remote sensing

Article

A New Multi-Scale Sliding Window LSTM Framework (MSSW-LSTM): A Case Study for GNSS Time-Series Prediction

Jian Wang [1], Weiping Jiang [2,*], Zhao Li [2] and Yang Lu [2]

[1] School of Geodesy and Geomatics, Wuhan University, Wuhan 430079, China; winner@whu.edu.cn
[2] GNSS Research Center, Wuhan University, Wuhan 430079, China; zhao.li@whu.edu.cn (Z.L.); yang.lu@whu.edu.cn (Y.L.)
* Correspondence: wpjiang@whu.edu.cn

Abstract: GNSS time-series prediction plays an important role in the monitoring of crustal plate movement, and dam or bridge deformation, and the maintenance of global or regional coordinate frames. Deep learning is a state-of-the-art approach for extracting high-level abstract features from big data without any prior knowledge. Moreover, long short-term memory (LSTM) networks are a form of recurrent neural networks that have significant potential for processing time series. In this study, a novel prediction framework was proposed by combining a multi-scale sliding window (MSSW) with LSTM. Specifically, MSSW was applied for data preprocessing to effectively extract the feature relationship at different scales and simultaneously mine the deep characteristics of the dataset. Then, multiple LSTM neural networks were used to predict and obtain the final result by weighting. To verify the performance of MSSW-LSTM, 1000 daily solutions of the XJSS station in the Up component were selected for prediction experiments. Compared with the traditional LSTM method, our results of three groups of controlled experiments showed that the RMSE value was reduced by 2.1%, 23.7%, and 20.1%, and MAE was decreased by 1.6%, 21.1%, and 22.2%, respectively. Our results showed that the MSSW-LSTM algorithm can achieve higher prediction accuracy and smaller error, and can be applied to GNSS time-series prediction.

Keywords: deep learning; long short-term memory; multi-scale sliding window; GNSS; time series; prediction

check for updates

Citation: Wang, J.; Jiang, W.; Li, Z.; Lu, Y. A New Multi-Scale Sliding Window LSTM Framework (MSSW-LSTM): A Case Study for GNSS Time-Series Prediction. *Remote Sens.* **2021**, *13*, 3328. https://doi.org/10.3390/rs13163328

Academic Editor: Nicola Cenni

Received: 1 August 2021
Accepted: 21 August 2021
Published: 23 August 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The long-term accumulated Global Navigational Satellite System (GNSS) coordinate time series provides valuable data for geodesy and geodynamic research [1–3]. These data not only reflect the long-term trend of change, but also represent nonlinear changes caused by geophysical effects. GNSS coordinate time series play an important role in the monitoring of crustal plate movements [4,5], dam or bridge deformation monitoring [6–10], and the maintenance of global or regional coordinate frames [11,12]. The coordinates of the successive time point can be predicted by analyzing the GNSS coordinate time series, thus providing an important basis for judging the motion trend. Therefore, the prediction of GNSS coordinate time series is a highly valuable work.

It is well known that the GNSS coordinate time series reflect both the deterministic law of motion and uncertain information, which may be caused by imperfect processing models, geophysical effects, and other factors that are difficult to model [13]. Two kinds of time-series analysis methods exist: physical modeling and numerical modeling. In the traditional physical and numerical modeling method, models of coordinate time series are constructed according to geophysics theory, the linear term, the periodic term, and gap information [14,15]. Usually, in these traditional modeling methods, the feature information and modeling parameters must be established artificially. The exclusion of elements will lead to systematic deviation and limitations in the results.

Deep learning is an emerging technology that forms a deep architecture by stacking learning modules in a hierarchical structure, and trains the whole network in an end-to-end manner according to gradient training. The deep learning algorithm does not need to artificially select the feature information, and automatically extracts the information suitable for the data characteristics by constructing a complex and precise network [16]. Due to the development of artificial intelligence (AI), an increasing number of powerful algorithms have been applied in different fields and have achieved excellent results. Among these, the recurrent neural network (RNN) is one of the most popular AI methods used for time-series prediction, and can process sequence information and regard the output of the current epoch as the input for the subsequent epoch [17,18]. Its data-driven characteristic can effectively memorize the information of the data. However, because the RNN is subject to the problem of the vanishing gradient, it cannot easily handle long sequences [19]. Thus, Hochreiter and Schmidhuber proposed long short-term memory (LSTM), which avoids the problem of gradient disappearance by optimizing memory cells via the use of gates [20]. LSTM has been widely used to deal with sequence learning problems such as natural language processing (NLP), and has shown significant potential for time-series prediction, such as air quality forecasting, weather forecasting, and traffic flow prediction [21–23].

Recently, LSTM has also been applied in the GNSS field, and has achieved remarkable results. In the monitoring of landslide deformation, Xing et al. proposed a model based on variational mode decomposition (VMD) and a stack LSTM, which had a higher forecast accuracy than that of LSTM and the EMD-LSTM network, in experiments conducted in Dashuitian [24]. Subsequently, Xing et al. combined the double moving average (DMA) method and LSTM to predict landslide displacement, and obtained high-quality confidence intervals [25]. Xie et al. used the LSTM algorithm to predict the periodic component of landslides, and showed that the performance of LSTM has good characteristics of dynamic features [26]. Wang et al. developed an attention mechanism LSTM model based on Complete Ensemble Empirical Mode Decomposition with Adaptive Noise (CEEMDAN-AMLSTM), and confirmed its validity for landslide displacement prediction [27]. Yang et al. used an LSTM model to predict the periodic displacement of landslides in the Three Gorges Reservoir Area, and found that the LSTM method can simulate the dynamic characteristics of landslides better than a static model due to full use of historical information [28].

In navigation and positioning, Tan et al. used LSTM as a de-noising filter and proposed the rEKF-LSTM method to significantly improve single-point positioning accuracy [29]. Jiang et al. proposed an LSTM-RNN algorithm to filter MEMS gyroscope outputs, and the results indicated that the method was effective for improving MEMS INS precision [30]. Kim et al. improved the accuracy and stability of GNSS absolute solutions for autonomous vehicle navigation using LSTM [31]. Tao et al. developed a CNN-LSTM method to mine the deep multipath features in GNSS coordinate series, and showed that the CNN-LSTM can effectively mitigate multi-GNSS multipath issues and reduce the average RMS of positioning errors [32]. In addition, Hoang et al. proposed an LSTM structure for WiFi fingerprinting of indoor localization, and achieved a smaller average localization error than that obtained from other algorithms [33]. Fang et al. used LSTM to support an inertial navigation system (INS), and confirmed that the algorithm can enhance the navigation accuracy compared with pure INS [34]. The above research shows that LSTM has produced good results in both deformation monitoring and positioning in the GNSS field, and the use of deep learning has gradually become more common, providing new ideas for research.

Prior to the use of LSTM, the data must be preprocessed. The traditional approach of a single sliding window is widely used in the existing research on data preprocessing. A review of studies of image processing shows that the multiscale sliding window is widely used in this area, and has achieved good results, because it can take into account information at different scales. The multiscale sliding window is a feature extraction method for image processing in the field of computer vision [35,36] that is able to consider the feature information at different scales. In this study, we applied the idea of the multiscale sliding window to one-dimensional time-series data. Furthermore, we applied the algorithm that was originally conceived for application to two-dimensional data, to one-dimensional data, thus providing a new idea for the use of LSTM.

In this study, we proposed a multiscale sliding window LSTM (MSSW-LSTM) approach for GNSS time-series prediction. The new method uses several different sliding windows for data preprocessing that can capture data information at different scales. Then, the preprocessed outputs are used as inputs into the corresponding LSTM, and each LSTM can be adjusted according to the data. The structure of this article is as follows: Section 2 details the methodology for the MSSW-LSTM. Then, the data and processing strategy are introduced in Section 3. Section 4 analyses the experimental results, and a discussion and conclusions are given in Section 5.

2. Methodology

2.1. LSTM

The traditional neural network model does not encompass the processing information of the previous time span, but only concerns information of the current time. In contrast, the RNN has a memory function, which provides information of the current moment to the subsequent moment. However, the long-term dependence of the RNN leads to gradient explosion. By comparison, LSTM can avoid the problem of gradient disappearance by optimizing memory cells, via the introduction of the concept of gates.

