Maneuvering Target Tracking with Recurrent Neural Networks for Radar Application

Chang Gao, Hongwei Liu, Shenghua Zhou, Hongtao Su, Bo Chen, Junkun Yan, Kuiying Yin†
National Laboratory of Radar Signal Processing, Xidian University, Xian 710071, China
Collaborative Innovation Center of Information Sensing and Understanding at Xidian University
† Nanjing Research Institute of Electronic Technology, Nanjing, China
Email: hwliu@xidian.edu.cn, xdugaochang@sina.com

***Abstract*—Maneuvering target tracking is a problem of state estimation where the system undergoes abrupt changes. Traditional model-based approaches to this problem are threatened by the performance degradation caused by the model mismatch and the estimator usually has limited statistical precision in practice. Developed from what the machine sees as mathematically optimal in the data, deep neural network-based methods are not sensitive to variant models as long as the interested motions are fully contained in the training data. Besides, having access to the true state while training a network can make it possible to break the limit of traditional estimation precision. To coincide with the sequential manner of target tracking, the recurrent neural network is proposed to estimate the true state. Simulation results show that the proposed network handles the target motion uncertainty problem well, meanwhile, the states are estimated more accurately.**

I. Introduction

Target tracking is essentially a state estimation problem. The key to successful target tracking lies in the effective extraction of useful information about the target's state from observations. It is well known that the so-called measurement origin uncertainty and target motion uncertainty are two major challenges in target tracking. This research deals only with the second uncertainty, leaving the techniques unique for the data association problems untouched.

Current tracking of airborne targets using noisy radar data typically requires the use of complex digital filtering techniques. A common method has been to model the target dynamics in a rectangular coordinate system which results in a linear set of state equations. The encountered nonlinear problem of measurements is usually solved by linearizing the measurement model or converting polar measurements to rectangular coordinates using debiased transformation. This system works fairly well until the target makes abrupt change in its trajectory. Conventional approaches to this problem can be grouped into two broad categories:

1) Single model with state augmentation.
2) Multiple models with Markovian jumps.

The first category requires maneuver detection and compensation procedures and attempts to reduce the filter bias that arises due to a change in the object maneuver mode [1]. However, these approaches do not give satisfactory tracking performance if the object acceleration changes rapidly. The second category is based on a stochastic hybrid system with multiple models for the object [2]. Among multiple model (MM) algorithms, the interacting multiple model (IMM) algorithm is one of the most computationally efficient approaches [3]. Nonetheless, all multiple model algorithms are limited to the fact that the MM estimator can provide reasonable performance only if the dynamic behavior of the underlying object can be approximated by a small set of models [4]. The inclusion of more models has the potential to degrade performance and increases computational load. Besides, the estimator from conventional model-based methods usually has a limited statistical precision caused by model mismatch and insufficient data for estimation.

Unlike most of the conventional methods that are based on models, deep neural network (DNN) based methods are developed from what the machine sees as mathematically optimal in data [5]. In some sense, the DNN can be considered to approximate a conditional density of outputs data given inputs with a deterministic mapping. Besides, its strong expressive power that any mapping can be approximated to any desired precision with a sufficiently complex network has been approved [6]. Although the training of a deep model is difficult, its implementation can be rather efficient and effective as long as the test data is fully contained in the training data.

For target tracking in radar fields, the motions of typically interested targets can be simulated according to the corresponding models [7], which means that massive data required in the training phase for a DNN is usually not problem. Meanwhile, a DNN with enough expressive power assures that the relation between measurements and true states can be extracted from massive training data consisting of interested motions of targets. Besides, the estimation accuracy can be better for that the network has the access to true states in training. Existing works using DNN-based methods in the radar fields concentrate mainly on the target recognition problem [8]. When it comes to the target tracking-related problem, the neural network is mainly used as a classifier. For instance, the convolutional neural network is used as a binary classifier to distinguish the real target tracks with hand-designed features in track initiation phase [9]. The tracks association between maritime radar and automatic identification system is proposed to solve with the neural network, which also acts as a binary classifier [10]. However, the recent rise of DNN is not limited to the application of classification.

