Simultaneous Multistep Transformer Architecture for Model Predictive Control

Junho Park[a], Mohammad Reza Babaei[a], Samuel Arce Munoz[a], Ashwin N. Venkat[b], John D. Hedengren[a]

[a]*Department of Chemical Engineering, Brigham Young University, Provo, Utah, USA*
[b]*Seeq Corporation, Seattle, Washington, USA*

Abstract

Transformer neural networks have revolutionized natural language processing by effectively addressing the vanishing gradient problem. This study focuses on applying Transformer models to time-series forecasting and customizing them for a simultaneous multistep-ahead prediction model in surrogate model predictive control (MPC). The proposed method showcases improved control performance and computational efficiency compared to LSTM-based MPC and one-step-ahead prediction models using both LSTM and Transformer networks. The study introduces three key contributions: (1) a new MPC system based on a Transformer time-series architecture, (2) a training method enabling multistep-ahead prediction for time-series machine learning models, and (3) validation of the enhanced time performance of multistep-ahead Transformer MPC compared to one-step-ahead LSTM networks. Case studies demonstrate a significant fifteen-fold improvement in computational speed compared to one-step-ahead LSTM, although this improvement may vary depending on MPC factors like the lookback window and prediction horizon.

Keywords: Transformer neural network architecture, Long short-term memory, model predictive control

1. Introduction

Machine learning technologies have gained attention in many industries and have grown in popularity, especially in areas requiring classification, such as pattern recognition for image and voice. Many researchers in process system engineering have recently renewed interest in machine learning technologies, such as deep learning. An overview of current trends and opportunities for machine learning technologies in process system engineering has been discussed in [1, 2, 3, 4]. Areas using machine learning technologies in process system engineering are control and optimization. Model predictive control (MPC) and real-time optimization (RTO) are two advanced automation systems that require process models. Many MPC application research studies have been conducted using artificial neural network (ANN) models: internal combustion engine [5], distillation column and continuous stirred tank reactor (CSTR) example [6, 7], Heating ventilation and air conditioning systems (HVAC) [8, 9], LSTM (Long short term memory network) MPC for continuous pharmaceutical reactors [10], and phthalic anhydride synthesis in a fixed-bed catalytic reactor [11]. A Hybrid model with a Hammerstein model structure, which consists of a static neural net model block and a linear dynamics model block in series, has been investigated in [12]. A gated recurrent unit (GRU)-based encoder–decoder architecture integrated with a physics-based model has been tested for thermal power plants [13]. An approximated MPC concept has also been investigated by learning control policy from the closed-loop performance data [14]. Reinforcement learning methods for RTO application have been investigated in [15, 16, 17].

Among the different types of ANNs, RNNs are a natural choice for modeling time-series data because the structure of RNN states has explicit temporal dependence. However, the sequential processing nature of the RNNs makes the model prediction computationally expensive by recursively updating the hidden state. A novel time-series ANN architecture, Transformer neural network, has been introduced to solve the issue by parallelizing the model prediction steps, making it possible to produce multistep predictions simultaneously [18]. State-of-the-art ANN architectures, especially attention mechanisms, improve natural language processing (NLP). Although dynamic process models can be built with time-series data like the NLP problems, the actual issues are vastly different in model usage for prediction in model-based control and optimization applications.

Previous research has recognized the Transformer network for its remarkable computational speed. A distinguishing feature of this network is its attention mechanism, which adeptly handles irregular temporal dependencies. This capability is particularly notable in the context of natural language data. This study further explores the potential for the application of the Transformer network within process control fields by integrating it into the MPC framework as a predictive model. In addition to creating a Transformer-based MPC system, the research introduces a multistep-ahead prediction structure to maximize the benefits derived from the Transformer advancements, particularly within the context of

MPC applications. Evidence from three case studies indicates that the multistep ahead Transformer MPC offers superior performance in terms of computational efficiency and setpoint tracking compared to the one-step ahead LSTM MPC.

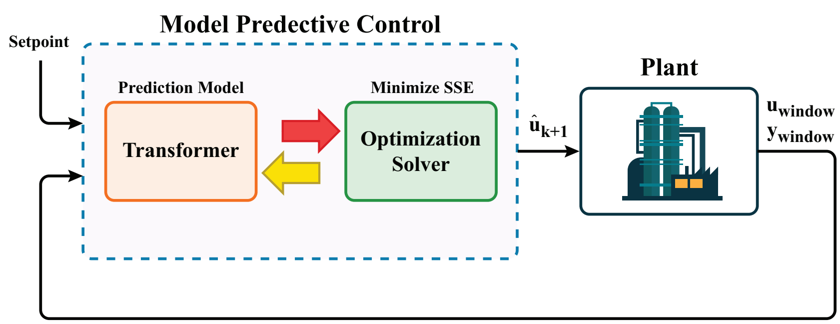
In an effort to contextualize these findings, the study contrasts the proposed multistep ahead Transformer MPC with the one-step ahead LSTM, which maintains the traditional data-driven time-series format within the surrogate MPC framework. Although all the case studies do not exhaustively examine two alternative model structures — multistep ahead LSTM and one-step ahead Transformer — the first case study (5.1) provides an indirect comparison of these models to the multistep ahead Transformer MPC in terms of prediction accuracy and computational efficiency. Further, sequential neural network models such as LSTM are recognized for a relative lack of prediction precision, particularly when dealing with irregular temporal dependencies prevalent in the multistep-ahead data structure as shown in Figure 7.

While this paper does not delve into a comprehensive investigation of all model structures, it lays the groundwork for future research. Further studies could provide valuable insights by examining the performance of various models in relation to complexity and ability to handle data irregularities. This could pave the way for a more understanding of model behaviors in different scenarios.

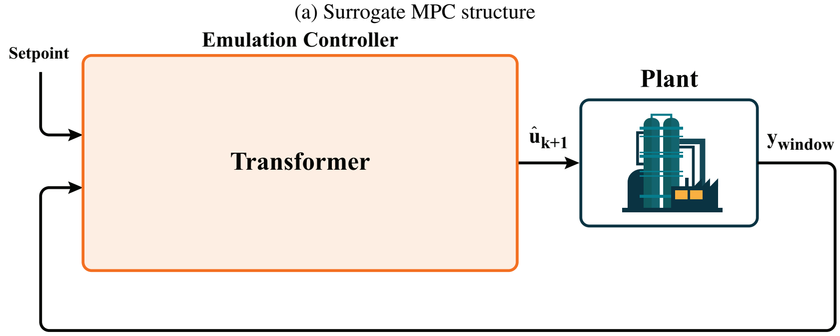
The objective of this work is to develop a simultaneous multistep-ahead prediction method using a Transformer architecture for MPC applications. Figure 1 gives an overview of two approaches for using Transformer models for model based-control. The Transformer model can emulate the process model (Figure 1a) or the entire model predictive control (Figure 1b). For the scope of this paper, the exploration primarily focuses on the former, investigating the surrogate MPC approach more intensively. A brief review of Transformer Network Architecture is discussed with details of the adaptations for time-series data. Source code is available from `https://github.com/BYU-PRISM/Transformer_MPC`

2. Transformer Network Architecture

The original concept of Transformer architecture in "Attention is all you need" [18] is dedicated to machine translation tasks (Figure 2). Thus, each element of the original design needs to be customized for the time-series data processing. This section discusses each part of the original Transformer design and the modifications for time-series data, especially for the simultaneous multistep-ahead prediction needed for MPC.



(a) Surrogate MPC structure



(b) Emulation control structure

Figure 1: Two scenarios for Transformers in model based control

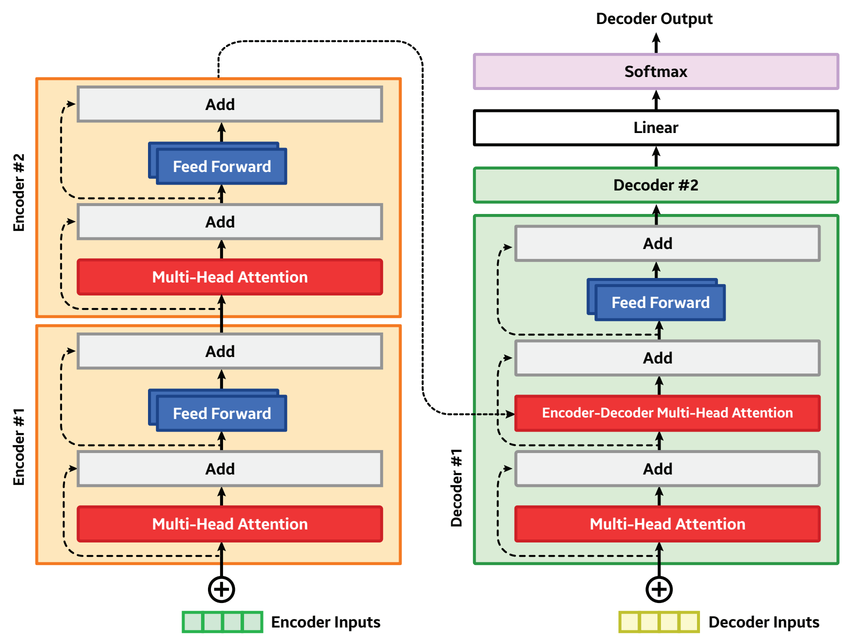


Figure 2: Transformer structure

2.1. *Encoder-Decoder*

The Encoder-Decoder concept is initially used in the Sequence to Sequence (Seq2seq) architecture that places two separate RNNs in a row. The Encoder RNN processes the input sequence to produce a context vector. This vector, which encapsulates the information from the input sequence, is then passed to the Decoder RNN to predict the output sequence. The Transformer architecture retains the encoder-decoder framework while using new concepts for specific components.

2.2. *Positional Encoding*

In the RNN models, the hidden state of the previous timesteps returns to the network as an additional input along with the actual input values for the next time step. This recursive data processing is not computationally efficient, but it has an advantage in that it can inherently recognize the order of the input sequence. However, as the Transformer model takes the whole input sequence simultaneously, the inherent temporal dependency in the RNN is no longer available in the Transformer. A positional embedding layer explicitly labels the sequence position to have the Transformer network recognize the sequence order. Trigonometric functions [18] is a method among many to embed the positional information into the input sequence. This method is particularly effective for sequences that have an irregular length and interval by assigning the unique index in the form of a real number vector. Thus, this method is commonly used for NLP tasks that vary the number of words in every sentence, resulting in different input vector lengths and time intervals. On the other hand, the time-series data usually has a fixed size and interval or can easily be converted to this form by signal processing.

