuel Arce Munoz, Ashwin N Venkat, and John D Hedengren. Simultaneous multistep transformer architecture for model predictive control. *Computers & Chemical Engineering*, 178:108396, 2023.

[98] Natalia Morlanés, Sai P Katikaneni, Stephen N Paglieri, Aadesh Harale, Bandar Solami, S Mani Sarathy, and Jorge Gascon. A technological roadmap to the ammonia energy economy: Current state and missing technologies. *Chemical Engineering Journal*, 408:127310, 2021.

[99] Vaclav Smil. *Enriching the Earth: Fritz Haber, Carl Bosch, and the Transformation of World Food Production*. MIT Press, 2001.

[100] Roland Rosen, Georg Von Wichert, George Lo, and Kurt D Bettenhausen. About the importance of autonomy and digital twins for the future of manufacturing. *IFAC-PapersOnLine*, 48(3):567–572, 2015.

[101] O. C. Zienkiewicz, R. L. Taylor, and J. Z. Zhu. *The Finite Element Method: Its Basis and Fundamentals*. Elsevier, 2005.

[102] M. P. Bendsøe and O. Sigmund. *Topology Optimization: Theory, Methods, and Applications*. Springer, 2003.

[103] Thiago Oliveira Cabral, Amirsalar Bagheri, and Davood B. Pourkargar. Learning-based model reduction and predictive control of an ammonia synthesis process. *Industrial & Engineering Chemistry Research*, 63(23):10325–10342, 2024. doi: 10.1021/acs.iecr.4c00340.

[104] D. Rindskopf and M. Shiyko. Measures of dispersion, skewness and kurtosis. In *International Encyclopedia of Education (Third Edition)*, pages 267–273. Elsevier, Oxford, third edition edition, 2010. ISBN 978-0-08-044894-7. doi: https://doi.org/10.1016/B978-0-08-044894-7.01344-0.

[105] Amirsalar Bagheri, Thiago Oliveira Cabral, and Davood B Pourkargar. Learning-based model predictive control of an ammonia synthesis reactor. In *2024 American Control Conference (ACC)*, pages 45–50. IEEE, 2024.

[106] A.F. Agarap. Deep learning using rectified linear units (relu). arxiv 2018. *arXiv preprint arXiv:1803.08375*, 1803.

[107] B. Xu, N. Wang, T. Chen, and M. Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[108] T. Chai and R.R. Draxler. Root mean square error (rmse) or mean absolute error (mae). *Geoscientific model development discussions*, 7(1):1525–1534, 2014.

[109] SGOPAL Patro and Kishore Kumar Sahu. Normalization: A preprocessing stage. *arXiv preprint arXiv:1503.06462*, 2015.

[110] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018.

[111] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghe-

mawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[112] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

[113] Mohammed Saad Faizan Bangi and Joseph Sang Kwon. Deep hybrid model-based predictive control with guarantees on domain of applicability. *AIChE Journal*, 69(5): e18012, 2023.

[114] Parth Shah, M Ziyan Sheriff, Mohammed Saad Faizan Bangi, Costas Kravaris, Joseph Sang Kwon, Chiranjivi Botre, and Junichi Hirota. Deep neural network-based hybrid modeling and experimental validation for an industry-scale fermentation process: Identification of time-varying dependencies among parameters. *Chemical Engineering Journal*, 441:135643, 2022.

[115] Mohammed Saad Faizan Bangi and Joseph S Kwon. Deep hybrid modeling of chemical process: application to hydraulic fracturing. *Computers & Chemical Engineering*, 134: 106696, 2020.

[116] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involv-

ing nonlinear partial differential equations. *Journal of Computational Physics*, 378: 686–707, 2019.

[117] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6): 422–440, 2021.

[118] Somdatta Goswami, Minglang Yin, Yue Yu, and George Em Karniadakis. A physics-informed variational deeponet for predicting crack path in quasi-brittle materials. *Computer Methods in Applied Mechanics and Engineering*, 391:114587, 2022.

[119] Jeremy Yu, Lu Lu, Xuhui Meng, and George Em Karniadakis. Gradient-enhanced physics-informed neural networks for forward and inverse pde problems. *Computer Methods in Applied Mechanics and Engineering*, 393:114823, 2022.

[120] James Hansen, Makiko Sato, and Reto Ruedy. Perception of climate change. *Proceedings of the National Academy of Sciences*, 109(37):E2415–E2423, 2012.

