International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Deep Learning Based-Model Observer For Prediction Of Nonlinear With Time-Delay System

Santo Wijaya[1], Muhammad Zarlis [2*], Ford Lumban Gaol [3], Antoni Wibowo[4]

Submitted: 18/10/2023 **Revised:** 10/12/2023 **Accepted:** 15/12/2023

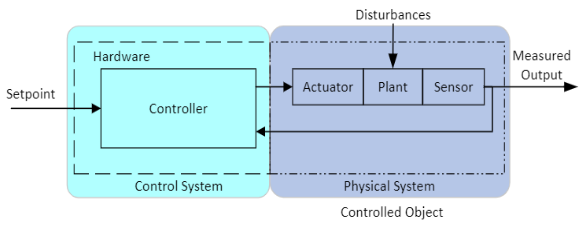
Abstract: Nonlinear time-delay systems, such as cloud-based control systems (CCS), have wide implementations, from robotics to multi-agent systems, but data-driven research on this complex system is still in its infancy. This study presents a model observer designed for the prediction of nonlinear time-delay systems utilizing Deep Learning (DL) methods. The time-delay Markov decision process is considered in the augmented state as an input feature for the model observer. Furthermore, additional input features include the dynamic rate of change to provide temporal nonlinearity and time delay for better prediction. The time-series prediction analysis is conducted on a dataset from an arbitrary nonlinear mass spring damper system induced by a time delay in the system input and output. This study thoroughly evaluates the performance of three DL networks, including the Feed Forward Neural Network (FFNN), Long Short Term Memory (LSTM), and Radial Basis Function Neural Network (RBFNN). While, the model prediction performance is evaluated with Mean Absolute Percentage Error (MAPE) and Root Mean Square Error (RMSE) performance metrics. Results show that the FFNN architecture of two hidden layers with 15 nodes in each layer and the LeakyReLU activation function achieves the best performance, outperforming other network layers with an average MAPE value of 8.4% and an average RMSE value of 0.0006038. The N-step ahead prediction performance of the model observer with the proposed features in this study serves as an important fundamental model for the development of control methods based on a data-driven approach for CCS.

Keywords: *Deep learning, Model observer, Nonlinear time delay system, N-step ahead prediction, Time series prediction.*

.

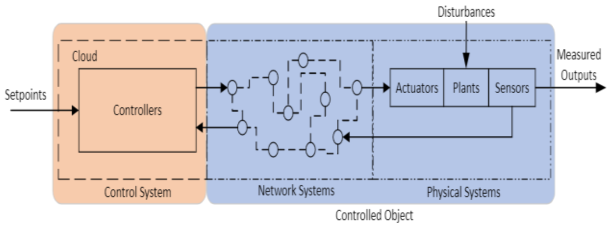
1. Introduction

In Industrial Revolution 4.0 and the Industrial Internet of Things (IIoT), the demand for networked-based control systems with scalability and interoperability according to the Cyber-Physical System (CPS) principle is growing rapidly [1]. Nevertheless, implementing CPS-based control in traditional control systems presents difficulties due to the fact that the hardware controllers and control systems, such as proportional integral derivative (PID) control, were not originally intended for such applications. The challenges involved in creating efficient control systems in this context encompass user friendliness, both local and global networking capabilities, interoperability between control systems, high-quality control methodologies, and the coordination of diverse controllers [2]. To overcome this challenge, Cloud-based Control Systems (CCS) have been developed [3]. CCS enables the replacement of hardware controllers as a solution for CPS-based industrial control systems [4], [5]. However, the migration from hardware controllers to CCS controllers poses issues in the context of the control system itself. In hardware control systems, the controlled entity only considers the process of the system; hence, the system's dynamics represent the controlled process. Conversely, in CCS, the controlled entity comprises both the controlled system and the interconnected internet network. As a result, there are three crucial attributes in CCS: (1) process nonlinearity; (2) communication of data with time delays [6-8]. Fig. 1. illustrates the difference in controlled entities between traditional/ hardware control systems and CCS control systems.



(a) Control system and controlled object in classic framework

[1] *Computer Science Department BINUS Graduate Program – Doctor of Computer Science, Bina Nusantara University, Jakarta 11480, Indonesia*
ORCID ID: 0000-0001-8653-7279
[2]*Information System Management Department BINUS Graduate Program – Master of Information System Management, Bina Nusantara University, Jakarta 11480, Indonesia*
ORCID ID: 0000-0003-0520-7273
[3]*Computer Science Department BINUS Graduate Program – Doctor of Computer Science, Bina Nusantara University, Jakarta 11480, Indonesia*
[4]*Computer Science Department BINUS Graduate Program – Master of Computer Science, Bina Nusantara University, Jakarta 11480, Indonesia*
ORCID ID: 0000-0002-5116-5708
** Corresponding Author Email: muhammad.zarlis@binus.edu*



(b) Control system and controlled object in CCS framework

Fig. 1. Illustration of controller and controlled object comparison.

Based on the outcomes of prior research [9], it was observed that classical control methods combined with a model-based control system approach have been employed in the design of CCS. In this approach, a model observer for the control system is developed using white box modeling. However, it's evident that the integration of internet networks with time delays and the nonlinearity of the physical system transform the control object of CCS into a complex system [10]. Complex systems are more effectively modeled using data driven-based modeling approaches [11]. This approach where decisions are made based on data, and the behavior of the system is controlled. This strategy is often used in various complex system, including weather patterns [12], traffic regulation systems [13], or education [14].