As shown in Figure 1a, a typical LSTM cell has three gates, i.e., input gate, forget gate, and output gate. The cell state and output hidden state are also cores of the LSTM cell.

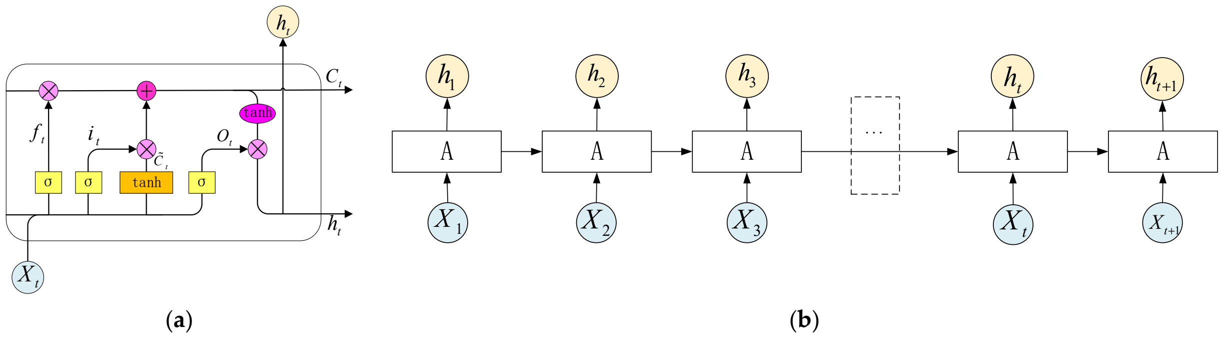


Figure 1. (**a**) Long short-term memory architecture; (**b**) typical structure of LSTM (1 layer).

The single-layer and multi-layer LSTM models are shown separately in Figures 1b and 2.

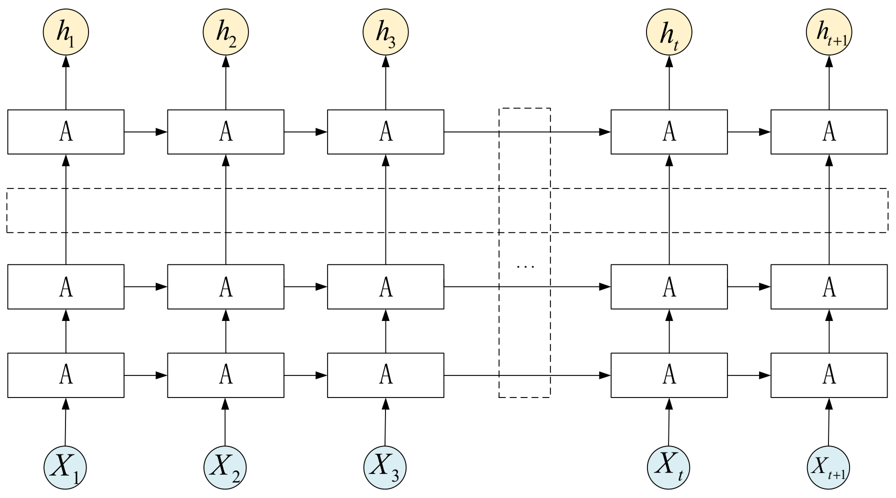


Figure 2. Multilayered neural networks of LSTM.

The definition of the forget gate can be written as:

$$f_t = \sigma(W_{fh}h_{t-1} + W_{fx}x_t + b_f) \tag{1}$$

where σ is the logistic sigmoid function, W_{fh}, W_{fx} are the weight matrix for transformation of information from cell to gate vectors, h_{t-1} is the input of the previous time, x_t is the input of the current time, b_f is the offset value of the forget gate, and f_t is the forget gate of the moment t. The forget gate combines the input h_{t-1} of the previous time with the input x_t of the current time to selectively forget the content.

The input gate can be shown as:

$$i_t = \sigma(W_{ih}h_{t-1} + W_{ix}x_t + b_i) \tag{2}$$

$$\widetilde{C}_t = \tanh(W_{ch}h_{t-1} + W_{cx}x_t + b_c) \tag{3}$$

where σ and tanh are activation functions, W_{ih}, W_{ix}, W_{ch}, W_{cx} are weight matrixes, h_{t-1} is the input of the previous time, x_t is the input of the current time, b_i and b_c are offset values of the input gate, and i_t and $\widetilde{C}_t$ are the input gates of the t moment. The input gate combines the input h_{t-1} of the previous time with the input x_t of the current time to selectively remember the content.

The definition of the cell state update can be written as:

$$C_t = f_tC_{t-1} + i_t\widetilde{C}_t \tag{4}$$

where f_t is the forget gate, C_{t-1} represents the information of the previous moment on the main line, and i_t is the input gate. $\widetilde{C}_t$ denotes information that should be memorized at time t, and C_t indicates the cell state of the main line. The main line cells selectively remember and forget the current input information. Finally, the output gate can be obtained by:

$$O_t = \sigma(W_{oh}h_{t-1} + W_{ox}x_t + b_o) \tag{5}$$

$$h_t = O_t\tanh(C_t) \tag{6}$$

where σ and tanh are activation functions, W_{oh} and W_{ox} are weight matrixes, h_{t-1} indicates the input of the previous time, x_t is the input of the current time, b_o denotes offset values of the input gate, O_t represents the output gate, C_t is the cell state of the main line, and h_t denotes the output of the t moment.

2.2. Multi-Scale Sliding Window LSTM

The sliding window, usually when dealing with two-dimensional images, is widely used in computer vision processing, such as in the fields of object detection and semantic segmentation. In this study, the concept of the sliding window was applied to data preprocessing. because GNSS coordinate time series are one dimensional, the sliding window was reduced to one dimension to construct the data sets. Traditional data preprocessing uses a single-scale sliding window to establish the initial data, as shown in Figure 3, among which the *length_x* and *length_y* are unique. The current LSTM research on time series uses a single-scale sliding window, or other transformations of the data. However, the information captured by a single scale at each time has a fixed scale, and this method of constructing a dataset is not perfect. The construction of the dataset may determine the accuracy of the model training. In this study, we proposed the method of a multiscale sliding window to input different scale information into the corresponding network, form a unified dimension, and integrate the existing research into a unified processing framework.

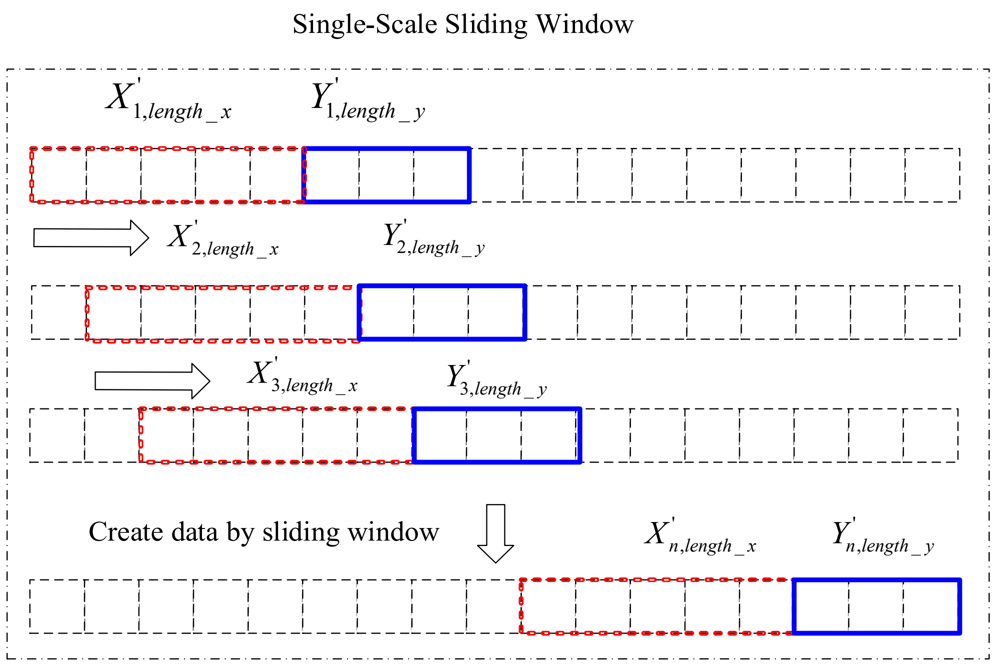


Figure 3. Single-scale sliding window.