Proposed to take advantage of the dependency between sequential inputs, the recurrent neural network (RNN) can be used to classify as well as generate a sequence according to the input. [5]. It is called recurrent for that it performs the same task for every element of a sequence, with the

output being depended on the previous computations. Every time the recurrent module receives an input, it can generate a corresponding output. With regard to target tracking, the filtered state or a prediction at the next scan is required once a new measurement is received. Besides, the RNN can be regarded as a deterministic approximation to a conditional density, which is also what a target tracking algorithm usually tries to calculate. Observing these similarities, we propose to use the RNN to resolve the target motion uncertainty in the context of radar tracking. The RNN is supposed to take sequential measurements of a target as input and generate corresponding true states or predictions.

The rest of the paper is organized as follows. The problem of maneuvering target tracking is illustrated in the section II. In the section III, the RNN and the proposed networks are described. Simulation results of a civil aircraft tracking example is presented in the section IV. Finally, conclusions are drawn in the section V.

II. Problem Formulation

The typical maneuvering target tracking is essentially a state estimation problem where the system undergoes abrupt changes. Conventional tracking methods are model based, in which the target motion and its observations are assumed to be represented by some known mathematical models sufficiently accurately. The target dynamics are assumed to belong to the set of models defined by

$$\mathbf{x}_{k+1} = f_{r_k}(\mathbf{x}_k) + \mathbf{w}_k,\ r_k \in \{1, 2, ..., d\} \tag{1}$$

where a target is treated as a point object and $\mathbf{x}_k$ is the target state vector. The process noise $\mathbf{w}_k$ is assumed addictive and f_{r_k} is the state transition function assumed to belong to the set of models, in which the impact of system input is included. The measurements are assumed to be model dependent and related to the true target states through

$$\mathbf{y}_k = h_k(\mathbf{x}_k) + \mathbf{v}_k, \tag{2}$$

where $\mathbf{y}_k$ is the measurement of a sensor. The measurement noise $\mathbf{v}_k$ is also assumed addictive and h_k is the measurement function determined by the sensor.

Let $\mathbf{y}^k = (\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_k)$ be a sequence of measurements. From a probabilistic view, the target tracking is mainly calculating the state that maximizes $p(\mathbf{x}_k|\mathbf{y}^{k-1})$ and $p(\mathbf{x}_k|\mathbf{y}^k)$, which correspond to the prediction procedure and filtering procedure, respectively. Amongst the two densities, the conditional density $p(\mathbf{x}_k|\mathbf{y}^k)$ is usually of primary concern in spite of that the predicted density $p(\mathbf{x}_k|\mathbf{y}^{k-1})$ is useful for associating subsequent measurement to the tracked target. Resorting to the Bayesian theory, the predicted density and conditional density of a target under the Markov assumption can be recursively calculated as follows:

$$\begin{aligned} p(\mathbf{x}_k|\mathbf{y}^{k-1}) &= \int p(\mathbf{x}_k|\mathbf{x}_{k-1}) p(\mathbf{x}_{k-1}|\mathbf{y}^{k-1}) d\mathbf{x}_{k-1} \\ p(\mathbf{x}_k|\mathbf{y}^k) &= \frac{1}{p(\mathbf{y}_k|\mathbf{y}^{k-1})} p(\mathbf{y}_k|\mathbf{x}_k) p(\mathbf{x}_k|\mathbf{y}^{k-1}), \end{aligned} \tag{3}$$

which correspond to the prediction step and filtering step, respectively. For a maneuvering target, the challenge of tracking arising from target motion uncertainty refers to the fact that an accurate dynamic model of the state is not available to the sensor. Specifically, the uncertainty of the transition function f_{r_k} and its transition between time series can complicate the calculation of the densities immensely.

