Hindawi
Mathematical Problems in Engineering
Volume 2019, Article ID 5491243, 9 pages
https://doi.org/10.1155/2019/5491243

Research Article

A MEMS Gyroscope Noise Suppressing Method Using Neural Architecture Search Neural Network

Zhenshu Zhu, Yuming Bo, and Changhui Jiang

School of Automation, Nanjing University of Science and Technology, Nanjing 210094, China

Correspondence should be addressed to Changhui Jiang; chiang_changhui@outlook.com

Received 15 August 2019; Revised 23 October 2019; Accepted 4 November 2019; Published 21 November 2019

Academic Editor: Ezequiel López-Rubio

Inertial measurement unit (IMU) (an IMU usually contains three gyroscopes and accelerometers) is the key sensor to construct a self-contained inertial navigation system (INS). IMU manufactured through the Micromechanics Electronics Manufacturing System (MEMS) technology becomes more popular, due to its smaller column, lower cost, and gradually improved accuracy. However, limited by the manufacturing technology, the MEMS IMU raw measurement signals experience complicated noises, which cause the INS navigation solution errors diverge dramatically over time. For addressing this problem, an advanced Neural Architecture Search Recurrent Neural Network (NAS-RNN) was employed in the MEMS gyroscope noise suppressing. NAS-RNN was the recently invented artificial intelligence method for time series problems in data science community. Different from conventional method, NAS-RNN was able to search a more feasible architecture for selected application. In this paper, a popular MEMS IMU STIM300 was employed in the testing experiment, and the sampling frequency was 125 Hz. The experiment results showed that the NAS-RNN was effective for MEMS gyroscope denoising; the standard deviation values of denoised three-axis gyroscope measurements decreased by 44.0%, 34.1%, and 39.3%, respectively. Compared with the Long Short-Term Memory Recurrent Neural Network (LSTM-RNN), the NAS-RNN obtained further decreases by 28.6%, 3.7%, and 8.8% in standard deviation (STD) values of the signals. In addition, the attitude errors decreased by 26.5%, 20.8%, and 16.4% while substituting the LSTM-RNN with the NAS-RNN.

1. Introduction

With the booming of the location-based service (LBS), the demand for position, velocity, and time (PVT) information has gained a significant increment [1–5]. Global Navigation Satellite System (GNSS) receiver has been the indispensable equipment for various vehicles, carriers, and smart devices, for instance, unmanned ground vehicles (UGV), unmanned aerial vehicle (UAV), smartphone, and so on [1–5]. With a GNSS receiver, these users are able to obtain accurate PVT information under an open-sky environment [6, 7].

According to the GNSS working principle, GNSS is the radio wireless signal-based navigation system. The receivers operate positioning function relying on receiving the wireless signals from the navigation satellites [8–10]. For GNSS, the navigation satellites in orbit emit the signals to the earth, and the receivers get the signal for obtaining distance between the user and satellite through measuring the transmitting time of the navigation signal. However, while the signal reaches the earth, it is quite weak and easy to be disturbed or interfered [8–10]. As illustrated in past publications, the following two reasons account for this phenomena: (1) the transmitting signal power is limited by the energy of the navigation satellite, saving the energy and for keeping the satellite life span, it is hard to enlarge the navigation signal strength will consume more energy; (2) for saving the cost of constructing the GNSS, while meeting the demand of covering the earth with fewer satellites, the orbit is far away from the earth and the signal transmits a long distance before reaching the earth [8–11]. Due to the above drawbacks, a standalone GNSS is usually unable to output seamless and ubiquitous navigation solutions. Thus, it is of significance for enhancing the performance of the GNSS in signal challenging environments.

In the past decades, researchers are devoted to improving the GNSS receiver performance under weak signals. The first approach is to extend the integration time, which is effective for weak signal tracking [12–14]. The second method is to utilize advanced signal tracking architecture, for instance, vector tracking loop (VTL) [15, 16]. However, apart from the weak signals, the receiver cannot receive all the available satellite signals due to the signal blockage. Under this condition, extending the integration time is usually ineffective. New approaches are proposed to address this situation. According to past investigations, integrating GNSS with other available sensors or navigation system for improving the availability of navigation solutions is the most popular solution [15–17]. Among them, the inertial navigation system (INS)/GNSS integrated navigation system is the most popular, due to the highly complementary characteristics between GNSS and INS [18–21].

The INS is a famous self-contained navigation system, which produces navigation solutions from processing data from inertial measurement unit (IMU). Commonly, an IMU is composed of three orthogonal gyroscopes and accelerometers [18–21]. Gyroscopes measure angular velocity; the accelerometers sense the specific force. Measurements from the IMU are processed together for outputting navigation solution (position, velocity and attitude, PVA) continuously. Due to the random noise contained in the raw signals, the INS navigation solutions diverge dramatically over time. GNSS has precise navigation solutions, which could be integrated with INS for providing more reliable navigation solutions [18–21]. As mentioned above, the INS and GNSS are complementary. Thus, GNSS/INS integration system is always the preferable selection for long-term accurate navigation solutions. While GNSS signal is normal, the GNSS can calibrate the INS and compensate the INS errors. While the signal is abnormal, the INS could provide short-term navigation solutions [18–21]. But, the errors diverge over time, which is determined by the noises contained in the IMU measurements. To further improve the performance of the GNSS/SINS integrated navigation system, it is critical to reducing the IMU measurement noise. Especially, for volume and cost-sensitive applications, Micromechanics Electronics Manufacturing System (MEMS) IMU is usually employed to construct the INS. In fact, there are several kinds of IMUs; among them, MEMS IMU is popular due to low cost and small size. MEMS IMU also performs the lowest accuracy. It is of great significance for reducing or suppressing MEMS IMU noises [18–21].

