From state observer to deep neural network: design, optimization, and application in bearing dynamics modeling

Yiliang Qian[1], Yan Wang[2], Diwang Ruan[3], Zhaorong Li[4], Jianping Yan[5], and Clemens Gühmann[6]

[1,3,4,5] *School of Aeronautics and Astronautics, Zhejiang University, Hangzhou, 310027, China*
yiliangqian@zju.edu.cn
ruandiwang607@163.com
lizhaorong@zju.edu.cn
jianping.yan@zju.edu.cn

[2] *School of Electrical Engineering and Computer Science, Technische Universität Berlin, Berlin, 10625, Germany*
yanwang_oliver@163.com

[6] *Chair of Electronic Measurement and Diagnostic Technology, Technische Universität Berlin, Berlin, 10625, Germany*
clemens.guehmann@tu-berlin.de

Abstract

Neural networks have been widely applied in system dynamics modeling. However, traditional neural networks still encounter limitations in capturing long-term dynamics, nonlinear modeling, and interpretability. To address these challenges, this study proposes a novel neural network architecture, Deep Stacked State-observer based Neural Network (DSSO-NN). Firstly, the state-space representation is introduced, integrating discretized state-space equations into the neural network design to leverage both system state information and deep learning capabilities. Subsequently, two optimization measures are employed to enhance the network's nonlinear modeling ability with activation functions and the state observer, respectively. Finally, DSSO-NN is validated using the Case Western Reserve University bearing dataset. Experimental results demonstrate that activation functions provide minimal improvement to model performance. In contrast, the incorporation of the state observer significantly enhances the DSSO-NN's ability to capture system dynamics behaviors and improves modeling accuracy. DSSO-NN exhibits higher precision and greater stability, offering a novel perspective on using the state observer as an alternative to traditional activation functions.

1. Introduction

In the context of the continuous development of neural network and deep learning theories, significant progress has been made in various fields, particularly in applications such as system dynamics modeling (Sarker, 2022) and fault diagnosis (Ruan, Chen, Gühmann, Yan, & Li, 2022). However, when dealing with nonlinear, time-varying, and multivariable systems, existing neural network architectures still face numerous challenges. Traditional neural networks often struggle to effectively capture the dynamics characteristics related to system states, failing to fully exploit the rich information inherent in the system. Despite certain breakthroughs in nonlinear modeling, current methods remain insufficient in addressing the complex dynamics in real-world systems, such as nonlinear interactions and time-varying parameters (Ren, Liu, Cheng, Ma, & Li, 2023). Additionally, the issues of interpretability and stability of neural networks in dynamics scenarios have not been effectively resolved, which limits their widespread application in safety-critical domains (Li, Zhang, Li, & Si, 2024). Therefore, enhancing the performance of neural networks in dynamics modeling, particularly in capturing complex time-varying behaviors, improving nonlinear modeling capabilities, and effectively integrating dynamics information from physical models into the learning process, remains a pressing scientific challenge.

The state-space model, with its ability to describe system dynamics in a structured and interpretable form, offers a promising foundation for addressing these challenges. This has led researchers to integrate conventional state-space models with neural network frameworks, culminating in the development of the State-space Neural Network (SS-NN) architecture (Amoura, Wira, & Djennoune, 2011). A notable benefit of state-space models is their capacity to explicitly delineate system state variables and dynamics processes, which frequently enhances interpretability. Amoura et al. (Amoura et al., 2011) proposed SS-NN for modeling nonlinear dynamics systems, proving its capability to estimate nonlinear relationships by substituting physical matrices with trainable ones. Rangapuram et al. (Rangapuram et al., 2018) proposed a state-space model for time-series forecasting, parameterized using a Recurrent Neural Network (RNN) trained with both time-series and dataset co-variants, confirming its applicability for bearing data as a type of time-series. Hauser et al. (Hauser, Gunn, Saab Jr., & Ray, 2019) applied state-space representation to understand deep neural networks, demonstrating that the Residual Neural Network (ResNet) (He, Zhang, Ren, & Sun, 2015) can be represented as a state-space neural network, with skip connections equivalent to first-order systems. Although SS-NN architecture has shown significant success in various fields, it still faces considerable challenges in modeling complex nonlinear and time-varying dynamics systems. Traditional state-space models typically assume linear system dynamics, whereas many real-world systems exhibit highly nonlinear behavior, making such models inadequate for capturing these complexities.

To overcome the limitations of current neural network modeling methods, improve the handling of dynamics system states, and enhance nonlinear modeling, interpretability, and generalization, this paper proposes an innovative modeling framework-Deep Stacked State-observer based Neural Network (DSSO-NN). DSSO-NN integrates the structural representation capability of state-space models with the powerful fitting ability of neural networks. By incorporating a state observer mechanism, DSSO-NN efficiently constructs complex dynamics system models while providing real-time estimation of internal states. The main contributions of this study are as follows.

- Innovative network architecture design: A novel neural network framework is proposed, enabling the seamless integration of dynamics system state information with the deep learning capabilities of neural networks.
- Enhanced nonlinear modeling via optimization: By comparing the effects of optimization strategies using activation functions and the state observer, DSSO-NN reveals robust nonlinear modeling capabilities, attributed to the inclusion of state observer mechanisms.
- Validation with experimental data: The effectiveness of the proposed method is validated through modeling experiments on the Case Western Reserve University (CWRU) bearing dataset, underscoring its potential for applications in system dynamics modeling.

