Neural Network State-Space Estimators

Minxing Sun, Li Miao, Qingyu Shen, Yao Mao Senior Member, IEEE, and Qiliang Bao

Abstract—Classical state estimation algorithms rely on predefined target's state-space model, which complicates model derivation and limits adaptability when system dynamics change. Neural network based estimators offer a data-driven alternative, but rarely fuse classical estimation theory into their structure and demand large, pre-computed training sets. To overcome these limitations, we propose a unified state-space structure without target's state-space model and treats both the input-layer activations and all network weights as latent states to be estimated online. We instantiate this nonlinear model with three canonical estimators—the extended Kalman estimator, the unscented Kalman estimator, and the particle estimator—to simulate different neural network and demonstrate its generality. We then benchmark our approach against seven leading neural network estimators across three representative scenarios. Results show that our neural network state-space estimators not only retain the robust learning capability, but also match or exceed the accuracy of both classical and pre-trained neural network methods. (Code, data, and more result: github.com/ShineMinxing/PaperNNSSE.git)

Index Terms—Kalman estimator, Neural network, Learning capability, Adaptability

I. Introduction

State estimation algorithms are extensively utilized across various engineering disciplines [1]–[3]. For example, in electro-optical tracking systems [4], [5], they are essential for mitigating optical path disturbances caused by atmospheric turbulence [6], compensating for delays in hardware and image processing pipelines [7], and filtering out signal noise generated by sensors and target detection mechanisms [8]. The fundamental principle of these algorithms comprises three stages: predicting the next state using the current estimate and target's state-space model, predicting the corresponding observation via the observation equation, and correcting the state estimate based on the deviations between the actual observation and its prediction.

Estimators can be categorized based on how they establish the relationship between observation deviations and state deviations. One category comprises methods such as the Extended Kalman Estimator (EKE), which computes partial derivatives from the state-space model. The primary limitation of EKE is the complexity involved in deriving analytical solutions for these partial derivatives, particularly in highly nonlinear models [9]–[13]. Another category includes the Unscented Kalman Estimator (UKE), which approximates the state distribution by assigning predefined deviations to each estimated state and evaluating the resulting variations in observations. However, UKE faces challenges in accurately capturing the relationships between coupled estimated states and observations [14]–[20]. The third category is represented by Particle Estimator (PE), which introduces random perturbations to the estimated states and assesses the resulting observation variations. While PE is better suited for handling nonlinearities, its main drawback is the high computational cost associated with the large number of particles required, which can impede real-time performance [21]–[24].

All three estimation methods rely on predefined state-space models, and even the robust estimators designed to mitigate model inaccuracies can only compensate for a limited range of errors [7], [25], [26]. In practical applications, defining an accurate state-space model for the target system is often inherently challenging, particularly when tracking non-cooperative targets.

The rapid advancement and widespread adoption of artificial neural network methodologies have profoundly impacted time-domain signal processing, enabling significant breakthroughs in estimation and prediction tasks [27]–[32]. Recurrent Neural Network (RNN), introduced by Hopfield [33], were enhanced by Rumelhart, Hinton, and Williams [34] with the backpropagation algorithm, facilitating more efficient training. Subsequent improvements by Jordan and Elman [35], [36] incorporated feedback loops, enhancing the ability to model temporal dependencies [37]–[39]. However, RNNs struggled with vanishing and exploding gradients, challenges ad-

Student author: M. Sun (sunminxing20@gmail.com).
*Corresponding author: Y. Mao (maoyao@ioe.ac.cn).
All authors are with the State Key Laboratory of Optical Field Manipulation Science and Technology, the Institute of Optics and Electronics, Chinese Academy of Sciences, Chengdu 610209, China.
All authors are with the Key Laboratory of Optical Engineering, the Institute of Optics and Electronics, Chinese Academy of Sciences, Chengdu 610209, China.
All authors are also with the School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences, Beijing 101408, China.

dressed by Hochreiter [40] with Long Short-Term Memory (LSTM) networks, and further refined by Schmidhuber [41] through the addition of forget gates [42], [43]. Gated Recurrent Unit (GRU), proposed by Kyunghyun [44] and validated by Junyoung [45], offered a simplified yet effective alternative to LSTM [46]–[48]. Convolutional Neural Network (CNN), initially developed by LeCun [49] for handwritten digit recognition, have been extended to Recurrent Convolutional Neural Networks [50] to capture spatial and temporal dependencies, significantly improving prediction accuracy in applications such as vehicle and human trajectory forecasting [51]–[54]. Temporal Convolutional Network (TCN), introduced by Colin [55], utilize pooling and upsampling to capture long-term motion patterns, enhancing capabilities in action segmentation and detection [56]–[59] (Chen, 2020; Kaiyu, 2022). Neural Ordinary Differential Equation (NeuralODE), proposed by Ricky [60], represent neural network layers through differential equations, facilitating deeper and more efficient networks for tasks like robotic path planning and travel time estimation [61]–[63]. Transformer architectures, introduced by Ashish [64], leverage self-attention mechanisms to model long-term dependencies while reducing computational complexity. Their applications span maneuvering target tracking, autonomous vehicle trajectory prediction, and 3D human pose estimation [65]–[69].

Traditional estimation algorithms have also been integrated into neural networks to optimize training processes. Early efforts by Singhal, Puskorius, and Pérez-Ortiz [70]–[73] applied EKE to train multilayer perceptrons and RNN, enhancing training efficiency. More recent integrations of UKE and PE have improved the training of feedforward and recurrent networks in applications such as voice classification and evapotranspiration estimation [74], [75]. However, in these studies, estimation algorithms primarily treated the backpropagation algorithm as state-space equations, serving an auxiliary role rather than being the primary mechanism for state estimation. Consequently, the effectiveness of estimation algorithms remains limited and warrants further enhancement [76].

To address these limitations, our proposed algorithm introduces a novel state-space model that treats input layer nodes and all weight parameters as estimated states, effectively simulating a neural network within a state estimation framework. The main contributions of this work are as follows:

- Neural Network State-Space Model (NNSSM): We propose a NNSSM that simulates a neural network, enabling real-time learning and pretraining. The estimation algorithm works wothout target's state-space models and can be tailored by adjusting the input layer length and prediction steps to meet engineering requirements.
- Neural Network State-Space Estimator (NNSSE): The versatility of the proposed NNSSM is validated through the implementation of three representative estimation algorithms—EKF, UKE, and PF—demonstrating its broad applicability.
- Flexible Architecture Configuration: The NNSSM can accommodate arbitrary neural network structures—adjusting the number of layers, the number of neurons per layer, connectivity patterns, and activation functions—to meet diverse application requirements.
- Comparative Performance Evaluation: We compare the estimation performance of our NNSSEs with various neural network architectures, including RNN, LSTM, GRU, TCN, NeuralODE, and Transformer.
- Comprehensive Testing: Extensive tests were conducted on generated sine trajectories, a laboratory dual-reflection mirror platform, and actual unmanned aerial vehicle trajectory data, demonstrating the robustness and effectiveness of our proposed method.

