Article

Not peer-reviewed version

Time Convolutional Network Based Maneuvering Target Tracking with Azimuth-Doppler Measurement

Huang Jianjun , Haoqiang Hu , Kang Li *

Posted Date: 22 November 2023

doi: 10.20944/preprints202311.1360.v1

Keywords: azimuth and Doppler; convolutional neural network; deep learning algorithm; maneuvering targets tracking

Article

Time Convolutional Network Based Maneuvering Target Tracking with Azimuth-Doppler Measurement

JianJun Huang, HaoQiang Hu and Li Kang *

School of Intelligent Information Processing, Shenzhen University, Shenzhen 518060, China; huangjj_atr@sina.com.cn (J.H.); 2100432050@email.szu.edu.cn (H.H.)

* Correspondence: kangli@szu.edu.cn

Abstract: In the field of maneuvering target tracking, the combined observations of azimuth and Doppler may cause weak observation or non-observation in the application of traditional target tracking algorithms. Additionally, traditional target-tracking algorithms require pre-defined multiple mathematical models to accurately capture the complex motion states of targets, while model mismatch and unavoidable measurement noise lead to significant errors in target state prediction. To address those above challenges, in recent years, the target-tracking algorithms based on neural networks, such as recurrent neural networks (RNNs), long short-term memory (LSTM) networks, and Transformer architectures, have been widely used for their unique advantages to achieve accurate predictions. To better model the nonlinear relationship between the observation time series and the target state time series, as well as the contextual relationship among time series points, we present a deep learning algorithm called recursive downsample-convolve-interact neural network (RDCINN) based on convolutional neural network (CNN) that downsamples time series into sub-sequences and extracts multi-resolution features to enable the modeling of complex relationships between time series, which overcomes the shortcomings of traditional target-tracking algorithms in using observation information inefficiently due to weak observation or non-observation. The experimental results show that our algorithm outperforms other existing algorithms in the scenario of strong maneuvering target tracking with the combined observations of azimuth and Doppler.

Keywords: azimuth and Doppler; convolutional neural network; deep learning algorithm; maneuvering targets tracking

---

1. Introduction

WITH the increasing application of radar maneuvering target tracking technology in both civil and military domains, how to use the data measured by radar sensors to obtain more accurate target state estimation has become a prominent research focus. Furthermore, the single-station passive positioning technology applied in target tracking utilizes the radar sensor that passively receives the source radiation signal to enable the positioning of the radiation source, and is highly regarded for its strong concealment and broad application [1]. In the past, pure azimuth passive positioning technology was commonly used for target localization and tracking [2]. However, relying solely on the nonlinear relationship between angle measurements and target states made it challenging to obtain accurate target state estimation. Subsequently, researchers discovered that introducing new observables such as angle change rate and Doppler frequency could enhance localization accuracy [3,4]. For instance, combining azimuth and Doppler observables enables more precise target state estimation.

However, using the combined observations of azimuth and Doppler has certain requirements on the relative positions of the observation point and the target trajectory. In specific cases, the observation point will experience weak observation or non-observation during some relative motion periods [5,6], as shown in Figure 1. When a maneuvering target moves with a uniform or uniformly variable speed along the line connecting it to the observation point, both the azimuth and Doppler measurements have a rate of change of 0. During these periods, the target becomes unobservable as there is no significant change in the measured information. Similarly, when the maneuvering target follows a uniform circular motion relative to the observation point, the Doppler measurement remains constant, and only the azimuth measurement provides some observable information, this observation is considered weak due to the lack of Doppler information, which limits the accuracy of target state estimation during such periods.

In short, the single station passive positioning technique based on the combination of azimuth and Doppler measurement has some limitations in the observation range of the maneuvering target tracking. Meanwhile, the maneuvering target states are always uncertain and complex in practically, and the single-model traditional algorithms like extended Kalman filter (EKF) and unscented Kalman filter (UKF) applied in the field of weak maneuvering target tracking are difficult to achieve the accurate estimation of such complex maneuvering target states [7]. To solve this kind of complex strong maneuvering target tracking problem, multiple models (MM) algorithms and their various variants have been proposed and widely used [8]. Taking the interactive multiple model algorithm (IMM) as an example [9,10], it usually needs to predefine a variety of target maneuvering models, model probabilities, and model transfer probabilities for the estimation of target states. However, in the specific application of this traditional target tracking algorithm, there is the problem of delayed model estimation when the target maneuvering state changes abruptly [11], coupled with the fact that the multi-model algorithm essentially needs to set up the motion models in advance just as the single-model target tracking algorithm does, and the inaccuracy of the preset motion model may also occur in the face of complex maneuvering situations, as well as the existence of observation noise in the radar sensors themselves, and various other adverse factors have a negative impact on this interactive multi-model algorithm [12].

In recent years, with the advancement of deep learning, the research on utilizing neural network modeling algorithms to break through the limitations of traditional target-tracking algorithms has significantly expanded [13]. Leveraging the distinctive advantages of neural networks [14], it becomes feasible to dispense with the requirement of predefining the motion models beforehand and accomplish end-to-end predictions between observations and maneuvering target states. A previous study with a maneuvering trajectory prediction method that employed a backpropagation neural network (BPNN) was introduced to combine the historical trajectory of the target to capture the target's motion patterns and generate predicted trajectories [15]. Furthermore, the articles [16–18] focus on addressing the challenges that arise from the inherent uncertainty in both maneuvering target states and the measurement information faced by traditional target tracking algorithms and presenting methodologies that better model the long-term dependence among sequence data through the gating mechanism. Subsequently, the articles [19,20] address the limitation of the long short-term memory (LSTM) model in capturing the global nature of the target maneuvering state by proposing the use of the transformer architecture which captures both long-term and short-term dependence of the target state, further enhance the accuracy of target tracking algorithms.