Firstly, inspired by the time sequence processing in data science community, Allan variance (AV) was employed to analyze the MEMS IMU error components, and then ARMA models are employed for modeling and representing the noise [22–31]. After this, some machine learning methods are also employed in this application, for instance, neural networks and support-vector machine (SVM). With the rapid development of the semiconductor technology and computing capacity, recently, deep learning (DL) gained a boom in data science community [22–31]. Artificial intelligence (AI) methods were employed in sequence data processing and obtained great advances while compared with the conventional machine learning methods [22–31].

In this paper, inspired by the excellent capability of deep neural networks in sequence data processing, the advanced Neural Architecture Search Recurrent Neural Network (NAS-RNN) was employed and tested for reducing the MEMS gyroscope noise (only the gyroscope was tested using the proposed method, and the accelerometers processing was similar to that of gyroscope). A commercial MEMS IMU STIM300 was employed in the experiments [22–31]. The sampling frequency was 125 Hz, and the testing time was approximately 800 seconds.

Specially, Long Short-Term Memory Recurrent Neural Network (LSTM-RNN) and Simple Reduced Unit Recurrent Neural Network (SRU-RNN) were employed in MEMS gyroscope noise suppressing. Both LSTM-RNN and SRU-RNN were popular variants of RNN, and they both performed better than conventional machine learning or regression method [32, 33]. However, it is interesting to explore a more feasible RNN structure in this application. NAS-RNN was able to automatically search for a more appropriate neural architecture for certain application. A popular commercial method was employed in this experiment for testing the NAS-RNN performance, and the LSTM-RNN was selected as the reference for demonstrating the NAS-RNN superiority compared with the LSTM-RNN.

Apart from the introduction section, the rest of the paper is organized as follows: (1) Section 2 introduces the model and the workflow of the employed NAS-RNN model; (2) Section 3 illustrates the NAS implementation, training, results, and the comparison details with the LSTM-RNN; (3) Section 4 concludes the paper and the discussion is added following this section.

2. Methods

In this section, we first introduce the basic architecture and the working flowchart of a popular LSTM-RNN, and then, an RNN generative method using NAS is presented. This section is divided into two subsections: (1) LSTM unit basic structure, relative mathematical equations, and workflow; (2) NAS-RNN mechanics and setting up, especially how NAS-RNN is trained and the logic.

2.1. Long Short-Term Memory Unit. Before introducing the NAS-RNN, Figure 1 gives the basic architecture of the LSTM unit, which is composed of three "gates" including "forget gate," "input gate," and "output gate." Basically, the three different gates have its unique functions, and the details are as following:

(1) "Forget gate" is the first gate, and its function is to decide what information will be thrown away from the previous cell state. As illustrated in Figure 1, the gate takes the variable h_{t-1} and x_t as the inputs of the function, which will generate values representing the forgetting degree of each number in the cell state. The operation equation f_t is as follows:

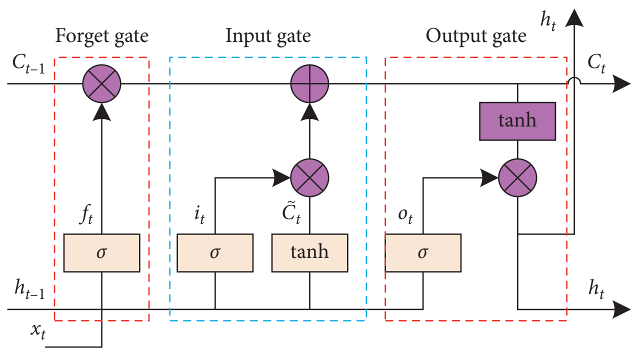


Figure 1: Basic architecture of an LSTM unit.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right), \quad (1)$$

where $\sigma(\cdot)$ is a sigmoid function, W_f is the updating weights, b_f is the bias, h_{t-1} is the hidden state, and x_t is the input vector.

(2) The next is the "input gate," which controls what new information should be inputted to the module. The relative equations are as follows:

$$\begin{aligned} i_t &= \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right), \\ \widetilde{C}_t &= \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right), \end{aligned} \quad (2)$$

here $\sigma(\cdot)$ is a sigmoid function, W_i is the updating weights, b_i is the bias in the input gate, h_{t-1} is the hidden state at time $t-1$, W_C is the updating weights, b_C is the bias, x_t is the input vector, and $\widetilde{C}_t$ is the new cell candidate values. This gate has two functions.

(3) The last part is the "output gate," and by its nature, this gate is designed for managing the outputs of the unit. Two functions are employed in this gate, and the details are given in equations (3) and (4).

$$o_t = \sigma\left(W_o \cdot [h_{t-1}, x_t] + b_o\right), \quad (3)$$

$$h_t = o_t * \tanh(C_t), \quad (4)$$

where W_o is the updating weights, b_o is the bias in the output gate, and C_t is the cell state at time t.