The GNSS coordinate time series are obtained and arranged in a unified dimension according to the time sequence:

$$x_1, x_2, x_3, \ldots, x_{m-2}, x_{m-1}, x_m, \text{ recorded as } X_m \tag{7}$$

where m is the length of X. The interval of the GNSS time series should a adopt uniform dimension, such as seconds, minutes, hours, days, weeks, months, or years. The construction the of multiscale sliding window is undertaken as follows:

Assume that the length of the front portion in the ith sliding window is $length\_xi$, and the length of the back portion is $length\_yi$. At each time, one unit is moved to sequentially construct the data, and the following conditions are required $v \le m - (length\_xi + length\_yi) + 1$. The data formats are as follows:

$$\begin{bmatrix} X^i_{1,length\_xi}, Y^i_{1,length\_yi} \\ X^i_{2,length\_xi}, Y^i_{2,length\_yi} \\ \ldots \\ X^i_{v,length\_xi}, Y^i_{v,length\_yi} \end{bmatrix} \tag{8}$$

In the multiscale mode, $k \ge i \ge 2$, where k represents a total of k scales. $length\_x1$, $length\_x2$, … , $length\_xk$ are not equal because it would be meaningless to construct duplicate data sets. However, $length\_y1$, $length\_y2$, and $length\_yk$ are equal, which is convenient for the final result of the weighting calculation.

The constructed data set is shown in Equation (9) and Figure 4:

$$\begin{bmatrix} X^1_{1,length\_x1}, Y^1_{1,length\_y1} \\ X^1_{2,length\_x1}, Y^1_{2,length\_y1} \\ \ldots \\ X^1_{u,length\_x1}, Y^1_{u,length\_y1} \end{bmatrix}, \begin{bmatrix} X^2_{1,length\_x2}, Y^2_{1,length\_y2} \\ X^2_{2,length\_x2}, Y^2_{2,length\_y2} \\ \ldots \\ X^2_{p,length\_x2}, Y^2_{p,length\_y2} \end{bmatrix}, \ldots, \begin{bmatrix} X^k_{1,length\_xk}, Y^k_{1,length\_yk} \\ X^k_{2,length\_xk}, Y^k_{2,length\_yk} \\ \ldots \\ X^k_{q,length\_xk}, Y^k_{q,length\_yk} \end{bmatrix} \tag{9}$$

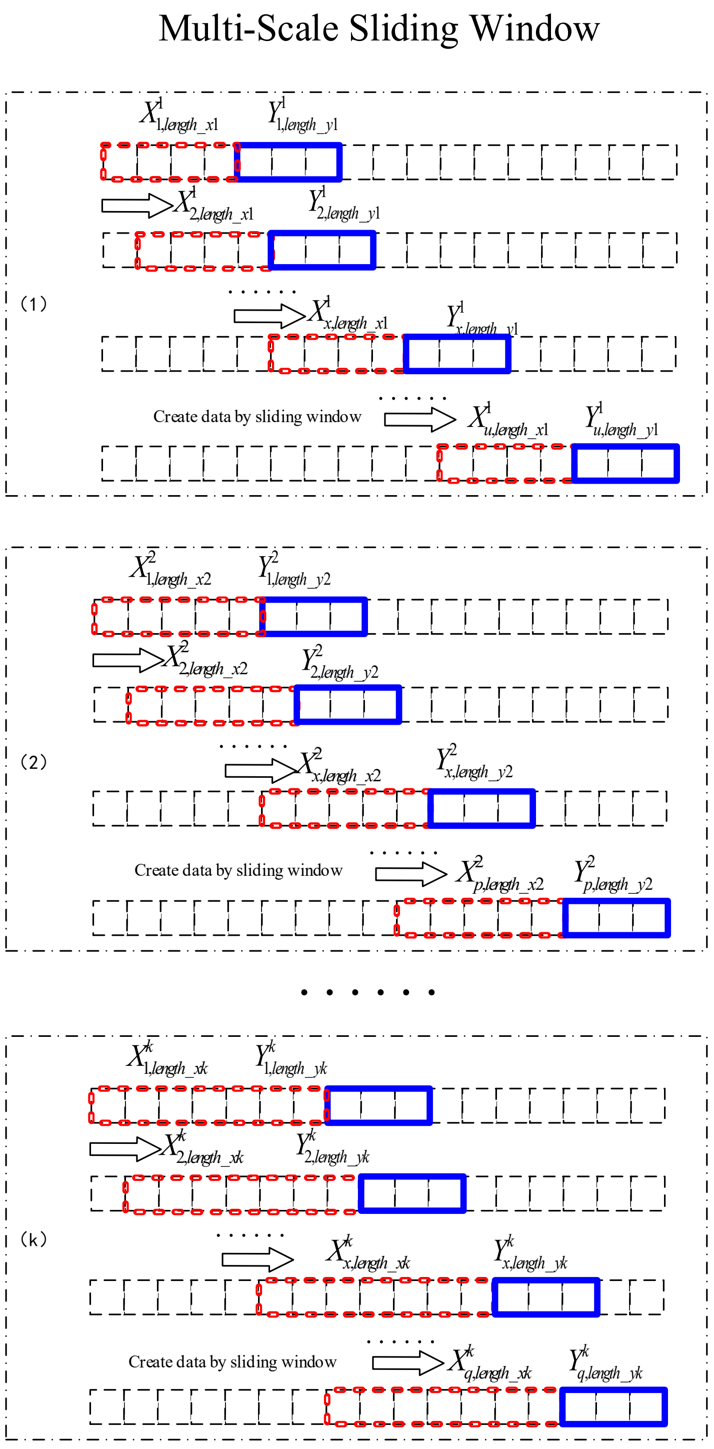


Figure 4. Constructing data with *k* sliding windows of different scales.

Figure 4 is a schematic diagram of K sliding windows of different scales. It can be seen that the sizes of the red sliding windows are different at different scales, and the sizes of the blue sliding windows are the same.

Thus, an MSSW-LSTM algorithm for GNSS time-series prediction was proposed. The overall processing flow of MSSW-LSTM is shown in Figure 5. First, the GNSS station coordinate time series is obtained, and different datasets are constructed using the multiscale window. Following the construction of the datasets, the corresponding LSTM subnetworks are established for each data set according to the actual situation of the data set.