To achieve an accurate estimation of strong maneuvering target states based on the combined observations of azimuth and Doppler, the first step is to use a time series of observations in the target motion state prediction to address the challenge of insufficient time information obtained from the azimuth-Doppler information at a single moment. The prediction of time series typically involves three fundamental structures, namely Recurrent Neural Networks (RNN), Transformer-based Networks (TBN), and Temporal Convolutional Networks (TCN) [21–23]. For leveraging the distinctive property where temporal relationships are largely preserved even after downsampling a time series into two subsequences, we propose a recursive downsample convolution interact-learning neural network (RDCINN) based on the Convolutional Neural Network (CNN) architecture [24] designed to address the challenge of motion states estimation. Our approach involves several key operations to extract motion features from the input observation time series. We first apply a full-connection layer and a position-coding layer to perform temporal coding operations on the input. Then, we proceed with recursive downsampling, temporal convolution, and interactive learning operations. In each layer, multiple convolutional filters are employed to extract motion features from the downsampling time series. By combining these rich features gathered from multiple resolutions, we can effectively address the issue of weak observation or non-observation encountered in traditional maneuvering target tracking algorithms based on the combined observations of azimuth and Doppler. Finally, the utilization of a binary tree structure in our model contributes to an increased depth of temporal feature extraction which allows for effective modeling of the nonlinear mapping relationship between high-noise observation time series and complex maneuvering states.

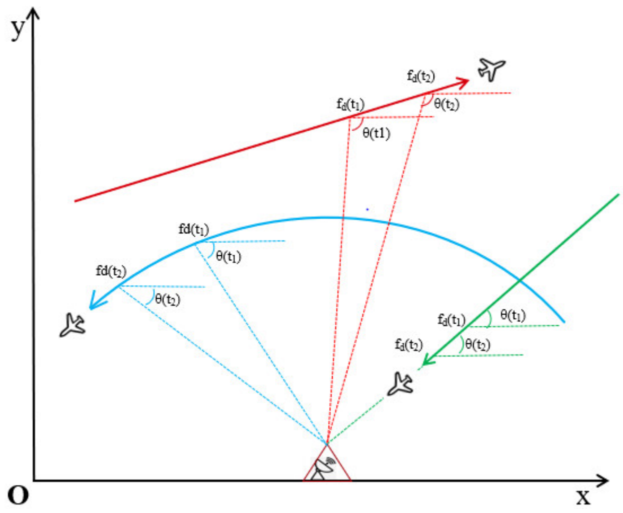

Figure 1. The relative positions of the observation point and the maneuvering target in the two-dimensional X-Y plane. The green trajectory illustrates the movement of a maneuvering target at a uniform or uniformly variable speed along the target and the observation point. The blue trajectory demonstrates the trajectory of a target moving circularly at a uniform velocity around the observation point. The red trajectory shows the movement of a maneuvering target relative to the observation point.

2. Problem Formulation

As in previous studies on maneuvering target tracking using deep learning approaches [17–19], our simulation scenarios are set on a 2D plane. In this setup, the radar observation point passively receives azimuth and Doppler information, and is positioned at the origin O. We assumed that s_k and z_k are the $k-th$ momentary maneuvering target state vector and observation vector, respectively. $(x_k,\ y_k)$ denotes the position of a $k-th$ moment maneuvering target in the $\mathrm{X}-\mathrm{Y}$ plane, $(\dot{x}_k,\ \dot{y}_k)$ represents the corresponding velocity. And $[\theta_k,\ d_k]^T$ represents the azimuth and Doppler information of the measurements z_k and is expressed as [25]:

$$\begin{bmatrix}\theta_k\\ d_k\end{bmatrix} = \begin{bmatrix} arctan\left(\frac{y_k}{x_k}\right)\\ \dot{x}_k\ cos\theta_k + \dot{y}_k sin\theta_k\end{bmatrix} + \begin{bmatrix} n_{\theta,k}\\ n_{d,k}\end{bmatrix} \tag{1}$$

where $n_{\theta,k} \sim N\,(0,\sigma_\theta^2)$, $n_{d,k} \sim N\,(0,\sigma_d^2)$, σ_θ、σ_d are the standard deviations of the Gaussian noise of the azimuth and Doppler measurement, respectively.

To perform target tracking using a deep learning approach, it is essential to generate an extensive dataset of trajectories and observations for training the network model [26]. This dataset enables the modeling of the nonlinear relationship between target states and observation information, ultimately facilitating the accurate estimation of maneuvering target states. Typically, the dataset is generated based on the state equations and observation equations as follow:

$$\begin{cases} s_k = Fs_{k-1} + n_{s,k}\\ z_k = h(s_k) + n_{z,k}\end{cases} \tag{2}$$

where $n_{s,k}$、$n_{z,k}$ denote state transfer noise and observation noise and are expressed as follow, respectively:

$$n_{z,k} = \left[n_{\theta,k}, n_{d,k}\right]^T \tag{3}$$

$$n_{s,k} = \left[n_{pos,k}, n_{pos,k}, n_{vel,k}, n_{vel,k}\right]^T \tag{4}$$

where $n_{pos,k} = \frac{T^2}{2}\alpha_k$, $n_{vel,k} = T\alpha_k$, T is the radar sensor sampling interval time. $\alpha_k \sim N\left(0, \sigma_s^2\right)$ denotes the maneuvering acceleration noise, which follows a Gaussian distribution. In the state equation $s_k = Fs_{k-1} + n_{s,k}$, we have incorporated two maneuver models, namely the constant velocity (CV) motion and the constant turning (CT) motion. The definitions of these models are as follows:

$$F_{CV} = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

$$F_{CT} = \begin{bmatrix} 1 & 0 & \frac{sin(wT)}{w} & \frac{cos(wT)-1}{w} \\ 0 & 1 & \frac{1-cos(wT)}{w} & \frac{sin(wT)}{w} \\ 0 & 0 & cos(wT) & -sin(wT) \\ 0 & 0 & sin(wT) & cos(wT) \end{bmatrix} \tag{6}$$