2.2. Neural Architecture Searching Recurrent Neural Network. The last subsection gives the basic LSTM unit computation in detail. In fact, it requires expert knowledge and takes time to design such basic recurrent unit. Recently, a neural architecture searching is proposed for generating neural networks with reinforcement learning [34–37]. Figure 2(a) is the overview of the NAS; the work is based on the observation that the structure and connectivity of a neural network can be typically specified by a variable-length string. It is therefore possible to use a recurrent network, the controller, to generate such string. Training the network specified by the string, the "child network," on the real data will result in accuracy on validation set. Using this accuracy as the renewal signal, we can compute the policy gradient to update the controller. As a result, in the next iteration, the controller will give higher probabilities to architectures that receive high accuracies.

In fact, the basic LSRN-RNN computations could be extended or illustrated as the tree of steps. Figure 3 shows an example of this tree structure, which is composed of two leaf nodes and one internal node. The leaf nodes are indexed by 0 and 1, and the internal node is indexed by 2. In this architecture, the controller RNN needs to predict 3 blocks first, each block specifying a combination method and activation function for each tree index. After this, it needs to predict the last 2 blocks that specify how to connect C_t and C_{t-1} to temporary variables inside the tree. Specifically, according to the predictions of the controller RNN in this example, the following computation steps will occur:

(1) The controller predicts Add and tanh for tree index 0, and this means that we need to compute $a_0 = \tanh(W_1 * x_t + W_2 * h_{t-1})$;

(2) The controller predicts ElemMult and ReLU for tree index 1, and this means that we need to compute $a_1 = \mathrm{ReLU}((W_3 * x_t) \cdot (W_4 * h_{t-1}))$;

(3) The controller predicts 0 for the second element of the "Cell Index"and Add and ReLU for elements in "Cell Inject," which means that we need to compute $a_0^{\mathrm{new}} = \mathrm{ReLU}(a_0 + C_{t-1})$. Notice that we do not have any learnable parameters for the internal nodes of the tree;

(4) The controller RNN predicts 1 for the first element for the "Cell Index," and this means that we should set C_t to the input of the tree at index 1 before the activation, $C_t = ((W_3 * x_t) \cdot (W_4 * h_{t-1}))$.

3. Experiments and Results

In this section, field tests were carried out for evaluating the performance of the proposed method. Moreover, a common LSTM-RNN was employed and compared with the proposed NAS-RNN in terms of the standard deviation of the gyroscope signals and the attitude errors. In this experiment, a popular MEMS IMU STIM300 was adopted, which was manufactured by Sensors AS Company from Norway. The IMU was presented in Figure 4, the gyroscope's full measurement range was $\pm 400/\mathrm{s}$, the gyroscope's bias instability was $0.3°/\mathrm{h}$, the angle random walk was $\leq 0.15°/\sqrt{hr}$, and the sampling frequency was 125 Hz [37].

Overall, this section is organized as follows:

(1) First subpart presented the setting up of the NAS-RNN, and the results of the NAS-RNN, the training loss, the standard deviation values, and attitude errors were employed as indicators for evaluating the performance of the NAS-RNN denoising

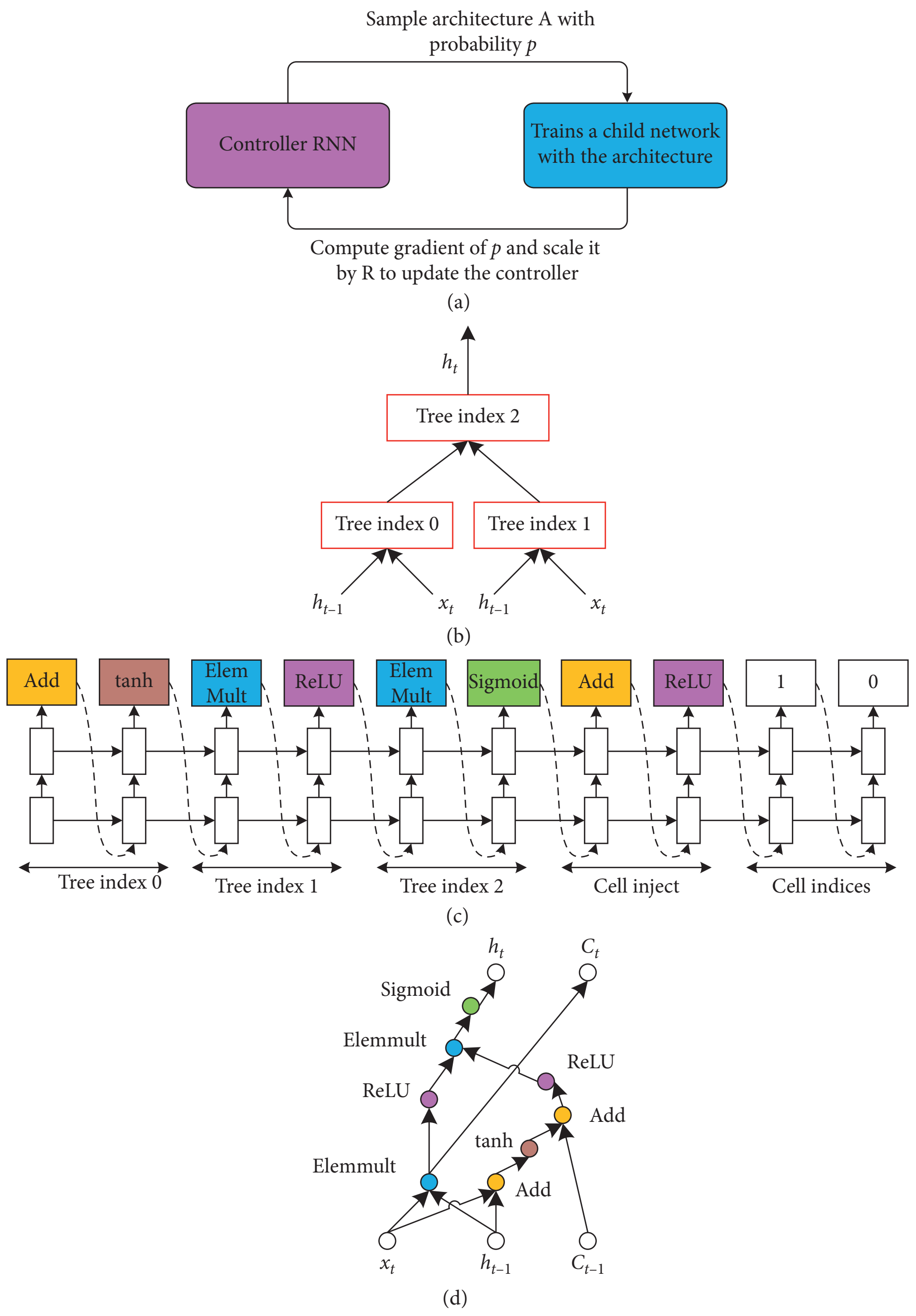
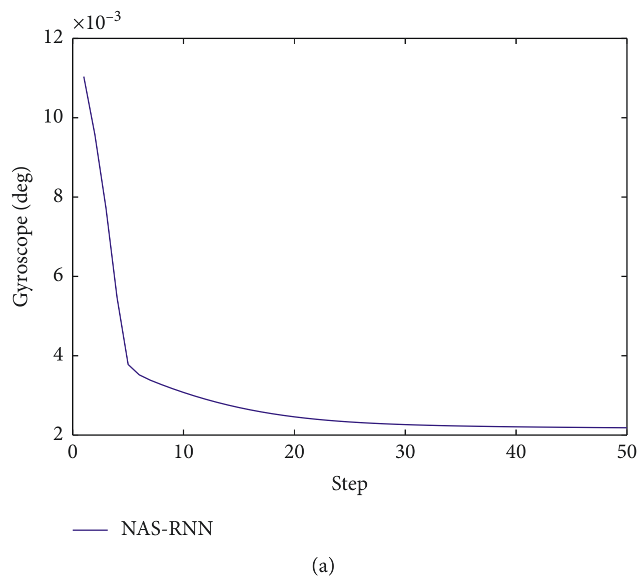