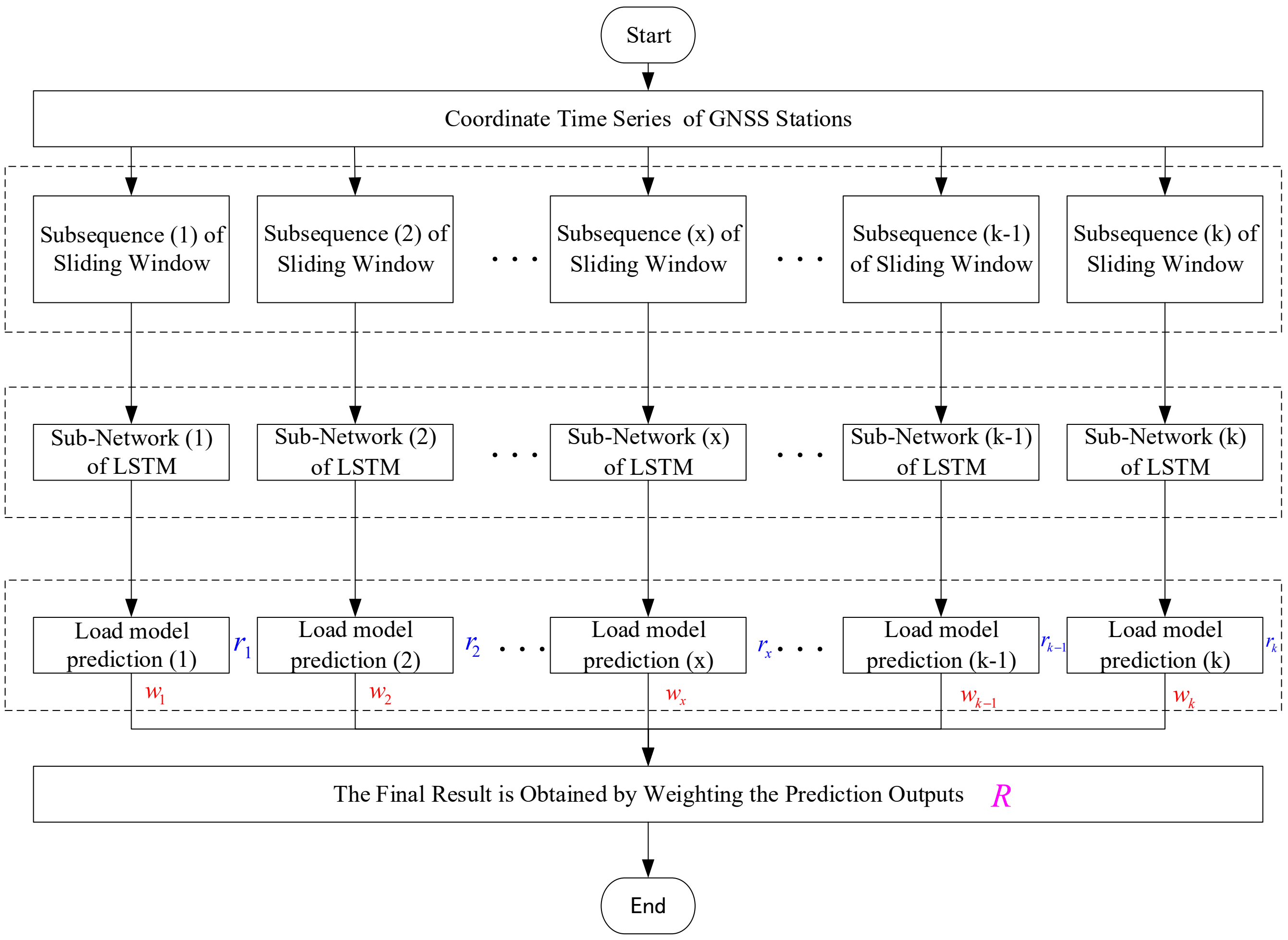

Start
Coordinate Time Series of GNSS Stations
Subsequence (1) of Sliding Window
Subsequence (2) of Sliding Window
Subsequence (x) of Sliding Window
Subsequence (k-1) of Sliding Window
Subsequence (k) of Sliding Window
Sub-Network (1) of LSTM
Sub-Network (2) of LSTM
Sub-Network (x) of LSTM
Sub-Network (k-1) of LSTM
Sub-Network (k) of LSTM
Load model prediction (1)
Load model prediction (2)
Load model prediction (x)
Load model prediction (k-1)
Load model prediction (k)
r_1
r_2
r_x
r_{k-1}
r_k
w_1
w_2
w_x
w_{k-1}
w_k
The Final Result is Obtained by Weighting the Prediction Outputs R
End

Figure 5. Framework of the multiscale sliding window LSTM (MSSW-LSTM).

Each LSTM sub network has its own weight matrix after training, adjustment, and optimization. The trained parameters are saved, and the model of each subnetwork is used for prediction. The prediction results of each subnetwork (1) r_1, subnetwork (2) r_2, . . . , sub network (k) r_k are then obtained.

The final prediction value R is the weighted value of each subnetwork prediction result, and the calculation formula is shown in Equation (10):

$$R = r_1 \times w_1 + r_2 \times w_2 + \ldots + r_k \times w_k \tag{10}$$

$$w_1 + w_2 + w_3 + \ldots + w_{k-1} + w_k = 1 \tag{11}$$

where $w_1, w_2, \ldots, w_{k-1}$ and w_k are the weights of the prediction results from each subnetwork. The sum of all weight values should be 1, as shown in Equation (11).

In general, if there is little difference between the subnetworks, the weight value of each subnetwork should be the same, as shown in Equation (12).

$$w_1 = w_2 = w_3 = \ldots = w_{k-1} = w_k = \frac{1}{k} \tag{12}$$

It should be noted that the MSSW-LSTM method has significant flexibility. For example, LSTM networks may be the same or different, and may consistent of a single layer or multiple layers. This flexibility is beneficial for researchers, who are able to select the most appropriate network model according to their own dataset characteristics and utilize the advantages of the network model.

2.3. Evaluation Criteria

To quantitatively evaluate the prediction accuracy of our proposed model, some indexes are used to calculate the difference between the real value and the predicted value. Here, the root mean square error (RMSE) and the mean absolute error (MAE) are used to evaluate the prediction accuracy [37], and the corresponding formulas are shown as below.

$$RMSE = \sqrt{\frac{1}{N} * \sum_{i=1}^{N} \left(y_i - \widehat{y}_i\right)^2} \tag{13}$$

$$MAE = \frac{1}{N} * \sum_{i=1}^{N} \left|y_i - \widehat{y}_i\right| \tag{14}$$

where N is the number of datasets, and y_i are true values and $\widehat{y}_i$ are predicted values.

3. Data and Processing Strategy

3.1. MSSW-LSTM Process Strategy

The data processing strategy proposed in this paper is shown in Figure 6a, and the main steps of the MSSW-LSTM algorithm are described in Table 1.