In the network model, the input consists of the measurement time series $z_{1:K} = \{z_1, z_2, \ldots, z_K\}$, while the output is the estimated state of the maneuver target $\tilde{s}_{1:K} = \{\tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_K\}$, K represents the total number of time steps in the input-output time series, and$w$is the turning rate. To evaluate the performance of the model, we employ the mean absolute error (MAE) as a measure. We denote the loss function with the estimated state of the maneuver target and its true state as:

$$Loss = \sqrt{\frac{1}{K}\sum_{k=1}^{K}\left|s_k - \tilde{s}_k\right|} \tag{7}$$

3. The Model for Maneuvering Target Tracking

In this section, we will discuss the deep learning algorithm applied to maneuvering target tracking based on the combined observations of azimuth and Doppler. The algorithm consists of two parts: the training phase and the testing phase. The training phase is depicted in Figure 2. Following a similar research scheme for maneuvering target tracking based on LSTM or TBN [18,19], we begin by using the observation equations and state equations to generate a large-scale trajectory dataset (LASTD) and then employ LASTD to train the neural network model. The LASTD is firstly followed by batch processing, and the normalized observations are input into the designed network model batch by batch. The output of the model is the predicted maneuvering target state which is converted back to its original form before. Finally, backpropagation and parameter updating of the network are performed using a loss function with the predicted states and the ground-truth states. We fine-tune the network to improve its performance in maneuvering target tracking.

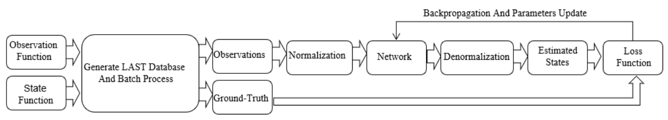

Figure 2. Flowchart of the algorithm training process. Generate target state and observation pairs for network training with motion and observation models first, then train with batch normalized observation as inputs, and update the parameters with back-propagation of the loss error between the predicted state and the true state iteratively.

3.1. Normalization Methods

To effectively utilize LASTD for network training and enhance the convergence speed, we normalize each generated trajectory data and its corresponding observations. In this study, we employ a hybrid normalization approach that takes into consideration the distributional characteristics of both the observation and target state data. Given that the observation data noise is assumed to be additive white noise, we adopt Gaussian normalization for the observation data. The normalization formula is expressed as follows:

$$z'_{1:k} = \frac{z_{1:k} - mean(z_{1:k})}{var(z_{1:k})} \tag{8}$$

where $z'_{1:k}$, $z_{1:k}$ denotes the normalized observation segment and the corresponding generated original observation segment respectively. $mean(\bullet)$ denotes the mean of the segment data and $var(\bullet)$ is the variance of the same. For the real target trajectory states, we normalize them using the min-max normalization method. The formula is denoted as:

$$s'_{1:k} = \frac{s_{1:k} - min(s_{1:k})}{max(s_{1:k}) - min(s_{1:k})} \tag{9}$$

where $s'_{1:k}$, $s_{1:k}$ denotes the normalized target state and the generated target state respectively. $min(\bullet)$ denotes the minimum value of the data, and $\max(\bullet)$ denotes the maximum value of the data. The partially normalized trajectory data distribution and the original trajectory data distribution are shown in Figure 3.

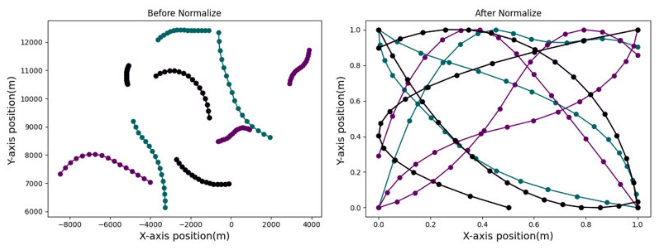

Figure 3. Min-max normalization of trajectory segments. The left side of the figure shows the original trajectories generated by 9 different motion models. On the right side, the distributions of the normalized trajectory data can be observed. The x and y ranges of the trajectories are restricted to the interval [0, 1] which ensures that the LASTD does not have excessively large ranges to pose difficulties during network training.

3.2. Proposed Model

Most previous studies tackling this problem of maneuvering target tracking utilized deep learning algorithms based on RNN or LSTM to model the nonlinear relationship between observation time series and target state time series. However, RNN may face challenges such as gradient explosion or vanishing during the training phase, while LSTM may not be very efficient with step-by-step prediction. Hence, the TBN was proposed for further accurate modeling with its advantages of parallel training and global processing of long sequences. Nevertheless, TBN may exhibit a weaker ability to extract local information compared to RNN and CNN [27].

To fully extract rich dynamic features from time series and overcome the challenges posed by the weak observation or non-observation in traditional algorithms, as well as accurately model the nonlinear relationship between observation sequences and target state sequences along with the contextual relationships among time series points in a complex motion environment, we propose a neural network called RDCINN based on one-dimensional CNN [24] as shown in Figure 4. This model architecture leverages a diverse array of convolutional filters to extract dynamic temporal features at multiple resolutions to learn the complex nonlinear relationships between temporal sequences and the contextual relationships within the temporal sequences.