The rest of this paper is organized as follows: Section 2 introduces the state-space model and the state observer. Section 3 describes the design of DSSO-NN, along with the dataset and pre-processing operations, and presents the results on the single state-space layer. Section 4 compares the impact of two optimization approaches on the performance of DSSO-NN. Section 5 is the discussion. Section 6 concludes the paper, summarizing the findings and outlining potential directions for future work.

2. Methodology

2.1. State-space model

By creating a neural network with time-related hidden state, the model could fit and predict certain nonlinear dynamics systems. The mathematical expression for a state-space representation is given by Eq. (1) (Luenberger, 1967):

$$\begin{cases} \dot{x} = A \cdot x + B \cdot u \\ y = C \cdot x + D \cdot u \end{cases}, \tag{1}$$

where x represents the hidden state of the model, u is the input, $\dot{x}$ is the differentiation in time, and y is system output. The number of state variables d in the state-space model represents the model's order. For a dth-order Multi-input Multi-output (MIMO) system with m-dimensions of input and n-dimensions of output, the dimensions of parameter matrices are calculated as: $A \in R^{d\times d}$, $B \in R^{d\times m}$, $C \in R^{n\times d}$, $D \in R^{n\times m}$. The matrix A, B, C, and D are called the state matrix, the input matrix, the output matrix, and the feed-forward matrix, respectively.

The state-space representation is based on continuous-timestep, to apply such structure to deep learning tasks, Euler's method is applied in this task (Londoño & Olivera, 2019). In Eq. (2), the differentiation of the hidden state is substituted with Euler's function, and ΔT is the sampling period of the system. Eq. (3) can be obtained by simplifying Eq. (2). Eq. (4) indicates the output function of the system. Eqs. (3) and (4) are summarized together to obtain Eq. (5), where the discretization of the parameter matrices is defined in Eq. (6), while the dimension of the parameter matrices remains the same.

$$\dot{x} = \frac{x(k+1) - x(k)}{\Delta T} = A \cdot x(k) + B \cdot u(k), \tag{2}$$

$$x(k+1) = (I + \Delta T \cdot A) \cdot x(k) + \Delta T \cdot B \cdot u(k), \tag{3}$$

$$y(k) = C \cdot x(k) + D \cdot u(k), \tag{4}$$

$$\begin{cases} x(k+1) = \Phi \cdot x(k) + G \cdot u(k) \\ \quad y(k) = H \cdot x(k) + J \cdot u(k) \end{cases}, \tag{5}$$

$$\Phi = I + \Delta T \cdot A, \quad G = \Delta T \cdot B, \quad H = C, \quad J = D. \tag{6}$$

2.2. State observer

A state observer is a system that can estimate the hidden state of a real-world system by observing its output. One of the classic types of state observers is the Luenberg observer (Luenberger, 1971), which is widely used in control theory. The goal of the Luenberg observer is to minimize the error between the estimated states and the real states through the feedback of the difference between the estimated and observed outputs.

The mathematical formulation of a Luenberg observer is defined in Eq. (7) (Luenberger, 1971), where $\hat{x}(t)$ and $x(t)$ represent the estimated and real hidden states at timestep t, respectively, and $\hat{y}(t)$ and $y(t)$ denote the estimated and real system outputs. L is the defined gain matrix to guarantee the desired system dynamics and convergence.

$$\begin{cases} \dot{\hat{x}}(t) = A \cdot \hat{x}(t) + B \cdot u(t) + L \cdot (y(t) - \hat{y}(t)) \\ \hat{y}(t) = C \cdot \hat{x}(t) + D \cdot u(t) \end{cases}. \quad (7)$$

3. Design of deep stacked state-observer based neural network (DSSO-NN)

3.1. From state-space model to neural network layer

As noted in Section 2.1, the hidden time-relevant dynamics of a system can be expressed using a state-space representation. The relationship of such dynamics is calculated with the iteration of the hidden state through the state function, and by using Eq. (1), the hidden state at each timestep can be transformed into the desired form of output. Thus, the system dynamics can be calculated using the parameter matrices A, B, C, and D and the initial state x_0.

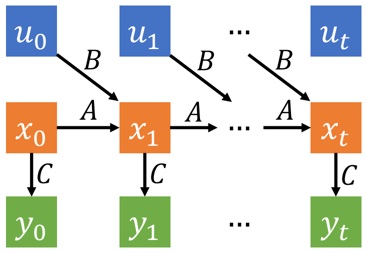


Figure 1. Basic structure of a DSSO-NN layer.

Based on this concept, the state-space model can be regarded as a special case of an RNN. As shown in Fig. 1, A, B, C, D indicate the four parameter matrices of the layer, $[u_0, ..., u_t]$ represents the input vector, $[y_0, ..., y_t]$ is the output vector of the layer, and $[x_0, ..., x_t]$ is the hidden state between the input and output. Hidden state at the first timestep x_0 is randomly initialized, the calculation of the hidden state at next timestep is determined by the input and the hidden state at current timestep. Meanwhile, the output is exported through the hidden state and the parameter matrix C. The computational process of the DSSO-NN layer is presented in Fig. 2.