2.3. Attention Mechanism

The attention mechanism is a crucial part of the Transformer model. It has been proposed to improve the long-term memory capability in the Seq2seq model [19]. The encoded input sequence in the Seq2seq model is saved in a fixed-length vector called context vector and passed to the decoder RNN [20]. Because the context vector in the Seq2seq model is just the last hidden state of the encoder RNN, it cannot accommodate the most critical correlation at earlier time steps caused by vanishing gradients. The memory issue of the RNN can be significant when the time series is long. The attention mechanism improves the memory of the context vector by introducing a probability distribution between a hidden state of the decoder RNN at a specific time step and the hidden states of every time step in the encoder RNN. This probability distribution helps the model focus more on a certain time step in the input sequence regardless of the order. It is converted further into a vector with the same size as the hidden state, called the attention value. This attention value acts as a context vector of the Seq2seq model, holding more useful long-term correlation information between the encoder and decoder. The Transformer architecture extensively employs the attention mechanism, which removes the inherent sequential processing in the RNN models. A conditional probability of a sequence data for the multistep-ahead prediction is shown in Equation 1.

$$\begin{aligned} &\mathbf{p}(\mathbf{y}_{k+1:k+P} \,|\mathbf{y}_{k-w:k},\, \mathbf{u}_{k-w:k+P}) \\ &\qquad = \prod_{t=k+1}^{k+P} \mathbf{p}(\mathbf{y}_t \mid \mathbf{y}_{t-w-1:t-1},\, \mathbf{u}_{t-w-1:t-1}), \end{aligned} \tag{1}$$

where, $\mathbf{p}(A \mid B)$ represents the probability of A under the given condition B. $\mathbf{y}$ and $\mathbf{u}$ denote the system output and system input, respectively. The past system output $\mathbf{y}$ is used as one of the NN model inputs, called auto-regressive input, while the system input $\mathbf{u}$ is called exogenous input. The future values ($t > k$) of $\mathbf{u}$ are also included in the exogenous input as known values for the NN model training, while the $\mathbf{y}$ only include the past values ($t < k$) as shown in LHS of Equation 1. k, w, and P denote the current time step, look-back window size, and prediction horizon size, respectively. Using the RNN models to achieve the multistep prediction, it is inevitable to build up the hidden state recursively from time step $k-w$ to k and repeat the same recursive computations for every prediction value in the prediction horizon P. However, the attention mechanism simultaneously computes the probability distribution, also called 'Attention distribution' between the input and output sequence, altering the recurrence in RNN with the matrix multiplications and $softmax$ activation function.

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{2}$$

where, Q, K, and V represent the weighted sequence data matrices, *Query*, *Key*, and *Value*. The same input sequence data, X, is assigned for all Q, K, and V matrices for the encoder self-attention layers as $Q = XW_Q$, $K = XW_K$, and $V = XW_V$, where, $X = [\mathbf{y}_{k-w\,:\,k+P};\ \mathbf{u}_{k-w\,:\,k+P}] \in \mathbb{R}^{b\times w+P\times l}$. The b and l denote the batch size and the number of variables in the input matrix, respectively. Note that the future values in sequence $\mathbf{y}$ in the X matrix ($\mathbf{y}_{k+1\,:\,k+P}$) need to be masked with the $\mathbf{y}$ value at the time step k ($\mathbf{y}_k$) to avoid including the prediction output in the input matrix (X) before training.

The prediction model, to be compatible with the MPC framework, requires both future marked $\mathbf{y}_k$ data points and $\mathbf{u}_{k+1:k+P}$ data points within the X matrix. The $\mathbf{u}_{k+1:k+P}$ data points, in particular, are vital to ensure the degrees of freedom for the MPC optimization solution. Conversely, while the MPC framework does not inherently require future $\mathbf{y}$ points, they are essential for maintaining consistent length between the $\mathbf{y}$ and $\mathbf{u}$ vectors within the X matrix. Including spurious $\mathbf{y}$ values in the prediction horizon portion of the $\mathbf{y}$ vector can lead to anomalous correlations between input and output. Nonetheless, this study demonstrates the adeptness of Transformer networks in managing such data irregularities.

3. Long-Short Term Memory Networks (LSTM)

The LSTM network is a type of RNN along with a GRU (Gated Recurrent Unit) network. It is composed of a cell, an input gate, an output gate, and a forget gate to reinforce the long-term memory proposed in [21]. LSTMs are well suited for making time-series predictions with longer-term sequences and dependencies with this unique structure. However, the inherent sequential computation in the LSTM leads to an extended processing time that limits the online MPC application as a prediction model where a specific cycle time must be met. From a train-ability point of view, the prediction accuracy with the capability of accommodating the special input-out data structure for multistep-ahead prediction is compared to the proposed Transformer model. The LSTM block and equations for each gate are shown in Figure 3.

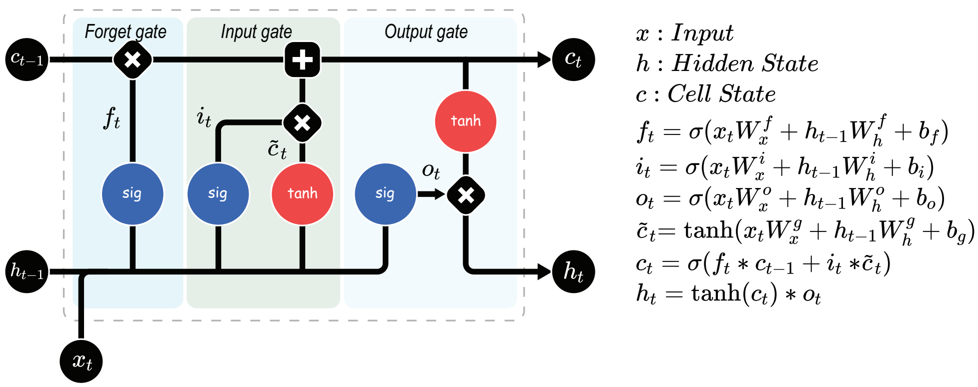


Figure 3: LSTM cell.

4. Transformer Model MPC Development

This section discusses the development of a Transformer model-based MPC system. It introduces the training data modification to obtain the multistep prediction model, layer configurations of the Transformer network, and an MPC framework that accommodates the Transformer prediction model.

4.1. Training Data Preparation

A simple single-input single-output (SISO) first-order plus dead-time (FOPDT) model is chosen for the process model for illustrative purposes. Equation 3 shows the continuous FOPDT equation between process input (u) and output (y), where K_p, τ_p, and θ_p represent the model parameters for process gain, time constant, and dead time, respectively. The parameter values used for the FOPDT model are also shown in Equation 3.

$$\tau_p \frac{dy(t)}{dt} = -y(t) + K_p u(t - \theta_p)$$
$$\text{where, } K_p = 1, \tau_p = 2, \text{ and } \theta_p = 0 \tag{3}$$

The training data is generated by simulating the FOPDT model with randomly created input (u) sequences for 1,600 data points. The simulated data is split into two sets, one for training and the other for validation. The validation set is used to monitor and prevent over-fitting of the NN model. An additional approach to handle correlated multivariate control inputs is to build a Principal Component Analysis (PCA) based model that reduces the input space or constrains the squared prediction error (SPE) to follow the PCA model during training. With unachievable setpoints, the PCA-based SPE constraint helps to maintain achievable setpoints [22, 23]. The data in this study does not have correlated inputs because they are randomly generated. The training and validation data are reshaped to fit the time-series NN model structure. Although the Transformer and LSTM models use entirely different algorithms, the dimension of the input and output vector is the same for the same sequence data. The three-dimensional vector, also called a tensor, can be formed with ($Batch size \times Length of one snapshot \times Number of variables$) and the size of a tensor varies depending on the size of prediction steps such as one-step (os) and multistep (ms) ahead prediction. Figure 4 visually describes the batch size or Number of snapshots (T), length of one snapshot ($w+P$), and number of variables, respectively. The lengths of the look-back window (w = 5) and the prediction horizon (P = 10) have been carefully selected to reflect the dynamic characteristics of the process. The mathematical expressions of the data structure are shown in Equation 4 and 5 for one-step- and multistep-ahead prediction models, respectively.

$$
\begin{aligned}
Y_{os} &= (y_{k+1}) \in \mathbb{R}^{b \times 1 \times l} \\
X_{os} &= ((u_{k-w}, \cdots u_k), (y_{k-w}, \cdots y_k)) \in \mathbb{R}^{b \times w \times f} \\
&\quad \text{where, } w \le k \le T-P, \\
&\qquad b = T-w-P
\end{aligned}
\tag{4}
$$

$$
\begin{aligned}
Y_{ms} &= (y_{k+1}, \cdots y_{k+P}) \in \mathbb{R}^{b \times P \times l} \\
X_{ms} &= \big((u_{k-w}, \cdots u_{k+P}), \big(y_{k-w}, \cdots y'_{k+1}, \cdots y'_{k+P}\big)\big) \\
&\qquad\qquad \in \mathbb{R}^{b \times (w+P) \times f} \\
&\quad \text{where, } w \le k \le T-P, \\
&\qquad b = T-w-P, \\
&\qquad y'_{k+i} = y_k \ (1 \le i \le P)
\end{aligned}
\tag{5}
$$

In Equations 4 and 5, Y and X represent the reshaped output and input of the NN training, which are also called $labels$ and $features$, respectively. On the other

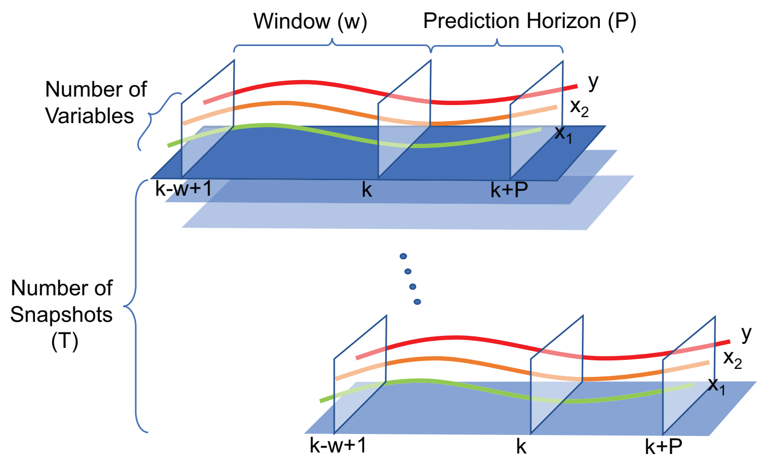


Figure 4: Training data preparation with receding window snapshots

hand, u and y are the process system input and output of the FOPDT model. As expressed in Equations 4 and 5, system input and output (u and y) are in the X as a linear auto-regressive and exogenous part of the NN model input, respectively. The symbols b, T, w, and P denote the batch size, entire time-series length, look-back window size, and prediction horizon length, respectively. Including all snapshots k, l, and f denote the current time step, the number of labels, and the number of features, respectively. A notable difference between one-step prediction and multistep prediction data structures is that the Y_{ms} extends the vector to the $k+P$ time step, including the full P-step-ahead values. In contrast, the Y_{os} includes only the first value in the prediction horizon. A similar difference appears in the input feature. The u and y vector in the X_{ms} extend to the $k+P$, while the ones in the X_{os} stop at the k. Important customization needs to be made for the y vector in the X_{ms}. The y values for future time steps are replaced with the y_k shown as y'_{k+i} in the Equation 5. Prior state values are included in the data row for training to maintain the vector shape consistent with the u vector in the X_{ms}. Figure 5 visually compares the data structures between one-step- and multistep-ahead models for one snapshot sample in the batch. The modified y vector for the multistep prediction model is illustrated in the dashed box with y_k. Figure 6 illustrates the training data modification for the multistep prediction models.