II. Problem Statement

In many real-world tracking problems, the target is non-cooperative, so an accurate state-space model cannot be specified a priori. A common expedient is the uniformly accelerated state-space model in (1), where only position can be observed while velocity and acceleration remain hidden:

$$\begin{aligned} \mathbf{x}_{i+1} &= \begin{bmatrix} 1 & T & \frac{T^2}{2} \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} \mathbf{x}_i + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{u}_i, \\ \mathbf{z}_i &= \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \mathbf{x}_i + \mathbf{v}_i. \end{aligned} \tag{1}$$

The state vector $\mathbf{x}_i = [p_i, v_i, a_i]^\mathsf{T}$ stacks position, velocity and acceleration at instant i. T is the time interval between successive estimates. The process noise u_i absorbs modelling errors and environmental perturbations, and the measurement noise v_i reflects sensor inaccuracies; both are assumed zero-mean Gaussian and seperately follow the covariance $\mathbf{Q}$, $\mathbf{R}$. Their second-order statistics together with the initial error covariance $\mathbf{\Pi}_{0|0}$ are summarised in (2), in which δ_{ij} denotes the Kronecker delta:

$$E\left(\begin{bmatrix}\mathbf{x}_0\\ \mathbf{u}_i\\ \mathbf{v}_i\end{bmatrix}\begin{bmatrix}\mathbf{x}_0\\ \mathbf{u}_j\\ \mathbf{v}_j\end{bmatrix}^{\mathsf{T}}\right)=\begin{bmatrix}\mathbf{\Pi}_{0|0} & 0 & 0\\ 0 & \mathbf{Q}\,\delta_{ij} & 0\\ 0 & 0 & \mathbf{R}\,\delta_{ij}\end{bmatrix}. \quad (2)$$

Because the true motion can be highly nonlinear, model (1) rarely yields the desired predictive accuracy. Although the Kalman estimator provides on-line estimates of v_i and a_i, they are ultimately reconstructed from past position measurements. With scalar gains $w_{v1}, w_{v2}, w_{a1}, w_{a2}$ from calculated Kalman gain, the recursive updates read

$$\begin{aligned}
v_i &= w_{v1}\frac{p_i-p_{i-1}}{T}+w_{v2}v_{i-1}\\
&= w_{v1}\frac{p_i-p_{i-1}}{T}\\
&\quad + w_{v1}w_{v2}\frac{p_{i-1}-p_{i-2}}{T}+w_{v1}w_{v2}^2\frac{p_{i-2}-p_{i-3}}{T}+\dots\\
&= w_{v1}\sum_{k=0}^{i-1}w_{v2}^k\,\frac{p_{\,i-k}-p_{\,i-k-1}}{T},\\
a_i &= w_{a1}\frac{p_i-2p_{i-1}+p_{i-2}}{T^2}+w_{a2}a_{i-1}\\
&= w_{a1}\frac{p_i-2p_{i-1}+p_{i-2}}{T^2}\\
&\quad + w_{a1}w_{a2}\frac{p_{i-1}-2p_{i-2}+p_{i-3}}{T^2}\\
&\quad + w_{a1}w_{a2}^2\frac{p_{i-2}-2p_{i-3}+p_{i-4}}{T^2}+\dots\\
&= w_{a1}\sum_{k=0}^{i-1}w_{a2}^k\,\frac{p_{\,i-k}-2p_{\,i-k-1}+p_{\,i-k-2}}{T^2}.
\end{aligned} \quad (3)$$

Given p_i, v_i, a_i, an n-step-ahead position prediction follows from kinematics:

$$p_{i+n}=p_i+v_i\,nT+\tfrac{1}{2}a_i\,n^2T^2. \quad (4)$$

Substituting (3) into (4) yields a weighted finite-difference expansion

$$\begin{aligned}
p_{i+n} &= p_i+nTw_{v1}\sum_{k=0}^{i-1}w_{v2}^k\frac{p_{\,i-k}-p_{\,i-k-1}}{T}\\
&\quad+\tfrac{1}{2}n^2T^2w_{a1}\sum_{k=0}^{i-1}w_{a2}^k\frac{p_{\,i-k}-2p_{\,i-k-1}+p_{\,i-k-2}}{T^2}\\
&= \sum_{k=0}^{i}w_kp_k,
\end{aligned} \quad (5)$$

Under model (1), long-term prediction ultimately reduces to a weighted sum of past positions, an intrinsic limitation when the target executes nonlinear manoeuvres. To validate the potential of position-only estimated states, we construct the corresponding state-space models.

We begin with the two-position estimator (E2P), based on

$$p_{i+1}=p_i+(p_i-p_{i-1}),$$

which yields

$$\begin{aligned}
\mathbf{x}_{i+1}&=\begin{bmatrix}2 & -1\\ 1 & 0\end{bmatrix}\mathbf{x}_i+\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\mathbf{u}_i,\\
\mathbf{z}_i&=\begin{bmatrix}1 & 0\end{bmatrix}\mathbf{x}_i+\mathbf{v}_i,
\end{aligned} \quad (6)$$

with $\mathbf{x}_i=[p_i,p_{i-1}]^{\mathsf{T}}$.

By extending the position history, we obtain higher-order variants. Defining $\mathbf{x}_i=[p_i,p_{i-1},p_{i-2}]^{\mathsf{T}}$ gives the three-position estimator (E3P), and $\mathbf{x}_i=[p_i,p_{i-1},p_{i-2},p_{i-3}]^{\mathsf{T}}$ yields the four-position estimator (E4P):

$$\begin{aligned}
p_{i+1}&=p_i+\tfrac{1}{2}(p_i-p_{i-1})+\tfrac{1}{2}(p_{i-1}-p_{i-2}),\\
p_{i+1}&=p_i+\tfrac{1}{3}(p_i-p_{i-1})+\tfrac{1}{3}(p_{i-1}-p_{i-2})\\
&\quad+\tfrac{1}{3}(p_{i-2}-p_{i-3}).
\end{aligned} \quad (7)$$

A variable-weight four-position estimator (E4PVW) further refines the weights:

$$\begin{aligned}
p_{i+1}&=p_i+\frac{3(p_i-p_{i-1})}{6}\\
&\quad+\frac{2(p_{i-1}-p_{i-2})}{6}+\frac{(p_{i-2}-p_{i-3})}{6}.
\end{aligned} \quad (8)$$

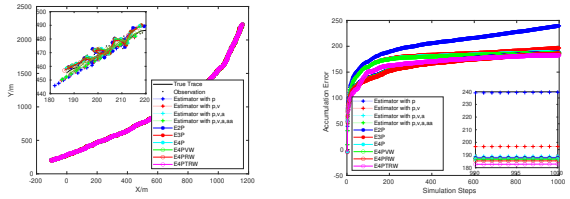
For even greater flexibility, the weights may be learned from data. The offline four-position estimator with regressed weights (E4PRW) uses

$$\begin{aligned}
p_{i+1}=&1.2668\,p_i-0.0152\,p_{i-1}\\
&+0.0103\,p_{i-2}-0.2618\,p_{i-3},
\end{aligned} \quad (9)$$

while its online counterpart (E4PTRW) updates these coefficients at each step using the most recent 50 samples.

Simulation results in Fig. 1 show that all position-stack estimators outperform classical Kalman filters built on $[p_i]$, $[p_i, v_i]$, $[p_i, v_i, a_i]$, and $[p_i, v_i, a_i, \dot{a}_i]$ (the latter including jerk $\dot{a}_i$). Accuracy increases with longer stacks (E3P, E4P) and peaks for variants with adaptive or regressed weights (E4PVW, E4PRW, E4PTRW), with E4PRW notably narrowing the error band and E4PTRW further enhancing adaptability.

Despite these gains, embedding a dedicated linear-regression step in each prediction is conceptually inelegant and risks overfitting when the target's motion pattern changes abruptly.



(a) Position prediction with different estimators.

(b) Accumulated prediction error.

Fig. 1. Comparison of classical Kalman filters with position-stack estimators.