3.2. Training and evaluation strategy

In this study, the Adam optimizer is employed for all training and validation processes to ensure efficient optimization. The model's trainable parameters are initialized with a uniform distribution with upper and lower bounds of [-1, 1]. As stated

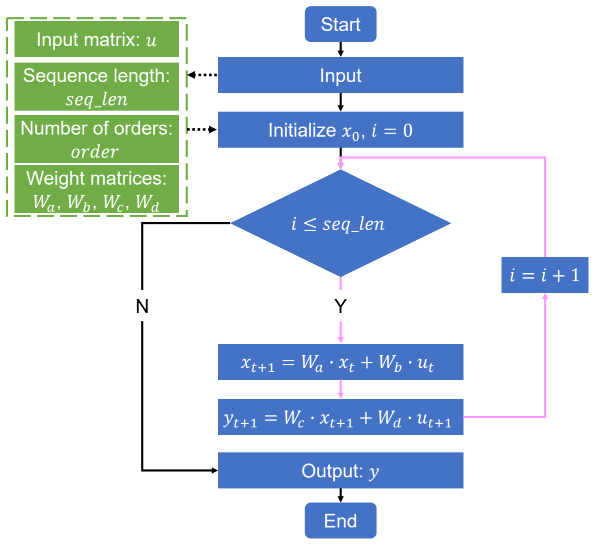

Start
Input
Initialize x_0, $i = 0$
$i \le seq\_len$
Y
N
$x_{t+1} = W_a \cdot x_t + W_b \cdot u_t$
$y_{t+1} = W_c \cdot x_{t+1} + W_d \cdot u_{t+1}$
$i = i + 1$
Output: y
End
Input matrix: u
Sequence length: seq_len
Number of orders: $order$
Weight matrices: W_a, W_b, W_c, W_d

Figure 2. Flowchart of DSSO-NN layer forward function.

in Section 3.1, the forward calculation is computed through multiple iterations of the hidden state. To avoid exponential gradient increases and Not a Number (NaN) outputs caused by randomly initialized parameter matrices during forward iterations, gradient clipping is applied during training. The maximum gradient is set to 30. Meanwhile, during the training process, an early-stopping function is used to terminate the training if the training loss does not decrease in three consecutive epochs.

The CWRU dataset discussed in Section 3.3 is bearing acceleration data collected in the time domain. Each single piece of data is a one-dimensional vector with the specified length based on the fault characteristic frequency. To test and evaluate such time-series data, the Root-Mean-Squared Error (RMSE) loss function is chosen for the training stage. The RMSE loss of the weights to be trained $\hat{\theta}$ is defined by the accumulation of the squared difference between predicted output $\hat{y}$ and the target output y at each timestep, as shown in Eq. (8) (Willmott & Matsuura, 2005). The RMSE loss alone cannot justify the performance of the model, the R-squared (R^2) score is also adopted as an evaluation metric for quantifying the modeling performance (Draper & Smith, 1966).

$$\mathrm{RMSE}(\hat{\theta}) = \sqrt{\frac{\sum_{i=1}^{N_t} (\hat{y}_i - y_i)^2}{N_t - 1}}. \quad (8)$$

3.3. Dataset introduction and pre-processing

As shown in Fig. 3, the CWRU test stand consists of the following components, including a 2 horsepower electric motor, a shaft with a torque transducer and encoder, a dynamometer, and an electronic control system. Both bearings are featured with four different fault diameters (fd), including 0.007 inches, 0.014 inches, 0.021 inches, and 0.028 inches. For this paper, the data acquired from the drive-end bearings and the 12 kHz drive end acceleration data is selected. Validation is conducted using samples from the CWRU dataset with outer race fault diameters of 0.007 inches and 0.021 inches (CWRU outer race, fd=0.007"/0.021").

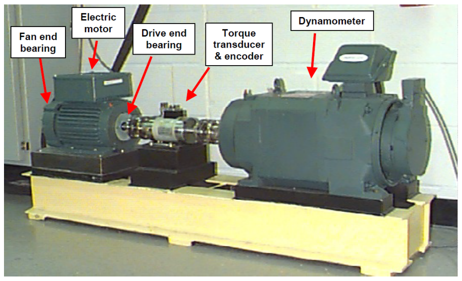


Figure 3. Test bench of CWRU data (Smith & Randall, 2015).

The dataset is raw signals obtained directly from the test bench. To utilize the data for bearing dynamics modeling, the dataset should be pre-processed. When a fault occurs, a periodic vibration signal will be generated from the failure part while the bearing is in operation. To obtain a signal that can fully represent the bearing failure cycle, the available dataset should be segmented. Starting from one fault spike of the vibration signal, with a fixed length of data, a complete failure cycle of the bearing is formed. The bearing failure period is calculated with Eq. (9).

$$n_{timestep} = \lceil \frac{1}{f_{BPFX}} \cdot f_s \rceil, n_{timestep} \in Z \tag{9}$$

where f_s indicates the sampling frequency, and $f_{BPFX} \in [f_{BPFI},\ f_{BPFO},\ f_{BSF}]$. These fault characteristic frequencies can be obtained once bearing geometry parameters and shaft rotation speed are given (Ruan, Wang, Yan, & Gühmann, 2023). Following Eq. (9), the timestep of the CWRU dataset for the out race fault is calculated as 116.

To minimize the impact of high-frequency noise on the signals, both datasets shall be filtered by a low-pass filter after segmentation of the dataset. The filtered signals are subsequently subjected to normalization. The speed of the shaft and the torque of the motor for the same working conditions can be considered constants for the bearing model since these two variables do not change during the operation of the bearing. The input shall become a two-dimensional vector with a corresponding length the same as the target vector. The dimensions of the input and output are shown in Table 1. To avoid the problem caused by the excessive value of the shaft speed (over 1000) compared to the order of magnitude of the torque ($0 \sim 5$), the shaft speed is divided by 60. The overview of pre-processed CWRU dataset are summarized in Table 2.