In each block module of RDCINN, the input undergoes a two-step process: splitting and interactive learning. The splitting step involves downsampling the input into two subsequences. These subsequences consist of the odd and even elements of the original sequence, respectively. This splitting operation takes advantage of the unique nature of time series, allowing the subsequences to retain most of the information present in the original sequence. The split odd and even subsequences are then individually processed by different convolutional kernels to extract valuable features and enhance the expressive power of the model. Following the splitting step, the interactive learning step compensates for any potential loss of information due to downsampling. This step involves the mutual learning of affine transform parameters between the two subsequences. The equations expressing the interactive learning step are as follows:

$$X_{odd}^{fst} = X_{odd} \otimes exp(\, conv1d(X_{even})) \tag{10}$$

$$X_{even}^{fst} = X_{even} \otimes exp(\, conv1d(X_{odd})) \tag{11}$$

$$X_{odd}^{sec} = X_{odd}^{fst} + conv1d(X_{even}^{fst}) \tag{12}$$

$$X_{even}^{sec} = X_{even}^{fst} - conv1d(X_{odd}^{fst}) \tag{13}$$

where X_{even} and X_{odd} represent the subsequence of even and odd elements obtained after splitting the original sequence respectively. $conv1d$ represents the one-dimensional convolution layer, exp is the exponential operation applied to the sequence after convolution, and $\otimes$ represents the Hadamard (element-wise) product operation. X_{odd}^{fst} and X_{even}^{fst} represent the parity subsequences obtained after the sequence splitting, convolution, exponential, and Hadamard operations. Finally, the module generates the two parity subsequences X_{odd}^{sec} and X_{even}^{sec} after the interactive learning step.

In RDCINN, an encoder-decoder architecture is employed. The encoder consists of multiple blocks organized in a binary tree structure. This structure enables the network model to have both local and global views of the entire time series, facilitating the extraction of useful temporal features. After the downsampling, convolution, and interaction operations within the encoder, the extracted features are reshaped into a new sequence representation. These reshaped features are then combined with the original time series for prediction. The decoder with a two-layer one-dimensional convolutional network layer, performs the prediction based on the combined representation of the reshaped features and the original time series. This encoder-decoder architecture allows RDCINN to leverage both local and global temporal information in the input time series, enhancing its ability to capture and model complex temporal relationships for accurate prediction.

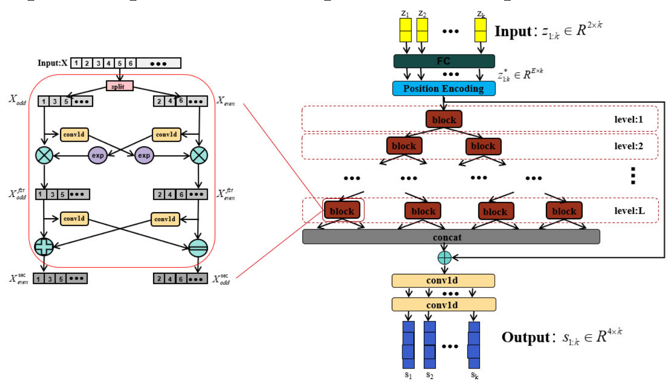

Figure 4. Model architecture of RDCINN. On the left is a diagram of a sub-block structure of the binary tree network architecture, which performs splitting, convolution, and interactive learning operations on the input sequence X sequentially, and finally outputs the odd and even subsequences X_{odd}^{sec} and X_{even}^{sec}. The input to the binary tree network is the 2-dimensional sequence of normalized observations $z_{1:k}$. It is first mapped through the fully connected layer into the E-dimension, and the final outputs obtained through the network as a 4-dimensional sequence of predicted target states $s_{1:k}$.

3.3. Test Reorganization Phase

We utilize the trained network to perform complex maneuvering target tracking on long trajectories. As illustrated in Figure 5, the process consists of the following steps: Divide the observation $z_{1:K}$ corresponding to the target states into L length segments using a sliding window approach. The window size ws is set to 16 with a moving stride st = 4. This division results $zs(n) = z_L$ with the length $L = 16$ is the nth time sequence of inputs, and L can be denoted as:

$$L = 1 + (n-1) * st{:}\, ws + (n-1) * st \tag{14}$$

where n=1,2..., (K-ws)/st+1, K is the total steps. Normalize each segment of the observation time sequence $zs(n)$. Input the normalized observation time sequence $zs(n)^*$ into the network to obtain the predicted target state outputs. Apply denormalization on the predicted target state output $ss(n)^*$ to obtain the corresponding predicted state sequence $ss(n) = s_L$.

Reconstruct each segment of the predicted target state sequences after denormalization. Process the overlapping length $overlap = ws - st$ among target state sequences $ss(n)$ by averaging them. The reconstruction steps are expressed mathematically as follows:

$$s(n)_{1:a} = cat(s(n-1)_{0:b}, \frac{(s(n)_{0:ol}+s(n-1)_{b:a})}{2}, s(n)_{ol:ws}) \quad (15)$$

where we denote $overlap$ as ol, $a = (n-1) * st + ws$, $b = t * (n-1)$, cat means that merge the state sequence $s(n-1)$ and $s(n)$ together. Additionally, we set the initial states $s(1) = ss(1)$, $s(2) = ss(2)$, while $s(n)$ is the merged result of state sequences.

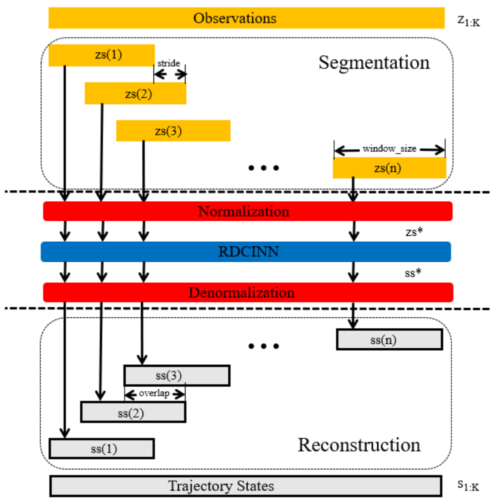

Figure 5. Trajectory segmentation and reconstruction consist of the following steps: Divide the observations $z_{1:K}$ corresponding to the target states using a sliding window approach. Normalize each segment of the observation time sequence$zs(n)$. Apply denormalization on the predicted target state outputs$ss(n)^*$to obtain the corresponding predicted state sequence $ss(n)$. Finally reconstruct each segment of predicted target state sequence after denormalization.