The target motions can usually be summed up to limited types or combinations of multiple types [7], [11], which is also the basis of the IMM algorithm. The actual motion of a target in reality is usually regarded as a nearly determined model plus fairly small noise representing the modeling error. For instance, the uniform motion can be described by a second-order nearly constant velocity (CV) model plus a zero-mean Gaussian white noise with an appropriate covariance, which represents the normally quite small linear accelerations. This nearly constant velocity model with addictive noise is given by

$$\mathbf{x}_{k+1} = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}_k + \begin{bmatrix} \frac{1}{2}T^2 & 0 \\ T & 0 \\ 0 & \frac{1}{2}T^2 \\ 0 & T \end{bmatrix} \mathbf{w}_{k+1}, \tag{4}$$

where T is the sampling interval and $\mathbf{x}_k = [\xi\ \dot{\xi}\ \eta\ \dot{\eta}]'$ denotes the coordinates and corresponding velocities in a 2D scene. In the same way, the turning of a target can be described as a nearly coordinated turn (CT) model or a nearly constant acceleration (CA) model plus some minor noise [7], etc.

III. RNN-Based Maneuvering Target Tacking

Inspired by the biological neural networks that constitute animal brains, artificial neural networks (ANNs) are constructed as a computing system to learn tasks by considering examples. An ANN with multiple hidden layers between the input and output layers quantifies a DNN. With a deeper structure, the network can extract more hierarchy feature and thus achieve more expressive power to transform the input to required output. A recurrent neural network (RNN) is a class of ANN where the output of the net is fed back into its input. In this way, an RNN can use its internal memory to process arbitrary sequences of inputs and generate corresponding sequences. Besides, the sequences can be processed and generated sequentially, which is in accord with the target tracking task.

A. Recurrent Neural Network

A conventional RNN is illustrated in Fig. 1. This network is defined by,

$$\begin{aligned} \mathbf{h}_k &= \phi_h(\mathbf{w}^{hi}\mathbf{i}_k + \mathbf{w}^{hh}\mathbf{h}_{k-1}) \\ \mathbf{o}_k &= \phi_o(\mathbf{w}^{oh}\mathbf{h}_k), \end{aligned} \tag{5}$$

where $\mathbf{w}^{hi}$, $\mathbf{w}^{hh}$ and $\mathbf{w}^{oh}$ are respectively input, transition and output matrices, and ϕ_h and ϕ_o are element-wise nonlinear functions. It is usual to use a saturating nonlinear function such as a logistic sigmoid function $\sigma(x) = 1/(1+e^{-x})$ or a hyperbolic tangent function $tanh(x) = (e^x - e^{-x})/(e^x + e^{-x})$. An ANN with a composition of several nonlinear computational layers is usually considered a deep neural network. In that sense, the RNN is already deep since that any RNN can be expressed as a composition of multiple nonlinear layers when unfolded in time. In addition, a commonly used deep version in the sense of the recurrent module is to stack multiple recurrent

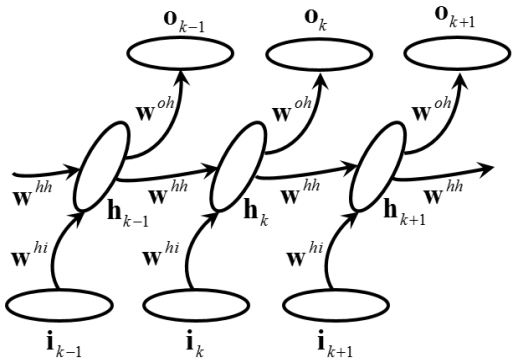

Fig. 1. Illustration of a recurrent neural network. The network consists of input layers, hidden layers and output layers, wherein the hidden layers act as the memory unit. The input layers and output layers receive and output sequences, respectively. Each recurrent module shares the same weights of input-hidden, hidden-output and hidden-hidden layers.

hidden layers on top of each other, which makes the RNN deep at each time step and thus increases its expressive power substantially. This network is usually called the stacked RNN (sRNN).