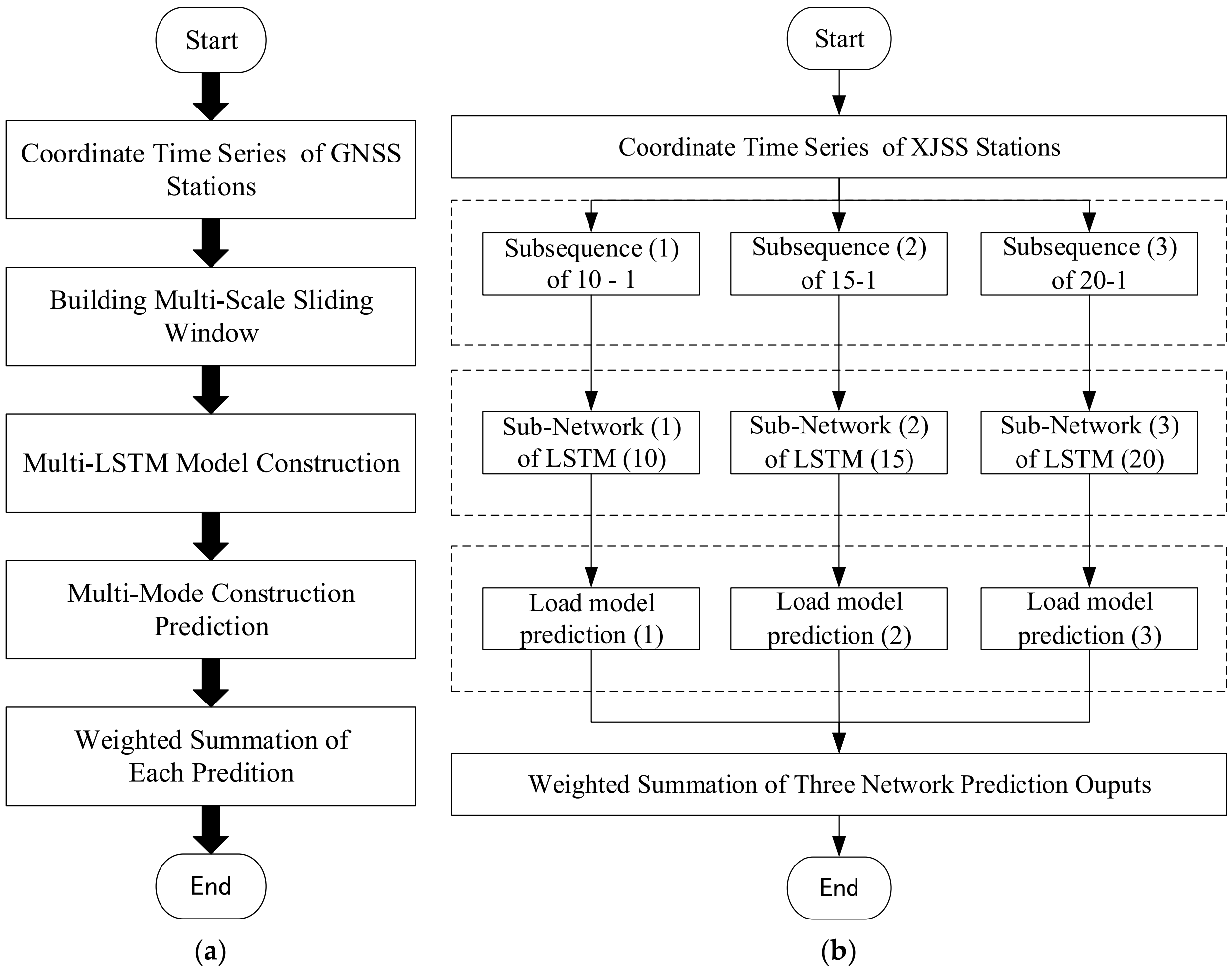

Start
Coordinate Time Series of GNSS Stations
Building Multi-Scale Sliding Window
Multi-LSTM Model Construction
Multi-Mode Construction Prediction
Weighted Summation of Each Predition
End
(a)
Start
Coordinate Time Series of XJSS Stations
Subsequence (1) of 10 - 1
Subsequence (2) of 15-1
Subsequence (3) of 20-1
Sub-Network (1) of LSTM (10)
Sub-Network (2) of LSTM (15)
Sub-Network (3) of LSTM (20)
Load model prediction (1)
Load model prediction (2)
Load model prediction (3)
Weighted Summation of Three Network Prediction Ouputs
End
(b)

Figure 6. (**a**) Flowchart of MSSW-LSTM; (**b**) data processing of XJSS using MSSW-LSTM.

Table 1. Training the MSSW-LSTM algorithm.

Step	Description
1	The GNSS coordinate time series is obtained by actual observation or solution, which should have dimensional consistency, such as weeks, days, hours, seconds. • Preparation of the data
2	The new sub-sequence is constructed by MSSW. Each sub-sequence can be divided into training and validation datasets. • Preparation of the multiscale sliding windows • Define the training and validation dataset • Data regularization
3	The sub-LSTM network is constructed for each corresponding data set, and the constructed network is trained and saved separately. In practice, to reduce running time and computing space, the network model should be simple and practical. • Define each LSTM layer and cell number • Define hyperparameters for each net
4	Training and preserving network structures The final prediction result is obtained by weighted summation. • Apply the well-trained networks to prediction • Define weighted values and calculate

Figure 6b shows the specific method of using the MSSW-LSTM algorithm to process the XJSS station. Three sliding windows with different scales were used to preprocess the GNSS time series. Accordingly, three sub-sequence sets were obtained, and three different LSTM networks were then established. For specific processing, please refer to Section 3.2.

3.2. MSSW-LSTM Processing for the XJSS Station

To more accurately verify the effectiveness of the MSSW-LSTM algorithm, we directly selected a real dataset, rather than simulated one, with a long time span. Through screening, a daily coordinate time series of the XJSS station in the Up component, representing a total of 1000 epochs with high data integrity, was finally selected as the experimental data. The data collection period was from 20,110,412 to 20,140,206, and data were obtained from the China Earthquake Networks Center. The overall data processing flow is shown in Figure 6b.

The experimental data $x_1,\ x_2,\ x_3,\ x_4,\ x_5, \ldots,\ x_{997},\ x_{998},\ x_{999},\ x_{1000}$ can be recorded as X_{1000}.

Then, we preprocessed the data and constructed a multiscale sliding window to form a new sub-sequence. Here, we first provide two definitions. The fixed sliding window length is the length of the training data entered at each time, and the predicted length denotes the data label, which represented the true value. In total, three sliding windows were constructed, as follows:

1. The fixed sliding window length was 10, and the predicted length was 1;
2. The fixed sliding window length was 15, and the predicted length was 1;
3. The fixed sliding window length was 20, and the predicted length was 1;

As shown below, the first sub-sequence had a fixed sliding window of 10 and a data label length of 1, resulting in the construction of 990 available datapoints:

$$\begin{bmatrix} [x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}] \\ [x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}] \\ \ldots \\ [x_{989}, x_{990}, x_{991}, x_{992}, x_{993}, x_{994}, x_{995}, x_{996}, x_{997}, x_{998}] \\ [x_{990}, x_{991}, x_{992}, x_{993}, x_{994}, x_{995}, x_{996}, x_{997}, x_{998}, x_{999}] \end{bmatrix} \text{ and } \begin{bmatrix} x_{11} \\ x_{12} \\ \ldots \\ x_{999} \\ x_{1000} \end{bmatrix} \tag{15}$$

The second sub-sequence had a fixed sliding window of 15 and a data label length of 1, resulting in the construction of 985 available datapoints, namely:

$$\begin{bmatrix} [x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}] \\ [x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_{16}] \\ \ldots \\ [x_{984}, x_{985}, x_{986}, x_{987}, x_{988}, x_{989}, x_{990}, x_{991}, x_{992}, x_{993}, x_{994}, x_{995}, x_{996}, x_{997}, x_{998}] \\ [x_{984}, x_{985}, x_{986}, x_{987}, x_{988}, x_{989}, x_{990}, x_{991}, x_{992}, x_{993}, x_{994}, x_{995}, x_{996}, x_{997}, x_{998}, x_{999}] \end{bmatrix} \text{ and } \begin{bmatrix} x_{16} \\ x_{17} \\ \ldots \\ x_{999} \\ x_{1000} \end{bmatrix} \tag{16}$$

Similarly, the third sub-sequence had a fixed sliding window of 20 and a data label length of 1, resulting in the construction of 980 available datapoints, namely:

$$\begin{bmatrix} [x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_{16}, x_{17}, x_{18}, x_{19}, x_{20}] \\ [x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_{16}, x_{17}, x_{18}, x_{19}, x_{20}, x_{21}] \\ \ldots \\ [x_{979}, x_{980}, x_{981}, x_{982}, x_{983}, x_{984}, x_{985}, x_{986}, \ldots, x_{993}, x_{994}, x_{995}, x_{996}, x_{997}, x_{998}] \\ [x_{980}, x_{981}, x_{982}, x_{983}, x_{984}, x_{985}, x_{986}, \ldots, x_{993}, x_{994}, x_{995}, x_{996}, x_{997}, x_{998}, x_{999}] \end{bmatrix} \text{ and } \begin{bmatrix} x_{21} \\ x_{22} \\ \ldots \\ x_{999} \\ x_{1000} \end{bmatrix} \tag{17}$$

During training, data normalization cannot be ignored. To ensure the stability of the data, they were preprocessed and normalized, and the attributes were scaled to between 0 and 1.

$$x' = (x - x.\text{min})/(x.\text{max} - x.\text{min}) \tag{18}$$

Following preparation of the datasets, they were divided into a training set and a validation set. Usually, the training set comprised 70% of the data and the verification set the remaining 30%. In our case, there were 990, 985, and 980 sub-sequence datasets in the first, second, and third groups, respectively. To ensure that prediction results of the three networks can be used when the final verification set is weighted, the number of verification sets should be consistent; thus, the final number of verification sets was 294 (980 × 0.3 = 294). The specific number of datapoints is shown in Table 2.

Table 2. Number of training and validation datapoints.

	Sub-Sequence (1)	**Sub-Sequence (2)**	**Sub-Sequence (3)**
Total	990	985	980
Training	696	691	686
Validation	294	294	294

Following the compilation of the dataset, the LSTM networks were constructed. In this experiment, a total of three sub-sequences were constructed, so three LSTM networks were required to be established correspondingly. The construction of the network should set reasonable parameters according to the actual situation. Through preliminary experiments, we found that the single-layer LSTM network was sufficient to train and simulate data. Therefore, to save operating costs and calculation space, a smaller model should be used in practice. The parameters of the three LSTM networks constructed in this study are shown in Table 3.

Table 3. Hyperparameter set.

	Sub-LSTM(1)	**Sub-LSTM(2)**	**Sub-LSTM(3)**
Layers	1	1	1
Hidden Cells	10	15	20
Learning Rate	0.01	0.01	0.01
Train Window	10	15	20
Predict Window	1	1	1
Epochs	5000	5000	5000

The numbers of hidden cells in the three subnetworks were 10, 15, and 20, respectively. It should be noted that it was coincidental that the number of hidden cells was consistent with the size of the training window. The learning rate and epoch of the three networks were the same, i.e., 0.01 and 5000, respectively, and the Adam Optimizer was chosen as the stochastic optimization algorithm.