Figure 2: An example of NAS-RNN generation. (a) Overview of the NAS. (b) Tree structure. (c) An example of a recurrent cell constructed from tree (two leaf nodes). (d) Specifications of the connections.

(2) In the second subpart, LSTM-RNN was compared with the proposed method for analyzing the difference between them

3.1. NAS-RNN Results. As aforementioned, Section 2 presented the diagram and the architecture of the NAS-RNN in detail. The NAS-RNN was implemented using the Python programming language (version 3.4) and the popular Google open source deep learning library TensorFlow (version 1.12.0) in this experiment. Among the NAS-RNN parameters, the learning rate was set to 0.01 and the training step was set 100. Input date length was 15, which means using the past 15 gyroscope measurements to predict the current gyroscope outputs. After this, the predicted outputs were employed for compensating the random noise contained in the raw signals. Before applying the NAS-RNN to predict noise, a training procedure was carried out for determining appropriate parameters and generating proper neural network architecture in NAS-RNN.

Figures 3, 5, and 6 list the training loss for the three-axis gyroscope outputs. It could be seen that the NAS-RNN training loss converged during the training procedure. In Figures 5(b), 3(b), and 6(b), the red line represents the NAS-



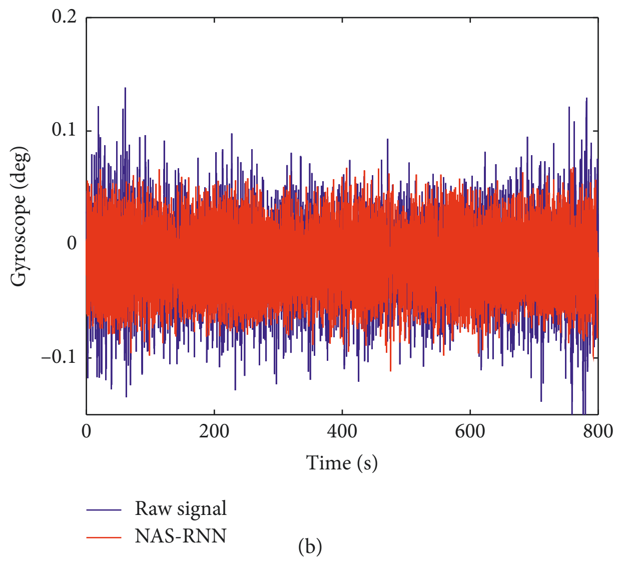
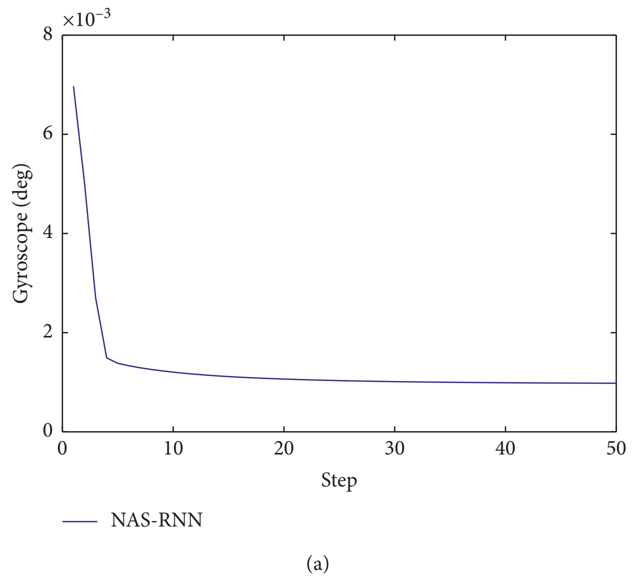


FIGURE 3: *X*-axis gyroscope single-layer GRU-RNN denoising results. (a) Training loss. (b) Comparison of NAS-RNN and raw signals.

