Time-Varying Direction-of-Arrival Estimation Exploiting Mamba Network

Saidur R. Pavel [1], Mirza A. Haider [1], Yimin D. Zhang [1]
Yanwu Ding [2], Dan Shen [3], Khanh Pham [4], and Genshe Chen [3]

[1] Department of Electrical and Computer Engineering, Temple University, Philadelphia, PA 19122, USA
[2] Department of Electrical and Computer Engineering, Wichita State University, Wichita, KS 67260, USA
[3] Intelligent Fusion Technology, Inc., Germantown, MD 20876, USA
[4] Air Force Research Laboratory, Kirtland Air Force Base, NM 87117, USA

Abstract—**Direction-of-arrival (DOA) estimation for moving targets presents a significant challenge in array signal processing. Traditional DOA estimation and tracking methods often encounter limitations due to the infeasibility of acquiring large volumes of stationary data and performing subspace-based processing over many snapshots, and lead to high computational costs. Recently, deep learning techniques have been effectively applied in DOA estimation, owing to their reduced complexity during inference. In this paper, we propose the use of Mamba network as a state-space model-based approach to estimate and track DOAs that vary snapshot-by-snapshot. The proposed network is interpretable and hardware-efficient, making it advantageous for training and real-time inference.**

keywords: Time-varying DOA estimation, state-space model, deep learning, Mamba.

I. Introduction

Direction-of-arrival (DOA) estimation is a fundamental technique in array signal processing, used to determine the spatial spectrum of incoming signals. It has diverse applications, including wireless communications, radar, autonomous vehicles, sonar, radio astronomy, and biomedical imaging [1]–[4]. Subspace-based methods, such as MUSIC [5] and ESPRIT [6], are commonly used for DOA estimation due to their low complexity and high accuracy. However, these methods are primarily suited for scenarios where targets remain stationary over an extended time period. For dynamic targets, where DOAs vary on a snapshot-by-snapshot basis, they are difficult to apply or the performance degrades significantly.

Many DOA estimation methods rely on subspace or eigen-based information derived from the eigenvalue decomposition of the covariance matrix or the singular value decomposition of the data matrix. This dependency has prompted the development of subspace-based tracking algorithms tailored for moving targets. One such algorithm is the projection approximation subspace tracking (PAST), an adaptive approach designed for efficient subspace estimation [7]. PAST uses recursive least squares (RLS) to approximate the signal subspace. Enhancements like PAST-ESPRIT [8] and the maximum correntropy criterion PAST-ESPRIT [9] have been introduced for increased robustness, particularly in noisy environments. However, these enhancements come with higher computational demands due to the additional steps for noise compensation and subspace updates.

In [10], a joint DOA estimation and source signal tracking, utilizing a combination of Kalman Filtering and a regularized QR-decomposition-based RLS (QRD-RLS) algorithm is proposed. The method models source signals as autoregressive (AR) processes, using the Kalman filter for dynamic tracking and QRD-RLS for estimating DOAs and AR coefficients. The effectiveness of this method depends on accurate AR modeling of the source signals and requires a high number of snapshots per coherent block for reliable DOA estimation.

The method proposed in [11] leverages a nested array structure alongside an innovative offset compensation technique based on a first-order Taylor expansion. Initially, DOAs are estimated using the discrete Fourier transform (DFT), followed by tracking of DOA changes through offset compensation between consecutive snapshots using a Taylor series approximation. The nested array design, combined with this compensation approach, improves tracking accuracy, particularly in dynamic environments and low SNR conditions, where traditional methods like PAST struggle. However, the approach requires a substantial number of snapshots per coherent interval for reliable estimation, leading to higher computational complexity compared to PAST and its variants.

Recently, machine learning and deep learning techniques have shown great potential in array signal processing, including DOA estimation [12]–[15]. For example, in [16], a U-Net-based fully convolutional neural network (FCNN) is applied for DOA trajectory localization, transforming input power spectra to high-resolution target maps for improved accuracy in DOA tracking. Recurrent neural networks (RNNs) and their variants, such as long short-term memory (LSTM) [17] and gated recurrent units (GRU) [18], have also been utilized for capturing temporal dependencies, making them suitable for tracking DOA changes over time. The transformer architecture [19], leveraging attention mechanisms, has been widely adopted in sequence modeling tasks, often outperforming RNN-based models. Both RNN variants and transformer models are popularly being used in tracking tasks [20]–[22].