A more principled solution is to regard the regression coefficients themselves as latent states, updating them in real time alongside the kinematic variables. Accordingly, we extend the state vector to include not only these coefficients but, when appropriate, the full set of weights of a neural network approximation of the target dynamics. These augmented states are then estimated online using a NNSSM.

III. Neural Network State-Space Model

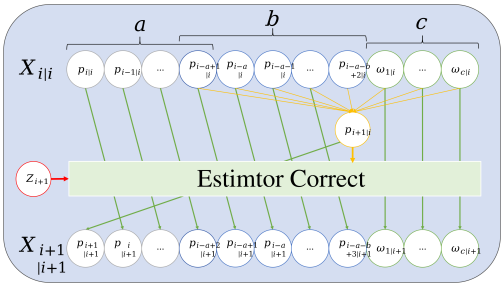


Fig. 2. Neural network state-space model.

Based on the concept of treating neural network weights as estimated states, a new state-space model to achieve state estimation for maneuverable targets is introduced as (10) and Fig. 2.

$$\begin{aligned}\mathbf{x}_{i+1} &= f_i(\mathbf{x}_i, \mathbf{u}_i) = f_i\big(\big[p_i, \ldots, p_{i-a+1}, \\ &\qquad p_{i-a-b+2}, w_1, \ldots, w_c\big]^{\mathsf{T}}, \mathbf{u}_i\big) \\ &= \big[p_{i+1}, p_i, \ldots, p_{i-a-b+3}, w_1, \ldots, w_c\big]^{\mathsf{T}} + \mathbf{u}_i \\ \mathbf{z}_i &= h_i(\mathbf{x}_i, \mathbf{v}_i) = p_i + \mathbf{v}_i\end{aligned} \tag{10}$$

In the proposed model, the state vector comprises $(a-1)+b+c$ elements in total, namely $a+b-1$ historical position states and c weight parameters. Here:

- a is the prediction horizon in discrete steps. For example, if the sampling interval is $T = 0.01$s and a 0.1s ahead prediction is required, then $a = 10$;
- b is the number of input-layer nodes of the neural network being simulated;
- c is the total number of weight parameters in that network.

These $(a-1)+b+c$ states are jointly estimated online via the NNSSM.

Let

$$\begin{aligned}\mathbf{P}_{i,[n:m]} &\triangleq \big[p_{i-n}, p_{i-n-1}, \ldots, p_{i-m}\big] \\ \mathbf{W} &\triangleq \big[w_1, w_2, \ldots, w_c\big],\end{aligned} \tag{11}$$

and denote by $f_{\mathrm{NN}}(\cdot)$ the neural network mapping from the previous state to the new predicted position p_{i+1}. A more detailed state-space model can then be written as

$$\begin{aligned}\mathbf{x}_{i+1} =& f_i(\mathbf{x}_i, \mathbf{u}_i) \\ =& \big[f_{\mathrm{NN}}\big(\mathbf{P}_{i,\,[(a-1):(a+b-2)]},\ \mathbf{W}\big), \\ & \mathbf{P}_{i,\,[0:(a+b-3)]},\ \mathbf{W}\big]^{\mathsf{T}} + \mathbf{u}_i \\ =& \big[p_{i+1},\ \mathbf{P}_{i,\,[0:(a+b-3)]},\ \mathbf{W}\big]^{\mathsf{T}}, \\ \mathbf{z}_i =& p_i + \mathbf{v}_i.\end{aligned} \tag{12}$$

The a-frame-ahead prediction function is therefore given by

$$p_{i+a} \;=\; f_{\mathrm{NN}}\big(\mathbf{P}_{i,\,[0:(b-1)]},\ \mathbf{W}\big). \tag{13}$$

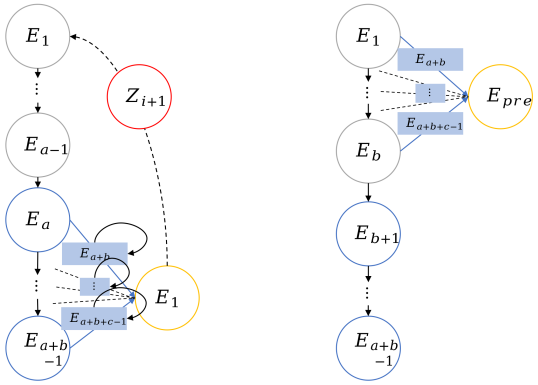


Fig. 3. Simplest neural network state-space model: weighted-sum iteration.

Fig. 3 illustrates the iterative structure of the simplest neural network—a single weighted-sum node—whose architecture defines the specific form of $f_{\mathrm{NN}}(\cdot)$. Let E_j denote the j-th component of the estimated

state vector $\mathbf{x}_i$. The corresponding state-space representation is

$$\begin{aligned}\mathbf{x}_{i+1} &= f_i\big(\mathbf{x}_i, \mathbf{u}_i\big)\\ &= \Big[\mathbf{P}_{i,[(a-1):(a+b-2)]}\,\mathbf{W}^{\mathsf{T}},\ \mathbf{P}_{i,[0:(a+b-3)]},\ \mathbf{W}\Big]^{\mathsf{T}} + \mathbf{u}_i\\ &= \big[p_{i+1},\ \mathbf{P}_{i,[0:a+b-3]},\ \mathbf{W}\big]^{\mathsf{T}},\\ \mathbf{z}_i &= p_i + \mathbf{v}_i.\end{aligned} \tag{14}$$

The needed position prediction formula is

$$\begin{aligned}E_{\text{pre}} = p_{i+a} &= f_{\text{NN}}\big(\mathbf{P}_{i,\,[0:(b-1)]},\ \mathbf{W}\big)\\ &= \mathbf{P}_{i,[0:(b-1)]}\,\mathbf{W}^{\mathsf{T}}.\end{aligned} \tag{15}$$

The weighted sum in (14) furnishes the prior one-step prediction E_1 in $\mathbf{x}_{i+1|i}$. After the new observation $\mathbf{z}_{i+1}$ is received, the estimator updates every state component E_j to obtain the posterior estimate $\mathbf{x}_{i+1|i+1}$.

For an a-frame look-ahead, the algorithm (15) uses the most recent b position estimates $E_1, \ldots, E_b$ and the current weight vector $E_{a+b}, \ldots, E_{a+b+c-1}$ into the same weighted-sum relation, producing the required multi-frame forecast E_{pre}.

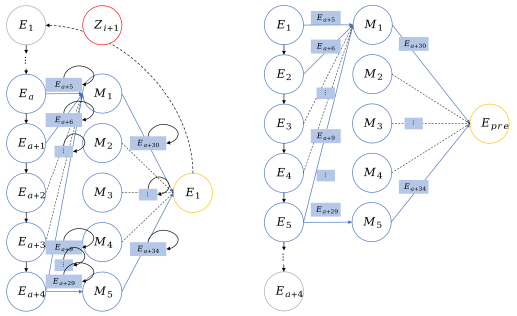


Fig. 4. Simple neural network state-space model: 5-5-1 architecture.

Although the weighted-sum model provides a conceptual bridge to neural networks, it lacks the representational richness of even a modest multilayer perceptron such as the 5-5-1 architecture shown in Fig. 4, which depicts a simple fully connected feedforward network with five input nodes, one hidden layer of five neurons, and a single output node (a "5-5-1" architecture).