The hyperparameters used to train all the models remain the same throughout the experiment, the specific hyperparameters are displayed in Table 3. All experiments are conducted on a computer with a CPU Intel i7-7700HQ and a GPU NVIDIA 1060. A portion of the training set is randomly chosen to serve as the validation set in addition to the pre-processed training set and the test set. The model is validated on the validation set after each epoch of training.

Table 1. In/output names and dimensions.

Symbol	Name	Dimension
u_1	Motor load	$[2, n]^*$
u_2	Motor speed / 60	
y_{obs}	Observation value (Target)	$[1, n]$

* Assume the length of target data is n.

Table 2. Summary of CWRU dataset.

Fault size	Outer race	
	Training set	Test set
fd=0.007"	600	80
fd=0.021"	600	80

3.4. Result on the single state-space layer

The model is validated on two different subsets of CWRU dataset. The data collected from the outer race are selected. All of the examined models in this paper are trained five times, and the results of each training are recorded.

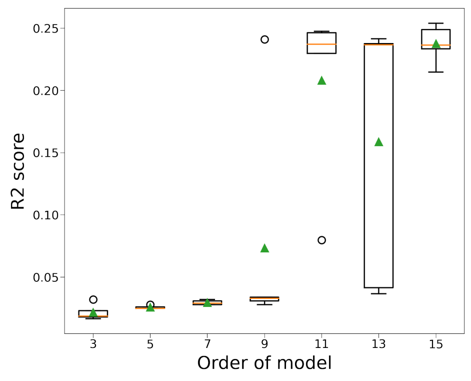
3.4.1. Case 1: CWRU outer race, fd=0.007"

The result is presented in Fig. 4a. Each box represents a group of data with the same label. The green arrow indicates the average value among the group, the top line and the bottom line represent the maximum and the minimum value, respectively. The irregular value is marked with a small circle. The orange line indicates the median value of this group. The R^2 score of each model indicates the average R^2 score of the test set. The sequence of a single layer model is set within the range [3, 5, 7..., 15]. As the order grows, the number of trainable parameters increases significantly.

As shown in Fig. 4a, while the number of trainable parameters grows, the R^2 score shows an overall increasing trend. When the model's order is low, its performance stabilizes at a very low level (close to 0). Additionally, although substantially improved, the model's performance at higher orders is still unstable. The maximum R^2 score is around 0.25. In Fig.

Table 3. Hyperparameters for training.

Hyperparameter	Setting
Optimizer	Adam
Learning rate	0.001
Early-stopping patience	3
Batch size	1


(a) CWRU outer race, fd=0.007".

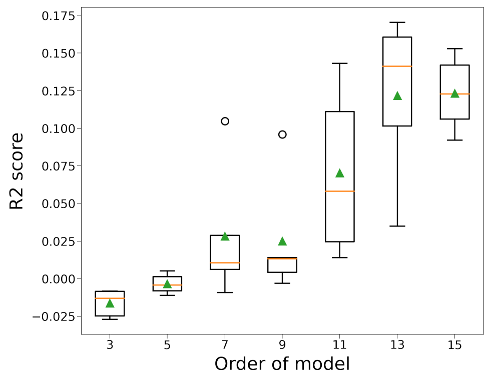
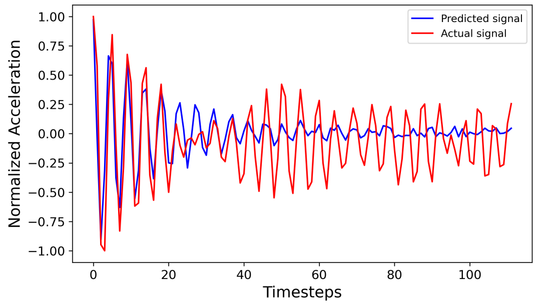

(b) CWRU outer race, fd=0.021".

Figure 4. R^2 score on the single layer model.


(a) CWRU outer race, fd=0.007".

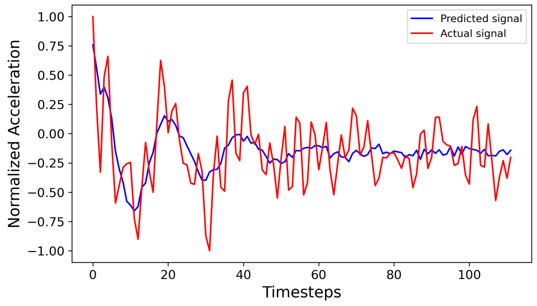

(b) CWRU outer race, fd=0.021".

Figure 5. Validation result with the best performance on the single layer model.

5a, the example of the predicted output of the model under 15th-order is plotted. The starting part of the bearing dynamics has already been expressed in the model, and the pattern of degrading vibration over time is also noticeable. However, since the model inputs are constant, the prediction cannot describe the nonlinearity of the signal, and the performance of the model is below satisfaction.

3.4.2. Case 2: CWRU outer race, fd=0.021"

As presented in Fig. 4b, the models also show an increasing trend in R^2 score with growing orders. The maximum of R^2 score is below 0.175, which indicates that the fitting ability is extremely restricted for the single state-space layer. Fig. 5b gives a detailed look at the qualitative result. The data with a larger fault diameter is much more complex, and the target outputs contain high-frequency noises despite the dataset already being pre-processed with a low-pass filter. Therefore, the predicted data can only follow the linear trend of the target and is incapable of modeling the nonlinear part of the target.