FIGURE 4: Inertial measurement unit (IMU).



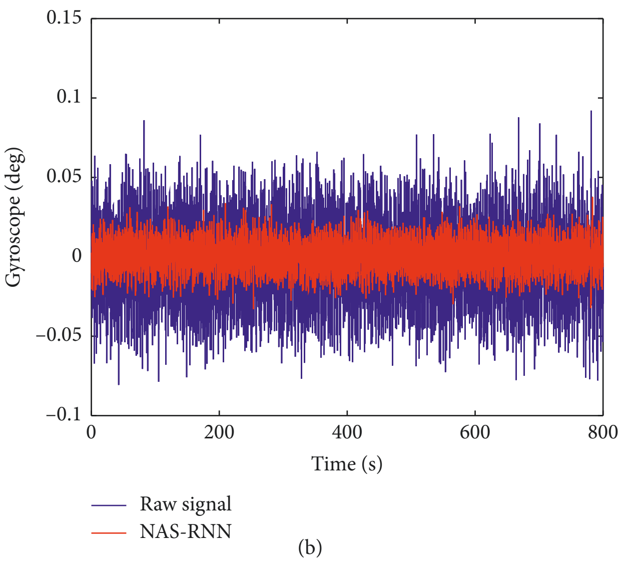
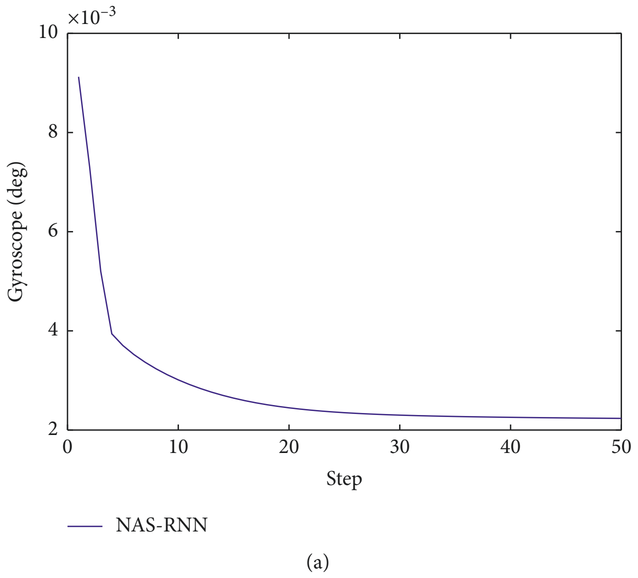


FIGURE 5: *X*-axis gyroscope signal NAS-RNN denoising results. (a) Training loss. (b) Comparison of NAS-RNN and raw signals.



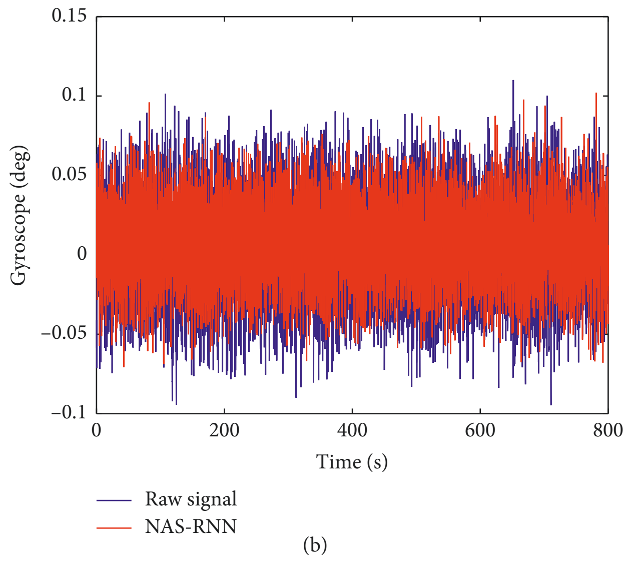


Figure 6: *X*-axis gyroscope single-layer GRU-RNN denoising results. (a) Training loss. (b) Comparison of NAS-RNN and raw signals.

Table 1: Comparison of raw signal and LSTM-RNN standard deviation values.

	X (degree)	*Y* (degree)	*Z* (degree)
Raw signal	0.025	0.041	0.056
LSTM-RNN	0.014	0.027	0.034
Percentage (%)	44.0	34.1	39.3

RNN denoised gyroscope signals and the green line represents the raw gyroscope signals. Furthermore, Table 1 lists the comparison of standard deviation values (SDV) between NAS-RNN and the raw gyroscope signals. Compared with the raw signals, the SDV of the NAS-RNN results decreased by 44.0%, 34.1%, and 39.3%, respectively. This result demonstrated the effectiveness of the NAS-RNN in this MEMS gyroscope noise suppressing application.