To perform an a-frame-ahead prediction, the augmented state vector must include $a+4$ past positions plus the network's weight parameters, for a total of $a + 34$ estimated states. Here, M_j denotes the activation of the jth hidden neuron. The mapping $f_{\text{NN}}(\cdot)$ in (12) is a nonlinear function implemented by a multi-layer recurrent architecture.

The precise layer sizes, connectivity pattern, and choice of activation functions should be determined based on the target application and available computational resources.

IV. Implementation of the Neural Network State-Space Model

Having formulated the NNSSM, we now describe its implementation with a classical Unscented Kalman Estimator, hereafter abbreviated NNSSE-UKE.

1. Parameter Initialization: For an n-dimensional state $\hat{\mathbf{x}}_{i|i}$ choose the spread α, the secondary scaling β, and the scaling factor κ. Define

$$\lambda = \alpha^2\,(n + \kappa) - n. \tag{16}$$

Initialise the state $\hat{\mathbf{x}}_0$, the a-posteriori covariance $\mathbf{\Pi}_{0|0}$, and the noise covariances $\mathbf{Q}$ (process) and $\mathbf{R}$ (measurement).

2. Sigma-Point Generation (Posterior State): With $\hat{\mathbf{x}}_{i|i}$ and $\mathbf{\Pi}_{i|i}$ form $2n+1$ sigma points:

$$\begin{aligned}&\mathbf{X}_0 = \hat{\mathbf{x}}_{i|i},\\ &\mathbf{X}_l = \hat{\mathbf{x}}_{i|i} \pm \big(\sqrt{(n+\lambda)\,\mathbf{\Pi}_{i|i}}\big)_l, \quad l = 1, \ldots, n.\end{aligned} \tag{17}$$

Weights for the state mean and covariance are

$$\begin{aligned}&w_S^{(0)} = \tfrac{\lambda}{n+\lambda}, && w_\Pi^{(0)} = w_S^{(0)} + 1 - \alpha^2 - \beta,\\ &w_S^{(l)} = w_\Pi^{(l)} = \tfrac{1}{2(n+\lambda)}, && l = 1, \ldots, 2n.\end{aligned} \tag{18}$$

3. Propagation through the State Equation: Propagate every sigma point through $f_i(\cdot)$ (see Eq. (12)):

$$\hat{\mathbf{X}}_l = f_i\big(\mathbf{X}_l\big), \qquad l = 0, \ldots, 2n. \tag{19}$$

4. Time Update:

$$\begin{aligned}\hat{\mathbf{x}}_{i+1|i} &= \sum_{l=0}^{2n} w_S^{(l)}\,\hat{\mathbf{X}}_l,\\ \Pi_{i+1|i} &= \sum_{l=0}^{2n} w_\Pi^{(l)}\big(\hat{\mathbf{X}}_l - \hat{\mathbf{x}}_{i+1|i}\big)\big(\hat{\mathbf{X}}_l - \hat{\mathbf{x}}_{i+1|i}\big)^{\mathsf{T}} + \mathbf{Q}.\end{aligned} \tag{20}$$

5. Sigma-Point Generation (Prior State) and Observation: Generate $\tilde{\mathbf{X}}_l$ from $\hat{\mathbf{x}}_{i+1|i}, \mathbf{\Pi}_{i+1|i}$ using (17); pass them through the measurement model $h(\cdot)$:

$$\mathbf{Z}_l = h\big(\tilde{\mathbf{X}}_l\big). \tag{21}$$

Compute

$$\begin{aligned}\hat{\mathbf{z}}_{i+1} &= \sum_{l=0}^{2n} w_S^{(l)}\mathbf{Z}_l,\\ \mathbf{\Pi}_{zz} &= \sum_{l=0}^{2n} w_{\mathbf{\Pi}}^{(l)}\big(\mathbf{Z}_l - \hat{\mathbf{z}}_{i+1}\big)\big(\mathbf{Z}_l - \hat{\mathbf{z}}_{i+1}\big)^{\mathsf{T}} + \mathbf{R},\\ \mathbf{\Pi}_{xz} &= \sum_{l=0}^{2n} w_{\mathbf{\Pi}}^{(l)}\big(\tilde{\mathbf{X}}_l - \hat{\mathbf{x}}_{i+1|i}\big)\big(\mathbf{Z}_l - \hat{\mathbf{Z}}_{i+1}\big)^{\mathsf{T}}.\end{aligned} \tag{22}$$

6. Measurement Update:

$$\begin{aligned}\mathbf{K} &= \mathbf{\Pi}_{xz}\,\mathbf{\Pi}_{zz}^{-1},\\ \hat{\mathbf{x}}_{i+1|i+1} &= \hat{\mathbf{x}}_{i+1|i} + \mathbf{K}\big(\mathbf{z}_{i+1} - \hat{\mathbf{z}}_{i+1}\big),\\ \mathbf{\Pi}_{i+1|i+1} &= \mathbf{\Pi}_{i+1|i} - \mathbf{K}\mathbf{\Pi}_{zz}\mathbf{K}^{\mathsf{T}}.\end{aligned} \tag{23}$$

7. Position Prediction: The updated state $\hat{\mathbf{x}}_{i+1|i+1}$ yields the position forecast $\hat{p}_{i+a}$ (13) and the algorithm returns to Step 2 for the next iteration.

V. Simulation and Experiment

The proposed NNSSM can be instantiated with diverse nonlinear estimators. In this section, we demonstrate its generality by constructing three NNSSEs based on representative EKE, UKE, and PE. By choosing prediction horizon $a = 3$, input dimension $b = 25$, and weight count $c = 25$, we obtain the variants NNSSE-EKE, NNSSE-UKE, and NNSSE-PE, all derived from (14).

To illustrate NNSSM's architectural flexibility, we fix $a = 3$ and employ UKE to realise estimators for different network topologies:

- NNSSE-5-5-1: 5 input nodes, one hidden layer of 5 neurons, and one output node (see Fig. 4);
- NNSSE-10-10-1: 10 input nodes, one hidden layer of 10 neurons, and one output node;
- NNSSE-Tanh: the 5-5-1 architecture with Tanh activations in the hidden layer;
- NNSSE-5-5-5-1: 5 input nodes, two hidden layers of 5 neurons each, and one output node.

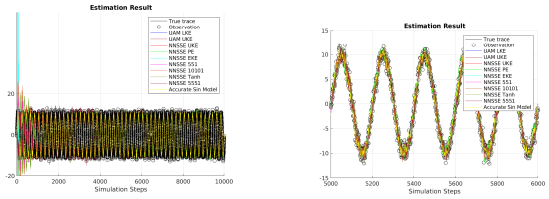
These configurations demonstrate the algorithm's ability to simulate and estimate a wide range of neural network structures.

In our experimental study, we also train established neural models—including RNN, LSTM, GRU, TCN, NeuralODE, Transformer, and an Online Transformer (trained adaptively at runtime)—to benchmark the adaptability and rapid learning capability of NNSSE against these baselines. All of these networks employ an input layer of 25 historical position values and between two and four hidden layers; their precise architectures are detailed in the code repository accompanying this paper.

A. Simulation

A noisy sine wave is used to emulate a manoeuvring target. The sensor sampling rate is 200 Hz with a latency of 0.15 s (three-frame prediction); the ground-truth motion is a 1 s sine with amplitude 10, and the observation noise is zero-mean Gaussian with unit covariance.

For comparison, a uniformly accelerated model (UAM) is estimated by both a linear Kalman estimator (UAM-LKE) and UKE (UAM-UKE). An Accurate-Sin-Model reference uses the exact sine model and LKE.



(a) True vs. estimated trajectories over the full simulation horizon.