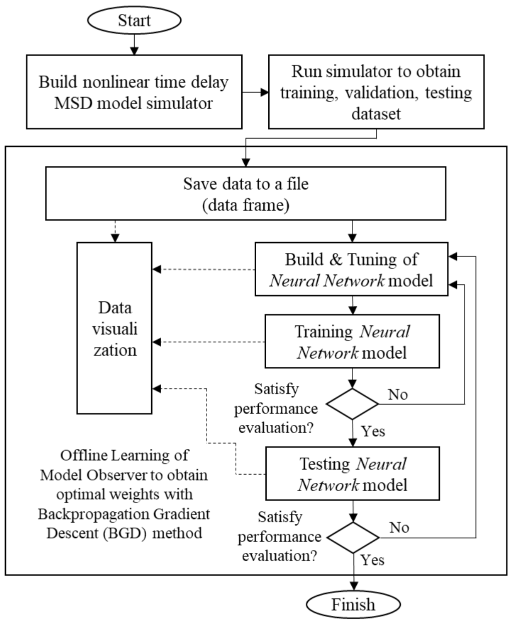
On the other hand, data-driven modeling research, such as neural network-based control for nonlinear time delay system utilizing adaptive backstepping or adaptive dynamic surface control methods, might suffer from the curse of dimensionality [15]. The Reinforcement Learning (RL) method has been studied in depth as a solution to the problem, and current research proposes Model-based RL (Mb-RL) for nonlinear time delay systems [16],17]. However, model architecture and Input-Output (I/O) features were not clearly defined. In our previous research [18], we clearly defined the I/O features with additional rate of change input and model architecture for the model that is used as an observer in the proposed Mb-RL scheme. The method is able to control a highly nonlinear Mass Spring Damper (MSD) system, but time delay was not considered in the system.In this work, we consider nonlinearity and time delay attributes in the MSD system. Time-delay phenomena are commonly encountered in real-world scenarios, such as systems involving network-based control. These time delays within nonlinear systems have been recognized as a factor contributing to the instability of control systems, as noted in previous studies [19]. The main objective of the paper is to derive model observer that is able to predict the nonlinear time delay system. We further explore the design of model observer in the following aspects: 1) formally define the model observer parameterization with input-output features to cope with nonlinear and time delay attributes, 2) thorough performance evaluation of different deep learning approaches: Feed Forward Neural Network (FFNN), Long-Short Term Memory (LSTM), and Radial Basis Function Neural Network (RBFNN) to find the best model observer architecture, 3) prediction evaluation of the model observer. The work in this paper is important as a solid foundation for Mb-RL scheme research in the future.

2. Method

2.1. Overall framework

Research framework of the paper is as shown in Fig. 2. To simulate the dynamics of an output system based on input supplied to the system, a model for the simulator is first created. To model the system, a nonlinear time-delay MSD model is built using differential equations that can each be solved numerically. To make training, validation, and testing datasets, a random seed controller is used to excite the simulator model. Data frames are used to store these datasets.

The next step is to perform data preprocessing, which includes normalization. The development of input-output features for the model observer and the construction of various neural network architectures with various hyperparameter settings. Using the Backpropagation Gradient Descent (BGD) method, the objective is to identify the best fit during the training and validation phases. The testing stage starts if the model produced by the selected hyperparameters satisfies the regression best-fit criteria. Predictions evaluations are conducted during the testing phase to choose the best model observer.



Start
Build nonlinear time delay MSD model simulator
Run simulator to obtain training, validation, testing dataset
Save data to a file (data frame)
Data visualization
Build & Tuning of Neural Network model
Training Neural Network model
Satisfy performance evaluation?
No
Yes
Testing Neural Network model
Offline Learning of Model Observer to obtain optimal weights with Backpropagation Gradient Descent (BGD) method
Satisfy performance evaluation?
No
Yes
Finish

Fig. 2. Research framework

2.2. System Modeling

In the paper, the MSD system is used to represent the nonlinearity attribute of the CCS. The nonlinear MSD system is an oscillatory system composed of mass springs and damping elements that store kinetic and potential energy. This model is widely used in various modeling approaches to describe system dynamics, such as vehicle-bridge interaction [20], microscopic traffic model [21]. Specifically, it is extensively studied in control theory. An arbitrary generic free body diagram of the system, complying with Newton's second law of motion, is depicted in Fig. 3(a).

In this context, it is assumed that the nonlinear MSD system comprises three masses, denoted as $m_i = 0.5\ kg$ where $i=\{1,2,3\}$, with the system's output being the positions of these masses, y_i (m). Notably, there is no measurement data available for the position of mass y_2. The system input is forces applied to masses u_j (N), where $j=\{1,3\}$ is the I/O indexes. The system parameters include damping dampers $d_i = 0.5\ Ns/m$ and nonlinear springs characterized by the function $k_i = k_L\, x + k_{NL}\, x^3$. The linear inertia has an uncertainty factor of $k_L = 200 \pm 5\ N/m$, while the nonlinear inertia is represented as $k_{NL} = 65\ N/m^3$. Additionally, there is a disturbance in the third inertia $||\ dist_3\ || \leq 200\ N$. The differential equations of the nonlinear MSD system, formulated according to Newton's laws, are provided in Eq. (1), (2) and (3).

$$m\ddot{y}_{1_t} = k_L\left(-2y_{1_t} + y_{2_t}\right) + k_{NL}\left(-{y_{1_t}}^3 + \left(y_{2_t} - y_{1_t}\right)^3\right) + d\left(\dot{y}_{2_t} - 2\dot{y}_{1_t}\right) + u_{1_t} \tag{1}$$

$$m\ddot{y}_{2_t} = k_L\left(y_{1_t} - 2y_{2_t} + y_{3_t}\right) + k_{NL}\left(\left(y_{3_t} - y_{2_t}\right)^3 - \left(y_{2_t} - y_{1_t}\right)^3\right) + d\left(\dot{y}_{1_t} - 2\dot{y}_{2_t} + \dot{y}_{3_t}\right) \tag{2}$$

$$m\ddot{y}_{3_t} = k_L\left(y_{2_t} - y_{3_t}\right) + k_{NL}\left(y_{2_t} - y_{3_t}\right)^3 + d\left(\dot{y}_{2_t} - \dot{y}_{3_t}\right) + u_{3_t} - dist_{3_t} \tag{3}$$

Time delay attribute representation and its effect on system I/O are illustrated in Fig. 3(b), where τ_u denotes the system input time delay and τ_s denotes the system output time delay. The input time delay affected the output of the system ($y_t|u_{(t-\tau u)}$) in such a way that the output is y_t given the input u_t with delay τ_u. The output time delay affected the output of the system ($y_{(t-\tau s)}|u_{(t-\tau u)}$) in such a way that the output when it arrived at the controller ($x_t|y_{(t-\tau s)}$) is x_t given the output y_t with delay τ_s. In the case of CCS, the output delay is a discrete delay and does not affect the physical system dynamics. Then, we rearranged the differential equation with time delay induced system I/O as provided in Eq. (4), (5), (6), (7), (8). In this work, we assume that $\tau_u = \tau_s = 1$ *timestep*.