4. Optimization of DSSO-NN

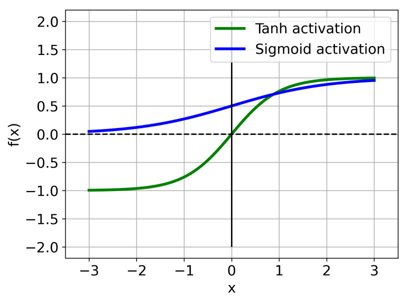
In Section 3.1, the fundamental structure of the network is presented. However, the validation results show that the fitting ability is not ideal. While the linear trend of the target signal is followed by the predicted signal, the nonlinear part of the target signal remains missing.

4.1. Activation function

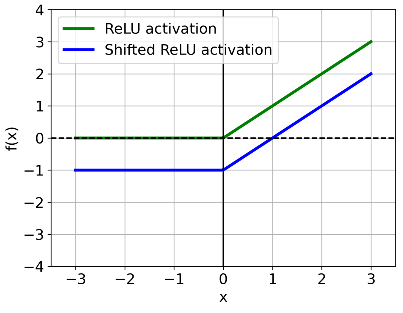
The simple state-space-based model is not able to fit the nonlinear part of the signal. The conventional solution is to use an activation function while computing the output. A nonlinear trend could be integrated by the activation function into the output of the model. In deep learning, some of the commonly used activation functions are Tanh, ReLU, Sigmoid, and SoftPlus.

Tanh function is widely used in time-series data training, which is defined by Eq. (10). The output of the Tanh function is restricted to [-1, 1] and centered at zero-point. The definition of the Sigmoid function is similar to Tanh (see Eq. (11)), while the output range of Sigmoid is [0, 1]. The Sigmoid function can be regarded as the result of a linear variation of the Tanh function. The shape of both functions are illustrated in Fig. 6a.

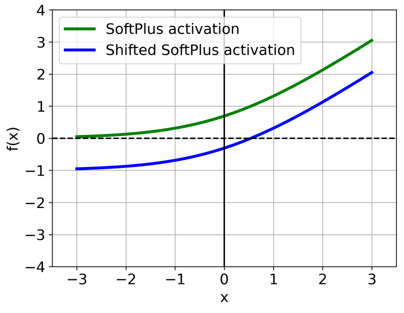
$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \tag{10}$$



(a) Sigmoid and Tanh activation.

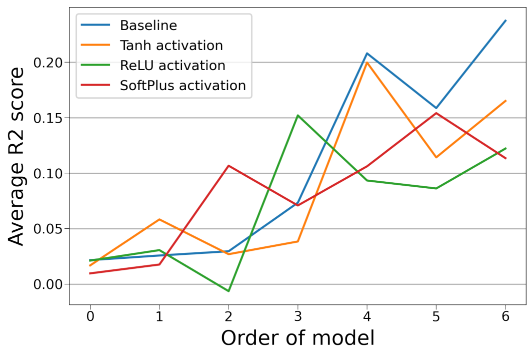


(b) ReLU activation.

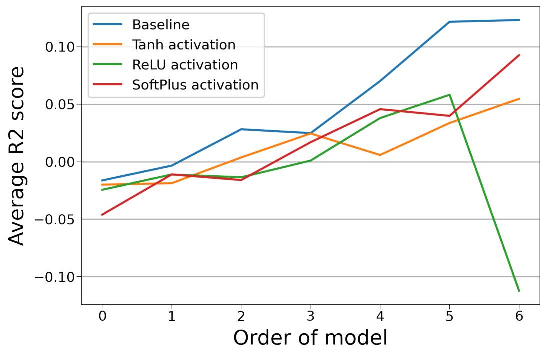


(c) SoftPlus activation.

Figure 6. Illustration of different activation functions.



(a) CWRU outer race, fd=0.007”.



(b) CWRU outer race, fd=0.021”.

Figure 7. Average R^2 score with different activation functions.

$$f(x) = \frac{1}{1+e^{-x}}. \tag{11}$$

ReLU function is calculated via Eq. (12), which only projects positive input, and 0 will be produced when the input is negative. This feature helps the model to operate at a faster rate, but also brings the problem of zero gradient when the input is negative. Thus, ReLU function used in this paper is shifted along the y-axis below, which enables the output of ReLU function to cover the whole number field of the dataset. Fig. 6b shows the shape of the original as well as the shifted ReLU function.

$$f(x) = \begin{cases} \max(0, x) & , x >= 0 \\ 0 & , x < 0 \end{cases} \tag{12}$$

SoftPlus activation is defined with Eq. (13). SoftPlus can be taken as a smoothing process for ReLU function. Similar to ReLU, the output of SoftPlus is restricted and the function used is also shifted. The shape of the SoftPlus function is demonstrated in Fig. 6c.

$$f(x) = \ln(1 + e^x). \tag{13}$$

4.2. Results with different activation functions

4.2.1. Case 1: CWRU outer race, fd=0.007”

The experiments are conducted by the models in increasing order. Models with different activation functions and orders are also trained five times, and the comparison of each kind of model is analyzed with the mean of the average R^2 score among the five trainings. In Fig. 7a, the results between the baseline model (without activation function) and models with selected activation functions are illustrated. With different activation functions, the fitting ability of the models shows inconspicuous improvement. At higher orders, the model with the activation function performs even worse than the model without it.