Despite their advantages, RNNs and transformers have inherent limitations. RNNs cannot be parallelized due to their sequential nature, posing challenges as sequence length grows. While transformers offer parallelization, their inference complexity increases quadratically with sequence length, unlike

This material is based upon work supported by the Air Force Research Laboratory (AFRL) under Contract No. FA9453-22-C-A127. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the AFRL.

the linear scaling of RNNs. Additionally, RNNs compress information into hidden states, which transformers bypass by directly leveraging the attention mechanism. Recently, state-space modeling (SSM) [23] has gained attention for its training and inference efficiency. SSMs can utilize parallelizable convolutional operations during training and function as recurrent models during inference. However, traditional SSMs are limited by their linear time-invariant (LTI) nature, which can be restrictive in complex, real-world scenarios. To address this, a selectivity mechanism was introduced in Mamba networks [24], enabling selective filtering of irrelevant information. This allows the model to efficiently compress context into a robust state, improving both its efficiency and robustness.

In this paper, we propose a Mamba network-based approach for tracking DOAs over time. Leveraging its efficient state representation, the Mamba network captures and preserves long-term temporal dependencies, which is crucial for tracking DOAs in dynamic scenarios. Unlike conventional approaches that assume coherent blocks or linear changes within each block and are thus unsuitable for fast-moving objects, the proposed approach can detect and track signal DOAs using a single snapshot. Specifically, we consider a scenario where the DOA changes at every snapshot, and the Mamba network is trained on multiple snapshots to learn the temporal relationships in the array data through latent states spanning consecutive snapshots, thereby enabling estimation of the true DOAs at each time step. Once trained, the network can detect the initial DOA from the very first snapshot and track subsequent DOA changes in every new snapshot. Its low computational complexity, particularly during inference, makes it well-suited for real-time applications. Additionally, Mamba's state-space structure enhances interpretability, as it aligns well with physical system models.

Notations: We use lower-case (upper-case) bold characters to describe vectors (matrices), whereas upper-case calligraphic characters are used to describe tensors. In particular, $(\cdot)^{\mathrm{T}}$ and $(\cdot)^{\mathrm{H}}$ respectively denote the transpose and conjugate transpose of a matrix or vector. $\jmath = \sqrt{-1}$ denotes the unit imaginary number. $\boldsymbol{I}_M$ stands for the $M \times M$ identity matrix. In addition, $\mathcal{R}(\cdot)$ and $\mathcal{I}(\cdot)$ denote the real part and imaginary part of a complex number, respectively. Finally, $\|\cdot\|_2$ defines the ℓ_2 norm of a vector.

II. Signal Model

A. Array Signal Model

Consider a D-element uniform linear array (ULA) receiving K uncorrelated, far-field narrowband target signals with time-varying DOAs $\boldsymbol{\theta}_t = [\theta_{1,t}, \theta_{2,t}, \ldots, \theta_{K,t}]^{\mathrm{T}}$ at time snapshot t. The $D \times 1$ signal vector received at the antenna array at time t is expressed as

$$\tilde{\boldsymbol{x}}_t = \sum_{k=1}^{K} s_{k,t}\boldsymbol{a}(\theta_{k,t}) + \boldsymbol{n}_t = \boldsymbol{A}_t\boldsymbol{s}_t + \boldsymbol{n}_t, \tag{1}$$

where

$$\boldsymbol{a}(\theta_{k,t}) = [1, e^{-\jmath\frac{2\pi}{\lambda}d\sin(\theta_{k,t})}, \cdots, e^{-\jmath\frac{2\pi}{\lambda}(D-1)d\sin(\theta_{k,t})}]^{\mathrm{T}} \tag{2}$$

is the $D \times 1$ steering vector associated with the DOA $\theta_{k,t}$, λ is the wavelength of the incident wave, $d = \lambda/2$ is the inter-element spacing, and

$$\boldsymbol{A}_t = [\boldsymbol{a}(\theta_{1,t}), \cdots, \boldsymbol{a}(\theta_{K,t})] \tag{3}$$

is the $D \times K$ array manifold matrix at time t. In addition, $\boldsymbol{s}_t = [s_{1,t}, \cdots, s_{K,t}]^{\mathrm{T}}$ is the $K \times 1$ signal waveform vector and $\boldsymbol{n}_t \sim \mathcal{CN}(\boldsymbol{0}, \sigma_n^2\boldsymbol{I})$ is the $D \times 1$ additive white Gaussian noise (AWGN) vector with noise power σ_n^2, whose elements are assumed to be independent and identically distributed, and are uncorrelated with the target signals.