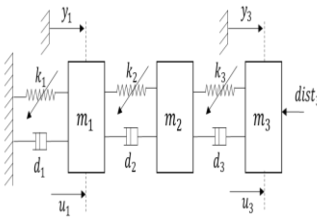
$$\dot{v}_{1_t} = \frac{1}{m}\left(k_L\left(-2y_{1_t} + y_{2_t}\right) + k_{NL}\left(-{y_{1_t}}^3 + \left(y_{2_t} - y_{1_t}\right)^3\right) + d\left(v_{2_t} - 2v_{1_t}\right) + u_{1(t-\tau_u)}\right) \tag{4}$$

$$\dot{v}_{2_t} = \frac{1}{m}\left(k_L\left(y_{1_t} - 2y_{2_t} + y_{3_t}\right) + k_{NL}\left(\left(y_{3_t} - y_{2_t}\right)^3 - \left(y_{2_t} - y_{1_t}\right)^3\right) + d\left(v_{1_t} - 2v_{2_t} +\right.\right. \tag{5}$$

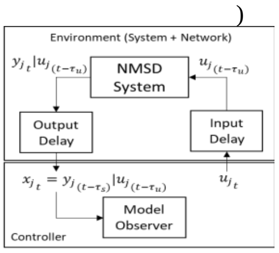
$$\dot{v}_{3_t} = \frac{1}{m}\left(k_L\left(y_{2_t} - y_{3_t}\right) + k_{NL}\left(y_{2_t} - y_{3_t}\right)^3 + d\left(v_{2_t} - v_{3_t}\right) + u_{3(t-\tau_u)} - dist_{3_t}\right) \tag{6}$$

$$\dot{y}_{1_t} = v_{1_t};\ \dot{y}_{2_t} = v_{2_t};\ \dot{y}_{3_t} = v_{3_t} \tag{7}$$

$$x_{j_t} = y_{j_{(t-\tau_s)}} | u_{j_{(t-\tau_u)}} \tag{8}$$



(a) Free-body diagram of the MSD model



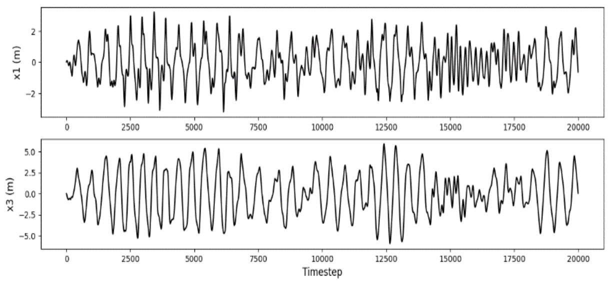
Environment (System + Network)
$y_{j_t}|u_{j_{(t-\tau_u)}}$
NMSD System
$u_{j_{(t-\tau_u)}}$
Output Delay
Input Delay
$x_{j_t} = y_{j_{(t-\tau_s)}}|u_{j_{(t-\tau_u)}}$
u_{j_t}
Model Observer
Controller

(b) Illustration of time delay induced system I/O

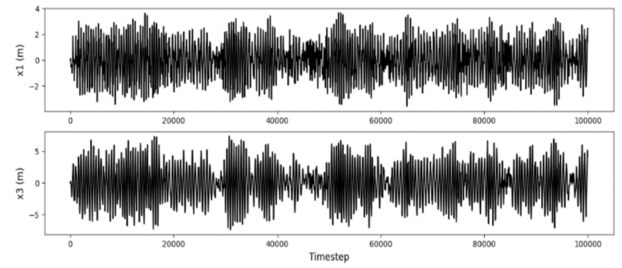
Fig. 3. System modeling of the MSD system and illutstration of time delay induced I/O

2.3. Data Collection

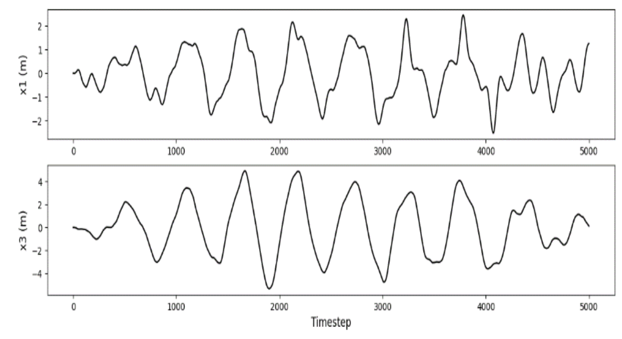
The equations are dynamically simulated using the SciPy library with a timestep $\Delta k = 0.001$ *seconds*, an absolute error of 1.0×10^{-8}, and a relative error of 1.0×10^{-6} as a simulator of the nonlinear time delay MSD model for data collection. The initial conditions of the system are steady-state with zero movement and $x_j = 0$. The training dataset is collected for *20.000 timesteps/ datapoints* with a random seed (Fibonacci no. 8) as input. The validation dataset is collected for *100.000 timesteps* with a random seed (Fibonacci no. 10) as input. The testing dataset is collected for *5.000 timesteps* with a random seed (Fibonacci no. 98) as input. Fig**.** 4 shows the dataset visualization obtained from the simulator. It is notable that the system is highly oscillated and unstable, which makes prediction difficult.



(a) Training dataset



(b) Validation dataset



(c) Testing dataset

Fig. 4. Dataset visualization obtained from the MSD system simulator**.**

2.4. Data Preprocessing

The paper employed Z-score normalization [22] to preprocess the input dataset, aiming to expedite the training of the LSTM model by standardizing it with a mean of 0 and a standard deviation of 1. This standardization process is represented as $x' = (x - \mu) / \sigma$, where x' represents the normalized data, x stands for the source data, μ is the dataset's mean, and σ represents the dataset's standard deviation. Z-score normalization was chosen because the dataset from the nonlinear time delay MSD system do not exhibit extreme outliers in its feature distribution.