4.2. Transformer Model Training

Prepared training data sets for one-step-ahead prediction (Y_{os}, X_{os}) and multistep-ahead prediction (Y_{ms}, X_{ms}) are used for training the Transformer model. Every

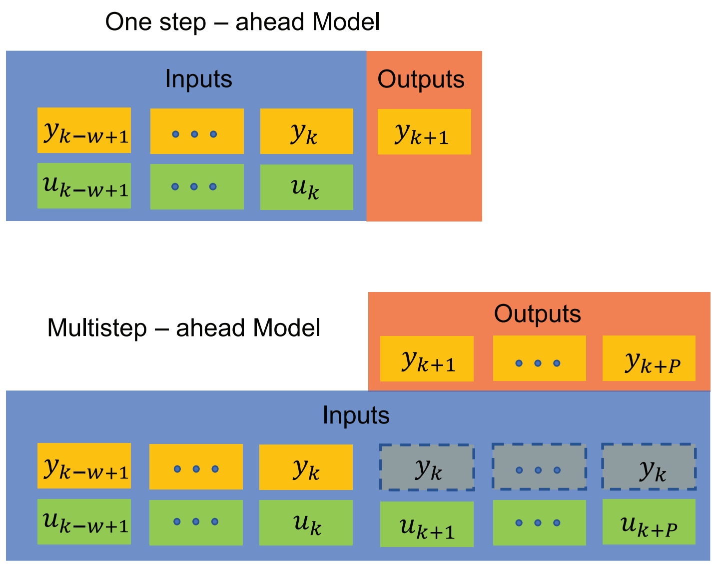


Figure 5: Training data structure comparison between one-step-ahead model and multistep-ahead model

training result is compared to the result of the counterpart LSTM model. The Transformer model consists of two encoders. The number of encoders and decoders for each system can be determined by hyperparameter optimization. Two encoders were used in this study. Hyperparameter optimization was not used in this study, but could be the subject of future work. Each encoder includes a multi-head attention layer, feedforward layer, dropout layer, and output layer. The multi-head attention layer consists of 10 heads with softmax activation function as described in Equation 2. The pre-processed inputs X_{os} and X_{ms} are used for Q, K, V vectors for the self-attention mechanism. The attention score vector is concatenated with the input (X) to pose the residual to feed it to the following feedforward layer. One feedforward layer consists of 100 neurons in the hidden layer with a hyperbolic tangent $(tanh)$ activation function, and the other output feedforward layer with the same number of neurons with the number of variables in feature X, which is 2 for this study. A dropout layer separates each feedforward layer with a dropout factor of 20%. The attention score vector, which also represents the output of the two encoder blocks, is fed to the final output feedforward layer to map it to the size of the label Y: 1 for the one-step-ahead model and P for the multistep-ahead model.

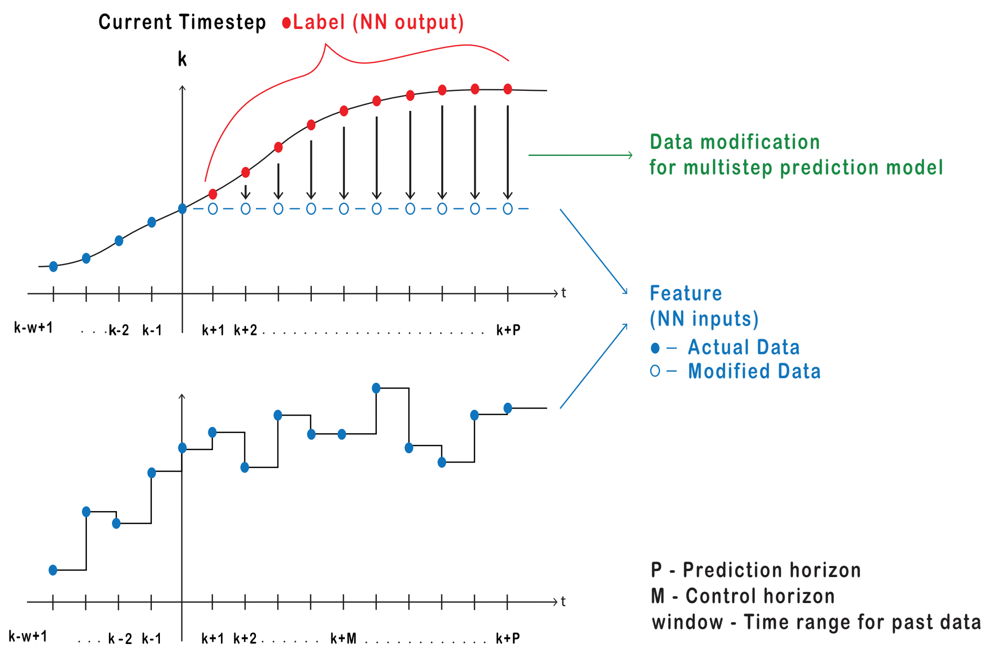


Figure 6: Training data modification for multistep-ahead prediction models

Meanwhile, the LSTM model consists of three LSTM layers and a dense layer, each separated by a dropout layer with a dropout factor of 20%. Each LSTM layer consists of 100 hidden states, and the current input and the 99 hidden states from the previous time step are fed into the LSTM layers at any time step. The final dense layer takes the 100 outputs generated by the third LSTM layer and maps them to the outputs. Both networks are trained using an Adam optimizer, with loss calculated as a mean-squared error (MSE). The summaries for multistep models are shown in Table 1 and 2. The number of trainable parameters of weight and bias is significantly less in the Transformer model than in LSTM.

Table 1: Summary of the Transformer model

Layer	Input Dimension	Output Dimension	Act. fn
MHA	$Batch \times 15 \times 2$	$Batch \times 15 \times 2$	$SoftMax$
Dense	$Batch \times 15 \times 2$	$Batch \times 15 \times 100$	$tanh$
Dense	$Batch \times 15 \times 100$	$Batch \times 15 \times 2$	$Linear$
MHA	$Batch \times 15 \times 2$	$Batch \times 15 \times 2$	$SoftMax$
Dense	$Batch \times 15 \times 2$	$Batch \times 15 \times 100$	$tanh$
Dense	$Batch \times 15 \times 100$	$Batch \times 15 \times 2$	$Linear$
Dense	$Batch \times 15 \times 2$	$Batch \times 10 \times 1$	$Linear$
Parameters		1,758	

Table 2: Summary of the LSTM model

Layer	Input Dimension	Output Dimension	Act. fn
LSTM	$Batch \times 15 \times 2$	$Batch \times 15 \times 100$	$tanh$
LSTM	$Batch \times 15 \times 100$	$Batch \times 15 \times 100$	$tanh$
LSTM	$Batch \times 15 \times 100$	$Batch \times 15 \times 100$	$tanh$
Dense	$Batch \times 15 \times 100$	$Batch \times 10 \times 1$	$Linear$
Parameters		203,010	

4.3. Transformer-Based Surrogate MPC

Two main parts of MPC are the prediction model and the optimization solver. In the NN-based surrogate MPC, NN models replace the classical transfer function model or physics-based model that generally uses differential equations. The optimization solver minimizes the MPC objective function by searching the best possible u sequences in the control horizon (M). The sum of squared error (SSE) objective function in the general MPC is shown in Equation 6.

$$
\begin{aligned}
\min_{\Delta \boldsymbol{U}} \quad & \Phi = (\hat{\boldsymbol{Y}} - \boldsymbol{Y}_{ref})^T \boldsymbol{Q} (\hat{\boldsymbol{Y}} - \boldsymbol{Y}_{ref}) + \Delta \boldsymbol{U}^T \boldsymbol{R} \Delta \boldsymbol{U} \\
s.t. \quad & 0 = f(\dot{x}, x, y, p, d, u) \\
& 0 = g(x, y, p, d, u) \\
& 0 \leq h(x, y, p, d, u)
\end{aligned}
\tag{6}
$$

where $\hat{\boldsymbol{Y}}$ and $\boldsymbol{Y}_{ref}$ are the column vectors for model prediction values and the reference trajectory from the time step $(k+1)$ to $(k+P)$. $\Delta \boldsymbol{U}$ is the column vector for the control moves for the future control horizon (M), from the time step $(k+1)$ to $(k+M)$. $\boldsymbol{Q}$ and $\boldsymbol{R}$ are the diagonal weighting matrices for multiple CVs (Controlled variables) and MVs (Manipulated variables). The detailed descriptions of the MPC equations can be found in [24].

The trained Transformer models are embedded into the MPC framework serving as prediction models providing the $\hat{\boldsymbol{Y}}$ in Equation 6. One-step-ahead prediction models for both LSTM and Transformer execute the models P times to get the full prediction for the defined prediction horizon. However, the multistep-ahead prediction models need only one execution for the same task, significantly improving computational efficiency.

5. Case Studies for Transformer Multistep-ahead Prediction Model MPC

This section discusses three MPC case studies with multistep Transformer models. The case studies include two experiments in the simulation environments and one with an actual temperature control test device, TCLab. The first case study examines one batch of MPC optimization calculation results for a SISO FOPDT process. The second and third case studies discuss continuous multi-input and multi-output (MIMO) MPC tests for a fluidized-bed roaster in gold ore recovery and with the TCLab device. The case studies contrast the improved computational efficiency of the proposed model among the different types of networks (LSTM) and prediction structures (one-step and multistep prediction).

5.1. Case I: First Order Dynamics Model (FOPDT)

Various types and structures of NN models are tested in the surrogate model MPC framework serving as prediction models, and the results are shown in Figure 7. Trained LSTM and Transformer models with one-step- and multistep-ahead prediction structures provide the model prediction values to the optimization solver. The gradient-based optimization solver iteratively searches the minimum objective function value (SSE). A sequential least-squares programming (SLSQP) solver is used in this study. The surrogate prediction models compute multiple prediction results for every control interval. Figure 7 shows the optimally predicted CV values based on the given setpoint of 1 and the computation time for different models for one control interval. The plot shows the result of one control interval in Figure 7 with the time steps in the prediction horizon ($P = 10$). Compared to the one-step-ahead models, there is a significant improvement in computation time for both LSTM and transformer with multistep-ahead prediction models. The multistep Transformer model shortens the computation time by about 20 times compared to the one-step LSTM model (from $10.85s$ to $0.53s$). Computation time in RNN-based deep learning models is substantially decreased by implementing two key modifications. Firstly, the sequence computation is replaced with a self-attention mechanism, thereby eliminating the recurrence inherent in the one-step ahead LSTM model. Secondly, a multistep ahead prediction structure is applied further enhancing the computational efficiency by calculating the whole length of the prediction simultaneously. The self-attention mechanism in the Transformer model accomplishes the same task with matrix operations reducing the internal sequences by w times less than the LSTM model. The other significant portion that contributes to the computation time is the simultaneous multistep-prediction structure. The multistep prediction structure simultaneously computes the model prediction values for the entire prediction horizon (from $k+1$ to $k+P$), which reduces the number of model calls by P times less than the one-step prediction model. The overall number of sequential computations occurring in one MPC control interval is shown in Table 3.

The number of iterations for MPC calculation (n) in Table 3 can be varied between different models as the numerical solver converges differently with every model. The traditional ODE (Ordinary Differential Equation) model-based MPC is also tested along with the NN models to provide the baseline performance. For a simple FOPDT model, the traditional MPC with a transfer function model outperforms NN models. As system complexity increases, similar NN models are capable of capturing complex, higher-order relationships. In contrast, the effort to build and the computation cost to execute physics-based models increases significantly. The use of multistep Transformers to model complex, nonlinear systems is a natural extension of this work and a topic of ongoing research.