4.2.2. Case 2: CWRU outer race, fd=0.021”

The dataset collected with a large fault diameter is more complicated. Data with a fault diameter of 0.021” are used for model training in different orders. Fig. 7b compares these models with the test R^2 score. The results are similar to those obtained from the data with a fault diameter of 0.007”, the model without the additional activation function outperforms the one with the additional activation function.

Generally, activation functions are used to add nonlinearity to model output. However, based on the experiments conducted above, the impact of the activation functions on the state-space-based models is insufficient. Hence, the activation functions are not selected as part of the models explored in this paper.

4.3. Linear state observer correction

As introduced in Section 2.2, the state observer is a system with state observation and output correction. These features can be utilized to improve the model's capability to fit nonlinearity. With this understanding, the gain vector L in Eq. (7) can be set to a trainable parameter. By training with the observation data, parameter matrices shall be updated with the optimizer, including L. Meanwhile, L vector promotes the correction of the predicted output value to approach the actual output. Finally, the DSSO-NN layer with linear state observer correction can be constructed (see Fig. 8). The pseudocode for the forward function of the DSSO-NN layer with a linear observer is shown in Algorithm 1.

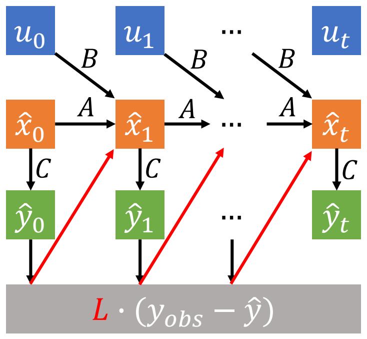


Figure 8. DSSO-NN layer with linear state observer.

4.4. Results with linear state observer

4.4.1. Case 1: CWRU outer race, fd=0.007”

In this experiment, the single state-space is selected as the baseline model. A state-space model with a linear observer is trained with the same hyperparameters five times as a comparison test. As illustrated in Fig. 9a, the performances of the models with a linear observer have a lower variance compared to the models without, meanwhile, the mean of R^2 score among the five trainings also reaches over 0.89. Compared to the results obtained from the models without the linear observer in Fig. 4a, the R^2 scores of each order's model increase significantly. Also, the trend of increasing R^2 scores along with the increasing order of the model remains. Fig. 10a presents the comparison between the results obtained from models trained on the same dataset with and without a linear observer, which more clearly shows the improvement in the performance of the model by the linear observer. The orange bars represent the results of the models with a linear observer and the blue bars are those without. The model's test R^2 scores at each order show distinct improvement, where the best average R^2 score of the models without linear observer is around 0.2, yet the minimum of the average R^2 score of the models with linear observer is over 0.89. Based on the analysis above, the models with linear observer correction show great progress in terms of R^2. The best performance of the model appears in the 11th-order model. Comparing with the best model without a linear observer, the validation R^2 is improved from 0.237 to 0.902.

The result of the validation set is plotted in Fig. 11a. With this model structure, there is a clear improvement in the qualitative results as well. The blue prediction line is almost identical to the target line. In Fig. 5a, the prediction computed by the model without a linear observer can still maintain a small error with the actual signal at the beginning, but with each iteration of the operation, the error gradually accumulates. Finally, the predicted value cannot match the actual signal at all. However, the prediction computed by the model with the linear observer not only follows the actual signal well at the beginning. As time goes on, the errors between the predicted signal and the actual signal are not accumulated but maintained within a normal range. This shows the power of the linear state observer, which brings the model's fitting ability to another level.

Algorithm 1 Pseudocode of DSSO-NN layer forward function with linear observer correction.

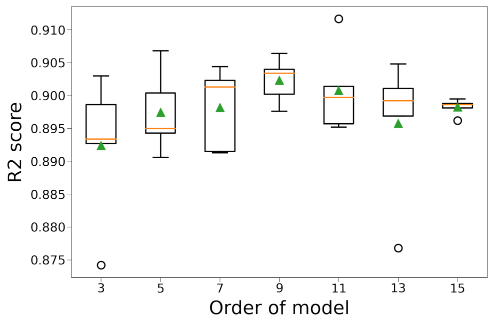
Input: u: input matrix, y_{obs}: observation value of output, seq_len: sequence length, $order$: number of orders, W_a, W_b, W_c, W_d: weight matrices, W_L: observer correction vector.

```
1: initialize hidden state $x_0$
2: for $i = 0$ to $seq\_len$ do
3:     $x_{t+1} = W_a \cdot x_t + W_b \cdot u_t$
4:     $y_{t+1} = W_c \cdot x_{t+1} + W_d \cdot u_{t+1}$
5:     if $y_{obs}$ then
6:         $x_{t+1} = x_{t+1} + W_L \cdot (y_{obs} - y_t)$
7:     end if
8: end for
```

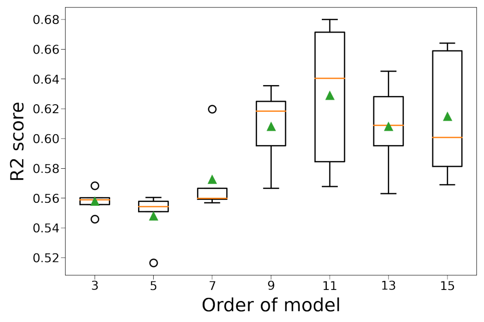
Output: estimated vector of y.