2.5. Model Architecture

The goal of this study is to make a model observer that can predict nonlinearity and time delay in systems like the MSD system that are meant for network-based control. This work explores two important parts of the data-driven model observer, which are the input-output features and the neural network architecture of the model. The design of input-output features of the model considers a Time Delay Markov Decision Process (TD-MDP), and the development of this model observer will be part of the Mb-RL research in the future. The TD-MDP formalism with regards to the environment and time delay in system input-output, as illustrated in Fig. 3(b) is stated in Definition 1 [23].

Definition 1. TD-MDP(E, τ_u , τ_s), E denotes the environment of the reinforcement learning agent/ controller, τ_u denotes the system input time delay, and τ_s denotes the system output time delay, represents a nonlinearity, and time delay of CCS can be parameterized with a 6-tuple $\langle X, U, \tilde{\mu}, \tilde{P}, \tau_s, \tau_u \rangle$ is an augmented version of standard MDP with a 4-tuple $\langle Y, U, \mu, P \rangle$:

- Augmented state-spaces: X = Y × U $^{\tau s+\tau u}$, where (τ_s, τ_u) ∈ N. (9)
- Action spaces: U = U (10)
- Initial state distribution:

$$\tilde{\mu}(x_0) = \tilde{\mu}\left(y_{0-\tau_s}, u_{0-1}, \cdots, u_{0-\tau_s-\tau_u}\right) = \mu\left(y_{0-\tau_s}\right) \prod_{i=1}^{\tau_u+\tau_s} \delta\left(u_{0-i} - c_{0-i}^{u}\right)$$

(11)

where $(c_{0-i}^{u})_{i=1:\tau_s+\tau_u}$ *is initial action sequence, and δ is the Dirac delta function.*

$$\tilde{P}(x_{t+1}|x_t, \boldsymbol{u}_t) = \tilde{P}\left(y_{t+1-\tau_s}, u_t^{(t+1)}, \cdots, u_{t-\tau_u-\tau_s}^{(t+1)} | y_{t-\tau_s}, u_{t-1}^{(t)}, \cdots, u_{t-\tau_u-\tau_s-1}^{(t)}, \boldsymbol{u}_t\right)$$

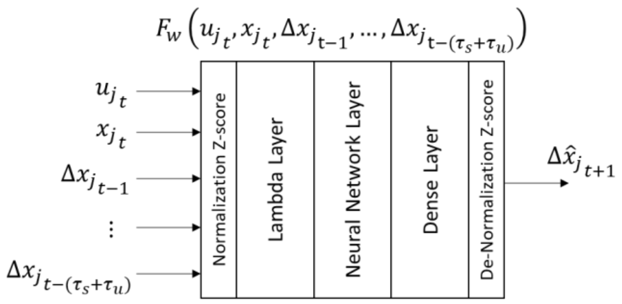
- *State transition distribution*: (12)

where y ∈ Y is the delayed system output, u ∈ U $^{\tau s+\tau u}$ *is a buffer of system input from the last* (τ_u + τ_s) *timestep delay. The reward function is excluded from the formalism due to the fact that the reward/ cost function parameters value can be provided by the model observer and not by the environment.*

With the TD-MDP frame, at each timestep, the model observer received the most recent available delayed system output, augmented with the buffer of system input from the last (τ_u + τ_s) timestep delay. Hence, we reformulate the system output received by the model observer from Eq. (8) into Eq. (9). The augmented state space provides important information about system outputs and the sequence of system inputs applied to the system as a result of past experience for the model observer. Based on the augmented state space formulation, we propose the input-output features and architecture layer of the model observer, as shown in **Error! Reference source not found.**.

$$x_{j\,t} = \left[y_{j\,(t-\tau_s)} \;\; u_{j\,(t-(\tau_s+\tau_u))} \;\; u_{j\,(t-(\tau_s+\tau_u)+1)} \;\; \cdots \;\; u_{j\,(t} \;\;)\right. \quad (9$$

The parameterization of the dynamic model observer function is defined as F_w (uj_t, xj_t, Δxj_{t-1},..., $\Delta xj_{t-(\tau s+\tau u)}$), where *w* is the neural network weights, and there are 4 inputs to the function with 1 output. uj_t is the control data/ input from the controller, xj_t is the system output received from the environment, Δxj_{t-1} is the rate of change of the system output from previous 1 timestep and formulate as $\Delta xj_{t-1} = xj_t - xj_{t-1}$, $\Delta xj_{t-(\tau s+\tau u)}$ is the rate of change of the system output from previous (τ_u + τ_s) timestep and formulate as $\Delta xj_{t-(\tau s+\tau u)} = xj_t - xj_{t-(\tau s+\tau u)}$. The use of dynamics rate of change to provide better prediction results for small time steps has been shown in previous work by Nagabandi [24]. We utilize the dynamics rate of change of the system output from the previous *1, ..., (*τ_u *+* τ_s*)* time step to provide temporal context of the nonlinearity and time delay characteristics for better model-observer prediction. Since in this work, we assume that $\tau_u = \tau_s =$ *1 timestep*, then there are totally 20 input features to the model, and 6 output features from the model.



$F_w\left(u_{j_t}, x_{j_t}, \Delta x_{j_{t-1}}, \ldots, \Delta x_{j_{t-(\tau_s+\tau_u)}}\right)$
u_{j_t}
x_{j_t}
$\Delta x_{j_{t-1}}$
⋮
$\Delta x_{j_{t-(\tau_s+\tau_u)}}$
Normalization Z-score
Lambda Layer
Neural Network Layer
Dense Layer
De-Normalization Z-score
$\Delta \hat{x}_{j_{t+1}}$

Fig. 5. Model observer input-output features and layer architecture

The model observer architecture consists of three layers: the lambda layer, the network layer, and the dense layer. There are three types of network layers that will be considered in the paper: FFNN, LSTM, and RBFNN. The FFNN or Multi Layer Perceptron (MLP) has been widely employed as a fundamental deep learning model. It has demonstrated favorable outcomes in capturing intricate relationships among variables and making good predictions even in the presence of noisy data [25], [26]. The LSTM model is a variant of the Recurrent Neural Network (RNN) architecture that is frequently employed in the domain of time-series forecasting. An advantageous characteristic of LSTM models in the context of time-series forecasting lies in their capability to effectively capture and model long-term dependencies within the data. This attribute renders LSTMs particularly valuable for accurately predicting trends and cycles [27]. The RBFNN model is a type of neural network extensively considered in neural controller design from the control theory community [28]. RBFNN uses a radial basis kernel as an activation function with excellent function approximation with fewer neurons and robustness to outliers.