The model accuracy is inferred from Figure 7. First, the u values are supposed

Table 3: Number of sequential computations occurring in one MPC calculation, where, w, P, and n represent a look-back window, prediction horizon, function evaluation of MPC optimization

Model Type	Sequential Computation	Multistep Prediction	MPC Iteration	Number of Executions
ODE (Sequential)	1	P	n	$P \times n$
LSTM-OS	w	P	n	$w \times P \times n$
LSTM-MS	$w+P$	1	n	$(w+P) \times n$
Transformer-OS	1	P	n	$P \times n$
Transformer-MS	1	1	n	n

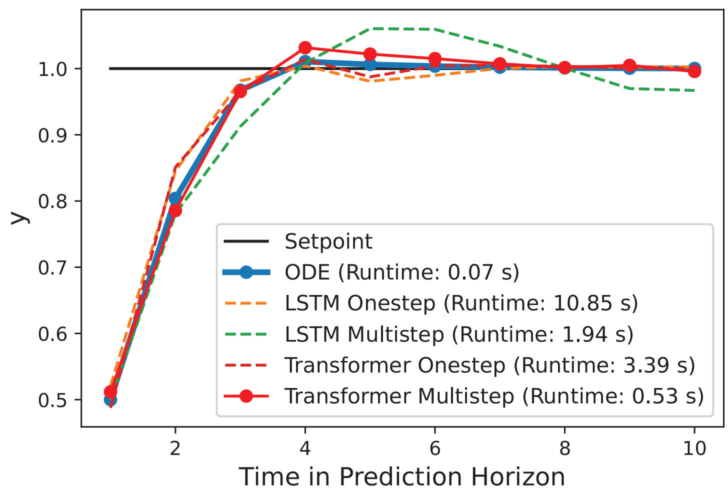
to be at a steady-state '1.0' like the ODE model result because the process gain (K_p) in the FOPDT model is defined as '1.0'. The steady-state u value must be the same as the y value with '1.0' as the output setpoint. However, the u results of the multistep-ahead LSTM model and the one-step-ahead Transformer model converge to slightly different values than 1.0, which means those models have some model mismatch with the process model. In addition, the LSTM multistep model shows a significant model mismatch on both y and u, while the multistep Transformer model follows the ODE model result. The different outcomes can be explained by two distinguished methods to learn the temporal dependency. The suggested training data structure for the multistep-ahead prediction models includes both the past measured data ($k-w:k$) and future prediction horizon data ($k+1:k+P$), especially for the feature (X) set. As discussed in Section 4.1, this is necessary not only for including the entire prediction horizon data to learn simultaneously but also to pass the control horizon part of the feature ($u_{k+1}:u_{k+M}$) to the SLSQP solver as decision variables to be solved. However, while the prediction horizon is part of the data in the feature set (X), the prediction horizon part of y data ($y_{k+1}:y_{k+P}$) is replaced with the currently measured data (y_k). This modification avoids including the labeled data in the feature set before training while maintaining a consistent data dimension. However, irrelevant data is included due to this modification, which introduces irregular temporal dependency for time-series data training. The way that the Transformer architecture learns the temporal dependency is completely different than RNNs. The self-attention mechanism employs a conditional probability, selectively including the useful relationship and excluding the irrelevant relationship between input and output data. Therefore, the Transformer is more effective in learning the irregular temporal dependency than LSTM that structurally includes the temporal dependency in a sequential manner.

5.2. Case II: Fluidized-Bed Roaster Model

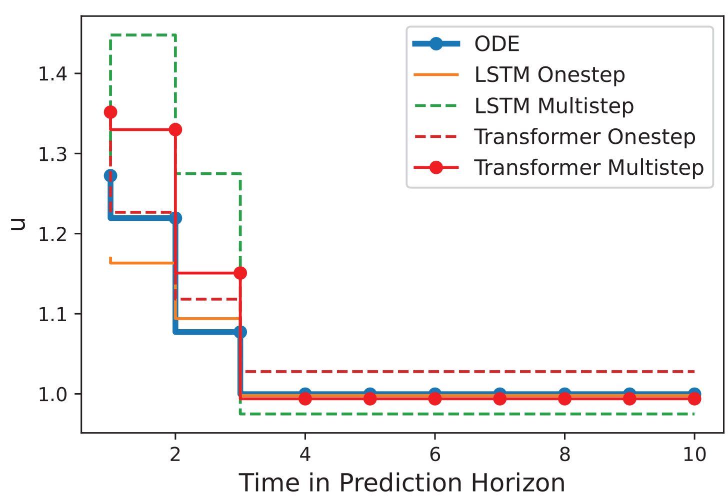
The roasting process is a part of gold production that recovers the gold from the ore. The substances that negatively affect the carbon-in-leach (CIL) process are removed by the oxidation reaction in the roasting process. The particular type of gold ore, called refractory gold ore, has several characteristics that interfere with satisfactory gold recovery in the traditional CIL. The iron sulfide in the refractory gold ore confines the gold in the minerals that prevent contact with the cyanide solution. The carbonaceous minerals in the ore tend to re-absorb the recovered gold component from the cyanide solution. Thus, the refractory gold ore needs to be pretreated in the roaster, converting the iron sulfide mineral and carbonaceous mineral to oxides. The roaster consists of two stages of fluidized bed reactors (FBRs) that oxidize sulfide compounds and organic carbon in the ore. Before the roasting process, the ore is crushed into fine particles with an average size of $75\mu m$ in diameter by passing them through a grinding process. The fine particles are fed into the roaster from the top and are fluidized by the upward force generated by the oxygen flow from the bottom of the roaster. Three critical factors that affect the oxidation reaction are ore particle size, oxygen content in the combustion gas, and bed temperature. Compared with traditional air-roasting, oxygen-roasting yields better gold recovery by improving reaction efficiency. Compared to oxygen roasting, air roasting requires a longer retention time at a higher temperature to maintain a high conversion of sulfidic metals and carbonaceous matter. These conditions easily over-roast and soften the ore particles (glassy flux), blocking the pores of the material from contacting the combustion gas. Over-roasting also affects the subsequent CIL process by encapsulating the gold in the glassy fluxed ore [25, 26, 27, 28]. The exothermic reaction generates most of the thermal energy required for the oxidation reaction. Additional heat is provided by feeding sulfur prills as fuel. The major reactions [27] are shown below and a diagram of the roasting process is shown in Figure 8.

Combustion of organic carbon:

$$\mathrm{C}(s) + \mathrm{O}_2(g) \rightarrow \mathrm{CO}_2(g)$$



(a) Setpoint tracking performance of CV and computation time



(b) Control movement of MV corresponding to given setpoint

Figure 7: Results of MPC calculation compared to the various types and structures of NN models. Setpoint tracking performance of CV and computation time (Upper), and control movement of MV corresponding to given setpoint (Lower)

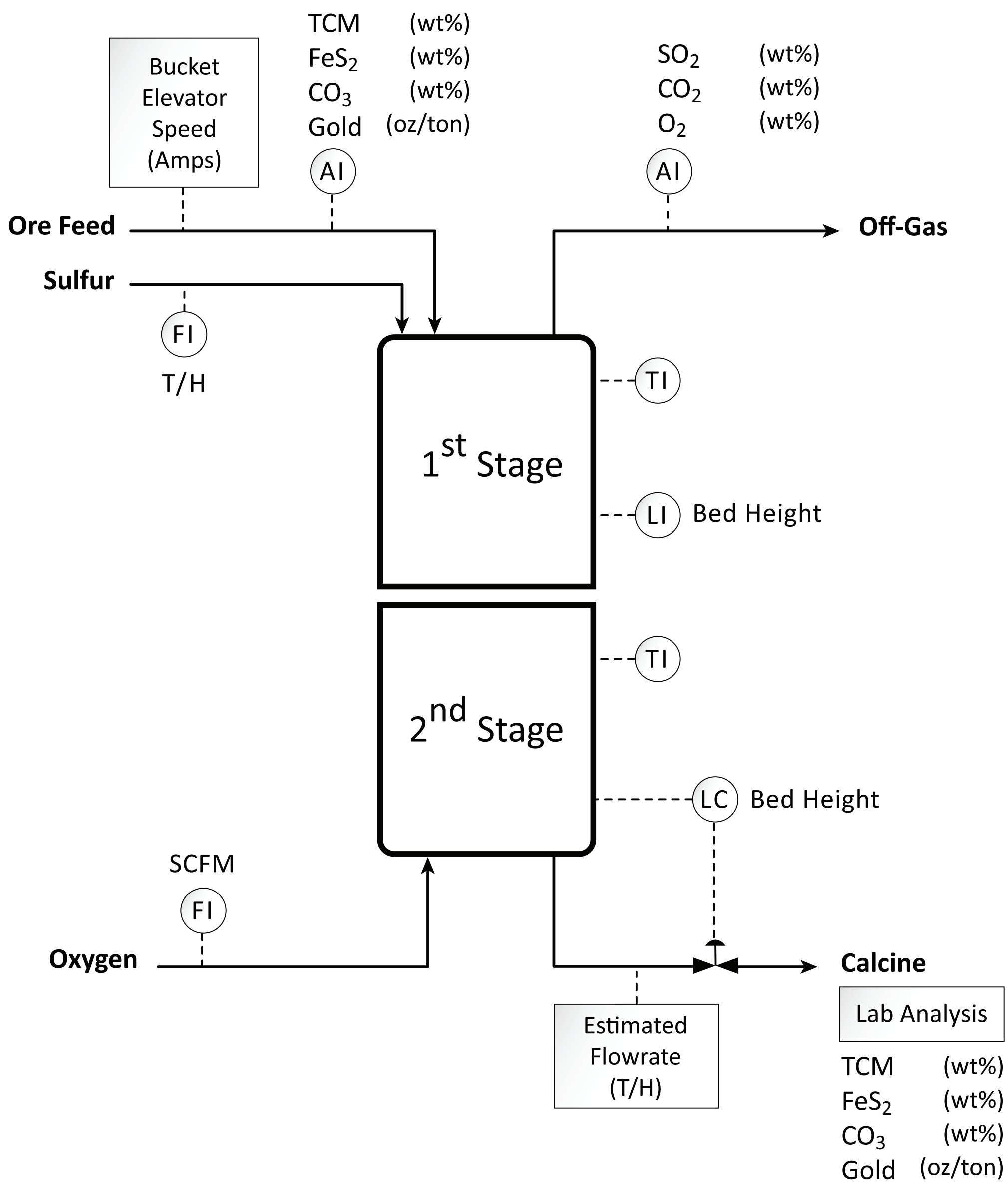


Figure 8: Schematic drawing of the roaster process

Combustion of iron sulfides (pyrite and pyrrhotite):

$$7\,FeS_2(s) + 6\,O_2(g) \rightarrow Fe_7S_8(s) + 6\,SO_2(g)$$
$$4\,Fe_7S_8(s) + 53\,O_2(g) \rightarrow 14\,Fe_2O_3(s) + 32\,SO_2(g)$$

Combustion of sulfur prills (fuel):

$$S(s) + O_2(g) \rightarrow SO_2(g)$$

Retention of Sulfur dioxide:

$$2\,CaCO_3(s) + 2\,SO_2(g) + O_2(g) \rightarrow 2\,CaSO_4(s) + 2\,CO_2(g)$$

The actual operation data is usually insufficient to train the NN surrogate models because the actual operations are often maintained in a narrow range that does not provide the information for the broader operation range where the optimal operation point may exist. Thus, a physics-based model is developed for process simulation experiments, including data generation for training NN models and emulating the actual process as a digital twin model during the controller performance test. The physics-based model for this research consists of heat and mass balance equations and reaction kinetics. The reaction kinetics uses a shrinking core model (SCM), whose reaction rate is dominated by pore diffusion.