A neural network can produce the required deterministic output only after trained. The RNN is assumed to take the sequence $\mathbf{i}_1, \mathbf{i}_2, ..., \mathbf{i}_k$ as input, and output $\mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_k$ correspondingly. Given a training set of N sequences $D = \{(\mathbf{i}_1^{(n)}, \mathbf{t}_1^{(n)}), (\mathbf{i}_2^{(n)}, \mathbf{t}_2^{(n)}), ..., (\mathbf{i}_{L_n}^{(n)}, \mathbf{t}_{L_n}^{(n)})\}_{n=1}^{N}$, consisting of input sequences $\mathbf{i}$ and the required output $\mathbf{t}$, the parameters of an RNN can be estimated by minimizing the cost function

$$J(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{L_n} d(\mathbf{o}_k, \mathbf{t}_k), \tag{6}$$

where $d(\mathbf{a}, \mathbf{b})$ is a predefined divergence measure between $\mathbf{a}$ and $\mathbf{b}$, such as Euclidean distance. $\boldsymbol{\theta} = \{\mathbf{w}^{hi}, \mathbf{w}^{hh}, \mathbf{w}^{oh}\}$ represents the network parameters. The parameters are usually estimated by stochastic gradient descent algorithm with the gradient of the cost function in (6) calculated by backpropagation through time (BPTT) [12].

From a probabilistic view, the RNN is to estimate the conditional probability $p(\mathbf{t}_1, \mathbf{t}_2, ..., \mathbf{t}_K | \mathbf{i}_1, \mathbf{i}_2, ..., \mathbf{i}_K)$ using deterministic functions. The computation of the RNN is to parameterize this conditional probability as a product of conditional probabilities such that

$$p(\mathbf{t}_1, \mathbf{t}_2, ..., \mathbf{t}_K | \mathbf{i}_1, \mathbf{i}_2, ..., \mathbf{i}_K) = \prod_{k=1}^{K} p(\mathbf{t}_k | \mathbf{i}^k), \tag{7}$$

where $\mathbf{i}^k = \{\mathbf{i}_1, \mathbf{i}_2, ...\mathbf{i}_k\}$. Each $p(\mathbf{t}_k | \mathbf{i}^k)$ is represented with a recurrent module.

B. RNN-Based Maneuvering Target Tracking

According to the Bayesian tracking theory in (3), the prediction and filtering procedure should be implemented recursively. However, this procedure is not necessary for the design of a neural network. Considering the Bayesian tracking procedure as a whole, the input to the tracking system is the sequential measurements while the output is the corresponding

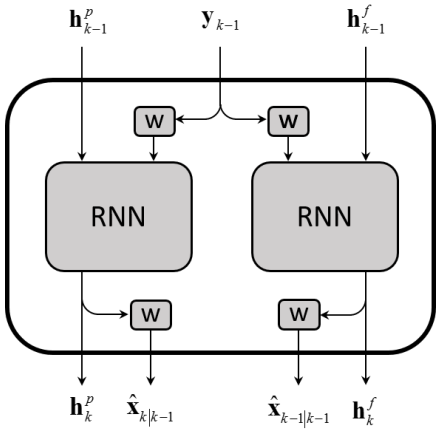

Fig. 2. An illustration of the proposed tracking net. $\mathbf{w}$s are also learnable parameters representing linear transformations to make the dimensions of the input and output the same as hidden layers. The right part uses $\mathbf{y}^{k-1}$ to estimate $\mathbf{x}_{k-1}$ while the left part is to predict $\mathbf{x}_k$ with $\mathbf{y}^{k-1}$. Each of the RNN module memorizes the history of details about prediction and filtering using separate hidden layers. The RNN modules can be replaced by sRNN modules as well.

true state and the predicted state. The information extracted from data stays the same even if the prediction and filtering procedure are implemented separately.

Based on this observation, we propose to use different RNNs to approximate the conditional density $p(\mathbf{x}_k | \mathbf{y}^k)$ and the predicted density $p(\mathbf{x}_k | \mathbf{y}^{k-1})$ separately. We rely on two different RNNs depicted in in Fig. 2 to learn the temporal dynamic model of targets. The input to the network $\mathbf{y}_{k-1}$ is the measurement of a target at time $k-1$. The outputs are the filtered state $\hat{\mathbf{x}}_{k-1|k-1}$ and the predicted state $\hat{\mathbf{x}}_{k|k-1}$. History information about prediction and filtering is memorized and passed forward by different hidden layers $\mathbf{h}_{\cdot}^p$ and $\mathbf{h}_{\cdot}^f$, respectively. The objective functions to optimize for the parameters of different networks are respectively defined by