To obtain the best-fit model observer for prediction of nonlinear and time-delay systems, comparative analyses of the neural network types are conducted. Hyperparameter tuning, employing random search, is employed to establish the optimal parameter values for various existing methods [29]. The parameter configurations for all methods are detailed in **Table 1**.

Table 1. Hyperparameter Settings

Hyperparameter	***Options***
Optimizer	Adam (Learning Rate = 0.0003)
Loss function	Mean Squared Error (MSE)
Training epochs / batch size	100 / 32
Activation function (all layers)	FFNN, LSTM - Sigmoid, Tanh, Leaky ReLU (0.5) RBFNN - Radial Basis Function
Number of nodes (each layer)	5, 10, 15, 25, 50, 75
Number of hidden layers	1, 2, 3

Source: Prepared by the author, (2023).

2.6. Evaluation

The evaluation process for a model observer is split into two stages: training and testing. During training, the model's performance is assessed using the Mean Squared Error (MSE) as a key metric, with a lower MSE indicating better accuracy in predicting data points, as seen in Eq. (10). In the testing stage, two performance metrics, Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE), are used, as seen in Eq. (11) and (12), respectively. RMSE helps identify outliers and measures data processing efficiency as it approaches zero. MAPE assesses model accuracy, with different categories like $MAPE < 10\%$, $10\% \leq MAPE < 20\%$, $20\% \leq MAPE < 50\%$, and $MAPE \geq 50\%$ indicating the prediction ability is very good, good, feasible, and poor, respectively. Here, N denotes the total dataset, t denotes the timestep, x denotes the groundtruth state, $\hat{x}$ denotes the predicted state.

$$MSE = \frac{1}{N}\sum_{t=1}^{N}(x_t - \hat{x}_t)^2 \qquad (10)$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(x_t - \hat{x}_t)^2} \qquad (11)$$

$$MAPE = \frac{1}{N}\sum_{t=1}^{N}\frac{|x_t - \hat{x}_t|}{x_t} \times 100\% \qquad (12)$$

3. Results and Discussion

The training dataset is standardized before being fed to the neural network layer. Both the mean and standard deviation are sampled from 20.000 data points. **Fig. 5** shows that all 20 input features have been normalized with a mean of 0 and a standard deviation of 1. The process of balancing the input states on a common scale facilitates efficient training of the neural network.

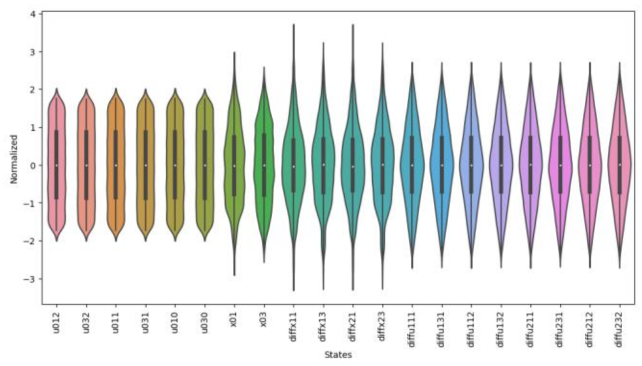
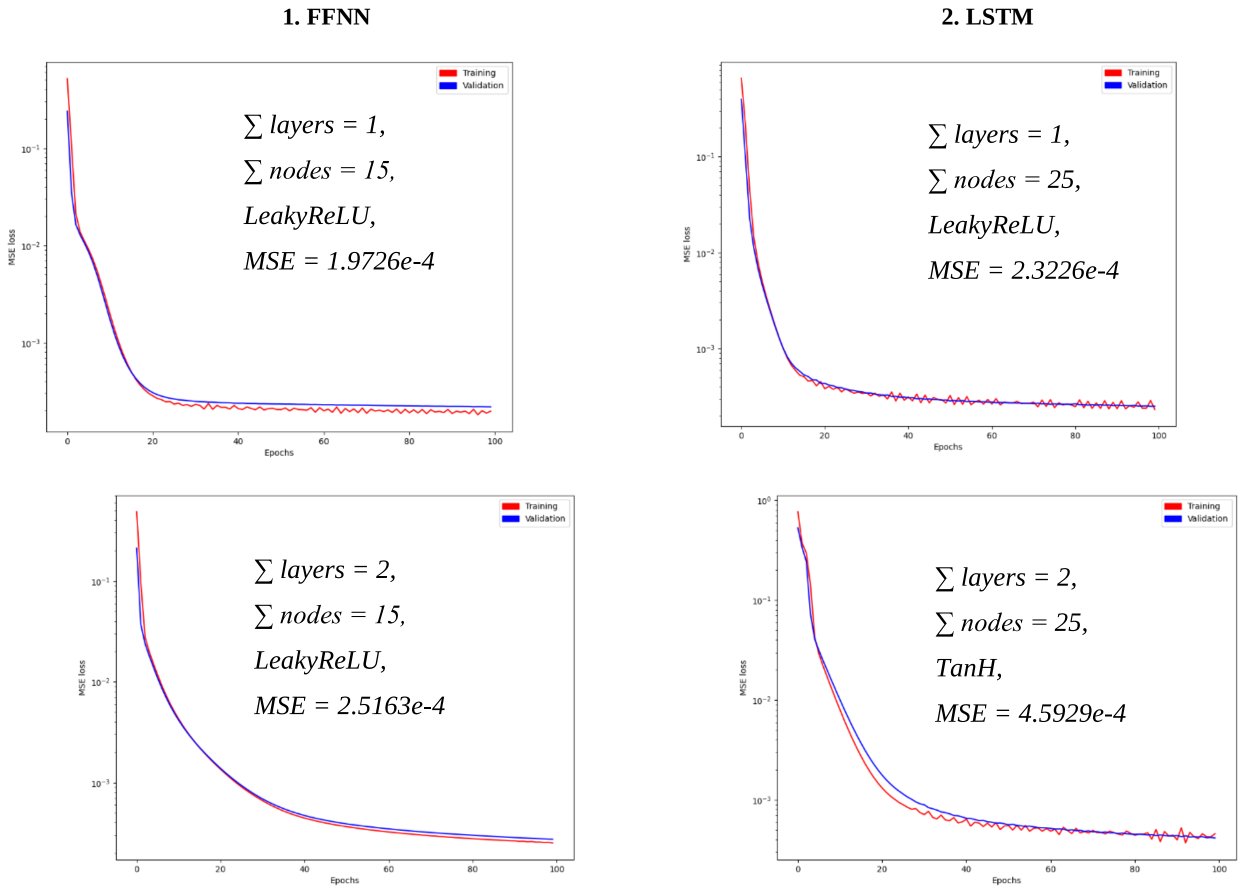
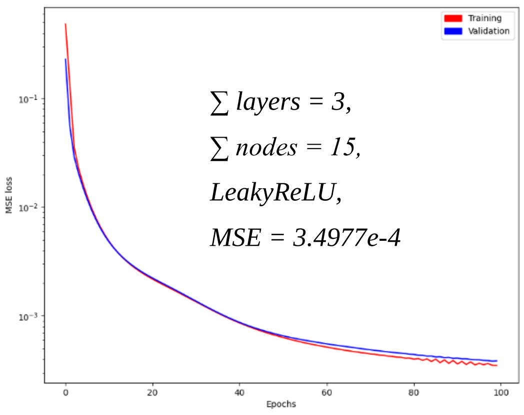