B. Data Pre-processing

The received signal vector $\tilde{\boldsymbol{x}}_t$ is first pre-processed in a certain order before being fed into the proposed Mamba network. We generate M different training samples, each with distinct initial DOA scenarios, and denote $\tilde{\boldsymbol{x}}_t^{(m)}$ as the received signal vector for the mth training sample. Since many functionalities of deep learning methods cannot efficiently handle complex-valued inputs, we decompose the complex-valued signal into real and imaginary components, resulting in a $2D \times 1$ real-valued vector as

$$\boldsymbol{x}_t^{(m)} = [\mathcal{R}^{\mathrm{T}}(\tilde{\boldsymbol{x}}_t^{(m)}), \quad \mathcal{I}^{\mathrm{T}}(\tilde{\boldsymbol{x}}_t)]^{\mathrm{T}}. \tag{4}$$

Let B denote the batch size for training. We define $\boldsymbol{X}^{(b)} = [\boldsymbol{x}_1^{(b)}, \boldsymbol{x}_2^{(b)}, \cdots, \boldsymbol{x}_T^{(b)}]^{\mathrm{T}} \in \mathbb{R}^{T \times 2D}$, where $\boldsymbol{x}^{(b)}$ represents the received signal from the bth sample of a particular batch, with $b = 1, 2, \cdots, B$. Then, a particular batch of data can be represented as

$$\boldsymbol{\mathcal{X}} = [\boldsymbol{X}^{(1)}, \boldsymbol{X}^{(2)}, \cdots, \boldsymbol{X}^{(B)}]_{\sqcup_1} \in \mathbb{R}^{B \times T \times 2D}, \tag{5}$$

where $[\cdot]_{\sqcup_i}$ denotes tensor concatenation along the ith dimension.

C. Problem Formulation

The objective is to sequentially estimate time-varying DOAs $\boldsymbol{\theta}_t$ for $t = 1, \ldots, T$, where T is the total number of snapshots. That is, given the available $\tilde{\boldsymbol{x}}_t$, we aim to determine

$$\hat{\boldsymbol{\theta}}_t = f(\tilde{\boldsymbol{x}}_t), \tag{6}$$

where $f(\cdot)$ is a mapping function. In this paper, we implement this mapping function using a Mamba network.

III. Mamba-based DOA estimation methodology

In this section, we describe the proposed Mamba network-based method for time-varying DOA estimation in a sequential manner.

A. State-Space Model

We first describe the state-space model (SSM) and selective SSM, which are the core of the Mamba network, in the continuous-time domain. To the notation with the discrete-time model, the continuous-time is denoted as (t).

The DOA estimation problem defined in Eq. (6) can be represented in the context of SSM as

$$\boldsymbol{h}_d'(t) = \boldsymbol{E}_d\boldsymbol{h}(t) + \boldsymbol{f}_d x_d(t) \tag{7a}$$

$$y_d(t) = \boldsymbol{g}_d\boldsymbol{h}_d(t), \tag{7b}$$

where $x_d(t) \in \mathbb{R}$ is a scaler observation corresponding to the dth dimension of $\boldsymbol{x}(t)$, $\boldsymbol{h}_d(t) \in \mathbb{R}^{N \times 1}$ is a latent state representation. The matrix $\boldsymbol{E}_d \in \mathbb{R}^{N \times N}$ represents the state transition, the vector $\boldsymbol{f}_d \in \mathbb{R}^{N \times 1}$ corresponds to the input, and $\boldsymbol{g}_d^{1 \times N}$ is a row vector associated with the output for the dth dimension. The sequence y_d is the estimated output corresponding to the dth dimension. $2D$ independent SSM are

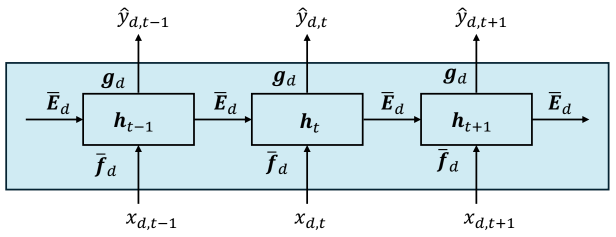


Fig. 1: Unrolling of the SSM.

applied to each dimension of the received signal to obtain the output sequence $\boldsymbol{y}(t) \in \mathbb{R}^{2D\times 1}$. The estimated DOAs $\hat{\boldsymbol{\theta}}$ then can be obtained by a linear projection of $\boldsymbol{y}(t)$.