4. Simulation Experiments

In this section, we design several experimental scenarios to evaluate the superiority of our algorithm in predicting the states of strong maneuvering radar targets with the combined observations of azimuth and Doppler. Additionally, we provide a detailed explanation of the specific parameters listed in each part of the experiment.

4.1. Parameter Setting Details

We utilize the LASTD which consists of 450000 trajectories with different motion laws and their corresponding observations for a comprehensive evaluation of the algorithms' performance in maneuvering target tracking tasks. The dataset was structured as follows: 150000 samples consist of 16s long trajectories of either uniform linear motion or uniform circular motion. Another 150000 samples are composed of 16s trajectories segmented into two 8s-long trajectories, every trajectory could be uniform linear motion or uniform circular motion. The remaining 150000 samples consist of 16s trajectories segmented into four 4s-long trajectories, while every trajectory could be either uniform linear motion or uniform circular motion. The sampling time T for each trajectory was set to 1s. The parameters of the LASTD are listed in Table 1.

Table 1. Parameters of LASTD.

Parameters	Value
Distance Range	[926m,18520m]
Angle Range	[-180°,180°]
Velocity Range	[-340m/s,340m/s]
Turn Rate (w)	[-10°/s,10°/s]
The Standard Deviation of Acceleration Noise (σ_s)	[8m/s^2,13m/s^2]
The Standard Deviation of Azimuth Noise (σ_θ)	[1°,1.8°]
The Standard Deviation of Doppler Noise (σ_d)	1m/s
Sampling Time Interval(T)	1s

In the training process, we set the following hyperparameters for our model: the dimension E of the fully connected layer is set to 64, the binary tree height is set to 2, the convolutional layer's kernel size, dilation rate, and group length is set to 5, 2, and 1, respectively. For the decoding layer, we have two one-dimensional convolutional layers with dimensions of 16 and 4, respectively. We use the Adam optimizer for the model training process. The weight decay rate is set to 1e-5. The learning rate is initially set to 7e-4, and it decays by 0.95 after each epoch. We trained 300 epochs with a batch size of 256 on a single NVIDIA 3090 GPU.

In our experiments, we compare our proposed algorithm with three existing algorithms: the LSTM network [17], the TBN model [19], and the traditional maneuvering target tracking method IMM-EKF [9]. We keep the model parameters of the LSTM network and the TBN model unchanged, as specified in their respective research papers, and train the deep learning models using the LASTD we have created.

4.2. Experimental Results

We first created a data set that consists of 1500 trajectories to evaluate the performance of each baseline neural network model, as well as our model.

The data set is similar in structure to the training set and consists of three types of trajectories with different motion patterns. Specifically, there are 500 samples of 16s uniform linear motion trajectory or uniform circular motion trajectory, 500 samples of two 8s uniform linear motion trajectories or uniform circular motion trajectories combined, and 500 samples of four 4s uniform linear motion trajectories or uniform circular motion trajectories combined. The trajectory tracking performance results are shown in Table 2.

Table 2. Tracking performance results of several neural network algorithms for trajectory segments.

	MAE of Position(m)	MAE of Velocity(m)
LSTM	58.73	8.84
TBN	44.16	6.82
RDCINN	42.76	6.35

Based on the results presented in Table 2, it can be observed that our network achieves lower position mean absolute error and velocity mean absolute error results compared to the other two baseline neural networks. This demonstrates that our model, applied to the strong maneuvering target tracking domain based on the combined observations of azimuth and Doppler, outperforms the previous target tracking networks.

After that, we utilize Monte Carlo simulation to generate a 16s strong maneuvering trajectory A. The initial state of A is [-4000m, 4000m, 50m/s, -66m/s]. This trajectory consists of four segments, each lasting 4s and employing different motion models, which reflect sudden changes in the motion target states in real-world scenarios. The first segment of the trajectory is a 4s uniform motion. The second segment is uniform circular motion with a turning rate w of -7°. The third segment is also a uniform circular motion but with a turning rate w of 7°. Finally, we set the last segment as a uniform motion. Additionally, we introduce azimuth observation noise as white noise with zero mean and standard deviation σ_θ of 1.8°, while the standard deviation of Doppler velocity observation noise σ_d is 1m/s. Additionally, the standard deviation of acceleration σ_s is set to 10m/s^2. To assess the tracking performance of trajectory A, we employ our own network model as well as three other baseline algorithms. Table 3 presents the evaluation results, while Figures 6–8 provides visual representations of these results.

Table 3. Tracking performance of several target tracking algorithms on trajectory A.

	MAE of Position(m)	MAE of Velocity(m)
IMM+EKF	219.53	11.17
LSTM	18.92	3.73
TBN	9.21	3.19
RDCINN	4.07	3.04

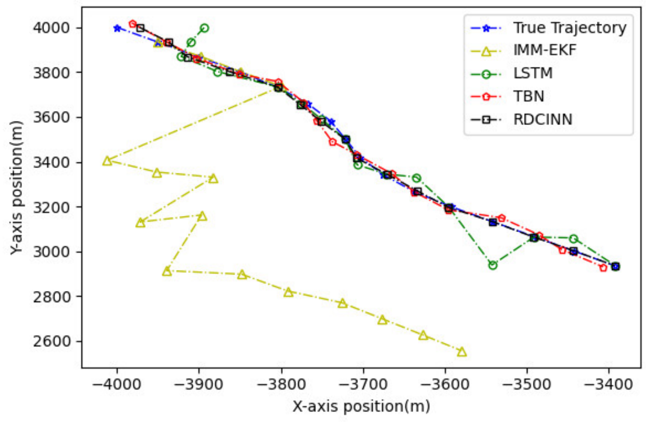

Figure 6. Tracking trajectory results of the trajectory A using different algorithms on the X-Y plane.