Fig. 5. Normalization result using Z-score method

Overall, the RBFNN architecture cannot converge to a lower MSE, and there was a wide gap in the MSE value between training and validation, given the hyperparameter setting values. The best performance result of training and validation given the hyperparameter settings in each layer is shown in **Fig. 6**. The selection criteria for the best result are not only based on a lower MSE but also on the rate of convergence and stability in training epochs. It is notable that the best-fit FFNN architecture has two hidden layers and 15 nodes for each layer with the LeakyReLU activation function. On the other hand, the best-fit LSTM architecture is one hidden layer and 25 nodes with the LeakyReLU activation function. In summary, considering the training result exhibits small fluctuations, the most stable FFNN architecture with two hidden layers and 15 nodes is selected to test model prediction performance.





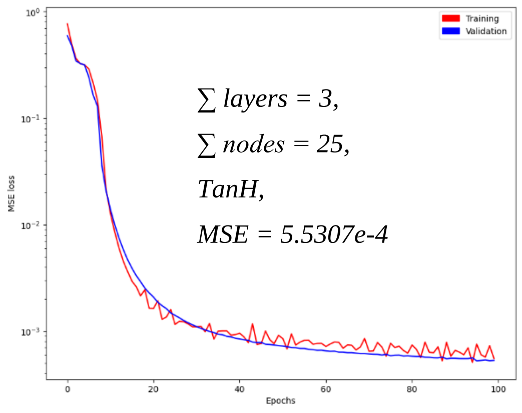
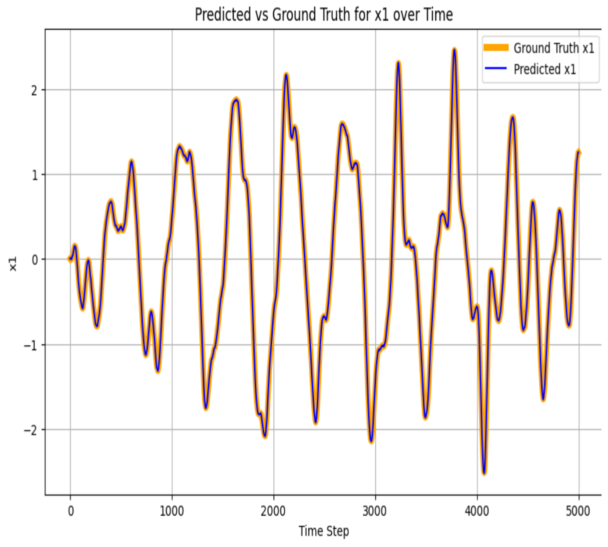


Fig. 6. Training and validation results

The prediction output for one step ahead of the model observer compared to the testing dataset from timestep 1 to 5000 is shown in **Fig. 7**. It is notable that the model prediction output follows ground truth for x_1 and x_3. The FFNN architecture achieved prediction performance for x_1 with a RMSE of 0.0003476 and a MAPE of 8.004%, and prediction performance for x_3 with a RMSE of 0.0008600 and a MAPE of 8.981%. Thus, according to the MAPE performance category, the model's prediction ability is very good.



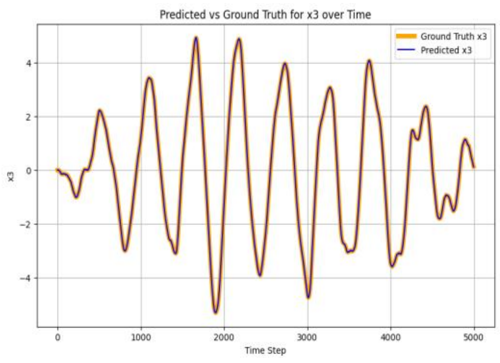


Fig. 7. One-step ahead prediction result compared to the testing dataset.

Additionally, we propagate (N_p - step ahead prediction) the learned dynamics forward in order to make multistep predictions, where N_p is the length of the propagation horizon. This is an important feature to be used for planning the optimal control signal for system input in the Mb-RL framework [30], [31]. The N_p-step ahead prediction of the model is shown in **Fig**. 8. In this case, Np = 5 is chosen as an example for propagation. It is notable that with Np = 5, there are visible errors. Further exploration of the optimal hyperparameter selection using heuristic algorithms, such as Particle Swarm Optimization (PSO) [32], is needed to minimize the propagation error and be able to do longer propagation. However, the model can predict the trend of the dynamic trajectories. Therefore, we can obtain knowledge on how long the prediction horizon can be used for control signal optimization planning in the Mb-RL framework. The Mb-RL uses temporal dynamics learning, in which propagation is done over and over again at each timestep to update the calculated control signal and make up for the model observer's long propagation error.