The linear SSM as described in Eq. (7) may suffer from the vanishing/exploding gradient problem as well as exhibits poor performance. To address this issue and preserve long-range dependencies, it is required to impose a certain structure in matrix $\boldsymbol{E}_d$. To do this, HiPPO theory of continuous-time memorization [25] is exploited. This theory specifies a class of certain matrices $\boldsymbol{E}_d$ that enables the states $\boldsymbol{h}_d(t)$ to memorize the history of the input $x(t)$. The structure of this HiPPO matrix can be defined as [23]

$$\boldsymbol{E}_{d_{n,k}} = -\begin{cases}(2n+1)^{\frac{1}{2}}(2k+1)^{\frac{1}{2}}, & \text{if } n > k,\\ n+1, & \text{if } n = k,\\ 0, & \text{if } n < k.\end{cases} \tag{8}$$

The state space representation is generally inspired by continuous systems that map a function or sequence $x(t)$ to $y(t)$ through the latent state $\boldsymbol{h}(t)$. Since the received signal for the underlying DOA estimation problem is obtained in discrete manner, the SSM is discretized using the zero-order hold (ZOH) technique [26]. In this technique, the continuous parameters $\boldsymbol{E}_d$ and $\boldsymbol{F}_d$ are discretized and are defined as

$$\bar{\boldsymbol{E}}_d = e^{\Delta \boldsymbol{E}_d}, \tag{9a}$$

$$\bar{\boldsymbol{f}}_d = (\Delta \boldsymbol{E}_d)^{-1}(e^{\Delta \boldsymbol{E}_d}\boldsymbol{I})\Delta \boldsymbol{f}_d, \tag{9b}$$

where Δ is a learnable parameter denoting the step size for discretization, whereas $\bar{\boldsymbol{E}}_d$ and $\bar{\boldsymbol{f}}_d$ are the discrete version of the parameters $\boldsymbol{E}_d$ and $\boldsymbol{f}_d$. As a result, the state equation can be represented as a recurrence of previous states, expressed as

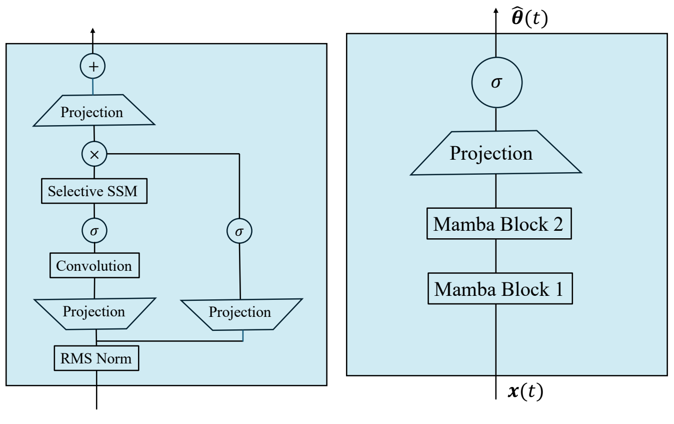
$$\boldsymbol{h}_{d,t} = \bar{\boldsymbol{E}}_d \boldsymbol{h}_{d,t-1} + \bar{\boldsymbol{f}}_d x_{d,t}, \tag{10a}$$

$$y_{d,t} = \boldsymbol{g}_d \boldsymbol{h}_{d,t}. \tag{10b}$$

These set of equations can be computed by recurrent neural network (RNN)-like structure as depicted in Fig. 1. From the figure, it is evident that, at a particular time instant t, the hidden latent state of the network depends on the new observation $x_{d,t}$ as well as the previous state $\boldsymbol{h}_{d,t-1}$, which captures the information of previous hidden states through the structured matrix $\bar{\boldsymbol{E}}_d$.