(b) Estimator predictions during the midle phase.

Fig. 5. Comparison of classical Kalman filters with position-stack estimators.

Fig. 5a-5b show the trajectories and estimation errors. NNSSE-EKE exhibits large transients in the first few dozen steps, whereas NNSSE-10-10-1 requires more data to converge owing to its larger parameter set.

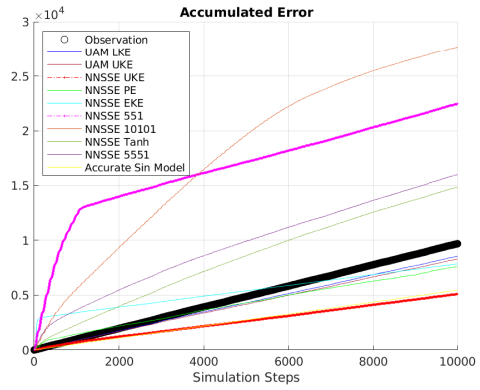


Fig. 6. Accumulated prediction error for all estimators over 10000 steps.

TABLE I
Accumulated error ($\times 10^4$) and run time for 10000 steps. Maximum values are red, minima green.

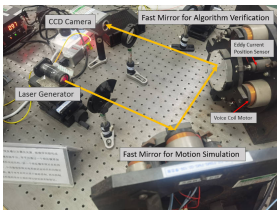
Estimator	Steps 1-10000	Steps 8000-10000	Time (s)
UAM-LKE	0.8554	0.1681	0.0718
UAM-UKE	0.8303	0.1633	0.2908
Accurate-Sin (ideal)	0.5465	0.1105	0.0687
NNSSE-UKE	0.5104	0.0985	7.0647
NNSSE-PE	0.7577	0.1327	22.8345
NNSSE-EKE	0.7852	0.0975	0.5203
NNSSE-5-5-1	2.2451	0.2137	6.7968
NNSSE-10-10-1	2.7693	0.2209	35.0905
NNSSE-Tanh	1.4850	0.2280	7.2935
NNSSE-5-5-5-1	1.6003	0.2353	16.6581

Fig. 6 plots the error after step 200, when all NNSSEs reach steady performance. NNSSE-UKE, NNSSE-PE, and NNSSE-EKE all surpass

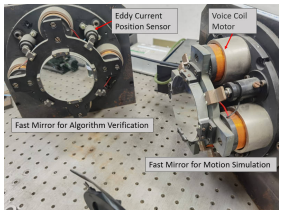
their UAM-based counterparts and even outperform the Accurate-Sin-Model reference despite having no model priors.

Table I summarises accumulated error and computation time. NNSSEs incur higher cost because of the enlarged state, yet the longest run (35.09 s for 10000 steps) remains practical. Among them, NNSSE-EKE is the most economical but suffers from early instability and Jacobian requirements, making UKE the preferred implementation.

B. Dual-Mirror Tracking Experiment



(a) Optical layout



(b) Mechanical assembly

Fig. 7. Dual fast mirror experimental platform.

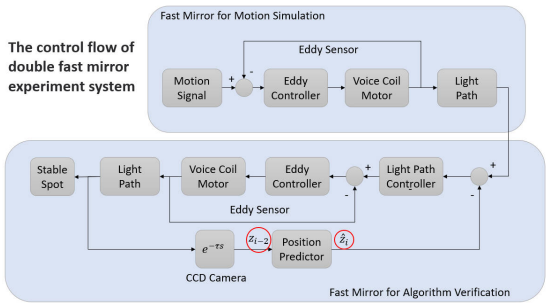
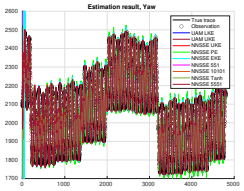


Fig. 8. Control flow of the dual-reflection mirror platform. A three-frame prediction compensates for the CCD latency.

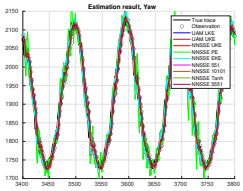
The proposed NNSSEs were further validated on a dual-reflection mirror platform. As depicted in Fig. 7, the bench comprises a laser source, a CCD camera, a fast steering mirror that emulates target motion, and a validation mirror for closed-loop testing. Each steering mirror is driven by four voice-coil actuators and monitored by four eddy-current displacement sensors.

The control objective is to maintain the laser spot at the centre of the CCD. Although the camera samples at 200 Hz, a 0.015 s latency exists, so a three-frame-ahead prediction is inserted in the feedback loop (Fig. 8). Extensive position sequences were first collected to train all baseline neural networks. A new sequences from the same source were then fed to both the trained networks and the NNSSEs for a fair comparison.

a) Overall tracking performance.: Fig. 9 shows the measured spot trajectory together with the estimates produced by the different NNSSEs; the right panel magnifies the central segment.



(a) Full record.



(b) Enlarged central segment.

Fig. 9. Spot trajectory and NNSSE estimates.

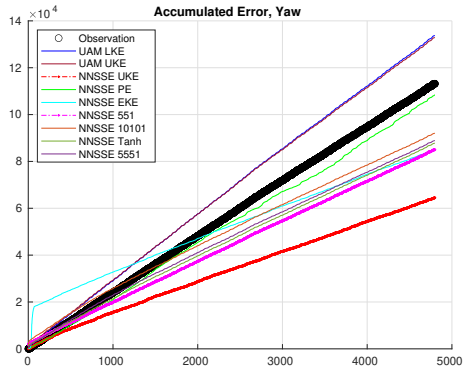


Fig. 10. Cumulative tracking error of competing estimators.

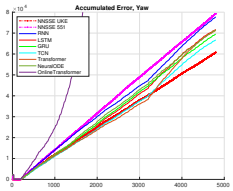
Cumulative-error curves are plotted in Fig. 10. Without artificially injected Gaussian noise in simulation, every NNSSE surpasses its UAM-based counterpart; quantitative results are summarised in Table II.

TABLE II
Dual-mirror experiment: cumulative error ($\times 10^5$). Maximum values are red, minima green.

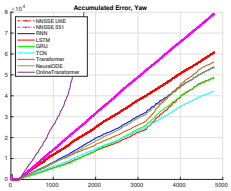
Estimator	Steps 1-4799	Steps 2000-4799	Time (s)
UAM-LKE	1.3374	0.7614	0.0357
UAM-UKE	1.3289	0.7533	0.1375
NNSSE-UKE	0.6450	0.3585	3.5825
NNSSE-PE	1.0842	0.6297	10.3143
NNSSE-EKE	0.8509	0.3802	0.2624
NNSSE-5-5-1	0.8504	0.4768	3.3630
NNSSE-10-10-1	0.9206	0.4805	16.3013
NNSSE-Tanh	0.8781	0.4812	3.4833
NNSSE-5-5-5-1	0.8892	0.4783	7.7643

b) Comparison with neural baselines.: We evaluate two training regimes: (i) limited training (10

epochs for most networks; 50 epochs for the Transformer) and (ii) full training (200+ epochs on 10000 samples). Fig. 11(a) and Fig. 11(b) show the cumulative prediction error under each regime, and Table III summarizes the error measured from step 200 onward.



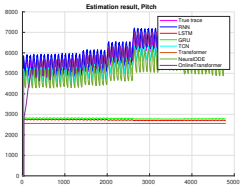
(a) Limited training



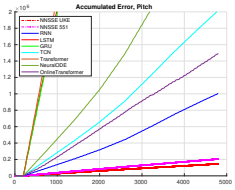
(b) Full training

Fig. 11. Cumulative prediction error of neural baselines under (a) limited and (b) full training regimes.