3.2. NAS-RNN vs. LSTM-RNN. For further analyzing the NAS-RNN performance in MEMS gyroscope denoising, and in this subsection, the LSTM-RNN was compared with NAS-RNN for exploring whether there was a more feasible architecture for this application. Table 2 lists the comparison results between raw signals and the NAS-RNN. As listed in Table 2, the SDV decreased by 60.0%, 36.6%, and 44.6% compared with the raw signals. Furthermore, Table 3 lists the comparison between LSTM-RNN and NAS-RNN. Compared with LSTM-RNN, the NAS-RNN realized a further decrease in SDV. More specifically, the three-axis SDV decreased by 28.6%, 3.7%, and 8.8% compared with LSTM-RNN.

In addition, Figures 7–9 present the training loss and testing loss comparison between LSTM-RNN and NAS-RNN. It could be seen that NAS-RNN converged to smaller values, but NAS-RNN performed at slower convergence speed. Actually, the NAS-RNN was more complicated than LSTM-RNN, and NAS-RNN had a more heavy computation

Table 2: Raw signal and NAS-RNN comparison.

	X (degree)	*Y* (degree)	*Z* (degree)
Raw signal	0.025	0.041	0.056
NAS-RNN	0.010	0.026	0.031
Percentage (%)	60.0	36.6	44.6

Table 3: LSTM-RNN and NAS-RNN comparison.

	X (degree)	*Y* (degree)	*Z* (degree)
LSTM-RNN	0.014	0.027	0.034
NAS-RNN	0.010	0.026	0.031
Percentage (%)	28.6	3.7	8.8

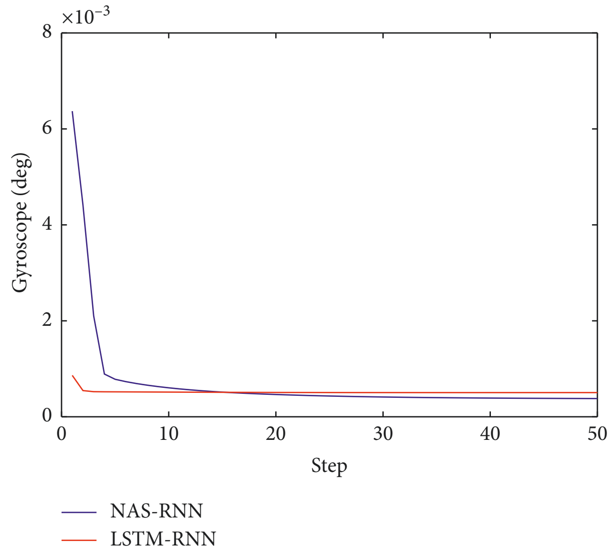


Figure 7: Comparison of *X*-axis gyroscope denoising results.

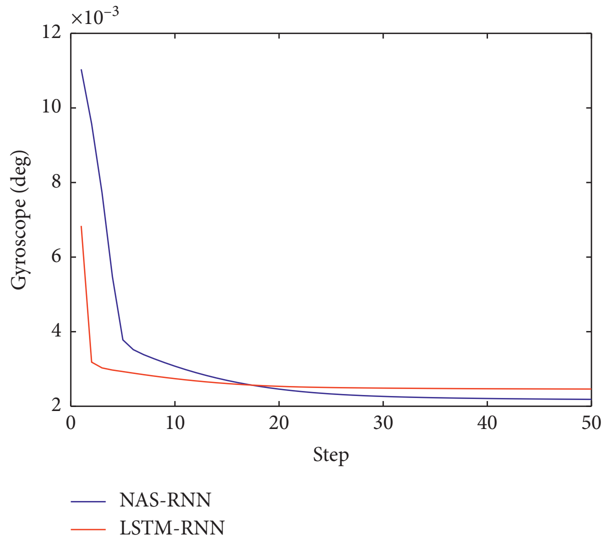


Figure 8: Comparison of Y-axis gyroscope denoising results.

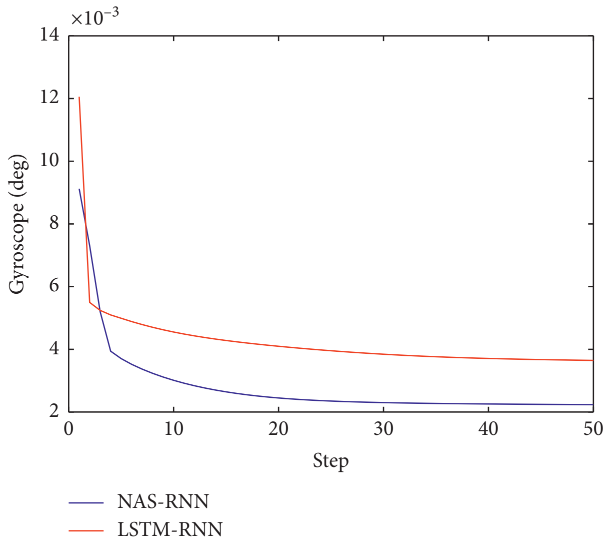


Figure 9: Comparison of Z-axis gyroscope denoising results.

load. More specifications were introduced in Section 2.2. Table 4 lists the comparison of attitude errors; the attitude errors obtained 26.5%, 20.8%, and 16.4% improvement in the three-axis attitude errors compared with LSTM-RNN.

4. Discussion

The motivation of this paper was to explore more feasible RNN for MEMS gyroscope noise modeling and compensation, and we think the following problems need more discussion:

(1) In this paper, the training procedure or processing data was limited by the computation capacity, and this might influence the generation capacity of the neural networks.