4. Experimental Result and Analysis

4.1. Experiment Results of Three Networks

In this study, data were collected from the XJSS station. The specific data preparation and distribution are shown in Table 2. The main hyperparameters of the three networks are shown in Table 3.

The training and prediction results of these three neural networks with different settings using different scale windows are shown in Figures 7–9, respectively.

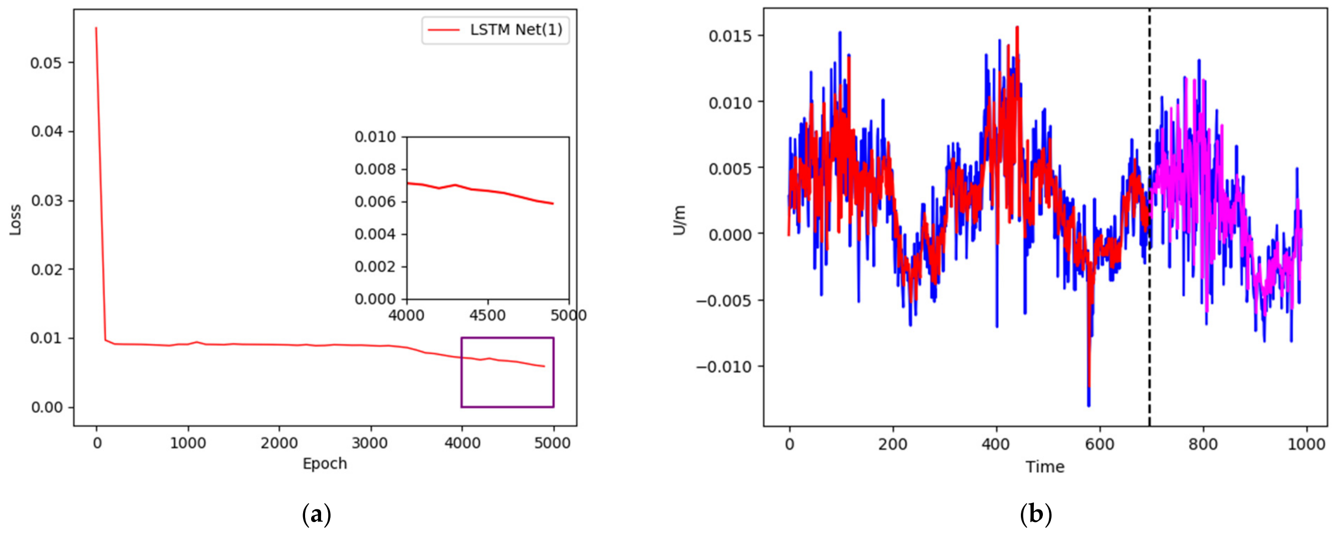
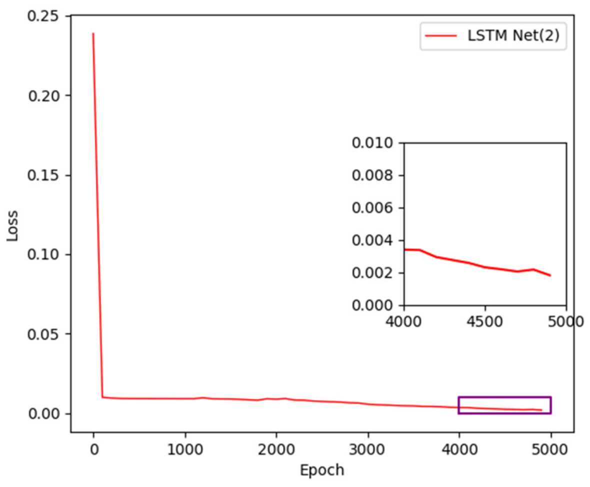


Figure 7. (**a**) Training loss curve; (**b**) data training and prediction of XJSS using LSTM Net(1).

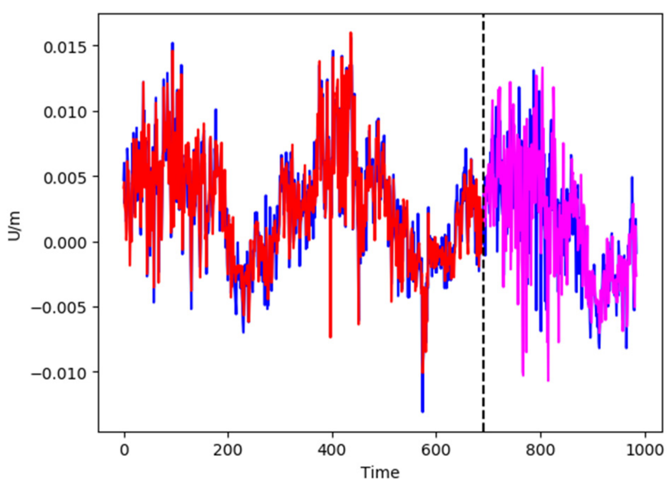
In the first experiment, a sliding window of 10-1 and 10 hidden cells of the single-layer LSTM were used as the network framework. The loss curve is shown in Figure 7a. We can observe that after 5000 training runs, the loss curve dropped to around 0.006. The training results and prediction results are shown in Figure 7b. The blue curve represents the original value, comprising 990 groups of data in total, the red curve denotes the training neural network, containing 696 groups of data, and the magenta curve shows the remaining 294 groups of data for prediction.

In the second experiment, a sliding window of 15-1 and 15 hidden cells of the single-layer LSTM were used as the network framework. The loss curve is shown in Figure 8a. We can observe that after 5000 training runs, the loss curve dropped to around 0.002. The training results and prediction results are shown in Figure 8b. The blue curve represents the original value, comprising a total of 985 groups of data, the red curve denotes the training neural network, containing 691 groups of data, and the magenta curve shows the remaining 294 groups of data for prediction.

In the third experiment, a sliding window of 20-1 and 20 hidden cells of the single-layer LSTM were used as the network framework. The loss curve is shown in Figure 9a. After 5000 training runs, the loss curve dropped to below 0.001. The training results and prediction results are shown in Figure 9b. The blue curve represents the original value, comprising a total of 980 groups of data, the red curve denotes the training neural network, containing 686 groups of data, and the magenta curve shows the remaining 294 groups of data for prediction.

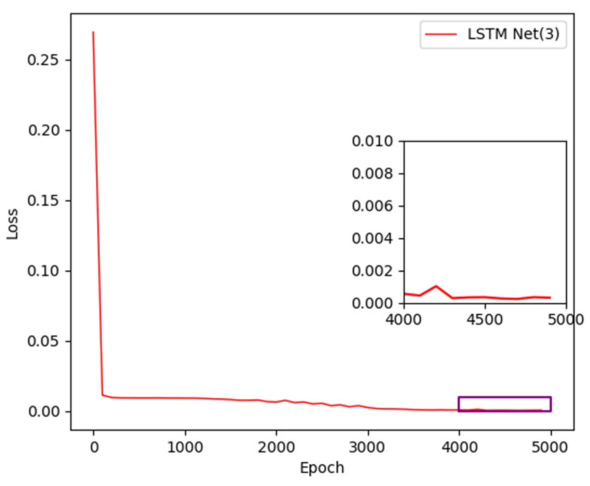


(**a**)

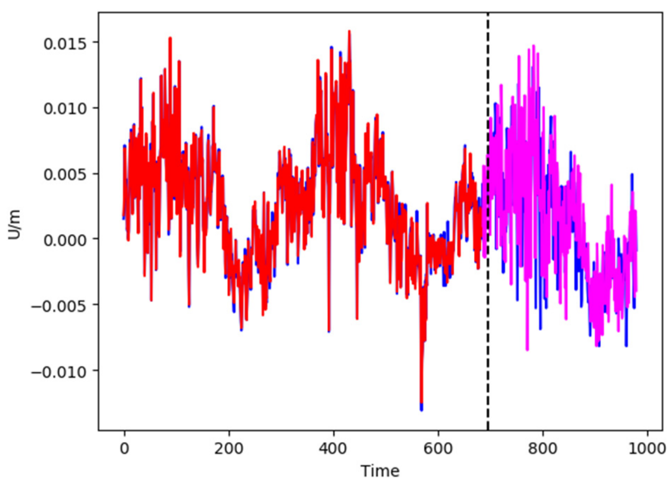


(**b**)

Figure 8. (**a**) Training loss curve; (**b**) data training and prediction of XJSS using LSTM Net(2).