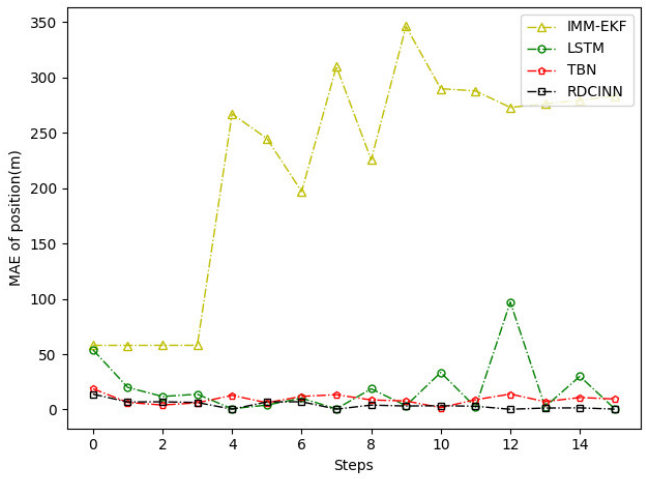

Figure 7. MAE results of position tracking by different algorithms for trajectory A.

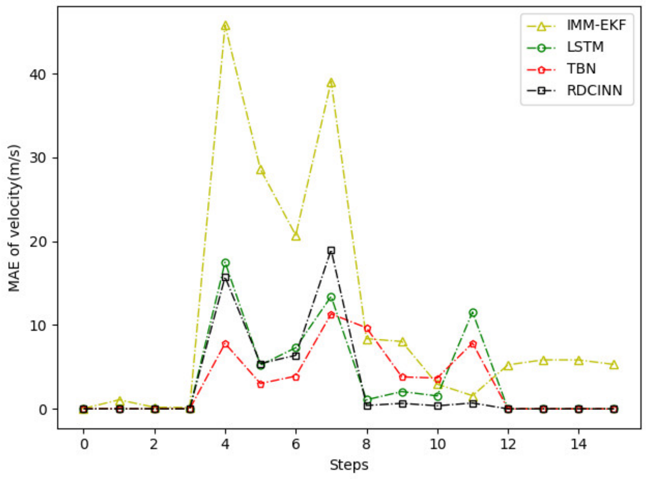

Figure 8. MAE results of velocity tracking by different algorithms for trajectory A.

In order to verify the applicability of our network model for tracking strong maneuvering trajectories with different step sizes, we generate trajectory B and trajectory C by conducting Monte Carlo simulations. Trajectory B is a 32s strong maneuvering trajectory with an initial state of [-8000m, 5000m, -30m/s, 21m/s]. It consists of four segments of 8s trajectories, each with a different model. The models for each segment are as follows: uniform circular motion with a turning rate w of 6°, uniform motion, uniform circular motion with a turning rate w of -5°, and uniform motion, respectively. Trajectory C is a 64s strong maneuvering trajectory with an initial state of [-5000m, 5000m, 30m/s, -23m/s]. It also consists of four segments of 16s trajectories with different motion models. The motion models for each 16s trajectory are as follows: uniform circular motion with a turning rate w of -1°, uniform motion, uniform circular motion with a turning rate w of 2°, and uniform motion, respectively. Keeping the standard deviation setup as what trajectory A set up as the same, we then evaluate the tracking performance of trajectories B and C using our network model and three baseline algorithms. The evaluation results are presented in Tables 4 and 5. Additionally, Figures 9–14 provide visual representations of these results.

Table 4. Tracking performance of several target tracking algorithms on trajectory B.

	MAE of Position(m)	MAE of Velocity(m)
IMM+EKF	184.08	10.25
LSTM	21.56	1.02
TBN	24.01	0.99
RDCINN	15.66	0.82

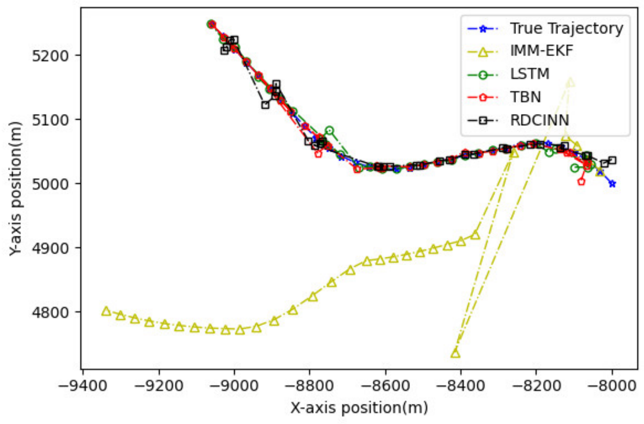

Figure 9. Tracking trajectory results of the trajectory B using different algorithms on the X-Y plane.