The NNSSEs attain prediction accuracy comparable to that of the neural baselines trained for only a few epochs. Although they lag behind the fully-trained networks in the interval from step 200 to step 4799, all NNSSE variants converge rapidly: after roughly 2000 learning steps their performance rivals that of the fully-trained models, falling significantly short only of the TCN.



(a) Spot trajectory and estimation



(b) Cumulative prediction error of neural baseline

Fig. 12. Estiamte pitch trajectory using yaw-trained networks.

To further test generalisation, we applied the yaw-trained networks to a new data set in the pitch axis. The pretrained RNN, LSTM, GRU, TCN, NeuralODE, Transformer, OnlineTransformer were unable to deliver useful forecasts. By contrast, all NNSSEs quickly adapted to the new motion pattern and maintained high accuracy. Fig. 12(a) shows the pitch trajectory and estimation result, and Fig. 12(b) plots the corresponding cumulative errors.

C. Drone-Tracking Experiment

The efficacy of the proposed approach was further evaluated on a real unmanned aerial vehicle trajectory.

Fig. 15(a) plots the yaw trajectory together with the estimates delivered by the various NNSSEs.

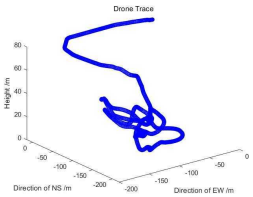
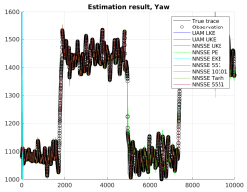
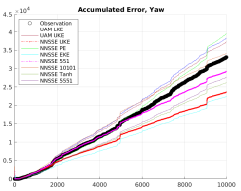


Fig. 13. Drone and part of its trajectory.



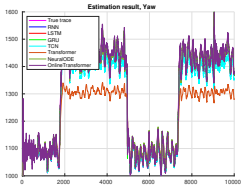
(a) Measured and estimated yaw trajectories



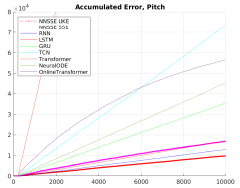
(b) Cumulative error of NNSSE variants

Fig. 14. Performance of NNSSEs on the UAV yaw data.

Unlike the quasi-sinusoidal laboratory path, the UAV executes complex, non-cooperative manoeuvres; the sharp 180° reversals arise from the zenith-crossing phenomenon (i.e., the target passes directly overhead and the yaw changes sign). Fig. 15(b) shows the corresponding cumulative errors: after a short learning phase, most NNSSEs attains a markedly lower error than the UAM methods baseline.



(a) Fully-trained neural predictions



(b) Cumulative error of neural baselines

Fig. 15. Neural baselines on the UAV yaw data.

For a fair comparison with data-driven baselines, we trained every neural model on 20000 historical samples. Fig. 15(a) presents the predictions obtained after full training, and Fig. 15(b) reports the resulting cumulative errors.

Benefiting from the rapid-learning capability of the NNSSM, NNSSE-UKE surpasses every pre-trained neural network in prediction accuracy. We anticipate that further optimisation of the network architecture and hyper-parameters will yield even higher precision.

TABLE III
Cumulative error ($\times 10^5$) for limited and full training. Maximum values are red, minima green.

	Limited Training		Full Training	
Method	Steps 200–4799	Steps 2000–4799	Steps 200–4799	Steps 2000–4799
NNSSE-UKE	0.6076	0.3585	0.6076	0.3585
NNSSE-5-5-1	0.7929	0.4768	0.7929	0.4768
RNN	0.7770	0.4759	0.5392	0.3416
LSTM	0.7172	0.4588	0.4882	0.3521
GRU	0.6947	0.4289	0.4858	0.3487
TCN	0.6673	0.4259	0.4237	0.2771
Transformer	0.7137	0.4768	0.5617	0.3968
NeuralODE	0.7152	0.4390	0.5356	0.3527
Online Transformer	4.6542	3.3732	4.6246	3.4425

VI. Conclusion

This paper has addressed the problem of tracking non-cooperative targets by first validating the feasibility of using a pure position sequence as the state vector in a classical state-space formulation. Building on this result, we introduced the Neural Network State-Space Model, which needn't target's state-space model and treats both the target's recent positions and every weight of a surrogate neural network as latent states. The resulting nonlinear model can be realised with different nonlinear estimator; in extensive trials, UKE provided the best overall performance, and its full algorithmic structure has been presented in detail.

To demonstrate generality, NNSSM was further instantiated with EKE, UKE, and PE, producing three baseline NNSSEs that outperform constant-acceleration Kalman filters on noisy sine trajectories, a dual-reflection mirror platform, and real UAV flights. Additional experiments showed that NNSSM can emulate a variety of network topologies (5-5-1, 10-10-1, 5-5-5-1, Tanh activations, etc.) simply by adjusting the augmented state. When compared with fully pre-trained RNN, LSTM, GRU, TCN, NeuralODE, Transformer, and Online-Transformer baselines, the proposed NNSSEs achieved comparable—or, after a short on-line learning phase, superior—accuracy while retaining the rapid adaptation capability of recursive estimators.

All source code, data and more result are openly available at https://github.com/ShineMinxing/PaperNNSSE.git, facilitating reproduction and further development.

Limitations and outlook:

1) Computational complexity. The Kalman update scales as $O(n^3)$ with the number of augmented states; simulating very deep or wide networks therefore becomes costly. Efficient sparsification or reduced-rank techniques are worth investigating.
2) Model design. The current NNSSM uses only fully connected layers and simple activations. Borrowing advanced architectural motifs (e.g. residual connections or attention) from modern deep learning may improve accuracy without enlarging the state excessively.
3) Estimator diversity. We evaluated the three foundational filters (EKE, UKE, PE). Incorporating more recent Bayesian filters or adaptive variants may yield further gains, particularly for high-dimensional weight vectors.

We believe that NNSSM enlarges the design space of state estimation by seamlessly blending data-driven neural representations with principled Bayesian filtering, thereby offering a practical alternative when explicit nonlinear modelling is intractable.

References

[1] Y. Bar-Shalom, T. E. Fortmann, and P. G. Cable, "Tracking and Data Association," The Journal of the Acoustical Society of America, vol. 87, no. 2, pp. 918–919, Feb. 1990.

[2] Xia Wenqiang, He Qiunong, Duan Qianwen, Zhou Xi, Deng Jiuqiang, and Mao Yao, "Equivalent acceleration feedforward based on sensor optimization and robust prediction," Opto-Electronic Engineering, vol. 48, no. 11, 2021.

[3] R. F. Stengel, Optimal Control and Estimation. Courier Corporation, 1994.

[4] Y. Kaymak, R. Rojas-Cessa, J. Feng, N. Ansari, M. Zhou, and T. Zhang, "A Survey on Acquisition, Tracking, and Pointing Mechanisms for Mobile Free-Space Optical Communications," IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, vol. 20, no. 2, pp. 1104–1123, 2018.

[5] J. Deng, W. Xue, W. Liang, X. Zhou, and Y. Mao, "On adjustable and lossless suppression to disturbances and uncertainties for nonminimum-phase laser pointing system," ISA TRANSACTIONS, vol. 136, pp. 727–741, May 2023.