B. Selective State Space Model

The parameters of a basic SSM are LTI and lack context-aware capabilities, as these parameters do not depend on the input. To address this issue, a selection mechanism was proposed in [24], which makes the parameters input-dependent by incorporating the sequence length and batch size of the input. This is achieved by learning different values of Δ, $\boldsymbol{\mathcal{F}}$, $\boldsymbol{\mathcal{G}}$ for each time instant t, and the shape of these tensors are adjusted accordingly. For a dataset with batch size B and total snapshots T, Δ now has dimensions of $B \times T \times 2D$, implying that, for each input example and dimension, there is a unique discretization rate for each time instant t. Note that we omit the subscript d from the tensors $\bar{\boldsymbol{\mathcal{E}}}$, $\bar{\boldsymbol{\mathcal{F}}}$, and $\bar{\boldsymbol{\mathcal{G}}}$, as their shapes now incorporate all dimensions. Intuitively, a larger Δ places more emphasis on the current input, while a smaller Δ persists in the current state, effectively ignoring the current input. Similarly, $\bar{\boldsymbol{\mathcal{F}}}$ and $\bar{\boldsymbol{\mathcal{G}}}$ are made input-dependent with dimensions $B \times T \times 2D \times N$, enabling finer-grained control over whether an input affects the state or whether the state affects the output. This allows the model to modulate its recurrent dynamics based on both content (input) and context (hidden states).



(a) A Mamba block (b) Stacking Mamba blocks to obtain Mamba network

Fig. 2: Mamba network.

In this model, parameters $\Delta, \boldsymbol{\mathcal{F}}, \boldsymbol{\mathcal{G}}$ are learned through linear layers of fully connected neural networks, defined as

$$s_F = \text{Linear}_N(x), \tag{11a}$$

$$s_G = \text{Linear}_N(x), \tag{11b}$$

$$s_\Delta = \text{Broadcast}_{2D}\left(\text{Linear}_1(x)\right), \tag{11c}$$

$$\tau_\Delta = \text{softplus}, \tag{11d}$$

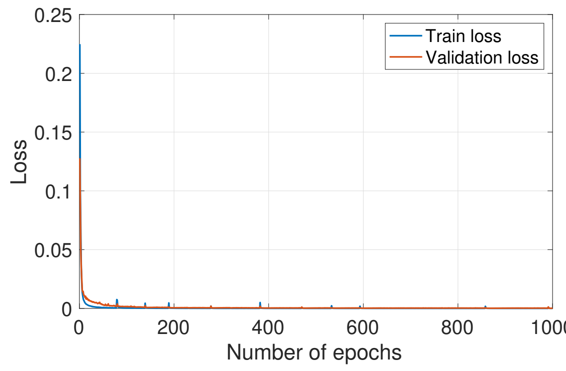
where Linear_d denotes a linear layer of neural network performing parameterized projection to dimension d, Broadcast_{2D} indicates replicating the single dimensional output form $\text{Linear}_1(x)$ across $2D$ dimension to match the shape of the input dimension, and the softplus operation is defined as

$$\tau_\Delta(x) = \log(1 + e^x). \tag{12}$$

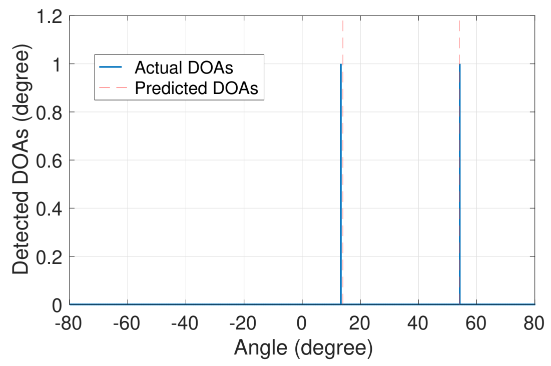
The parameters $\boldsymbol{\mathcal{F}}$ and $\boldsymbol{\mathcal{G}}$ parameters are learned through s_F and s_G operations, whereas Δ is learned through the function $\tau_\Delta(s_\Delta)$. In each epoch, the parameter Δ is applied to $\boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{F}}$ and $\boldsymbol{\mathcal{G}}$ to obtain their discrete versions, $\bar{\boldsymbol{\mathcal{E}}}, \bar{\boldsymbol{\mathcal{F}}}$ and $\bar{\boldsymbol{\mathcal{G}}}$.