$$\begin{aligned} J_f(\boldsymbol{\theta}_f) &= \sum_{k=1}^{K} (\hat{\mathbf{x}}_{k|k} - \mathbf{x}_k)^2 \\ J_p(\boldsymbol{\theta}_p) &= \sum_{k=1}^{K} (\hat{\mathbf{x}}_{k|k-1} - \mathbf{x}_k)^2, \end{aligned} \tag{8}$$

where $\mathbf{x}_k$ is the true state of a target, which is known in the training phase. It should be noted that the cost function of a neural network is usually non-convex, which means that many local minima may exit. However, these local minima are usually not a problematic form of non-convexity for that most local minima have a low cost function value, especially for large neural networks. As a result, it is usually not important to find a true global minimum rather than to find a point in parameter space that has low but not minimal cost [5].

C. Network Training

The ANN is usually trained to achieve network parameters using the backpropagation algorithm [13]. As for an RNN, it is usually trained by backpropagation through time (BPTT) because of its temporal structure. Given input and output sequences, the calculation of BPTT is the same as a normal

backpropagation algorithm after the temporal structure of the RNN is unfolded [12]. On account of that the ANN is still a black box to us, some empirical tricks are helpful for the training. The tricks of training an RNN are virtually the same as training an ANN for that RNN is a special kind of ANN, In this section, their training tricks are not distinguished deliberately.

1) Training Data: The ANNs are usually troubled by the overfitting problem that the training data is not enough for the excessive expressive power of the networks. Consequently, different kinds of motions of interested targets should be produced abundantly. As for the design of target motions and their locations, all interested motions ought to be measured sufficiently in the observation area of interest. This is to make sure that the training distribution matches the test distribution, or the models with high training accuracy can do poorly on test data.

2) Dropout: Another way to reduce the overfitting in ANNs is to use the dropout trick, which is also a regularization technique [14]. As the term says, some of the units are dropped out randomly with a certain predetermined probability in every training epoch. Dropout training can also be viewed as a form of ensemble learning. A scaling of the weights admits an approximate computation of geometric mean of the ensemble predictions, as well as avoids the prohibitively expensive calculation of the enormous ensemble learning [14], [15].

3) Clipping Gradients: When training an RNN using BPTT, the gradients can be exponentially large from being multiplied by numbers larger that 1 when propagated back in time, which is usually called the exploding gradient problem [16]. Large gradients can aggravate the learning of parameters in that a gradient descent parameter update could throw the parameters very far, into a region where the objective function is larger, undoing much of the work that had been done to reach the current solution. Gradient clipping will clip the gradients between two numbers to prevent them from getting too large [5].

IV. Simulation Results

In order to show the performance of the proposed tracking net, numerical results will be presented in this section. Here the target motions of interest are the routine maneuvers of civil aircraft, namely the nearly constant velocity movement accomplished by constant acceleration or coordinated turn at a steady turning rate. The interested area is assumed to be observed by a 2-D radar and its extent is $15km \sim 150km$ in range and $0 \sim 90^\circ$ in azimuth. The target is observed every 10 seconds in the tracking stage, while the measurements of range and azimuth are corrupted by white Gaussian noise with standard deviation $30m$ and 0.5°, respectively.

As aforementioned, the proposed tracking net should be trained before used. The tracking networks with recurrent module RNN and sRNN are trained and tested using the same data, respectively. In the training phase, measurements of target motions with initial speed $100m/s \sim 300m/s$ and maximum acceleration $30m/s^2$ or maximum turning rate $4.5^\circ/s$ in the total interested area are produced. The directions of the velocity and acceleration are random. Without loss of generality, the interested target state is assumed to be its location. Precisely the inputs to tracking networks are the sequential measurement while the outputs are trained to be close to the true states. The number of hidden units of the two networks are both 256 and the sRNN has 6 hidden layers. The interacting model with constant velocity model and constant acceleration model (IMM) is used as a baseline for comparison. The set of parameters refers to [17]. Here the measurements in the polar coordinates are converted to rectangular coordinates using the debiased consistent transformation [18].