[6] J. Chen, H. Guo, P. Liu, and Y. Wang, "The summary on atmospheric disturbance problems in the motion imaging of high resolution earth observation system," in Proceedings of 2011 International Conference on Electronic

& Mechanical Engineering and Information Technology, vol. 8, Aug. 2011, pp. 3999–4003.
[7] Z. Li, M. Sun, Q. Duan, and Y. Mao, "Robust State Estimation for Uncertain Discrete Linear Systems with Delayed Measurements," MATHEMATICS, vol. 10, no. 9, p. 1365, May 2022.
[8] M. Sun, Q. Duan, W. Xia, Q. Bao, and Y. Mao, "Multiple adaptive factors based interacting multiple model estimator," IET Control Theory & Applications, vol. 18, no. 8, pp. 1059–1069, 2024.
[9] R. E. Kalman, "A new approach to linear filtering and prediction problems," Journal of Basic Engineering, vol. 82, no. 1, pp. 35–45, Mar. 1960.
[10] M. Betke and L. Gurvits, "Mobile robot localization using landmarks," IEEE Transactions on Robotics and Automation, vol. 13, no. 2, pp. 251–263, Apr. 1997.
[11] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, Estimation with Applications to Tracking and Navigation: Theory Algorithms and Software. John Wiley & Sons, 2001.
[12] M. S. Grewal and A. P. Andrews, Kalman Filtering : Theory and Practice Using MATLAB®, fourth edition ed. Hoboken, New Jersey: Wiley Hoboken, New Jersey, 2015.
[13] T. Qin, P. Li, and S. Shen, "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator," IEEE Transactions on Robotics, vol. 34, no. 4, pp. 1004–1020, Aug. 2018.
[14] S. J. Julier and J. K. Uhlmann, "A new extension of the kalman filter to nonlinear systems." Bellingham, WA: Society of Photo-Optical Instrumentation Engineers (SPIE Proceedings. Vol. 3068), 1997, pp. 182–193.
[15] E. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373), Oct. 2000, pp. 153–158.
[16] R. van der Merwe, A. Doucet, N. de Freitas, and E. Wan, "The Unscented Particle Filter," in Advances in Neural Information Processing Systems, vol. 13. MIT Press, 2000.
[17] S. Julier and J. Uhlmann, "Unscented filtering and nonlinear estimation," Proceedings of the IEEE, vol. 92, no. 3, pp. 401–422, Mar. 2004.
[18] J. Zhao and L. Mili, "Robust Unscented Kalman Filter for Power System Dynamic State Estimation With Unknown Noise Statistics," IEEE Transactions on Smart Grid, vol. 10, no. 2, pp. 1215–1224, Mar. 2019.
[19] R. Xiong, J. Tian, W. Shen, and F. Sun, "A Novel Fractional Order Model for State of Charge Estimation in Lithium Ion Batteries," IEEE Transactions on Vehicular Technology, vol. 68, no. 5, pp. 4130–4139, May 2019.
[20] D. Feng, C. Wang, C. He, Y. Zhuang, and X.-G. Xia, "Kalman-Filter-Based Integration of IMU and UWB for High-Accuracy Indoor Positioning and Navigation," IEEE Internet of Things Journal, vol. 7, no. 4, pp. 3133–3146, Apr. 2020.
[21] N. Gordon, D. Salmond, and A. Smith, "Novel-Approach to Nonlinear Non-Gaussian Bayesian State Estimation," IEE PROCEEDINGS-F RADAR AND SIGNAL PROCESSING, vol. 140, no. 2, pp. 107–113, Apr. 1993.
[22] D. Fox, S. Thrun, W. Burgard, and F. Dellaert, "Particle Filters for Mobile Robot Localization," in Sequential Monte Carlo Methods in Practice, ser. Statistics for Engineering and Information Science, A. Doucet, N. de Freitas, and N. Gordon, Eds. New York, NY: Springer, 2001, pp. 401–428.
[23] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," IEEE Transactions on Signal Processing, vol. 50, no. 2, pp. 174–188, Feb. 2002.
[24] B. Ristic, S. Arulampalam, and N. Gordon, Beyond the Kalman Filter: Particle Filters for Tracking Applications. Artech house, 2003.
[25] S. S. Ge and J. Wang, "Robust adaptive tracking for time-varying uncertain nonlinear systems with unknown control coefficients," IEEE TRANSACTIONS ON AUTOMATIC CONTROL, vol. 48, no. 8, pp. 1463–1469, Aug. 2003.
[26] M. Sun, Y. Mao, and H. Liu, "A robust state estimator with adaptive factor," IEEE ACCESS, vol. 8, pp. 144 514–144 521, 2020.
[27] N. Aloysius and M. Geetha, "A review on deep convolutional neural networks," in 2017 International Conference on Communication and Signal Processing (ICCSP), Apr. 2017, pp. 0588–0592.
[28] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," Heliyon, vol. 4, no. 11, p. e00938, Nov. 2018.
[29] S. Feofilov and D. Khapkin, "Application of Recurrent Neural Networks in Closed Loop Tracking Systems for Controlling Essentially Nonlinear Objects," in 2021 3rd International Conference on Control Systems, Mathematical Modeling, Automation and Energy Efficiency (SUMMA), Nov. 2021, pp. 467–472.
[30] A. Kovacs, B. Bogdandy, and Z. Toth, "Predict Stock Market Prices with Recurrent Neural Networks using NASDAQ Data Stream," in 2021 IEEE 15th International Symposium on Applied Computational Intelligence and Informatics (SACI), May 2021, pp. 000 449–000 454.
[31] X. Fang, L. Wang, and K. Zhang, "Adaptive recurrent neural network intelligent sliding mode control of permanent magnet linear synchronous motor," Neural Computing and Applications, vol. 36, no. 1, pp. 349–363, Jan. 2024.
[32] W. Tian, F. Luo, and K. Shen, "PSRUNet: A recurrent neural network for spatiotemporal sequence forecasting based on parallel simple recurrent unit," Machine Vision and Applications, vol. 35, no. 3, p. 55, Apr. 2024.
[33] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities." Proceedings of the National Academy of Sciences, vol. 79, no. 8, pp. 2554–2558, Apr. 1982.
[34] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," Nature, vol. 323, no. 6088, pp. 533–536, Oct. 1986.
[35] M. Jordan, "Serial Order: A Parallel Distributed Processing Approach," in Advances in Psychology. North-Holland, Jan. 1997, vol. 121, pp. 471–495.
[36] J. L. Elman, "Finding Structure in Time," Cognitive Science, vol. 14, no. 2, pp. 179–211, 1990.
[37] H. K. Mahajan, J. Kaur, S. Singh, and S. Banerjee, "Recurrent Neural Network for Estimation of Aerodynamic Parameters," in 2021 8th International Conference on Signal Processing and Integrated Networks (SPIN), Aug. 2021, pp. 150–155.
[38] M. P. Belov, N. Van Lanh, and T. D. Khoa, "State Observer based Elman Recurrent Neural Network for Electric Drive of Optical-Mechanical Complexes," in 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus), Jan. 2021, pp. 802–805.
[39] J. Kaur, H. K. Mahajan, S. Singh, and S. Banerjee, "Recurrent Neural Network for Aircraft Parameter Estimation," in 2021 IEEE 6th International Conference on Computing, Communication and Automation (ICCCA), Dec. 2021, pp. 604–608.
[40] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.
[41] F. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," in 1999 Ninth

International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470), vol. 2, Sep. 1999, pp. 850–855 vol.2.

[42] C. Wang and Y. Fu, "Ship Trajectory Prediction Based on Attention in Bidirectional Recurrent Neural Networks," in 2020 5th International Conference on Information Science, Computer Technology and Transportation (ISCTT), Nov. 2020, pp. 529–533.