5.2.1. Physics-based Model of a Roaster Process

A physics-based model for the roaster is developed to generate the Transformer model training and validate the Transformer model-based MPC performance. The SCM is used for reaction kinetics of ore particle oxidation reaction shown in Equation 7. The rate equation of the SCM consists of three parts: Diffusion through gas film controls, diffusion through ash layer control, and chemical reaction controls. The diffusion through gas film control is negligible for engineering applications because it has an effect only for a short period of the initial stage. Once the ash layer starts forming on the particle surface, the ash layer diffusion step controls the overall reaction rate [29, 30]. The ash layer diffusion term requires a calculation of the effective diffusivity, $\mathscr{D}_e$. The Chapman-Enskog equation is used for calculating the effective diffusivity ($\mathscr{D}_e$) shown in Equations 8 - 10.

$$\begin{aligned} -r_i &= \frac{A_p C_\mathrm{i}}{\frac{1}{k_g} + \frac{R_p}{6\mathscr{D}_e} + \frac{1}{k''}} \\ &= \frac{A_p C_\mathrm{i}}{\frac{R_p}{6\mathscr{D}_e} + \frac{1}{k''}} \end{aligned} \tag{7}$$

$$\mathscr{D}_{AM} = 0.0018583\sqrt{T^3\left(\frac{1}{M_{O_2}}+\frac{1}{M_{O_2}}\right)}\frac{1}{p\sigma_{AB}^2\Omega_{\mathscr{D},AB}} \tag{8}$$

$$\Omega_{\mathscr{D},AB} = \frac{1.06036}{T^{*0.15610}} + \frac{0.19300}{\exp(0.47635T^*)} + \frac{1.03587}{\exp(1.52996T^*)} + \frac{1.76474}{\exp(3.89411T^*)}$$
$$where,\ T^* = \kappa T/\varepsilon \tag{9}$$

$$\mathscr{D}_e = \frac{\mathscr{D}_{AM}\varepsilon_p}{\tau},\ \ where\ \ \tau = \varepsilon_p^{-0.41} \tag{10}$$

$$k'' = A_p T \exp(-E_r/RT) \tag{11}$$

$$\frac{dN_i}{dt} = F_{i,in} - F_{i,out} + r_i \tag{12}$$

$$NC_p\frac{dT_i}{dt} = F_{i,0}H_{in} - F_{i,out}H_{out} + \Delta H_{reaction} \tag{13}$$

Table 4: Lumped Parameters from Physics-based Roaster Model

Quantity	Value
Average surface area of ore particle (A_p)	0.0176 m^2
Average radius of ore particle (R_p)	37.5 μm
Atmospheric pressure (p)	1 atm
Molecular weight of Oxygen (M_{O_2})	32 g/mol
Lennard-Jones potential parameter for Oxygen molecules (σ)	3.433 Å
Lennard-Jones potential parameter for Oxygen molecules (ε/K)	113 K
Average porosity of ore particles (ε_p)	0.5
Effective diffusivity ($\mathscr{D}_e$)	
Diffusivity for gases ($\mathscr{D}_{AM}$)	
Mass transfer coefficient between air and particle (k_g)	
Rate constant for the surface reaction (k'')	
Reactor temperature (T)	
Tortuosity (τ)	
Collision Integral for diffusivity (Ω)	

5.2.2. Train Transformer Model for Roaster MPC

Training data is generated by simulating the physics-based roaster model. The roaster simulation model has fourteen main operation variables, including six input and eight output variables. The first set of input variables is related to the quantity of reactor feed, such as the quantity of ore feed, the mass flow rate of sulfur prill, and the volume flow rate of oxygen. These are the primary operation variables of roaster operation and are the candidates for an advanced process control system manipulated variables (MV). The second set of input variables is the quality of the ore feed, such as the contents of a specific component. Compared to the first set of inputs, the second set is not adjustable for the process operation; thus, it is considered an advanced process control system disturbance variable (DV). The roaster model in this research includes organic carbon, iron sulfide, and carbonate content in the feed ore. The random input signals for the six reactor input variables are generated and simulated in the reactor model, which creates the associated output variable responses. The random input signals and output variable responses are then used for training the Transformer model. The reactor output variables in this case study include the off-gas and calcine specifications and the two reactor temperatures. The input and output variables of the roaster process are shown in Table 5 and the roaster operation range for the random input signals refer to [27]. Fifteen thousand sample points are generated for training the Transformer model, and a short segment of the input and output data is shown in Figures 9 and 10. Figure 10 also displays the Transformer model fitting. The blue dashed line in the figure represents the sampled data from the roaster physics-based simulation model, and the red line shows the prediction results with the Transformer model. The model fitting results validate that the Transformer model can provide accurate model predictions for control.

The network configuration for the Transformer model is shown in Table 6. It consists of Multi-head attention (MHA) layers followed by a set of Feedforward layers, as depicted in Figure 2 in the previous section 2. The tensor dimension for each layer represents: $(Batch) \times (Length\ of\ one\ snapshot) \times (Number\ of\ variables)$, where the $Batch$ size for the roaster training is the same as the length of the time series. Each snapshot includes the past data with a length of look-back window size, of six, plus the prediction horizon length, of ten, which forms the second dimension of MHA input tensor, sixteen. The output dimension of the last feed-forward layer ($Dense$ layer) is reduced to the size of the multistep prediction result with the length of the prediction horizon, ten, and the number of system output variables, eight: $(Batch \times 10 \times 8)$.