(2) Only a STIM300 was employed in testing the proposed method; it was of significance for evaluating the performance of the NAS-RNN in different MEMS IMUs since different IMUs performed different noise characteristics.

Table 4: Comparison of LSTM-RNN and NAS-RNN attitude errors.

	X (degree)	Y (degree)	Z (degree)
LSTM-RNN	0.412	0.524	0.622
NAS-RNN	0.303	0.415	0.520
Percentage (%)	26.5%	20.8%	16.4%

5. Conclusions

In this paper, an advanced variant of RNN was employed in MEMS gyroscope noise modeling and compensating. A commercial MEMS IMU product STIM300 was employed in this experiment for evaluating the performance of the proposed NAS-RNN-based method. Compared with the raw signals, the three-axis NAS-RNN denoised signals performed 60.0%, 36.6%, and 44.6%, decreasing in standard deviation individually. In addition, compared with popular LSTM-RNN, the NAS-RNN obtained 28.6%, 3.7%, and 8.8% further decreasing for the three-axis gyroscope signals. Basically, the NAS-RNN was effective for this application, and it performed better than LSTM-RNN. We hope these results might provide reference for embedding a deep learning module in MEMS gyroscope for improving the accuracy of the MEMS gyroscope.

Data Availability

The source codes used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

The authors gratefully acknowledge the financial support from China Scholarship Council (CSC) and Excellent Doctor Training Fund of Nanjing University of Science and Technology (NJUST). This study was supported by the Fundamental Research Funds for the Central Universities (grant no. 30917011105), Jiangsu Planned Projects for Postdoctoral Research Funds (grant no. 1501050B), and China Postdoctoral Science Foundation (grant no. 2015M580434). Also, this study received the special grade of the financial support from the China Postdoctoral Science Foundation (grant no. 2016T90461).

References

[1] Y. Chen, J. Tang, C. Jiang et al., “The accuracy comparison of three simultaneous localization and mapping (SLAM)-Based indoor mapping technologies,” *Sensors*, vol. 18, no. 10, p. 3228, 2018.

[2] C. Jiang, S. Chen, Y. Bo, Z. Sun, and Q. Lu, “Implementation and performance evaluation of a fast relocation method in a

GPS/SINS/CSAC integrated navigation system hardware prototype," *IEICE Electronics Express*, vol. 14, no. 6, Article ID 20170121, 2017.

[3] T. Gädeke, J. Schmid, M. Zahnlecker, W. Stork, and K. D. Müller-Glaser, "Smartphone pedestrian navigation by foot-IMU sensor fusion," in *Proceedings of the 2012 Ubiquitous Positioning, Indoor Navigation, and Location Based Service (UPINLBS)*, pp. 1–8, Helsinki, Finland, October 2012.

[4] J. Tang, Y. Chen, L. Chen et al., "Fast fingerprint database maintenance for indoor positioning based on UGV SLAM," *Sensors*, vol. 15, no. 3, pp. 5311–5330, 2015.

[5] L. Ruotsalainen, H. Kuusniemi, M. Z. H. Bhuiyan, L. Chen, and R. Chen, "A two-dimensional pedestrian navigation solution aided with a visual gyroscope and a visual odometer," *GPS Solutions*, vol. 17, no. 4, pp. 575–586, 2013.

[6] C. Jiang, Y. Chen, S. Chen et al., "A mixed deep recurrent neural network for MEMS gyroscope noise suppressing," *Electronics*, vol. 8, no. 2, p. 181, 2019.

[7] C. Jiang, S. Chen, Y. Chen, and Y. Bo, "Research on a chip scale atomic clock aided vector tracking loop," *IET Radar, Sonar & Navigation*, vol. 13, no. 7, pp. 1101–1106, 2019.

[8] C. Jiang, S. Chen, Y. Bo, Y. Wang, and Z. Sun, "Performance improvement of GPS/SINS ultra-tightly integrated navigation system utilizing a robust cubature kalman filter," *Journal of Aeronautics, Astronautics and Aviation*, vol. 49, no. 1, pp. 49–55, 2017.

[9] M. Lashley, D. M. Bevly, and J. Y. Hung, "Performance analysis of vector tracking algorithms for weak GPS signals in high dynamics," *IEEE Journal of Selected Topics in Signal Processing*, vol. 3, no. 4, pp. 661–673, 2009.

[10] C. Jiang, S. Chen, Y. Chen, Y. Bo, C. Wang, and W. Tao, "Performance analysis of GNSS vector tracking loop based GNSS/CSAC integrated navigation system," *Journal of Aeronautics, Astronautics and Aviation*, vol. 49, pp. 289–297, 2017.

[11] R. Chen, Y. Chen, L. Pei et al., "A DSP-based multi-sensor multi-network positioning platform," in *Proceedings of the 22nd International Technical Meeting of The Satellite Division of the Institute of Navigation*, vol. 22–25, pp. 615–621, Savannah, GA, USA, September 2009.

[12] M. G. Petovello, C. O'Driscoll, and G. Lachapelle, "Weak signal carrier tracking using extended coherent integration with an ultra-tight GNSS/IMU receiver," in *Proceedings of the ENC-GNSS*, Toulouse, France, April 2008.

[13] J. Tian, L. Yang, and B. Hang, "A novel GNSS weak signal acquisition using wavelet de-noising method," in *Proceedings of the NTM ION*, San Diego, CA, USA, January 2008.