(**a**)



(**b**)

Figure 9. (**a**) Training loss curve; (**b**) data training and prediction of XJSS using LSTM Net(3).

During the training of the neural network, the lower the loss value, the better. Although a smaller loss value indicates a stronger fitting performance of the neural network to the training data, it also results in less generalization ability. In practical application, the hyperparameters can be adjusted according to the data and the neural network structure; for example, a loss value of around 0.005 can maintain a good fit and generalization ability. In addition, the use of 5000 training runs in this experiment was also based on empirical values obtained after multiple training runs.

4.2. Experiment Summary

By weighted summation of the three groups of prediction results, 294 corresponding MSSW-LSTM prediction results were obtained, as shown in the Figure 10 below.

Figure 10 shows the comparison between the forecast results of MSSW-LSTM (blue) and the actual time series (red). It can be seen that MSSW-LSTM forecast results are consistent with the true values.

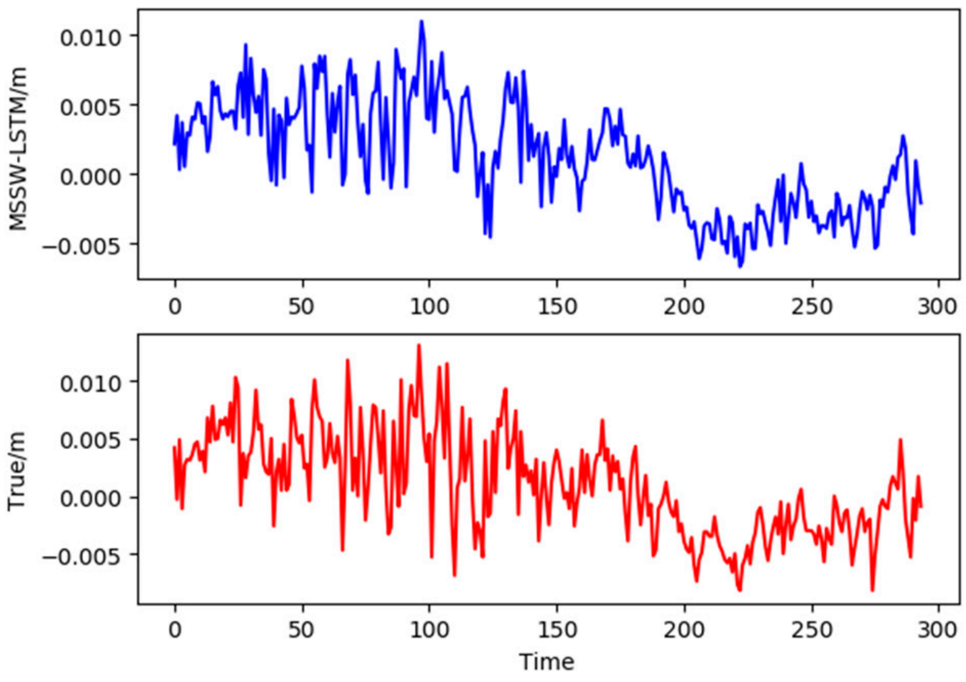


Figure 10. Comparison of MSSW-LSTM prediction and the true values.

To quantitatively evaluate the prediction accuracy of our proposed model, indexes (RMSE and MAE) were used to calculate the difference between the real and predicted values. In the three LSTM experiments, the network model was obtained by training nearly 700 datapoints, and the remaining 294 datapoints were predicted. The statistical results are shown in Table 4, among which the RMSE of LSTM(1), LSTM(2), and LSTM(3) are 3.2292, 4.1424, and 3.9810, respectively, and the MAE of these three experiments are 2.4252, 3.0239, and 3.0679.

Table 4. Performance comparison of traditional LSTM and proposed MSSW-LSTM.

	RMSE	MAE
LSTM(1)	3.2292	2.4253
LSTM(2)	4.1434	3.0239
LSTM(3)	3.9810	3.0679
MSSW-LSTM	3.1628	2.3864

We can observe that, although the network model may become more complex, the RMSE and MAE may not necessarily decrease. For example, although LSTM(2) is more complex than LSTM(1), its RMSE value is greater. It is obvious that both RMSE and MAE of MSSW-LSTM reach the best values. Specifically, RMSE is reduced by 2.1%, 23.7%, and 20.1% and MAE is decreased by 1.6%, 21.1%, and 22.2%, respectively.

In theory, through a single network, it is difficult to achieve optimal results even after multiple training runs. However, the MSSW-LSTM algorithm can combine the advantages of multiple networks and the data characteristics at different scales. The principle of this advantage is that the framework adopts the idea of measurement adjustment.

5. Discussion and Conclusions

In this study, a new forecasting framework, named MSSW-LSTM, comprising a multiscale sliding window (MSSW) and LSTM, was proposed for predicting GNSS time series. In the data preprocessing stage, the multiscale sliding window is used to form different training subsets, which can effectively extract the feature relationship under different scales, and facilitates mining the deep features of the data. The LSTM network can then effectively avoid the problem of gradient disappearance in the process of parameter solving. The MSSW-LSTM can use multiple LSTM networks to make simultaneous predictions, and obtains final results by weighting.

To verify the effectiveness of the MSSW-LSTM algorithm, 1000 daily solutions of the XJSS station in the Up component were selected for prediction experiments. The results of three groups of controlled experiments showed that the RMSE was reduced by 2.1%, 23.7%, and 20.1%, and MAE was decreased by 1.6%, 21.1%, and 22.2%, respectively. The experimental results showed that the proposed framework has a higher prediction accuracy and a smaller error.

It should be noted that the MSSW-LSTM method has significant flexibility. Researchers can easily construct appropriate subspace subsets formed by multiscale windows according to different data characteristics. In addition, LSTM networks may be the same or different, and may comprise single layer or multiple layers. This feature is beneficial to researchers for the selection of the most appropriate network model according to their own dataset characteristics, and for use of the advantages of the network model. MSSW-LSTM is a general framework for prediction that can be extended to other fields, such as traffic flow prediction, weather forecasting, and air quality forecasting.

Author Contributions: Conceptualization, J.W. and W.J.; methodology, J.W. and W.J.; software, J.W.; validation, J.W. and Z.L.; formal analysis, J.W. and Z.L.; investigation, J.W. and Y.L.; resources, J.W.; data curation, J.W.; writing—original draft preparation, J.W. and Z.L.; writing—review and editing, J.W., Z.L. and Y.L.; All authors have read and agreed to the published version of the manuscript.