[121] G Clarke Topp, GW Parkin, Ty PA Ferré, MR Carter, and EG Gregorich. Soil water content. *Soil sampling and methods of analysis*, 2:939–962, 2008.

[122] Catriona MK Gardner, David Robinson, Ken Blyth, and J David Cooper. Soil water content. In *Soil and environmental analysis*, pages 13–76. CRC Press, 2000.

[123] D. Entekhabi, I. Rodriguez-Iturbe, and F. Castelli. Remote sensing of soil moisture availability and vegetation water use: The smap mission. *Water Resources Research*, 46(3):W03523, 2010.

[124] Marco Bittelli. Measuring soil water content: A review. *HortTechnology*, 21(3):293–300, 2011.

[125] L. A. Richards. Capillary conduction of liquids through porous mediums. *Physics*, 1 (5):318–333, 1931.

[126] Jacob Bear and Alexander H.-D. Cheng. *Modeling Groundwater Flow and Contaminant Transport*. Springer Netherlands, 2010.

[127] Daniel Hillel. *Introduction to Environmental Soil Physics*. Elsevier Academic Press, 2004.

[128] Harry Vereecken, Andrea Schnepf, Jan W Hopmans, Mathieu Javaux, Dani Or, Tiina Roose, Jan Vanderborght, Michael H Young, Wulf Amelung, Matt Aitkenhead, and et al. Modeling soil processes: Review, key challenges, and new perspectives. *Vadose Zone Journal*, 15(5):vzj2015.09.0131, 2016.

[129] Hamlyn G Jones. The role of soil moisture in controlling water use by plants. *Journal of Experimental Botany*, 52(359):1481–1490, 2001.

[130] Amilcare Porporato, Edoardo Daly, and Ignacio Rodriguez-Iturbe. Soil moisture and plant stress dynamics along the kalahari precipitation gradient. *Journal of Geophysical Research: Atmospheres*, 108(D3):8–1, 2002.

[131] Jasper A. Vrugt, Hoshin V. Gupta, Stefan C. Dekker, Soroosh Sorooshian, Thorsten Wagener, and Willem Bouten. Application of stochastic parameter optimization to the sacramento soil moisture accounting model. *Journal of Hydrology*, 325(1):288–307, 2006.

[132] Hongxiang Yan, Caleb M. DeChant, and Hamid Moradkhani. Improving soil moisture profile prediction with the particle filter-markov chain monte carlo method. *IEEE Transactions on Geoscience and Remote Sensing*, 53(11):6134–6147, 2015.

[133] WT Crow and Dongryeol Ryu. A new data assimilation approach for improving runoff prediction using remotely-sensed soil moisture retrievals. *Hydrology and Earth System Sciences*, 13(1):1–16, 2009.

[134] Shikha Prakash, Animesh Sharma, and Sitanshu Shekhar Sahu. Soil moisture prediction using machine learning. In *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, pages 1–6. IEEE, 2018.

[135] Gursimran Singh, Deepak Sharma, Amarendra Goap, Sugandha Sehgal, AK Shukla, and Satish Kumar. Machine learning based soil moisture prediction for internet of things based smart irrigation system. In *2019 5th International Conference on Signal Processing, Computing and Control (ISPCC)*, pages 175–180. IEEE, 2019.

[136] Briana M Wyatt, Tyson E Ochsner, William G Brown, D Cole Diggins, Bradley G Illston, and Christopher A Fiebrich. Mesosoil v2. 0: An updated soil physical property database for the oklahoma mesonet. *Vadose Zone Journal*, 20(4):e20134, 2021.

[137] Yonggen Zhang and Marcel G Schaap. Weighted recalibration of the rosetta pedotransfer model with improved estimates of hydraulic parameter distributions and summary statistics (rosetta3). *Journal of Hydrology*, 547:39–53, 2017.

[138] Fatemeh Karandish and Jiří Šimnek. A comparison of numerical and machine-learning modeling of soil water content with limited input data. *Journal of Hydrology*, 543: 892–909, 2016.

[139] Zhihao Hong, Z Kalbarczyk, and Ravishankar K Iyer. A data-driven approach to soil moisture collection and prediction. In *2016 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 1–6. IEEE, 2016.

[140] Patrignani-Andres Ghanbarian Behzad and Pourkargar Davood B Bagheri, Amirsalar. A physics-informed machine learning approach to predict soil water content for agricultural decision-making. In *2024 American Control Conference (ACC)*, pages 2–7. IEEE, 2024.

[141] Lutz Prechelt. Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer, 2002.

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