[43] A. Ip, L. Irio, and R. Oliveira, "Vehicle Trajectory Prediction based on LSTM Recurrent Neural Networks," in 2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring), Apr. 2021, pp. 1–5.

[44] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," Sep. 2014.

[45] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," Dec. 2014.

[46] A. B. Adege, H.-P. Lin, and L.-C. Wang, "Mobility Predictions for IoT Devices Using Gated Recurrent Unit Network," IEEE Internet of Things Journal, vol. 7, no. 1, pp. 505–517, Jan. 2020.

[47] Y. Gao, J. Tebbe, and A. Zell, "A Model-free Approach to Stroke Learning for Robotic Table Tennis," in 2022 International Joint Conference on Neural Networks (IJCNN), Jul. 2022, pp. 1–8.

[48] D. Guan, N. Ren, K. Wang, Q. Wang, and H. Zhang, "Checkpoint data-driven GCN-GRU vehicle trajectory and traffic flow prediction," Scientific Reports, vol. 14, no. 1, p. 30409, Dec. 2024.

[49] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation Applied to Handwritten Zip Code Recognition," Neural Computation, vol. 1, no. 4, pp. 541–551, Dec. 1989.

[50] M. Liang and X. Hu, "Recurrent Convolutional Neural Network for Object Recognition," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3367–3375.

[51] L. Ma and S. Tian, "A Hybrid CNN-LSTM Model for Aircraft 4D Trajectory Prediction," IEEE Access, vol. 8, pp. 134 668–134 680, 2020.

[52] X. Mo, Y. Xing, and C. Lv, "Interaction-Aware Trajectory Prediction of Connected Vehicles using CNN-LSTM Networks," in IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society, Oct. 2020, pp. 5057–5062.

[53] D. Zhao and J. Oh, "Noticing Motion Patterns: A Temporal CNN With a Novel Convolution Operator for Human Trajectory Prediction," IEEE Robotics and Automation Letters, vol. 6, no. 2, pp. 628–634, Apr. 2021.

[54] G. Yang, G. Wang, Y. Li, and W. Yu, "ACBL:Attentive CNN-BiLSTM Model For Trajectory Prediction," in 2024 43rd Chinese Control Conference (CCC), Jul. 2024, pp. 8243–8248.

[55] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal Convolutional Networks for Action Segmentation and Detection," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 156–165.

[56] H. Chen, "STGCN For Modeling Vehicle Trajectory in Highway Scenario," in 2020 5th International Conference on Mechanical, Control and Computer Engineering (ICMCCE), Dec. 2020, pp. 1115–1118.

[57] K. Zhang, C. Zhang, Q. Zhu, Y. Gu, and C. Liu, "Ship Trajectory D-TCN Prediction Method Based on Satellite Remote Sensing Positioning Data," in 2022 China Automation Congress (CAC), Nov. 2022, pp. 1419–1424.

[58] H. Wang, X. Shi, and M. Sun, "A multimodal vehicle trajectory prediction method with spatio-temporal feature fusion," in 2023 7th International Conference on Transportation Information and Safety (ICTIS), Aug. 2023, pp. 2053–2058.

[59] H. Yuan, J. Zhang, L. Zhang, Z. Zhang, and Z. Wang, "Vehicle Trajectory Prediction Based on Posterior Distributions Fitting and TCN-Transformer," IEEE Transactions on Transportation Electrification, vol. 10, no. 3, pp. 7160–7173, Sep. 2024.

[60] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural Ordinary Differential Equations," in Advances in Neural Information Processing Systems, vol. 31. Curran Associates, Inc., 2018.

[61] S. Park, M. Schwartz, and J. Park, "NODEIK: Solving Inverse Kinematics with Neural Ordinary Differential Equations for Path Planning," in 2022 22nd International Conference on Control, Automation and Systems (ICCAS), Nov. 2022, pp. 944–949.

[62] Z. Chang, S. Liu, R. Qiu, S. Song, Z. Cai, and G. Tu, "Time-aware neural ordinary differential equations for incomplete time series modeling," The Journal of Supercomputing, vol. 79, no. 16, pp. 18 699–18 727, Nov. 2023.

[63] K. Ke, J. Yang, Y. Liu, M. Chen, X. Wei, and X. Tang, "Social Lode: Human Trajectory Prediction with Latent Odes," in ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Apr. 2024, pp. 5360–5364.

[64] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in Advances in Neural Information Processing Systems, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[65] J. Qiu, L. Chen, X. Gu, F. P.-W. Lo, Y.-Y. Tsai, J. Sun, J. Liu, and B. Lo, "Egocentric Human Trajectory Forecasting With a Wearable Camera and Multi-Modal Fusion," IEEE Robotics and Automation Letters, vol. 7, no. 4, pp. 8799–8806, Oct. 2022.

[66] Y. Zhang, G. Li, X.-P. Zhang, and Y. He, "Transformer-based tracking Network for Maneuvering Targets," in ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Jun. 2023, pp. 1–5.

[67] T. Salzmann, H.-T. L. Chiang, M. Ryll, D. Sadigh, C. Parada, and A. Bewley, "Robots That Can See: Leveraging Human Pose for Trajectory Prediction," IEEE Robotics and Automation Letters, vol. 8, no. 11, pp. 7090–7097, Nov. 2023.

[68] J. Peng, Y. Zhou, and P. Y. Mok, "KTPFormer: Kinematics and Trajectory Prior Knowledge-Enhanced Transformer for 3D Human Pose Estimation," in 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 2024, pp. 1123–1132.

[69] L. Ouyang, T. Zheng, X. Xu, H. Zhang, and Z. Zhang, "A Transformer-based Trajectory Prediction Method for a Near-space Flight Vehicle," in 2024 43rd Chinese Control Conference (CCC), Jul. 2024, pp. 4000–4005.

[70] S. Singhal and L. Wu, "Training Multilayer Perceptrons with the Extended Kalman Algorithm," in Advances in Neural Information Processing Systems, vol. 1. Morgan-Kaufmann, 1988.

[71] G. Puskorius and L. Feldkamp, "Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks," IEEE Transactions on Neural Networks, vol. 5, no. 2, pp. 279–297, Mar. 1994.

[72] J. A. Pérez-Ortiz, F. A. Gers, D. Eck, and H. Schmidhuber, "Kalman filters improve LSTM network performance in problems unsolvable by traditional recurrent nets," NEURAL NETWORKS, vol. 16, no. 2, pp. 241–250, Mar. 2003.

[73] Z. Darojah and E. S. Ningrum, "The extended Kalman filter algorithm for improving neural network performance in voice recognition classification," in 2016 International

Seminar on Intelligent Technology and Its Applications (ISITIA), 2016, pp. 225–230.
[74] Z. Darojah, E. S. Ningrum, and D. S. Purnomo, "The training of feedforward neural network using the unscented Kalman filter for voice classification application," in 2017 International Electronics Symposium on Knowledge Creation and Intelligent Computing (IES-KCIC), Sep. 2017, pp. 21–25.
[75] M. Nazari and S. Shamshirband, "The particle filter-based back propagation neural network for evapotranspiration estimation," Taylor & Francis, Jul. 2020.
[76] N. Alsadi, S. A. Gadsden, and J. Yawney, "Intelligent estimation: A review of theory, applications, and recent advances," DIGITAL SIGNAL PROCESSING, vol. 135, p. 103966, Apr. 2023.