C. Mamba Network

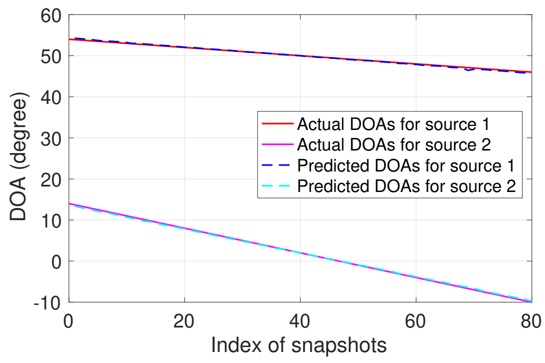
The Mamba architecture is shown in Fig. 2, while Fig. 2(a) illustrates the structure of a single Mamba block. Note that the Mamba network in Fig. 2 provides a more detailed neural network depiction of the system shown in Fig. 1 at time instant t, incorporating the selectivity mechanism. Initially,



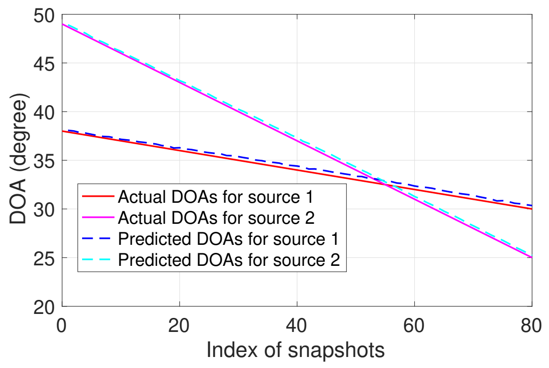
(a) Loss versus number of epochs



(b) Single snapshot



(c) Time varying DOA estimation for scenario 1



(d) Time varying DOA estimation for scenario 2

Fig. 3: DOA estimation performance.

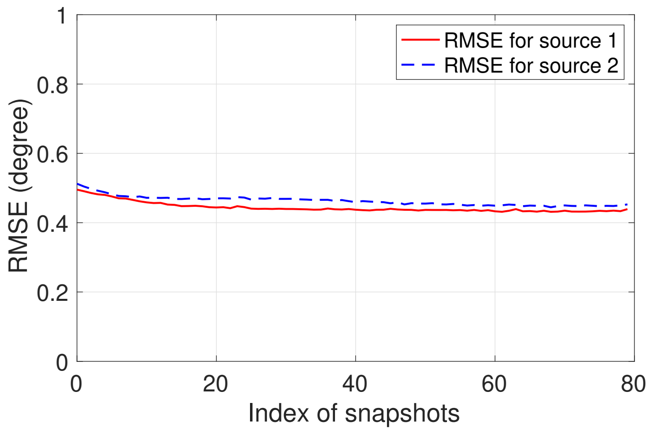


Fig. 4: RMSE versus time.

root mean square (RMS) layer normalization is applied within the block to stabilize neural network training by normalizing neuron dynamics within a layer [27]. Following this, a linear layer projects the data into a high-dimensional space to capture complex data relationships. A one-dimensional convolutional layer is then employed to capture the local dependencies of the input sequence. Subsequently, the selective SSM is applied to maintain long-term dependencies in the input data.

The Mamba block also incorporates a residual connection, which enhances the flow of information by allowing the original data to bypass the main processing path. This feature helps preserve essential information that might otherwise be lost or significantly altered by the transformations within the main branch [28]. The residual connection also results stable training in deep neural networks. Finally, another linear projection is used to return the data to its original dimension. The Sigmoid Linear Unit (SiLU) activation function is applied within the Mamba block. In Fig. 2(b), multiple Mamba blocks are stacked together to form the complete network. Additionally, it is necessary to project the output from the Mamba blocks to match the dimension of $\boldsymbol{\theta}$.

The network is trained using minibatch processing. Let $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{B\times T\times 2D}$ represent a batch of data, and let $\boldsymbol{\theta}^{(\text{batch})} \in \mathbb{R}^{B\times T\times K}$ and $\hat{\boldsymbol{\theta}}^{(\text{batch})} \in \mathbb{R}^{B\times T\times K}$ denote the true and estimated DOAs for the batch, respectively. The network is trained with a mean-squared-error loss function, defined as

$$\text{Loss} = \frac{1}{BT}\sum_{b=1}^{B}\sum_{t=1}^{T}\left\|\boldsymbol{\theta}_{b,t}^{(\text{batch})} - \hat{\boldsymbol{\theta}}_{b,t}^{(\text{batch})}\right\|_2^2, \tag{13}$$

where $\boldsymbol{\theta}_{b,t}^{(\text{batch})}$ and $\hat{\boldsymbol{\theta}}_{b,t}^{(\text{batch})}$ are the true and predicted DOAs for the tth time snapshot and the bth example within a batch.