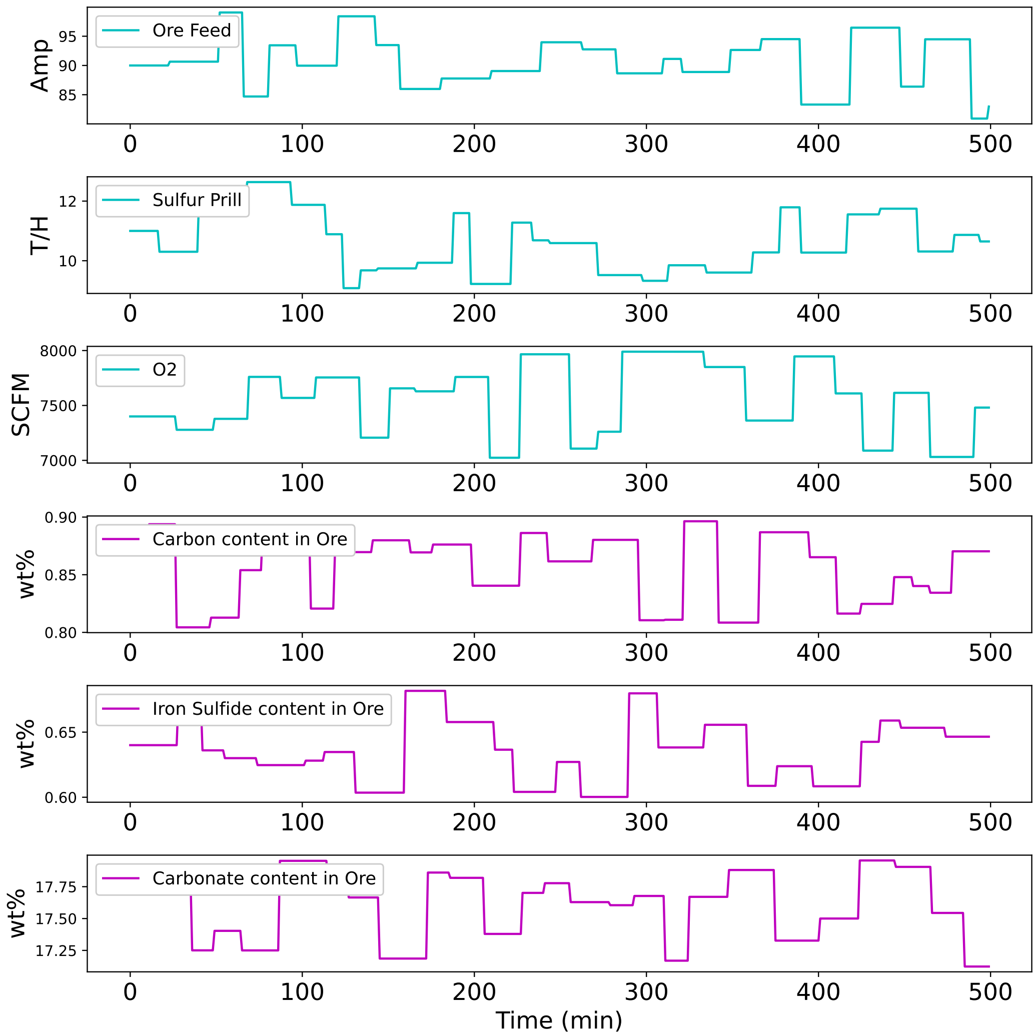


Figure 9: Roaster input data for Transformer model training

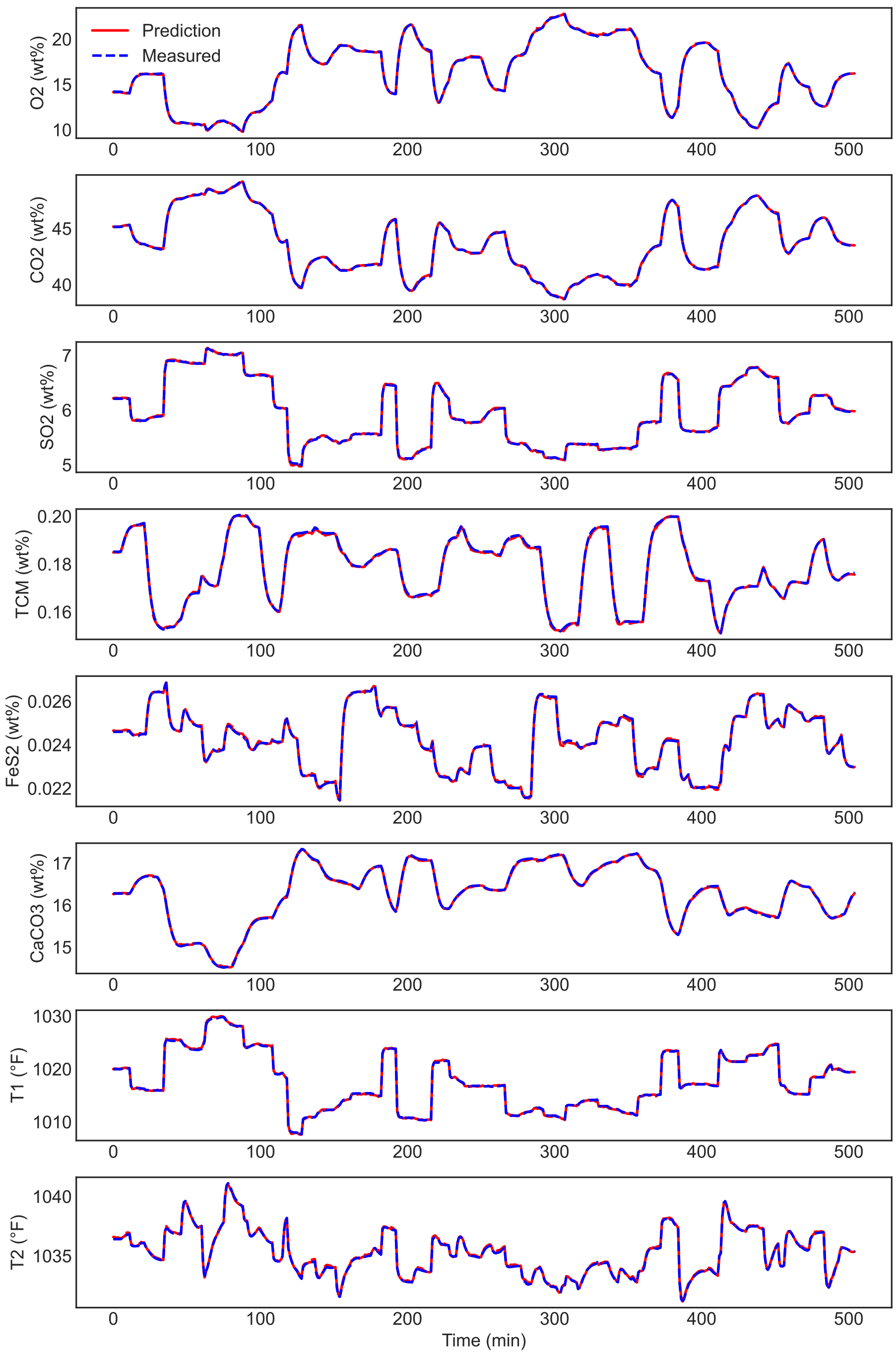


Figure 10: Transformer model training result for Roaster

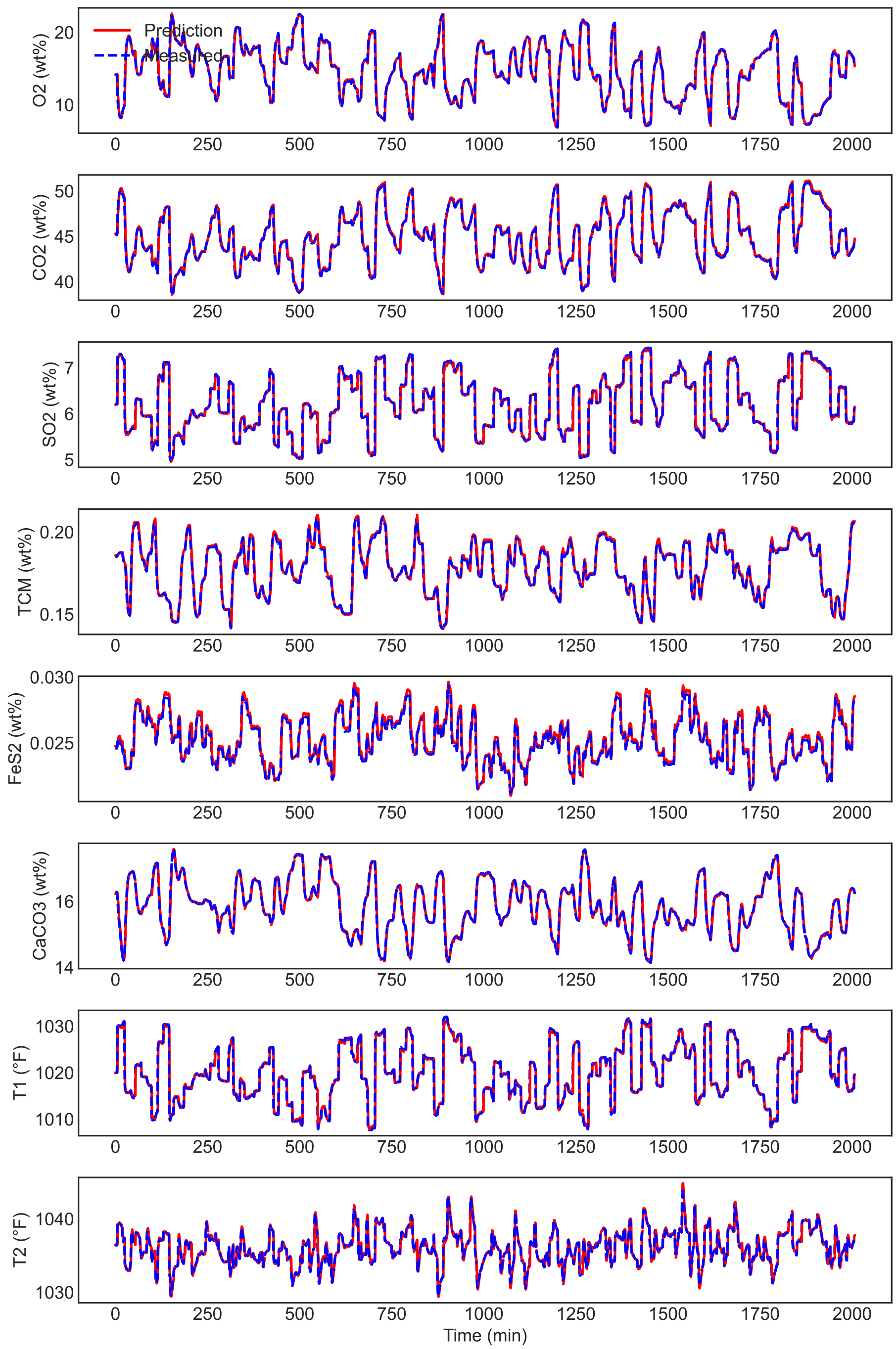


Figure 11: Transformer model validation result for Roaster

Table 5: Operation variables of roaster simulation model

Variables in Roaster model	
Input variables	Output variables
	O_2 in Offgas (wt%)
Ore feed amount (Amp)	CO_2 in Offgas (wt%)
Sulfur prill (T/H)	SO_2 in Offgas (wt%)
Oxygen (SCFM)	TCM in Calcine (wt%)
Carbon content in Ore (wt%)	FeS_2 in Calcine (wt%)
Iron Sulfide content in Ore (wt%)	$CaCO_3$ in Calcine (wt%)
Carbonate content in Ore (wt%)	Stage1 Temperature (F)
	Stage2 Temperature (F)

Table 6: Summary of the Transformer model for roaster data

Layer	Input Dimension	Output Dimension	Act. fn
MHA	$Batch \times 16 \times 14$	$Batch \times 16 \times 14$	$SoftMax$
Dense	$Batch \times 16 \times 14$	$Batch \times 16 \times 100$	$tanh$
Dense	$Batch \times 16 \times 100$	$Batch \times 16 \times 14$	$Linear$
MHA	$Batch \times 16 \times 14$	$Batch \times 16 \times 14$	$SoftMax$
Dense	$Batch \times 16 \times 14$	$Batch \times 16 \times 100$	$tanh$
Dense	$Batch \times 16 \times 100$	$Batch \times 16 \times 14$	$Linear$
Dense	$Batch \times 16 \times 14$	$Batch \times 10 \times 8$	$Linear$
Parameter		26,216	

5.2.3. Control Results for Roaster

An MPC controller is designed as a case study to test the prediction accuracy and solution time of the Transformer based model in the MPC system. This case study selects two controlled variables (CVs) and two manipulated variables (MVs). The two reactor temperatures at each stage of the roaster are chosen for two CVs, and ore feed rate and sulfur prill inlet flows are chosen for the two MVs. The reactor temperature is one of the most critical operation variables directly affected by the reaction status. Thus, it is commonly selected as a major CV in many advanced process control applications inferring the conversions. The ore particles are transported by the bucket elevator to the 1st stage roaster, and the feed rate is adjusted by changing the speed of the bucket elevator. The sulfur prill feed rate is assigned to the second MV that maintains the reaction temperature requirement to ensure high sulfide sulfur and organic carbon conversion. The other process input variables, including the ore feed composition and oxygen inlet flow, are retained at a constant value during the experiment. The MPC settings are shown in Table 7.

Table 7: Settings of Roaster MPC

Parameters	Setting
Prediction Horizon (P)	10 min
Control Horizon (M)	6 min
CV Weight for SSE (q_{T1})	$1x10^3$
CV Weight for SSE (q_{T2})	$1x10^2$
MV Weight for rate of change ($r_{Ore\ feed}$)	$1x10^3$
MV Weight for rate of change ($r_{Sulfur\ prill}$)	1
MV Max move (Ore feed)	1 Amp
MV Max move (Sulfur prill)	0.2 T/H
MV Bounds for SLSQP (Ore feed)	80 - 100 Amp
MV Bounds for SLSQP (Sulfur prill)	9 - 13 T/H

A simulated MPC control result for the roaster process is shown in Figure 12. In this test, two temperature setpoints are changed to validate the setpoint tracking performance of the Transformer model-based MPC. At 20 minutes, MPC is activated and starts controlling the temperatures. Ore and sulfur feed rates as MVs move to meet the temperature setpoints based on the MPC control calculation. To increase the temperatures at a given condition, MPC decreases ore feed and increases the sulfur prill feed because the sulfur prill has a higher reaction rate than ore particle as it does not go through the ash layer diffusion. This demonstrates that the Transformer controller provides proper prediction by taking into account the reaction kinetics of gold ore roasting. Although this case study only includes a subset of operation variables, the complete set of variables can be included in future research studies not only for the MPC application but also for the real-time optimization application that deals with an economic objective and steady-state targets.

5.3. Case III: Temperature Control Lab Device

In this section, an experiment is conducted using a temperature control lab (TCLab). The device consists of two heater inputs and two temperature output measurements. This device has been used for control education and research

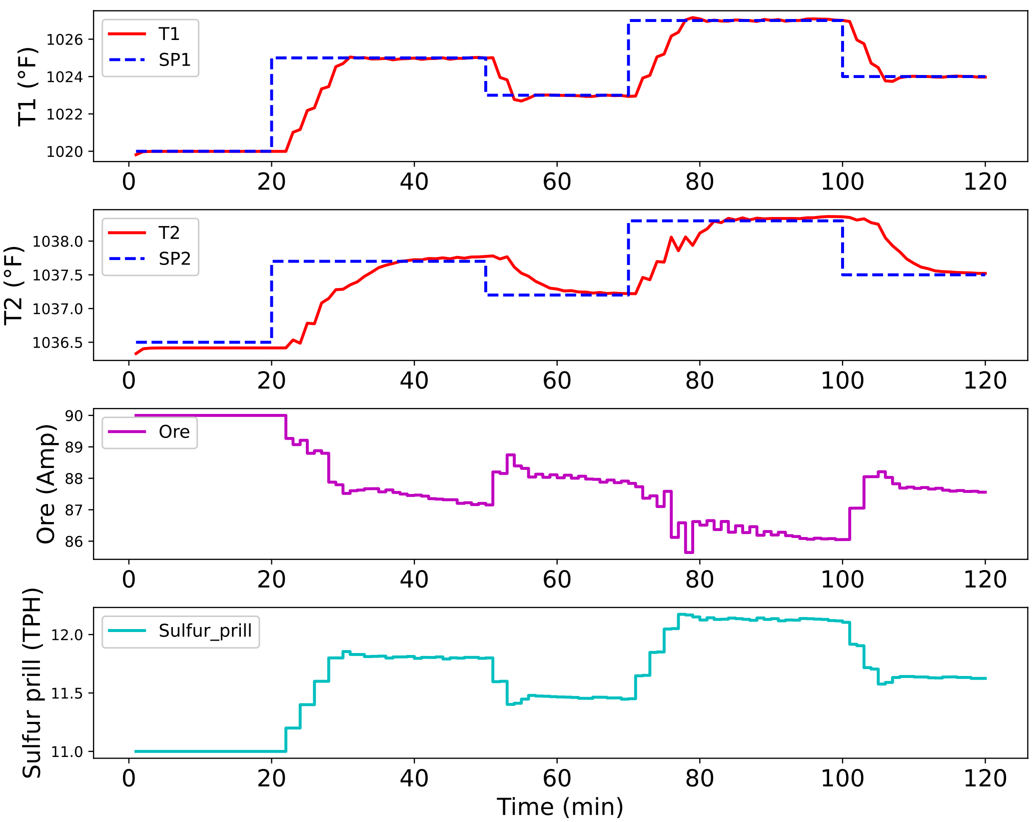


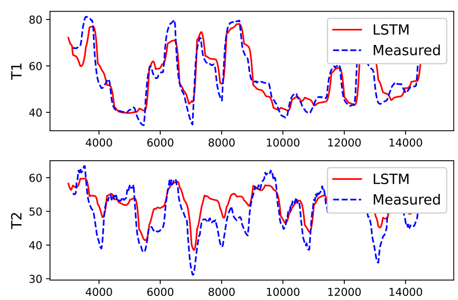
Figure 12: MPC result for Roaster temperature control

purposes, demonstrating control theory and machine learning-based automation methods [31].