Funding: This study is partially supported by the Natural Science Innovation Group Foundation of China (No. 41721003), and National key R&D Program of China (2018YFC1503601).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors gratefully acknowledge the China Earthquake Networks Center, National Earthquake Data Center (http://data.earthquake.cn, accessed on 16 August 2021) for providing GNSS data.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Geoffrey, L.; Blewitt, D. Effect of annual signals on geodetic velocity. *J. Geophys. Res. Solid Earth* **2002**, *107*, ETG 9-1–ETG 9-11.
2. Ohta, Y.; Kobayashi, T.; Tsushima, H.; Miura, S.; Hino, R.; Takasu, T.; Fujimoto, H.; Iinuma, T.; Tachibana, K.; Demachi, T.; et al. Quasi real-time fault model estimation for near-field tsunami forecasting based on RTK-GPS analysis: Application to the 2011 Tohoku-Oki earthquake (Mw 9.0). *J. Geophys. Res. Solid Earth* **2012**, *117*. [CrossRef]
3. Deng, L.; Jiang, W.; Li, Z.; Chen, H.; Wang, K.H.; Ma, Y.F. Assessment of second-and third-order ionospheric effects on regional networks: Case study in China with longer CMONOC GPS coordinate time series. *J. Geod.* **2017**, *91*, 207–227. [CrossRef]
4. Bevis, M.; Alsdorf, D.; Kendrick, E.; Fortes, L.P.; Forsberg, B.; Smalley, R.; Becker, J. Seasonal fluctuations in the mass of the Amazon River system and Earth's elastic response. *Geophys. Res. Lett.* **2005**, *32*. [CrossRef]
5. Montillet, J.P.; Williams, S.; Koulali, A.; McClusky, S.C. Estimation of offsets in GPS time-series and application to the detection of earthquake deformation in the far-field. *Geophys. J. Int.* **2015**, *200*, 1207–1221. [CrossRef]
6. Meng, X.; Roberts, G.W.; Dodson, A.H.; Cosser, E.; Barnes, J.; Rizos, C. Impact of GPS satellite and pseudolite geometry on structural deformation monitoring: Analytical and empirical studies. *J. Geod.* **2004**, *77*, 809–822. [CrossRef]
7. Yi, T.H.; Li, H.N.; Gu, M. Experimental assessment of high-rate GPS receivers for deformation monitoring of bridge. *Meas. J. Int. Meas. Confed.* **2013**, *46*, 420–432. [CrossRef]
8. Yu, J.; Meng, X.; Shao, X.; Yan, B.; Yang, L. Identification of dynamic displacements and modal frequencies of a medium-span suspension bridge using multimode GNSS processing. *Eng. Struct.* **2014**, *81*, 432–443. [CrossRef]
9. Xi, R.; Jiang, W.; Meng, X.; Zhou, X.; He, Q. Rapid initialization method in real-time deformation monitoring of bridges with triple-frequency BDS and GPS measurements. *Adv. Space Res.* **2018**, *62*, 976–989. [CrossRef]
10. Chen, Q.; Jiang, W.; Meng, X.; Jiang, P.; Wang, K.; Xie, Y.; Ye, J. Vertical deformation monitoring of the suspension bridge tower using GNSS: A case study of the forth road bridge in the UK. *Remote Sens.* **2018**, *10*, 364. [CrossRef]

11. Altamimi, Z.; Rebischung, P.; Métivier, L.; Collilieux, X. ITRF2014: A new release of the International Terrestrial Reference Frame modeling nonlinear station motions. *J. Geophys. Res. Solid Earth* **2016**, *121*, 6109–6131. [CrossRef]
12. Li, Z.; Chen, W.; van Dam, T.; Rebischung, P.; Altamimi, Z. Comparative analysis of different atmospheric surface pressure models and their impacts on daily ITRF2014 GNSS residual time series. *J. Geod.* **2020**, *94*, 1–20. [CrossRef]
13. Zhang, J.; Bock, Y.; Johnson, H.; Fang, P.; Williams, S.; Genrich, J.; Wdowinski, S.; Behr, J. Southern California permanent GPS geodetic array: Error analysis of daily position estimates and site velocities. *J. Geophys. Res. Solid Earth* **1997**, *102*, 18035–18055. [CrossRef]
14. He, X.; Montillet, J.P.; Fernandes, R.; Bos, M.; Yu, K.; Hua, X.; Jiang, W. Review of current GPS methodologies for producing accurate time series and their error sources. *J. Geodyn.* **2017**, *106*, 12–29. [CrossRef]
15. Klos, A.; Olivares, G.; Teferle, F.N.; Hunegnaw, A.; Bogusz, J. On the combined effect of periodic signals and colored noise on velocity uncertainties. *GPS Solut.* **2018**, *22*, 1–13. [CrossRef]
16. Hinton, G.E.; Salakhutdinov, R.R. Reducing the Dimensionality of Data with Neural Networks. *Science* **2006**, *313*, 504–507. [CrossRef]
17. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [CrossRef]
18. Elman, J.L. Finding structure in time. *Cogn. Sci.* **1990**, *14*, 179–211. [CrossRef]
19. Bengio, Y.; Simard, P.; Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw.* **1994**, *5*, 157–166. [CrossRef]
20. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
21. Freeman, B.S.; Taylor, G.; Gharabaghi, B.; Thé, J. Forecasting air quality time series using deep learning. *J. Air Waste Manag. Assoc.* **2018**, *68*, 866–886. [CrossRef]
22. Karevan, Z.; Suykens, J. Transductive LSTM for time-series prediction: An application to weather forecasting. *Neural Netw.* **2020**, *125*, 1–9. [CrossRef]
23. Tian, Y.; Pan, L. Predicting short-term traffic flow by long shortterm memory recurrent neural network. In Proceedings of the 2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity), Chengdu, China, 19–21 December 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 153–158.
24. Xing, Y.; Yue, J.; Chen, C.; Cong, K.L.; Zhu, S.L.; Bian, Y.K. Dynamic Displacement Forecasting of Dashuitian Landslide in China Using Variational Mode Decomposition and Stack Long Short-Term Memory Network. *Appl. Sci.* **2019**, *9*, 2951. [CrossRef]
25. Xing, Y.; Yue, J.; Chen, C. Interval Estimation of Landslide Displacement Prediction Based on Time Series Decomposition and Long Short-Term Memory Network. *IEEE Access* **2020**, *8*, 3187–3196. [CrossRef]
26. Xie, P.; Zhou, A.; Chai, B. The Application of Long Short-Term Memory(LSTM) Method on Displacement Prediction of Multifactor-Induced Landslides. *IEEE Access* **2019**, *7*, 54305–54311. [CrossRef]
27. Wang, J.; Nie, G.; Gao, S.; Wu, S.; Li, H.; Ren, X. Landslide Deformation Prediction Based on a GNSS Time Series Analysis and Recurrent Neural Network Model. *Remote Sens.* **2021**, *13*, 1055. [CrossRef]
28. Yang, B.; Yin, K.; Lacasse, S.; Liu, Z.Q. Time series analysis and long short-term memory neural network to predict landslide displacement. *Landslides* **2019**, *16*, 677–694. [CrossRef]
29. Tan, T.N.; Khenchaf, A.; Comblet, F.; Franck, P.; Champeyroux, J.M.; Reichert, O. Robust-Extended Kalman Filter and Long Short-Term Memory Combination to Enhance the Quality of Single Point Positioning. *Appl. Sci.* **2020**, *10*, 4335. [CrossRef]
30. Jiang, C.; Chen, S.; Chen, Y.; Zhang, B.; Feng, Z.; Zhou, H.; Bo, Y. A MEMS IMU De-Noising Method Using Long Short Term Memory Recurrent Neural Networks (LSTM-RNN). *Sensors* **2018**, *18*, 3470. [CrossRef]
31. Kim, H.U.; Bae, T.S. Deep Learning-Based GNSS Network-Based Real-Time Kinematic Improvement for Autonomous Ground Vehicle Navigation. *J. Sens.* **2019**, *2019*, 1–8. [CrossRef]
32. Tao, Y.; Liu, C.; Chen, T.; Zhao, X.W.; Liu, C.Y.; Hu, H.J.; Zhou, T.F.; Xin, H.Q. Real-Time Multipath Mitigation in Multi-GNSS Short Baseline Positioning via CNN-LSTM Method. *Math. Probl. Eng.* **2021**, *2021*, 1–12.
33. Hoang, M.T.; Yuen, B.; Dong, X.; Lu, T.; Westendorp, R.; Reddy, K. Recurrent Neural Networks for Accurate RSSI Indoor Localization. *IEEE Internet Things J.* **2019**, *6*, 10639–10651. [CrossRef]
34. Fang, W.; Jiang, J.; Lu, S.; Gong, Y.; Tao, Y.; Tang, Y.; Yang, P.; Luo, H.; Liu, J. A LSTM Algorithm Estimating Pseudo Measurements for Aiding INS during GNSS Signal Outages. *Remote Sens.* **2020**, *12*, 256. [CrossRef]
35. Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. In Proceedings of the IEEE Conference on Computer Vision & Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 6517–6525.
36. Ren, S.; He, K.; Girshick, R.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [CrossRef]
37. Plutowski, M.; Cottrell, G.; White, H. Experience with selecting exemplars from clean data. *Neural Netw. Off. J. Int. Neural Netw. Soc.* **1996**, *9*, 273–294. [CrossRef]