4.4.2. Case 2: CWRU outer race, fd=0.021”

As demonstrated in Fig. 9b, the average R^2 score increases as the order increases. The mean R^2 score of each model is higher at high order. The highest average R^2 score is 0.628, obtained from the 11th-order models. Compared to the results obtained from the models without a linear observer in Fig. 4b, the R^2 scores of each order's model increase significantly. Fig. 10b presents the comparison between the models with/without a linear observer. The average R^2 score at each order of the model from the model without a linear observer contains values less than 0 and the maximum is only around 0.1. The models with the linear state observer present a significant improvement compared with the baseline mod-

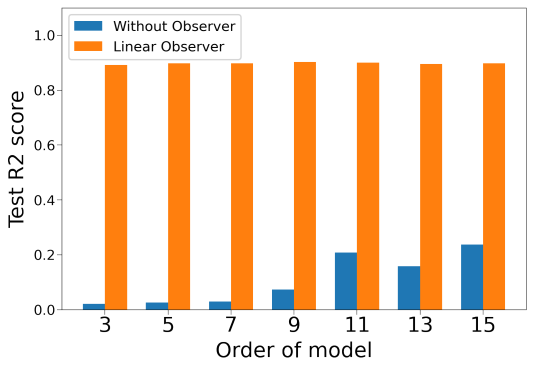


(a) CWRU outer race, fd=0.007".

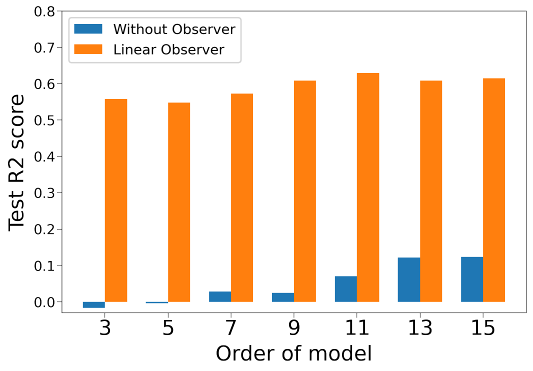


(b) CWRU outer race, fd=0.021".

Figure 9. Results with linear state observer.



(a) CWRU outer race, fd=0.007".



(b) CWRU outer race, fd=0.021".

Figure 10. Comparison between models with/without linear state observer.

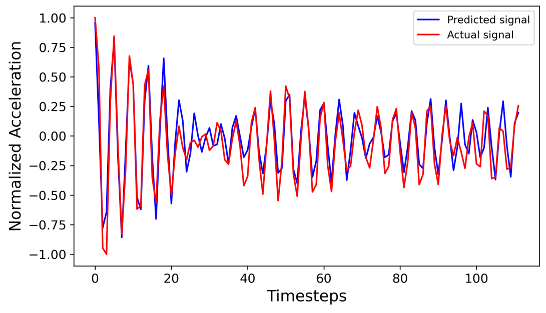
els. The best performance of the models appears in the 11th-order model. The validation R^2 score is raised from 0.123 to 0.628.

Fig. 11b shows the validation results with the best performance. Though the data from the CWRU outer race (fd=0.021") contain much more noise, the model is capable of following not only the linear trend but also the nonlinear trend. The predicted signal can maintain a stable error with the actual signal most of the time. However, the linear state observer does not perform well when the signals suddenly increase or decrease on a large scale. The signal decreases from -0.25 to -1 and increases back to over -0.2 in a very short time, which means the correction provided by the linear observer ($L \cdot (y_{obs} - y_{pred})$) does not vary enough while the difference increases or decreases, further optimization should be undertaken.

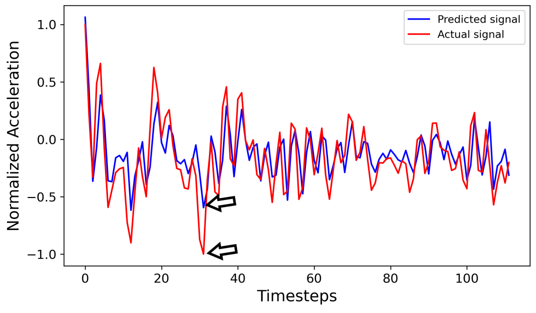
Nevertheless, DSSO-NN demonstrates admirable performance across a range of metrics in terms of tracking the expected outcomes. Therefore, employing DSSO-NN for bearing dynamics modeling is deemed feasible.

5. Discussion

Activation functions are widely regarded as critical components in enhancing the nonlinearity of neural networks, enabling them to model complex input-output relationships (Apicella, Donnarumma, Isgrò, & Prevete, 2021). However, our experiments on DSSO-NN reveal that the contribution of activation functions to model performance improvement is negligible, as shown in Fig. 7. This finding challenges the traditional perspective that activation functions are indispensable for increasing nonlinearity and improving performance. Current research often focuses on optimizing activation functions or their variants, such as ReLU, Leaky ReLU, and Sigmoid, to enhance neural network performance (Ramachandran, Zoph, & Le, 2017; Deng, Wang, & Lin, 2024). However, these activation functions have inherent limitations. On the one hand, their behavior is static and predefined, lacking adaptability to specific system dynamics or signal characteristics. On the other hand, in tasks involving high-dimensional data or complex signals, activation functions may amplify noise, leading to instability or reduced robustness in model outputs. These challenges are particularly pronounced in the domain of system dynamics model-



(a) CWRU outer race, fd=0.007".



(b) CWRU outer race, fd=0.021".

Figure 11. Validation results with the best performance using the linear state observer.

ing, where the dynamics characteristics of the process being modeled are both diverse and crucial.