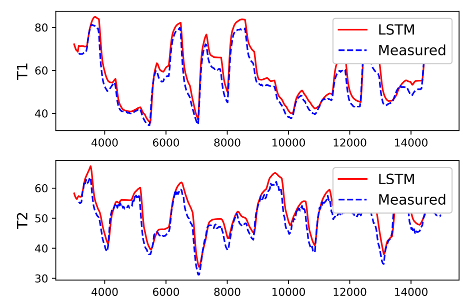
In the experiment, the new Transformer multistep-ahead prediction model is developed for the TCLab device and embedded into the MPC algorithm. The newly proposed Transformer multistep-ahead prediction model is compared with three other options, including the Transformer one-step-ahead prediction model and one-step and multistep-ahead prediction models with the LSTM network. In the TCLab case study, all four different model types are tested for offline model development to show model fitness. Then, two of the four models are chosen to validate the online control performance: the multistep Transformer model and the one-step LSTM model. The two models are compared for the computational efficiency and setpoint tracking performance in the online control comparison.

The training data is generated with the TCLab device with random step test input signals for the two heaters. The step-test random signals are generated with a combination of two individual random number generators. The first regulates the amplitude of step changes, while the second generates the number of intervals between the step changes. The amplitude range for the first random number generator is zero to one hundred percent of the maximum heater value of the TCLab. The step interval range for the second random number generator is in the range of three hundred to six hundred seconds to account for the dynamic behavior of the TCLab. This step-change interval distribution ensures the collection of various input-output relationships with variable frequency responses.

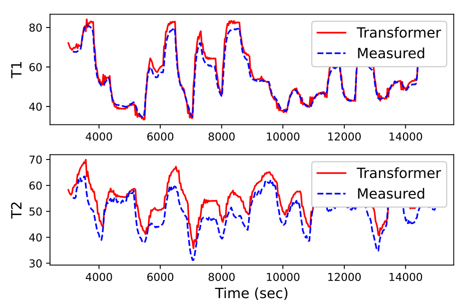
Data is collected every second for twenty-five thousand seconds and down-sampled every 30 seconds balancing the control calculation speed and control performance in the MPC application. The first training data set includes the first 3,000 seconds of the data to represent a case of insufficient training data. The other training session uses the full 25,000 seconds of data. The validation results with MAE values for each model are shown in Table 8. In addition, our case studies have certain limitations when confirming the LSTM model performance with irregular temporal dependencies. While this has been repeatedly demonstrated in the field of natural language processing, the model prediction accuracy shown in Table 8 does not fully encapsulate the MS LSTM viability as an MPC prediction model. Thus, a more extensive study with a primary focus on prediction performance could be an interesting topic for future research. Figure 13 depicts the model training results of using either a small data set or large data to train one-step LSTM models and multistep Transformer models on TCLab data. Figure 13a, shows the one-step model training result using the small three thousand seconds of data for training. The model predicts the general shape of the measured data but is not able to accurately predict the measured values. Figure 13b shows the training result of the one-step LSTM model trained on the large twenty-five thousand seconds of data. The model accurately predicts both the shape of the measured data and the values of the data over the training range. Figure 13c shows the training result for the Transformer one-step model using three thousand seconds of data. Compared with the LSTM one-step model trained in Figure 13a on the smaller data range, overall accuracy is greatly improved. In addition, when Figure 13c is compared with the LSTM model trained on the large set of data depicted in Figure 13b, the Transformer model, despite having less training data, is more accurate in certain intervals than the LSTM. Figure 13d shows the results for using the twenty-five thousand-second data set on the Transformer one-step model. Of the four models in Figure 13, it is best able to predict the measured data across the entire range.



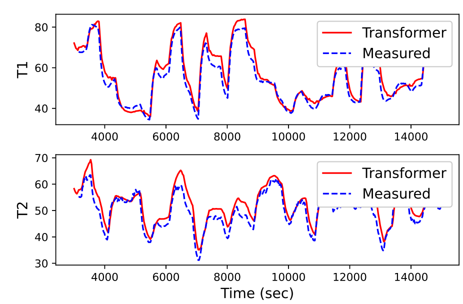
(a) LSTM (one-step model): using 3,000 seconds of data



(b) LSTM (one-step model): using 25,000 seconds of data)



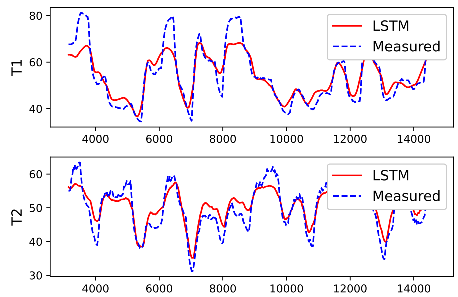
(c) Transformer (one-step model): using 3,000 seconds of data)



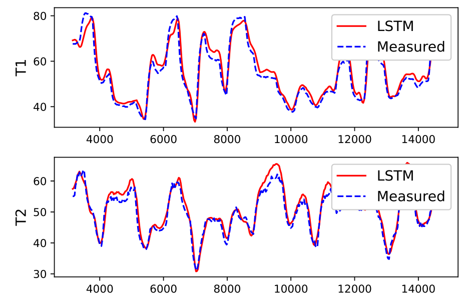
(d) Transformer (one-step model): using 25,000 seconds of data)

Figure 13: One-step Models validation for LSTM and Transformer: in comparison with using 3,000 (a, c) and 25,000 seconds data (b, d)

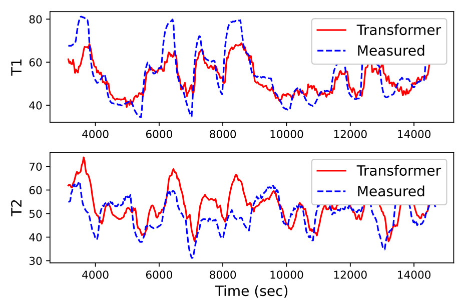
Figure 14 depicts the results of using either a small data set or large data to



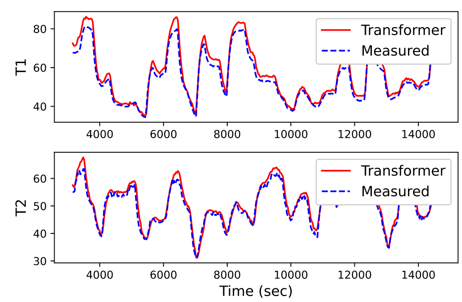
(a) LSTM (Multistep model): using 3,000 seconds of data



(b) LSTM (Multistep model): using 25,000 seconds of data)



(c) Transformer (Multistep model): using 3,000 seconds of data)



(d) Transformer (Multistep model): using 25,000 seconds of data)

Figure 14: Multi-step Models validation for LSTM and Transformer: in comparison with using 3,000 (a, c) and 25,000 seconds data (b, d)

Table 8: Training result comparison

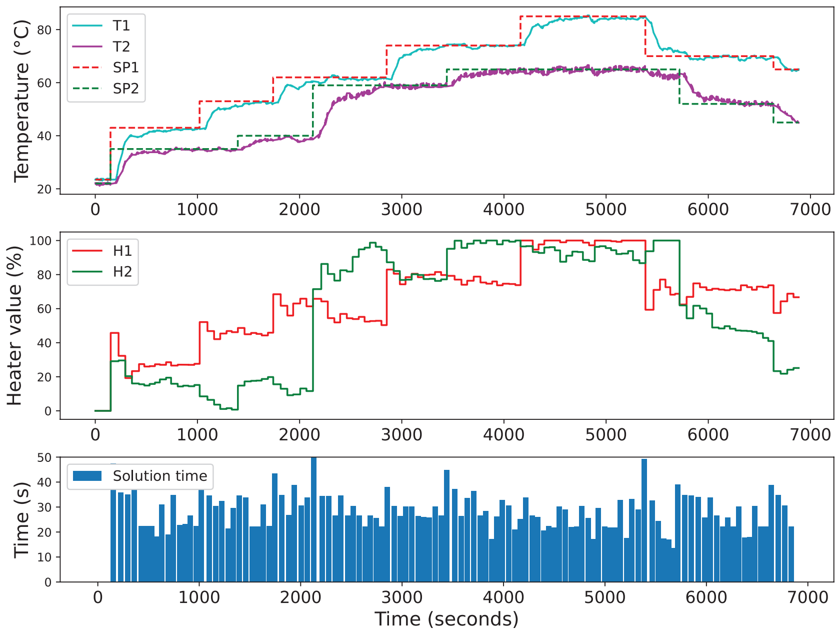
			MAE	
			Quantity of training data	
Prediction Type	Network Type	Variables	3,000 sec	24,000 sec
One-step	LSTM	T_1 ($^\circ C$)	4.03	3.82
		T_2 ($^\circ C$)	3.88	2.64
	Transformer	T_1 ($^\circ C$)	2.16	3.32
		T_2 ($^\circ C$)	5.26	5.63
Multistep	LSTM	T_1 ($^\circ C$)	3.59	2.59
		T_2 ($^\circ C$)	2.70	1.57
	Transformer	T_1 ($^\circ C$)	2.89	2.61
		T_2 ($^\circ C$)	2.87	1.47

train multi-step LSTM models and Transformer models on TCLab data. Figure 14a shows the results of using the smaller three thousand second data to train a multistep LSTM model. Figure 14b depicts the results of using the larger twenty-five thousand data set to train the multistep LSTM model. The model accurately predicts the measured data. Figure 14c show the results of training a multistep Transformer model on the small three thousand second data. It is more accurate than the multistep LSTM trained on 3,000 seconds of data shown in Figure 14a, and in certain ranges, it is more accurate than the multistep LSTM model trained on twenty-five thousands seconds of data shown in Figure 14b. Figure 14d depicts the result of using the twenty-five thousand seconds data set to train the multistep Transformer model. This model is the most accurate of the four models, as expected.