IV. Simulation Results

We consider a ULA consisting of $D = 10$ receive antennas. $K = 2$ narrow-band, uncorrelated target signals are assumed. The training dataset is generated by sampling the initial DOAs at $t = 0$ from a uniform distribution within the range of $[-60^\circ, 60^\circ]$. After this, the initial DOAs are assumed to change at a constant rate between $[0.1^\circ, 0.5^\circ]$ per snapshot. The total number of available snapshots is $T = 100$.

Based on these conditions, we generated 50,000 example DOA scenarios along with their corresponding array received signal vectors for $t = 1$ to $t = 100$, using a signal-to-noise ratio (SNR) of 10 dB. Among them 80% data is used for training, and the remaining 20% is used for validation purposes. We set the batch size to $B = 256$, so each batch of real-valued input data for the network has dimensions of $256 \times 100 \times 20$, and the corresponding target labels have dimensions of $256 \times 100 \times 2$. We used ADAM optimizer with a weight decay of 10^{-4} to improve generalization and a learning rate scheduler with an initial learning rate 0.001, decreasing it by a factor of 0.1 if validation loss does not decrease within the next 15 epochs. The hidden state size of the Mamba network is set to 128, and training is performed over 1000 epochs. Fig. 3(a) shows the reduction in training and validation loss over the epochs, with both losses converging smoothly as epochs increase.

Figs. 3(b) and 3(c) illustrate the DOA estimation performance for a test scenario. In this case, the initial DOAs are $54°$ and $14°$ at $t = 0$, with source 1 changing by $0.1°$ per snapshot and source 2 by $0.3°$ per snapshot. Fig. 3(b) illustrates that the estimation of the initial DOA, indicating the network's effectiveness in estimating DOAs from a single snapshot when trained. Fig. 3(d) depicts a more challenging scenario, where the sources move towards each other, intersect, and then move apart. The proposed model successfully tracks this scenario, although a slight error is noticeable when the two sources are close, due to the difficulty in distinguishing closely located DOAs. We further evaluate the model's performance by computing the root-mean-squared error (RMSE) between the true and estimated DOAs across all time samples for $N = 1500$ test scenarios, defined as

$$\text{RMSE}_{k,t} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}\left(\theta_{k,t} - \hat{\theta}_{k,t}\right)^2}, \tag{14}$$

where $\text{RMSE}_{k,t}$ defines the RMSE value for the kth source at time t, $\theta_{k,t}$ and $\hat{\theta}_{k,t}$ are the true and predicted DOAs for the kth source at time t. From Fig. 4, it is evident the RMSE values across time remains between 0.4 to 0.6 degrees.

V. Conclusion

In this paper, we developed a Mamba neural network for estimating DOAs that change with each time snapshot. The selection mechanism within the Mamba network effectively captures the temporal dependencies of the input sequence, enabling efficient tracking. The proposed approach can detect DOAs from single-snapshot data and subsequently track changes in DOAs across successive snapshots. The effectiveness of the proposed method is validated through simulation results.

References

[1] H. L. Van Trees, *Optimum Array Processing: Part IV of Detection, Estimation, and Modulation Theory.* Wiley, 2002.

[2] T. E. Tuncer and B. Friedlander, *Classical and Modern Direction-of-Arrival Estimation.* Academic Press, 2009.

[3] Y. Zhang, W. Mu, and M. G. Amin, "Subspace analysis of spatial time-frequency distribution matrices," *IEEE Trans. Signal Process.*, vol. 49, no. 4, pp. 747–759, 2001.

[4] S. Sun and Y. D. Zhang, "4D automotive radar sensing for autonomous vehicles: A sparsity-oriented approach," *IEEE J. Sel. Topics Signal Process.*, vol. 15, no. 4, pp. 879–891, 2021.

[5] R. Schmidt, "Multiple emitter location and signal parameter estimation," *IEEE Trans. Antennas Propag.*, vol. 34, no. 3, pp. 276–280, 1986.

[6] R. Roy and T. Kailath, "ESPRIT–estimation of signal parameters via rotational invariance techniques," *IEEE Trans. Acousti., Speech, Signal Process.*, vol. 37, no. 7, pp. 984–995, 1989.