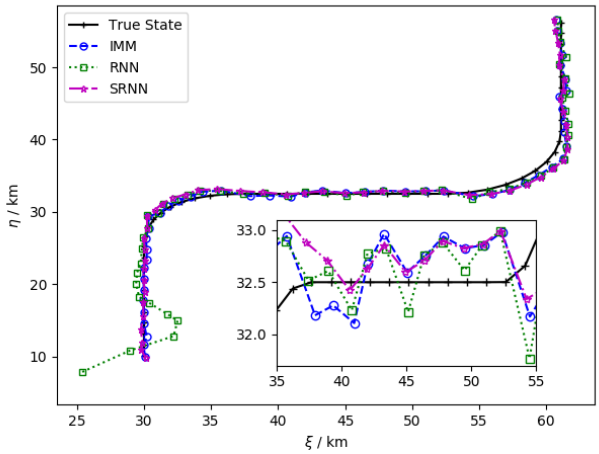

Fig. 3. Tracking at a single time for the proposed networks and the IMM method. The enlarged view is about the tracking between the first and the second maneuver.

In the test phase, measurements of a target with the same certain motion parameters are reproduced. The target is tracked for 50 steps with initial state

$$\mathbf{x}_0 = [30000m\ 0\ 10000m\ 150m/s]', \tag{9}$$

representing the location and velocity of the target in a two-dimensional Cartesian coordinate system, respectively. The constant velocity motion continues for 100 seconds. The first turn is the result of acceleration inputs

$$u_\xi = -u_\eta = 1.5m/s^2, \quad 100s \le t \le 200s. \tag{10}$$

Then the target keeps the velocity constant for another 100 seconds and makes the second turn with turning rate $1^\circ/s$. The turn continues for 90 seconds until the target keeps another constant velocity motion for the last 110 seconds.

The result of tracking at a single time is illustrated in Fig 3. As a result of the simplistic structure, the tracking of the RNN net deviates significantly from the true state at the very start. With a deeper structure to extract the feature from data, the sRNN net can track the target better. Besides, after several steps both of the proposed networks can achieve a better tracking accuracy than the IMM method in this instance.

To be more persuasive, the root means square (RMS) position error is also compared to each other over 5000 Monte Carlo runs, as shown in Fig. 4. The sRNN yields better accuracy and more stable tracking because of its deeper structure. After the first few steps of unstable tracking, the proposed networks can achieve a far better estimation accuracy than the

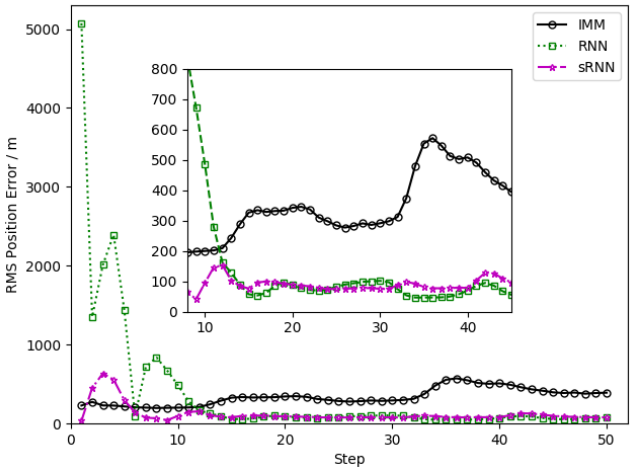

Fig. 4. The filtered RMS position error for the proposed networks and IMM method as a function of the tracking steps. The enlarged view is about the RMS error from the 8th step to the 45th step. Note that the two maneuvers occur from the 10th to the 20th step and from 31st to the 39th step, respectively.

IMM method, which can be regarded to have broke the limit of conventional estimation limit. Besides, the tracking accuracy of the both proposed networks is not as easily influenced by the maneuvers as the IMM method.

At beginning, the inaccurate estimation of the both networks can be caused by the initiation of the hidden states for an RNN, where the hidden states are all set to be zero in our experiments. In the case of that the RNN can achieve better accuracy than the sRNN at a few steps, it can be a result of the increased difficulty of training brought by a deeper structure. Unfortunately, the variance of estimated states in our networks cannot be given as traditional model-based methods. On one hand, it's not appropriate to make the network output the estimation variance like states as corresponding trained targets of variance is hard to achieve. On the other hand, the parameters of a well trained network are achieved through non-convex optimization methods, which complicate the analysis of how the output will respond to unseen inputs.