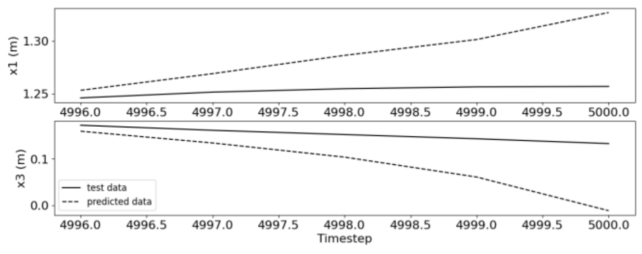


Fig. 8. 5-step ahead prediction result compared to the testing dataset.

4. Conclusion

Nonlinear time-delay systems, such as cloud-based control systems (CCS), have widespread applications, including robotics and multi-agent systems, but data-driven research that provides an advantage in this complex system is in its infancy. Using Deep Learning (DL) techniques, this study presents a model observer for the prediction of nonlinear time-delay systems. In the context of the model observer, the augmented state incorporates the time-delay Markov decision process as one of its input features. Additionally, the model also takes into account other input features, including the dynamic rate of change. These additional features are included to introduce temporal nonlinearity and account for time delays, ultimately improving prediction accuracy. Comparative study of DL networks, including FFNN, LSTM, RBFNN provide evaluation of predictive performance of the model observer. In case of the nonlinear time delay mass spring damper system, the FFNN performed better thanother DL networks and the model observer also shown to be robust under different time delay environment, but it has small fluctuations in the predicted data. The study focused on the design of input-output features and DL networks of the model observer using arbitrary system. Future research should include real-world system modeled into the simulator, and utilization of Extended Kalman Filter as a conjunction with a trained model to post-process the model's predictions to improve their smoothness, and also to estimate state that is unmeasurable, in this case state y_2.

Acknowledgment

This research is funded by the DRTPM PDD and Bina Nusantara University with Grant Number [149/VR.RTT/VII/2023].

Author contribution

Conceptualization: Santo Wijaya, Muhammad Zarlis; Methodology: Santo Wijaya, Muhammad Zarlis; Literature Search and Data Analysis: Santo Wijaya; Writing: Santo Wijaya; Review and Editing: Santo Wijaya, Muhammad Zarlis, Ford Lumban Gaol, Antoni Wibowo; Supervision: Muhammad Zarlis.

Funding statement

This research is funded by the DRTPM PDD and Bina Nusantara University with Grant Number [149/VR.RTT/VII/2023].

Conflict of interest

The authors declare that they have no known competing financial interests or personal relationships that could be perceived as having influenced the work described in this paper.

Additional information

No additional information is available for this paper.

Data and Software Availability Statements

All of the material is owned by the authors, and/or no permissions are required. The authors will provide the data and software repository link in case it is requested by the reviewers.

References

[1] P. Zheng *et al.*, “Smart manufacturing systems for Industry 4.0: Conceptual framework, scenarios, and future perspectives,” *Frontiers of Mechanical Engineering*, vol. 13, no. 2, pp. 137–150, 2018, doi: 10.1007/s11465-018-0499-5.

[2] R. Langmann and M. Stiller, “The PLC as a smart service in industry 4.0 production systems,” *Applied Sciences*, vol. 9, no. 18, p. 3815, Sep. 2019, doi: 10.3390/app9183815.

[3] Y. Xia, “Cloud control systems,” *IEEE/CAA Journal of Automatica Sinica*, vol. 2, no. 2, pp. 134–142, Apr. 2015, doi: 10.1109/jas.2015.7081652.

[4] P. Papcun, E. Kajati, C. Liu, R. Y. Zhong, J. Koziorek, and I. Zolotova, “Cloud-based control of industrial cyber-physical systems,” *Proceedings of International Conference on Computers and Industrial Engineering, CIE*, pp. 1–14, 2018.

[5] P. Skarin and K.-E. Årzén, “Explicit MPC recovery for cloud control systems,” *2021 60th IEEE Conference on Decision and Control (CDC)*, pp. 5394–5401, Dec. 2021, doi: 10.1109/cdc45484.2021.9683307.

[6] M. S. Darup and T. Jager, “Encrypted Cloud-based Control using Secret Sharing with One-time Pads,” *2019 IEEE 58th Conference on Decision and Control (CDC)*, pp. 7215–7221, Dec. 2019, doi: 10.1109/cdc40024.2019.9029342.

[7] J. Zhang, Y. Xia, Z. Sun, and D. Chen, “Event-Triggered Disturbance Rejection control for CPSs,” in *CRC Press eBooks*, 2021, pp. 153–180. doi: 10.1201/9781003260882-12.

[8] H. Yang, S. Ju, Y. Xia, and J. Zhang, “Predictive cloud control for networked multiagent systems with quantized signals under DOS attacks,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 51, no. 2, pp. 1345–1353, Feb. 2021, doi: 10.1109/tsmc.2019.2896087.

[9] S. Wijaya, A. Ramadhan, and A. Andhika, “Cloud-based control systems: a systematic literature review,” *International Journal of Reconfigurable & Embedded Systems (IJRES)*, vol. 12, no. 1, p. 135, Mar. 2023, doi: 10.11591/ijres.v12.i1.pp135-148.

[10] A. Dong, Z. Du, and Z. Yan, “Round trip time prediction using recurrent neural networks with minimal gated unit,” *IEEE Communications Letters*, vol. 23, no. 4, pp. 584–587, Apr. 2019, doi: 10.1109/lcomm.2019.2899603.