[7] B. Yang, "Projection approximation subspace tracking," *IEEE Trans. Signal Process.*, vol. 43, no. 1, pp. 95–107, 1995.

[8] H. Wu and X. Zhang, "DOD and DOA tracking algorithm for bistatic MIMO radar using PASTD without additional angles pairing," in *Proc. IEEE Int. Conf. Adv. Comput. Intell. (ICACI)*, 2012, pp. 1132–1136.

[9] S. Li, B. Lin, and Z. Che, "Unmanned ship tracking algorithm based on MIMO radar," in *Proc. IEEE/CIC Int. Conf. Commun. China (ICCC)*, 2024, pp. 874–878.

[10] J.-F. Gu, S. Chan, W.-P. Zhu, and M. Swamy, "Joint DOA estimation and source signal tracking with kalman filtering and regularized QRD RLS algorithm," *IEEE Trans. Circuits Systems II: Express Briefs*, vol. 60, no. 1, pp. 46–50, 2013.

[11] Q. Zhang, J. Li, Y. Li, and P. Li, "A DOA tracking method based on offset compensation using nested array," *IEEE Trans. Circuits Systems II: Express Briefs*, vol. 69, no. 3, pp. 1917–1921, 2021.

[12] H. Huang, J. Yang, H. Huang, Y. Song, and G. Gui, "Deep learning for super-resolution channel estimation and doa estimation based massive mimo system," *IEEE Trans. Veh. Technol.*, vol. 67, no. 9, pp. 8549–8560, 2018.

[13] S. R. Pavel, M. W. T. Chowdhury, Y. D. Zhang, D. Shen, and G. Chen, "Machine learning-based direction-of-arrival estimation exploiting distributed sparse arrays," in *Proc. Asilomar Conf. Signals, Syst., Comput.*, 2021, pp. 241–245.

[14] L. Wu, Z.-M. Liu, and Z.-T. Huang, "Deep convolution network for direction of arrival estimation with sparse prior," *IEEE Signal Process. Lett.*, vol. 26, no. 11, pp. 1688–1692, 2019.

[15] Z.-M. Liu, C. Zhang, and S. Y. Philip, "Direction-of-arrival estimation based on deep neural networks with robustness to array imperfections," *IEEE Trans. Antennas Propag.*, vol. 66, no. 12, pp. 7315–7327, 2018.

[16] S. Jaiswal, R. Pandey, and S. Nannuru, "Deep architecture for DOA trajectory localization," in *Proc. IEEE Int. Conf. Acoustics, Speech Signal Process. (ICASSP).* IEEE, 2023, pp. 1–5.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, 1997.

[18] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Łukasz Kaiser, and I. Polosukhin., "Attention is all you need," *Adv. Neural Inform. Process. Syst.*, vol. 30, pp. 1–11, 2017.

[20] J. Liu, Z. Wang, and M. Xu, "DeepMTT: A deep learning maneuvering target-tracking algorithm based on bidirectional LSTM network," *Inform. Fusion*, vol. 53, pp. 289–304, 2020.

[21] C. Gao, J. Yan, S. Zhou, P. K. Varshney, and H. Liu, "Long short-term memory-based deep recurrent neural networks for target tracking," *Inform. Sciences*, vol. 502, pp. 279–296, 2019.

[22] X. Chen, B. Yan, J. Zhu, D. Wang, X. Yang, and H. Lu, "Transformer tracking," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recog.*, 2021, pp. 8126–8135.

[23] A. Gu, K. Goel, and C. Ré, "Efficiently modeling long sequences with structured state spaces," *arXiv preprint arXiv:2111.00396*, 2021.

[24] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," *arXiv preprint arXiv:2312.00752*, 2023.

[25] A. Gu, T. Dao, S. Ermon, A. Rudra, and C. Ré, "Hippo: Recurrent memory with optimal polynomial projections," *Adv. Neural Inform. Process. Syst.*, vol. 33, pp. 1474–1487, 2020.

[26] T. J. Moir and T. J. Moir, *Rudiments of Signal Processing and Systems*. Springer, 2022.

[27] B. Zhang and R. Sennrich, "Root mean square layer normalization," *Adv. Neural Inform. Process. Syst.*, vol. 32, p. 12360–12371, 2019.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE conf. Comput. Vision Pattern Recogn.*, 2016, pp. 770–778.