V. Conclusion

In this paper we proposed to use the RNN to resolve the target motion uncertainty in target tracking. The core idea is that the neural network with a sequential structure, the RNN, is trusted to learn the transformation between measurements and estimated states from large amounts of data, which comprises different kinds of motions of interested targets in the interested area. Numerical results show that, compared to the IMM method, the proposed networks can achieve much better estimation accuracy after few steps of unstable tracking. Besides, the state estimation accuracy of the proposed networks is not as sensitive to the maneuvers as the IMM method. However, the network is hard to understand because of the involved non-convex optimization procedures while training, which directly leads to the difficulty of a quantitative analysis of how good the estimation can be in application. As a result, how to achieve more stable estimations and quantitative analyses of the estimation accuracy need more research in the future.

Acknowledgment

This work is partially supported by the National Science Fund for Distinguished Young Scholars (61525105), the Fund for Foreign Scholars in University Research and Teaching Programs (the 111 Project No. B18039), the Fundamental Research Funds for the Central Universities (JB180215) the National Natural Science Foundation of China (61601340, 61501351).

References

[1] R. Singer, R. Sea, and K. Housewright, "Derivation and evaluation of improved tracking filter for use in dense multitarget environments," *IEEE Transactions on Information Theory*, vol. 20, no. 4, pp. 423–432, 1974.

[2] M. Farooq, S. Bruder, T. Quach, and S. Lim, "Adaptive filtering techniques for manoeuvring targets," in *Circuits and Systems, 1991., Proceedings of the 34th Midwest Symposium on*. IEEE, 1991, pp. 31–34.

[3] E. Mazor, A. Averbuch, Y. Bar-Shalom, and J. Dayan, "Interacting multiple model methods in target tracking: a survey," *IEEE Transactions on aerospace and electronic systems*, vol. 34, no. 1, pp. 103–123, 1998.

[4] S. Challa, *Fundamentals of object tracking*. Cambridge University Press, 2011.

[5] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[6] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[7] X. R. Li and V. P. Jilkov, "Survey of maneuvering target tracking. part i. dynamic models," *IEEE Transactions on aerospace and electronic systems*, vol. 39, no. 4, pp. 1333–1364, 2003.

[8] J. Ding, B. Chen, H. Liu, and M. Huang, "Convolutional neural network with data augmentation for sar target recognition," *IEEE Geoscience and remote sensing letters*, vol. 13, no. 3, pp. 364–368, 2016.

[9] Y. Zhang, S. Yang, H. Li, and H. Mu, "A novel multi-target track initiation method based on convolution neural network," in *Remote Sensing with Intelligent Processing (RSIP), 2017 International Workshop on*. IEEE, 2017, pp. 1–5.

[10] W. Kazimierski, "Proposal of neural approach to maritime radar and automatic identification system tracks association," *IET Radar, Sonar & Navigation*, vol. 11, no. 5, pp. 729–735, 2016.

[11] X. R. Li and V. P. Jilkov, "Survey of maneuvering target tracking: Ii. ballistic target models," in *Signal and Data Processing of Small Targets 2001*, vol. 4473. International Society for Optics and Photonics, 2001, pp. 559–582.

[12] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[13] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, p. 533, 1986.

[14] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *arXiv preprint arXiv:1409.2329*, 2014.

[15] D. Warde-Farley, I. J. Goodfellow, A. Courville, and Y. Bengio, "An empirical analysis of dropout in piecewise linear networks," *arXiv preprint arXiv:1312.6197*, 2013.

[16] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[17] Y. Bar-Shalom, K. Chang, and H. A. Blom, "Tracking a maneuvering target using input estimation versus the interacting multiple model algorithm," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 25, no. 2, pp. 296–300, 1989.

[18] D. Lerro and Y. Bar-Shalom, "Tracking with debiased consistent converted measurements versus ekf," *IEEE transactions on aerospace and electronic systems*, vol. 29, no. 3, pp. 1015–1022, 1993.