[14] T. Pany and B. Eissfeller, "Use of a vector delay lock loop receiver for GNSS signal power analysis in bad signal conditions," in *Proceedings of the 2006 IEEE/ION Position, Location, And Navigation Symposium*, pp. 893–903, Coronado, CA, USA, April 2006.

[15] K.-H. Kim, G.-I. Jee, and S.-H. Im, "Adaptive vector-tracking loop for low-quality GPS signals," *International Journal of Control, Automation and Systems*, vol. 9, no. 4, pp. 709–715, 2011.

[16] M. Lashley and D. M. Bevly, "Analysis of discriminator based vector tracking algorithms," in *Proceedings of the National Technical Meeting of the Institute of Navigation*, vol. 2007, pp. 570–576, San Diego, CA, USA, January 2007.

[17] X. Chen, X. Wang, and Y. Xu, "Performance enhancement for a GPS vector-tracking loop utilizing an adaptive iterated extended Kalman filter," *Sensors*, vol. 14, no. 12, pp. 23630–23649, 2014.

[18] Z. F. Syed, P. Aggarwal, C. Goodall, X. Niu, and N. El-Sheimy, "A new multi-position calibration method for MEMS inertial navigation systems," *Measurement Science and Technology*, vol. 18, no. 7, pp. 1897–1907, 2007.

[19] C. H. Jiang, S. Chen, Y. Y. Chen, and Y. M. Bo, "Research on chip scale atomic clock driven GNSS/SINS deeply coupled navigation system for augmented performance," *IET Radar, Sonar Navigation*, vol. 13, no. 2, pp. 326–331, 2019.

[20] X. Niu, S. Nassar, and N. El-Sheimy, "An accurate land-vehicle MEMS IMU/GPS navigation system using 3D auxiliary velocity updates," *Navigation*, vol. 54, no. 3, pp. 177–188, 2007.

[21] W. Li and J. Wang, "Effective adaptive Kalman filter for MEMS-IMU/magnetometers integrated attitude and heading reference systems," *Journal of Navigation*, vol. 66, no. 1, pp. 99–113, 2013.

[22] D. Bhatt, P. Aggarwal, V. Devabhaktuni, and P. Bhattacharya, "A novel hybrid fusion algorithm to bridge the period of GPS outages using low-cost INS," *Expert Systems with Applications*, vol. 41, no. 5, pp. 2166–2173, 2014.

[23] N. El-Sheimy, H. Hou, and X. Niu, "Analysis and modeling of inertial sensors using Allan variance," *IEEE Transactions on Instrumentation and Measurement*, vol. 57, no. 1, pp. 140–149, 2008.

[24] D. W. Allan, "Historicity, strengths, and weaknesses of allan variances and their general applications," *Gyroscopy and Navigation*, vol. 7, no. 1, pp. 1–17, 2016.

[25] A. Radi, S. Nassar, and N. El-Sheimy, "Stochastic error modeling of smartphone inertial sensors for navigation in varying dynamic conditions," *Gyroscopy and Navigation*, vol. 9, no. 1, pp. 76–95, 2018.

[26] D. Wang, Y. Dong, Q. Li, Z. Li, and J. Wu, "Using allan variance to improve stochastic modeling for accurate GNSS/INS integrated navigation," *GPS Solutions*, vol. 22, p. 53, 2018.

[27] L. Wang, C. Zhang, S. Gao, T. Wang, T. Lin, and X. Li, "Application of fast dynamic allan variance for the characterization of FOGs-Based measurement while drilling," *Sensors*, vol. 16, no. 12, p. 2078, 2016.

[28] M. Khashei and M. Bijari, "A novel hybridization of artificial neural networks and ARIMA models for time series forecasting," *Applied Soft Computing*, vol. 11, no. 2, pp. 2664–2675, 2011.

[29] A. Waegli, J. Skaloud, S. Guerrier, M. E. Parés, and I. Colomina, "Noise reduction and estimation in multiple micro-electro-mechanical inertial systems," *Measurement Science and Technology*, vol. 21, no. 6, pp. 156–158, 2010.

[30] D. Bhatt, P. Aggarwal, P. Bhattacharya, and V. Devabhaktuni, "An enhanced mems error modeling approach based on nu-support vector regression," *Sensors*, vol. 12, no. 7, pp. 9448–9466, 2012.

[31] H. Xing, B. Hou, Z. Lin, and M. Guo, "Modeling and compensation of random drift of MEMS gyroscopes based on least squares support vector machine optimized by chaotic particle swarm optimization," *Sensors*, vol. 17, no. 10, p. 2335, 2017.

[32] C. Jiang, S. Chen, Y. Chen et al., "A MEMS IMU de-noising method using long short term memory recurrent neural networks (LSTM-RNN)," *Sensors*, vol. 18, no. 10, p. 3470, 2018.

[33] C. Jiang, S. Chen, Y. Chen et al., "Performance analysis of a deep Simple recurrent unit recurrent neural network (SRU-RNN) in MEMS gyroscope de-noising," *Sensors*, vol. 18, no. 12, p. 4471, 2018.

[34] 2015, https://colah.github.io/posts/20150–8-Understanding-LSTMs/.

[35] F. Ordóñez and D. Roggen, "Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition," *Sensors*, vol. 16, no. 1, p. 115, 2016.
[36] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 2016, https://arxiv.org/abs/1611.01578.
[37] https://www.sensonor.com/media/1132/ts1524r9-datasheet-stim300.pdf.