In this study, we propose replacing activation functions with state observers. Unlike activation functions, state observers incorporate system dynamics into the modeling process by introducing nonlinearity through real-time state information. This mechanism aligns the network's responses more closely with the physical system being modeled, thereby enhancing its ability to capture dynamics features. Our experiments demonstrate that substituting activation functions with state observers in bearing dynamics modeling achieves superior performance. The findings presented herein pave the way for the development of novel deep learning layers grounded in state observers.

By standardizing state-observer-based layers as replacements for traditional activation functions and applying them across various architectures, a more interpretable and robust method for introducing nonlinearity can be established. This approach also reduces the reliance on hyperparameter tuning of activation functions, improving training stability and convergence. Moreover, state-observer-based layers, being more closely aligned with the fundamental dynamics of the system, enhance the generalization capabilities of deep learning models, resulting in better performance in practical applications.

Finally, while this study represents an initial exploration of replacing activation functions with state observers, it underscores the feasibility and value of this research direction. By standardizing state-observer-based layers, future neural network designs could reduce reliance on data-driven optimization while achieving greater interpretability and computational efficiency. Nevertheless, this study still requires further optimization. For instance, future work could employ visualization analysis to elucidate the underlying mechanism by which the state observer outperforms activation functions, explore whether integrating the two yields superior performance compared to a standalone state observer, and investigate more diverse forms of state observer modules for deep learning networks by drawing inspiration from the design principles of various activation functions.

6. Conclusion

To address the challenges posed by neural networks in modeling complex dynamics systems, a novel DSSO-NN is proposed in this paper. The core idea involves constructing neurons based on traditional state-space equations and state observers to form the network layers. This architecture enables DSSO-NN to effectively overcome the limitations of existing methods in handling nonlinear and time-varying dynamics systems. Experimental validation demonstrates the superior performance of this framework in bearing dynamics modeling.

In conclusion, the incorporation of state observers significantly enhances the nonlinear modeling capabilities of DSSO-NN and offers a promising alternative when traditional activation functions fail. Experimental results on the CWRU dataset further validate the advantages of DSSO-NN in dynamics modeling, highlighting its potential in tackling complex nonlinear systems.

Future work includes introducing nonlinear state observers and exploring optimization strategies such as serial and parallel structures to further improve model performance. Mechanism-based modeling could be utilized to investigate the weight parameters in DSSO-NN, aiming to reduce the number of identifiable parameters and enhance training efficiency. Additionally, DSSO-NN could be extended to more complex dynamics systems to assess its scalability and robustness across various industrial and scientific applications.

References

Amoura, K., Wira, P., & Djennoune, S. (2011, 01). A state-space neural network for modeling dynamical nonlinear systems. *NCTA 2011 - Proceedings of the International Conference on Neural Computation Theory and Applications*, 369-376.

Apicella, A., Donnarumma, F., Isgrò, F., & Prevete, R. (2021). A survey on modern trainable activation functions. *Neural Networks*, *138*, 14–32.

Deng, Q., Wang, C., & Lin, H. (2024). Memristive hopfield neural network dynamics with heterogeneous activation functions and its application. *Chaos, Solitons & Fractals*, *178*, 114387.

Draper, N., & Smith, H. (1966). *Applied regression analysis*. Wiley.

Hauser, M., Gunn, S., Saab Jr., S., & Ray, A. (2019). State-space representations of deep neural networks. *Neural Computation*, *31*(3), 538-554.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition.

Li, H., Zhang, Z., Li, T., & Si, X. (2024). A review on physics-informed data-driven remaining useful life prediction: Challenges and opportunities. *Mechanical Systems and Signal Processing*, *209*, 111120.

Londoño, J. D., & Olivera, C. (2019). Discretization by euler's method for regular lagrangian flow.

Luenberger, D. (1967). Canonical forms for linear multivariable systems. *IEEE Transactions on Automatic Control*, *12*(3), 290-293.

Luenberger, D. (1971). An introduction to observers. *IEEE Transactions on Automatic Control*, *16*(6), 596-602.

Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for activation functions.

Rangapuram, S. S., Seeger, M. W., Gasthaus, J., Stella, L., Wang, Y., & Januschowski, T. (2018). Deep state space models for time series forecasting. In *Advances in neural information processing systems* (Vol. 31). Curran Associates, Inc.

Ren, H., Liu, R., Cheng, Z., Ma, H., & Li, H. (2023). Data-driven event-triggered control for nonlinear multi-agent systems with uniform quantization. *IEEE Transactions on Circuits and Systems II: Express Briefs*.

Ruan, D., Chen, Y., Gühmann, C., Yan, J., & Li, Z. (2022). Dynamics modeling of bearing with defect in modelica and application in direct transfer learning from simulation to test bench for bearing fault diagnosis. *Electronics*, *11*(4), 622.

Ruan, D., Wang, J., Yan, J., & Gühmann, C. (2023). Cnn parameter design based on fault signal analysis and its application in bearing fault diagnosis. *Advanced Engineering Informatics*, *55*, 101877.

Sarker, I. H. (2022). Ai-based modeling: techniques, applications and research issues towards automation, intelligent and smart systems. *SN Computer Science*, *3*(2), 158.

Smith, W., & Randall, R. (2015, 05). Rolling element bearing diagnostics using the case western reserve university data: A benchmark study. *Mechanical Systems and Signal Processing*, *64-65*.

Willmott, C., & Matsuura, K. (2005). Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate Research*, *30*, 79–82.