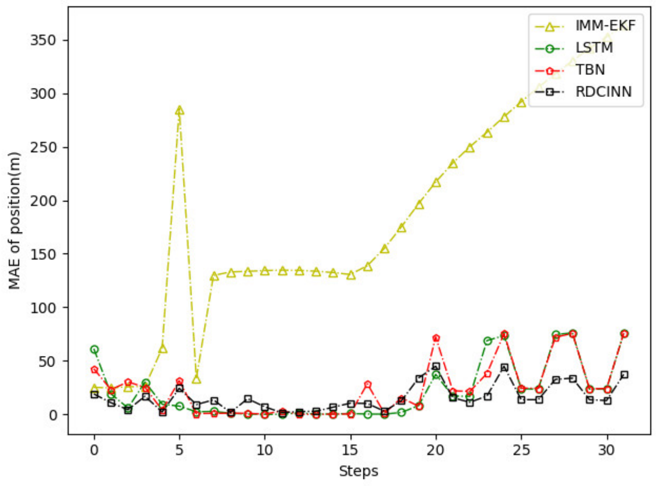

Figure 10. MAE results of position tracking by different algorithms for trajectory B.

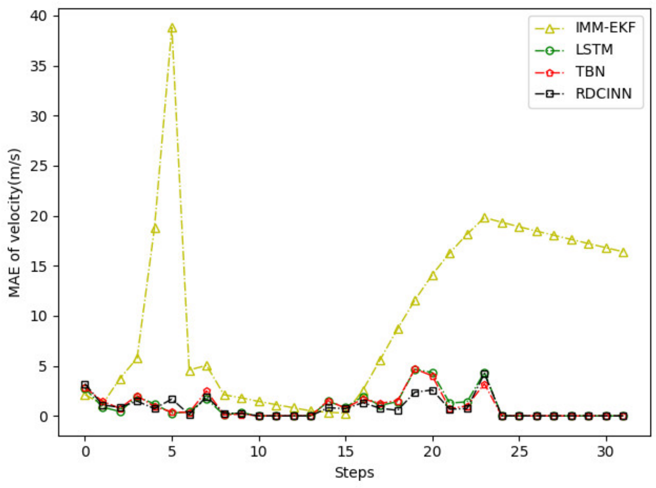

Figure 11. MAE results of velocity tracking by different algorithms for trajectory B.

Table 5. Tracking performance of several target tracking algorithms on trajectory C.

	MAE of Position(m)	MAE of Velocity(m)
IMM+EKF	60.82	5.73
LSTM	13.65	0.16
TBN	15.02	0.19
RDCINN	13.33	0.13

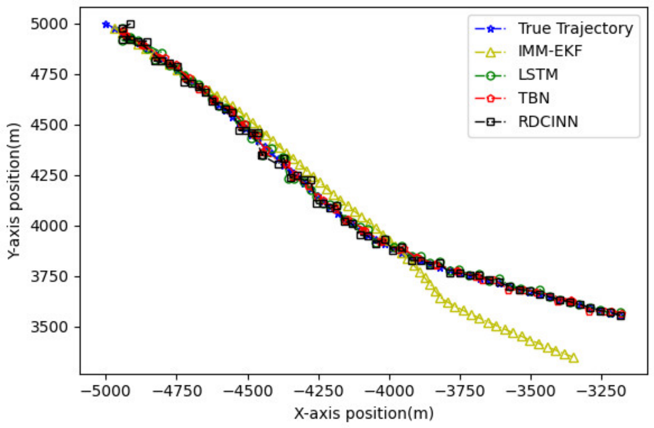

Figure 12. MAE results of position tracking by different algorithms for trajectory C.

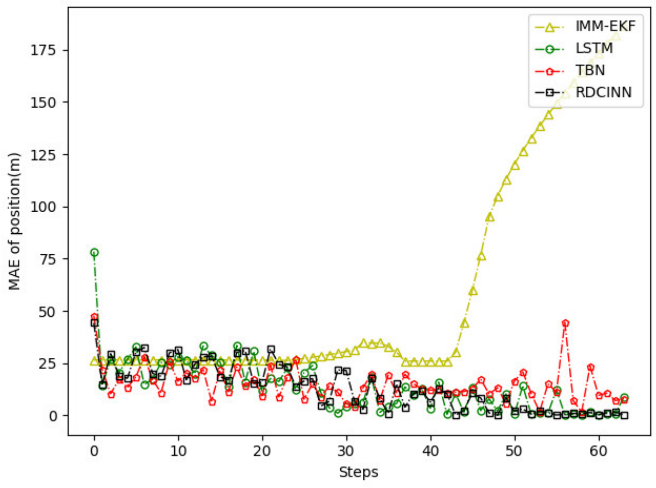

Figure 13. MAE results of position tracking by different algorithms for trajectory C.

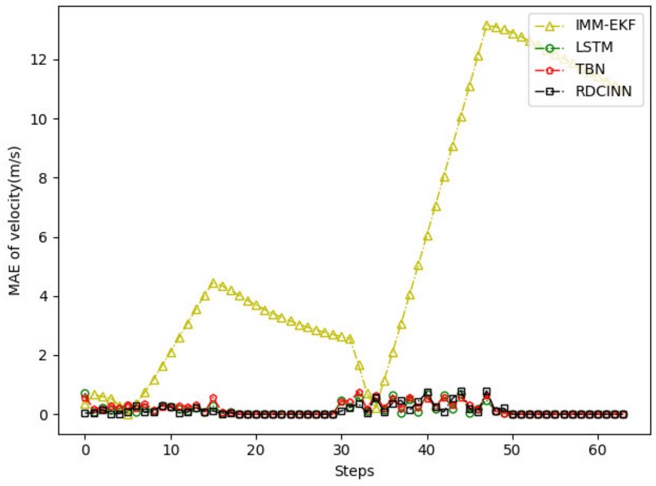

Figure 14. MAE results of velocity tracking by different algorithms for trajectory C.

The experimental results demonstrate that our model achieves superior trajectory tracking performance compared to other algorithms. This is particularly noticeable when tracking strong maneuvering targets under the combined observations of azimuth and Doppler.