[11] W. Hao and Y. Han, “Data Driven Control with Learned Dynamics: Model-Based versus Model-Free Approach.,” *arXiv (Cornell University)*, Jun. 2020, [Online]. doi: 10.48550/ARXIV.2006.09543

[12] N. Verba, K.-M. Chao, J. Lewandowski, N. Shah, A. James, and F. Tian, “Modeling industry 4.0 based fog computing environments for application analysis and deployment,” *Future Generation Computer Systems*, vol. 91, pp. 48–60, Feb. 2019, doi: 10.1016/j.future.2018.08.043.

[13] N. N. Hasibuan, M. Zarlis, and S. Efendi, “Detection and tracking different type of cars with YOLO model combination and deep sort algorithm based on computer vision of traffic controlling,” *Jurnal Dan Penelitian Teknik Informatika*, vol. 6, no. 1, pp. 210–221, 2021, doi: 10.33395/sinkron.v6i1.11231.

[14] F. Izhari, M. Zarlis, and S. Sutarman, “Analysis of backpropagation neural neural network algorithm on student ability based cognitive aspects,” *IOP Conference Series*, vol. 725, no. 1, p. 012103, 2020, doi: 10.1088/1757-899x/725/1/012103.

[15] S. F. Wijaya, F. L. Gaol, A. Wibowo, and L. Wang, “Neural controller design for a class of nonlinear systems with time-varying delay: A bibliometric analysis-based short literature review,” *2023 International Conference on Computer Science, Information Technology and Engineering (ICCoSITE)*, Feb. 2023, doi: 10.1109/iccosite57641.2023.10127776.

[16] S. Ramstedt, “Real-Time Reinforcement learning,” *arXiv.org*, Nov. 11, 2019. http://arxiv.org/abs/1911.04448

[17] B. Chen, M. Xu, L. Zhang, and D. Zhao, “Delay-aware model-based reinforcement learning for continuous control,” *Neurocomputing*, vol. 450, pp. 119–128, Aug. 2021, doi: 10.1016/j.neucom.2021.04.015.

[18] S. F. Wijaya, Y. Heryadi, Y. Arifin, W. Suparta, and L. Lukas, “Long short-term memory (LSTM) model-based reinforcement learning for nonlinear mass spring damper system control,” *Procedia Computer Science*, vol. 216, pp. 213–220, Jan. 2023, doi: 10.1016/j.procs.2022.12.129.

[19] S. J. Yoo, “Neural-Network-Based adaptive resilient dynamic surface control against unknown deception attacks of uncertain nonlinear Time-Delay cyberphysical systems,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 10, pp. 4341–4353, Oct. 2020, doi: 10.1109/tnnls.2019.2955132.

[20] J. P. Yang, “Theoretical formulation of Three-Mass vehicle model for Vehicle–Bridge Interaction,” *International Journal of Structural Stability and Dynamics*, vol. 21, no. 07, p. 2171004, Apr. 2021, doi: 10.1142/s0219455421710048.

[21] Z. Li, F. A. Khasawneh, X. Yin, A. Li, and Z. Song, “A new microscopic traffic model using a Spring-Mass-Damper-Clutch system,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 8, pp. 3322–3331, Aug. 2020, doi: 10.1109/tits.2019.2926146.

[22] E. I. Altman, “Applications of Distress Prediction Models: What Have We Learned After 50 Years from the Z-Score Models?,”

International Journal of Financial Studies, vol. 6, no. 3, p. 70, Aug. 2018, doi: 10.3390/ijfs6030070.

[23] LiTeng, XuZhiyuan, TangJian, and WangYanzhi, "Model-free control for distributed stream data processing using deep reinforcement learning," *Proceedings of the VLDB Endowment*, vol. 11, no. 6, pp. 705–718, Feb. 2018, doi: 10.14778/3199517.3199521.

[24] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, doi: 10.1109/icra.2018.8463189.

[25] F. Kurniawan, S. Sulaiman, S. Konate, and M. A. A. Abdalla, "Deep learning approaches for MIMO time-series analysis," *International Journal of Advances in Intelligent Informatics*, vol. 9, no. 2, p. 286, Jul. 2023, doi: 10.26555/ijain.v9i2.1092.

[26] A. H. Fath, F. Madanifar, and M. Abbasi, "Implementation of multilayer perceptron (MLP) and radial basis function (RBF) neural networks to predict solution gas-oil ratio of crude oil systems," *Petroleum*, vol. 6, no. 1, pp. 80–91, Mar. 2020, doi: 10.1016/j.petlm.2018.12.002.

[27] Y. Mao, A. Pranolo, A. P. Wibawa, A. B. P. Utama, and F. A. Dwiyanto, "Robust LSTM with Tuned-PSO and Bifold-Attention mechanism for analyzing multivariate Time-Series," *IEEE Access*, vol. 10, pp. 78423–78434, Jan. 2022, doi: 10.1109/access.2022.3193643.

[28] B. Jiang, D. Liu, H. R. Karimi, and B. Li, "RBF Neural Network Sliding Mode Control for Passification of Nonlinear Time-Varying Delay Systems with Application to Offshore Cranes," *Sensors*, vol. 22, no. 14, p. 5253, Jul. 2022, doi: 10.3390/s22145253.

[29] L. Wu, G. Perin, and S. Picek, "I choose you: Automated hyperparameter tuning for deep learning-based side-channel analysis," *IEEE Transactions on Emerging Topics in Computing*, pp. 1–12, Jan. 2022, doi: 10.1109/tetc.2022.3218372.

[30] Z. Wu, A. Tran, D. Rincón, and P. D. Christofides, "Machine learning-based predictive control of nonlinear processes. Part I: Theory," *AIChE Journal*, vol. 65, no. 11, Aug. 2019, doi: 10.1002/aic.16729.

[31] J. Arroyo, C. Manna, F. Spiessens, and L. Helsen, "Reinforced model predictive control (RL-MPC) for building energy management," *Applied Energy*, vol. 309, p. 118346, Mar. 2022, doi: 10.1016/j.apenergy.2021.118346.

[32] S. Sabzevari, R. Heydari, M. Mohiti, M. Savaghebi, and J. Rodríguez, "Model-Free neural Network-Based predictive control for robust operation of power converters," *Energies*, vol. 14, no. 8, p. 2325, Apr. 2021, doi: 10.3390/en14082325.