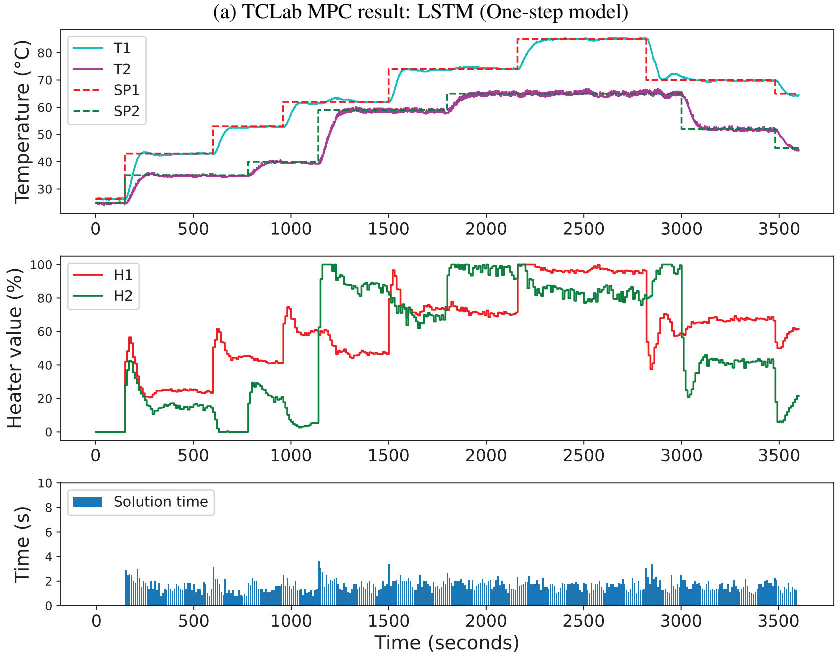
The LSTM one-step prediction model and the Transformer multistep prediction model are chosen to validate the online control performance, as shown in Figure 15. As observed in the FOPDT model experiment in Section 5.1, a significant difference in MPC calculation time is observed between LSTM one-step model and the Transformer multistep model. The solution time at each control interval is also shown in Figure 15a and 15b, along with the TCLab system variables. The average solution time of the Transformer multistep model is 1.64 seconds. In contrast, the LSTM one-step model takes 28.5 seconds on average, which is more than fifteen times slower than the Transformer multistep model. The MPC solution time affects the control performance in many different ways. The slower solution time must be considered to select the control interval during the MPC design. In this test, for example, the control interval for the Transformer multistep model can be set to 2 seconds to account for overhead communication time and variable solver iterations. However, the LSTM one-step model control interval is set to 30 seconds as it needs longer than 28.5 seconds of average MPC solution time. This slower MPC solution time also affects the overall control performance. The mean absolute error (MAE) value between setpoint and temperatures measures the setpoint tracking performance of each type of controller and is observed visually in Figure 15a and 15b. Note that the total simulation time to fulfill the identical setpoint sequences for both controllers is different: sixty-eight hundred seconds for LSTM one-step model and thirty-six hundred seconds for the Transformer multistep model. The surrogate MPC efficiency affects setpoint tracking performance or disturbance compensating performance. For example, the rise time, the amount of time that the controlled variable (CV) reaches the setpoint for the first time since the latest setpoint has been changed, is 100 seconds, while the rise time of the LSTM takes more than 500 seconds on average. The weights for the MPC objective function for both controllers are set to the same settings. Table 9 summarizes the control performance metrics.

Table 9: MPC performance comparison between LSTM and Transformer models

Model Type		LSTM One-step	Transformer Multistep
Simulation time		6885s	3600s
Control interval		30s	10s
Number of MPC executions		115	345
	Min	13.61s	0.79s
MPC solution time	Max	51.07s	3.60s
	Avg	28.54s	1.64s
MAE (Mean absolute error)	T1 ($^{\circ}C$)	10.97	8.26
	T2 ($^{\circ}C$)	11.03	8.01



(a) TCLab MPC result: LSTM (One-step model)



(b) TCLab MPC result: Transformer (Multistep model)

Figure 15: TCLab MPC control result comparison between one-step LSTM model and Multistep Transformer model

6. Conclusions

A novel Transformer with a multistep prediction model structure is proposed for MPC. The new multistep prediction model structure is tested with the Transformer NN architecture. The Transformer model demonstrates improved computation time with the multistep prediction structure compared to the one-step prediction structures with the LSTM model. The parallel data processing of the Transformer architecture eliminates the sequential computation nature of RNN-based models, including LSTM. Additionally, the simultaneous multistep prediction structure removes the recursive procedure of the one-step prediction model that updates the prediction horizon value one step at a time. As a result, the proposed multistep Transformer model calculates the entire length of the model prediction with a single execution of the forward propagation. The increased computational speed of a control system, such as with MPC, substantially improves the control performance by enabling a shorter control interval in real-time control practice.

In addition, the self-attention mechanism in the Transformer model learns the irregular temporal dependencies more effectively than the sequential learning mechanism in the LSTM. The irregular temporal dependencies introduced in the multistep prediction structure are properly handled with the Transformer model compared to the LSTM, which is briefly shown in the first case study result (Figure 7). The multistep Transformer model shows more dependable accuracy than the multistep LSTM model, even with a smaller model size, containing fewer trainable parameters (Table 1 and 2).

This study does not carry out a thorough evaluation of the model quality across a variety of problem complexities or examine how different types of models, including alternatives like MS-LSTM and OS-Transformer, would perform under these conditions. Such an investigation could serve as an intriguing focus for future research endeavors.

References

[1] J. H. Lee, J. Shin, M. J. Realff, Machine learning: Overview of the recent progresses and implications for the process systems engineering field, Computers and Chemical Engineering 114 (2018) 111–121. doi:10.1016/j.compchemeng.2017.10.008.
URL http://dx.doi.org/10.1016/j.compchemeng.2017.10.008

[2] S. J. Qin, L. H. Chiang, Advances and opportunities in machine learning for process data analytics, Computers and Chemical Engineering 126 (2019) 465–473. doi:10.1016/j.compchemeng.2019.04.003.

[3] V. Venkatasubramanian, The promise of artificial intelligence in chemical engineering: Is it here, finally?, AIChE Journal 65 (2) (2019) 466–478. doi:10.1002/aic.16489.

[4] R. B. Gopaluni, A. Tulsyan, B. Chachuat, B. Huang, J. M. Lee, F. Amjad, S. K. Damarla, J. Woo Kim, N. P. Lawrence, Modern machine learning tools for monitoring and control of industrial processes: A survey, IFAC-PapersOnLine 53 (2) (2020) 218–229. doi:10.1016/j.ifacol.2020.12.126.
URL https://doi.org/10.1016/j.ifacol.2020.12.126

[5] S. W. Wang, D. L. Yu, J. B. Gomm, G. F. Page, S. S. Douglas, Adaptive neural network model based predictive control of an internal combustion engine with a new optimization algorithm, Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering (2006). doi:10.1243/095440706X72754.

[6] M. Lee, S. Park, A new scheme combining neural feedforward control with model-predictive control, AIChE Journal 38 (2) (1992) 193–200. doi:10.1002/aic.690380204.

[7] R. K. Al Seyab, Y. Cao, Differential recurrent neural network based predictive control, Computers and Chemical Engineering 32 (7) (2008) 1533–1545. doi:10.1016/j.compchemeng.2007.07.007.

[8] Y. Wang, K. Velswamy, B. Huang, A long-short term memory recurrent neural network based reinforcement learning controller for office heating ventilation and air conditioning systems, Processes 5 (3) (9 2017). doi:10.3390/pr5030046.

[9] J. Drgoňa, A. R. Tuor, V. Chandan, D. L. Vrabie, Physics-constrained deep learning of multi-zone building thermal dynamics, Energy and Buildings 243 (2021). doi:10.1016/j.enbuild.2021.110992.

[10] W. C. Wong, J. Li, X. Wang, E. Chee, J. Li, X. Wang, Recurrent neural network-based model predictive control for continuous pharmaceutical man-

ufacturing, Mathematics 6 (11) (7 2018). doi:10.3390/math6110242.
URL http://arxiv.org/abs/1807.09556

[11] Z. Wu, A. Tran, Y. M. Ren, C. S. Barnes, S. Chen, P. D. Christofides, Model predictive control of phthalic anhydride synthesis in a fixed-bed catalytic reactor via machine learning modeling, Chemical Engineering Research and Design 145 (2019) 173–183. doi:10.1016/j.cherd.2019.02.016.

[12] J. Park, R. A. Martin, J. D. Kelly, J. D. Hedengren, Benchmark temperature microcontroller for process dynamics and control, Computers & Chemical Engineering 135 (2020) 106736. doi:10.1016/j.compchemeng.2020.106736.
URL https://linkinghub.elsevier.com/retrieve/pii/S0098135419310129

[13] D. Machalek, J. Tuttle, K. Andersson, K. M. Powell, Dynamic energy system modeling using hybrid physics-based and machine learning encoder–decoder models, Energy and AI 9 (2022) 100172. doi:10.1016/j.egyai.2022.100172.

[14] A. Mesbah, K. P. Wabersich, A. P. Schoellig, M. N. Zeilinger, S. Lucia, T. A. Badgwell, J. A. Paulson, Fusion of Machine Learning and MPC under Uncertainty: What Advances Are on the Horizon?, in: Proc. of the American Control Conference (ACC), 2022, pp. 342–357.

[15] B. K. M. Powell, D. Machalek, T. Quah, Real-time optimization using reinforcement learning, Computers and Chemical Engineering 143 (2020) 107077. doi:10.1016/j.compchemeng.2020.107077.
URL https://doi.org/10.1016/j.compchemeng.2020.107077

[16] D. Machalek, T. Quah, K. M. Powell, A Novel Implicit Hybrid Machine Learning Model and Its Application for Reinforcement Learning, Computers & Chemical Engineering 155 (2021) 107496. doi:10.1016/j.compchemeng.2021.107496.
URL https://doi.org/10.1016/j.compchemeng.2021.107496

[17] H. Yoo, H. E. Byun, D. Han, J. H. Lee, Annual Reviews in Control Reinforcement learning for batch process control : Review and perspectives, Annual Reviews in Control 52 (July) (2021). doi:10.1016/j.arcontrol.

2021.10.006.
URL https://doi.org/10.1016/j.arcontrol.2021.10.006

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, Advances in Neural Information Processing Systems 2017-Dec (Nips) (2017) 5999–6009.

[19] D. Bahdanau, K. H. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings (2015) 1–15.

[20] K. Cho, M. B. Van, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation, Journal of Clinical Microbiology 28 (4) (2014) 828–829. doi:10.1128/jcm.28.4.828-829.1990.

[21] S. Hochreiter, Long Short-Term Memory, Neural computation 1780 (1997) 1735–1780.

[22] H. Hassanpour, B. Corbett, P. Mhaskar, Integrating dynamic neural network models with principal component analysis for adaptive model predictive control, Chemical Engineering Research and Design 161 (2020) 26–37. doi:https://doi.org/10.1016/j.cherd.2020.03.031.

[23] H. Hassanpour, B. Corbett, P. Mhaskar, Artificial neural network-based model predictive control using correlated data, Industrial & Engineering Chemistry Research 61 (8) (2022) 3075–3090.

[24] J. Park, C. Price, D. Pixton, M. Aghito, R. Nybø, K. Bjørkevoll, J. D. Hedengren, Model predictive control and estimation of managed pressure drilling using a real-time high fidelity flow model, ISA Transactions 105 (2020) 256–268. doi:10.1016/j.isatra.2020.05.035.
URL https://doi.org/10.1016/j.isatra.2020.05.035

[25] C. W. Jr, Oxygen Engiched Atmosphere Roasting, Ph.D. thesis, Montana School of Mines (1948).

[26] J. C. Smith, T. H. McCord, G. R. O'Neil, Treating refractory gold ores via oxygen-enriched roasting (1990).

[27] K. Thomas, A. Cole, Roasting Developments – Especially Oxygenated Roasting, Ken Thomas & Murray Pearson, 2016. doi:10.1016/b978-0-444-63658-4.00023-2.
URL http://dx.doi.org/10.1016/B978-0-444-63658-4.00023-2

[28] H. Qin, X. Guo, Q. Tian, D. Yu, L. Zhang, Recovery of gold from sulfide refractory gold ore: Oxidation roasting pretreatment and gold extraction, Minerals Engineering 164 (January) (2021). doi:10.1016/j.mineng.2021.106822.

[29] D. Kunii, O. Levenspiel, Fluidization engineering, Butterworth-Heinemann, 1991.

[30] H. Ahn, S. Choi, A comparison of the shrinking core model and the grain model for the iron ore pellet indurator simulation, Computers and Chemical Engineering 97 (2017) 13–26. doi:10.1016/j.compchemeng.2016.11.005.

[31] J. Park, R. A. Martin, J. D. Kelly, J. D. Hedengren, Benchmark temperature microcontroller for process dynamics and control, Computers and Chemical Engineering 135 (4 2020). doi:10.1016/j.compchemeng.2020.106736.