5. Conclusions

In this paper, we propose a deep learning algorithm called RDCINN for tracking strong maneuvering targets with the combined observations of azimuth and Doppler. We utilize LASTD generated by the motion models to train RDCINN to learn the nonlinear mapping relationship between observations and target states and facilitate accurate offline estimation of target states in complex maneuvering scenarios despite noisy observations. Simulation results demonstrate that our algorithm not only addresses the limitations of traditional target tracking algorithms, which struggle to update target states due to weak observation or non-observation but also outperforms two previous deep learning algorithms applied to maneuvering target tracking. It is important to note that our algorithm is currently limited to two-dimensional target tracking scenarios, and future work will focus on extending its application to three-dimensional scenarios. Furthermore, there is relatively limited research on the utilization of temporal convolutional networks in the field of target tracking. Future work will involve gaining a deeper understanding of temporal convolutional networks and further improving state estimation accuracy.

Funding: This work was supported in part by National Natural Science Foundation of China (62171287), Shenzhen Science and Technology Program (Grant No. JCYJ 20220818100004008).

References

1. Chunlai Y U. Research on key technologies of single-station passive positioning and tracking using spatial frequency domain information [D][J]. 2008.
2. V. J. Aidala, "Kalman Filter Behavior in Bearings-Only Tracking Applications," in IEEE Transactions on Aerospace and Electronic Systems, vol. AES-15, no. 1, pp. 29-39, Jan. 1979, doi: 10.1109/TAES.1979.308793.
3. Blazek J, Jiranek J, Bajer J. Indoor passive positioning technique using ultrawide band modules[C]//2019 International Conference on Military Technologies (ICMT). IEEE, 2019: 1-5.
4. Becker K. Passive localization of frequency-agile radars from angle and frequency measurements[J]. IEEE Transactions on Aerospace and Electronic Systems, 1999, 35(4): 1129-1144.
5. Fu Q, Tian S, Mao X. Target Locating and Tracking Based on the Azimuth and the Change Rate of Doppler Frequency[C]//Proceedings of 2017 Chinese Intelligent Automation Conference. Springer Singapore, 2018: 709-719.
6. WANG J, LV K, LIU M. Maneuvering Target Passive Tracking Algorithm Based on Doppler Frequency Rate and Azimuth[J]. Command Control and Simulation, 2018, 40(4): 38-44.
7. Wu P, Li X. Passive multi-sensor maneuvering target tracking based on UKF-IMM algorithm[C]//2009 WASE International Conference on Information Engineering. IEEE, 2009, 2: 135-138.

8. Liu G F, Gu X F, Wang H N. Design and comparison of two MM algorithms for strong maneuvering target tracking[J]. Journal of System Simulation, 2009, 21(4): 965-968.
9. Blom H A P, Bar-Shalom Y. The interacting multiple model algorithm for systems with Markovian switching coefficients[J]. IEEE transactions on Automatic Control, 1988, 33(8): 780-783.
10. Shuli G, Honglan W, Cheng T, et al. Tracking maneuvering target on airport surface based on IMM-UKF algorithm[C]//2010 International Conference on Optoelectronics and Image Processing. IEEE, 2010, 2: 671-675.
11. Li B, Pang F, Liang C, et al. Improved interactive multiple model filter for maneuvering target tracking[C]//Proceedings of the 33rd Chinese Control Conference. IEEE, 2014: 7312-7316.
12. Gao L, **ng J, Ma Z, et al. Improved IMM algorithm for nonlinear maneuvering target tracking[J]. Procedia Engineering, 2012, 29: 4117-4123.
13. Park Y, Dang L M, Lee S, et al. Multiple object tracking in deep learning approaches: A survey[J]. Electronics, 2021, 10(19): 2406.
14. Li H. Deep learning for natural language processing: advantages and challenges[J]. National Science Review, 2018, 5(1): 24-26.
15. Song L, Shengli W, Dingbao X. Radar track prediction method based on BP neural network[J]. The Journal of Engineering, 2019, 2019(21): 8051-8055.
16. Gao C, Yan J, Zhou S, et al. Long short-term memory-based recurrent neural networks for nonlinear target tracking[J]. Signal Processing, 2019, 164: 67-73.
17. Liu J, Wang Z, Xu M. DeepMTT: A deep learning maneuvering target-tracking algorithm based on bidirectional LSTM network[J]. Information Fusion, 2020, 53: 289-304.
18. Yu W, Yu H, Du J, et al. DeepGTT: A general trajectory tracking deep learning algorithm based on dynamic law learning[J]. IET Radar, Sonar & Navigation, 2021, 15(9): 1125-1150.
19. Zhao G, Wang Z, Huang Y, et al. Transformer-Based Maneuvering Target Tracking[J]. Sensors, 2022, 22(21): 8482.
20. Zhang Y, Li G, Zhang X P, et al. Transformer-based tracking Network for Maneuvering Targets[C]//ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2023: 1-5.
21. Bai S, Kolter J Z, Koltun V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling[J]. ar**v preprint ar**v:1803.01271, 2018.
22. Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.
23. Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J]. Advances in neural information processing systems, 2017, 30.
24. Liu M, Zeng A, Chen M, et al. Scinet: Time series modeling and forecasting with sample convolution and interaction[J]. Advances in Neural Information Processing Systems, 2022, 35: 5816-5828.
25. Liu J, Wang Z, Xu M. A Kalman estimation based rao-blackwellized particle filtering for radar tracking[J]. IEEE Access, 2017, 5: 8162-8174.
26. Li X R, Bar-Shalom Y. Design of interacting multiple model algorithm for tracking in air traffic control systems[C]//Proceedings of 32nd IEEE Conference on Decision and Control. IEEE, 1993: 906-911.
27. Tang G, Müller M, Rios A, et al. Why self-attention? a targeted evaluation of neural machine translation architectures[J]. 2018.DOI:10.18653/v1/D18-1458.