Short term prediction of wireless traffic based on tensor decomposition and recurrent neural network

Tao Deng¹ · Mengxuan Wan¹ · Kaiwen Shi¹ · Ling Zhu¹ · Xichen Wang¹ · Xuchu Jiang¹

Received: 29 April 2021 / Accepted: 9 August 2021

Published online: 20 August 2021

© The Author(s) 2021

Abstract

This paper proposes a wireless network traffic prediction model based on Bayesian Gaussian tensor decomposition and recurrent neural network with rectified linear unit (BGCP-RNN-ReLU model), which can effectively predict the changes in the upstream and downstream network traffic in a short period of time in the future. The research is divided into two parts: (i) The missing observations are imputed by an algorithm based on Bayesian Gaussian tensor decomposition. (ii) The recurrent neural network is used to forecast the true observations only rather than both true and estimated observations. The results show that, compared with other combined models of missing data imputation and neural networks, the BGCP-RNN-ReLU model proposed in this paper has the smallest prediction error for both the upstream and downstream traffic. The new model achieves better forecasting precision, and thus can help to regulate the load of communication station to reduce resource consumption.

Highlights

- The problem of forecasting wireless network traffic with missing values is divided in two stages to handle.
- A newly proposed method can more efficiently impute missing values in wireless network traffic data.
- Simple recurrent neural network obtains better prediction performance than other complex networks.

Keywords Wireless traffic · Time series prediction · Tensor decomposition · RNN

1 Introduction

At present, mobile communication technology has entered the 5G from the 4G. While the rapid development of the mobile Internet has brought many conveniences to people, it has also brought great traffic pressure to communication base stations. The usage of wireless network traffic in residential areas has obvious periodic changes, that is, the usage is higher in some periods and lower in some periods [1], which provides the possibility to predict its future changes. If the future changes in the up-down-traffic of the cell covered by the base station can be

accurately predicted, then a part of the carrier frequency of the base station can be turned off when the traffic load is at a low level, thereby greatly reducing energy and energy without affecting the normal use of users. The surrounding environment of wireless base stations is complex, and due to various natural or man-made factors such as severe weather, equipment aging, and normal maintenance, there may be missing traffic data for some periods of time. These missing values must be treated in some way because neither traditional time series models nor neural networks can deal with the time series with missing values directly. Therefore, how to accurately predict time series

✉ Xuchu Jiang, xuchujiang@zuel.edu.cn | ¹Zhongnan University of Economics and Law, Wuhan 430073, China.

with missing values is an important issue in the field of time series analysis.

For missing values in the time series, a straightforward way is to delete them directly, but this is likely to cause the loss of useful information. Che et al. [2] chose not to explicitly fill in the missing values, but to comprehensively use the original data and its missing pattern representation. But the usual method is to fill in the missing values before modeling and forecasting. Yi [3] discussed the current mainstream imputation methods for missing values in time series. Commonly applied methods are filling with historical mean, historical median or the most recent valid value, etc. The probability matrix decomposition in recommendation system can also be used to fill in missing values in time series. In recent years, researchers have proposed a variety of new imputation methods including local average of nearest neighbors [4] and vector autoregressive model [5]. Tensor factorization is a high-order generalization of matrix factorization. It has three main purposes: dimensionality reduction, implicit relationship mining and missing data filling. Commonly used tensor decomposition algorithms include Tucker decomposition, Candecomp/Parafac decomposition (CP decomposition), tensor singular value decomposition (t-SVD), etc. [6, 7]. Chen et al. [8] extended the Bayesian matrix decomposition model to higher-order tensors, and proposed Bayesian Gaussian CP decomposition (BGCP). Compared with traditional probability matrix decomposition, BGCP has greatly improved its performance.

According to the different scale ranges of the forecast time, network traffic forecast can be divided into short-term forecast and long-term forecast. Shafiq et al. [9] analyzed traditional short-term prediction methods, including Markov models, Poisson models, autoregressive (AR) models, autoregressive moving average (ARMA) models, Bayesian methods, support vector machines, etc. In recent years, with the rise of deep learning methods, a variety of neural networks represented by recurrent neural network (RNN) have been applied to network traffic prediction tasks, and the prediction effect is significantly better than that of standard statistical models such as autoregressive integrated moving average (ARIMA) [10]. M Li [11] compared the accuracy of stacking autoencoders and long short-term memory (LSTM) for up wireless network traffic prediction, and found that the LSTM has higher accuracy. Laisen Nie et al. [12] decomposed wireless network traffic into low-pass components and high-pass components, using deep belief networks to predict low-pass components describing long-term dependence, and using Gaussian model modeling for high-pass components that represent irregular fluctuations. Vinayakumar R et al. [13] compared the performance of LSTM with other

RNNs and found that although LSTM performed best, the gap with other RNNs was not obvious.

Real network traffic data is likely to contain missing values, and current research using neural networks to predict network traffic usually uses data without missing values, which is somewhat different from the actual situation. Combining tensor decomposition with neural networks to process time series forecasting tasks is one of the research hotspots in the field of deep learning, but most studies use tensor decomposition to obtain low-dimensional data representations, thereby reducing the number of parameters while maintaining model performance, Improve the convergence speed of the network [14–16]. There are relatively few researches on filling in missing values in time series through tensor decomposition, and then using neural network prediction.

This paper proposes a prediction model BGCP-RNN for time series with missing values based on BGCP and RNNs. Compared with previous papers which also use tensor decomposition and neural networks to predict network traffic, the biggest difference of this paper is that a new tensor decomposition approach is applied to missing value processing, which effectively solves the problem that the general neural network cannot directly process the data with missing values. First, the original data with missing values is expressed as a tensor containing dependencies or correlations of the original dataset. In this paper, the BGCP method is used to estimate a full dataset from the original dataset with missing values. The BGCP method proposed by Chen X et al. [8] uses CP decomposition to decompose the tensor into the sum of multiple single-rank tensors, and uses Bayesian inference for parameter estimation, sampling and updating the samples, which can effectively fill the missing values in the raw data. Then, the standard RNN is used to train and predict on the data that has filled in the missing values. In experiments based on real wireless network traffic data, compared with other time series prediction models that deal with missing values, BGCP-RNN has a smaller number of parameters, and achieves the best results on both the up-down-traffic short-term prediction tasks, which means our model is significantly more accurate and faster than others.

The paper is organized as follows. Data sources and data preprocessing are introduced in Sect. 2. Section 3 briefly demonstrates the basic principle of tensor decomposition and the flow of BGCP algorithm. Section 4 describes how recurrent neural networks can predict time series. Our proposed BGCP-RNN model and how to apply it are illustrated in Sect. 5. Empirical analysis in Sect. 6 compares the prediction ability of BGCP-RNN with other explicit imputation models and GRU-D. In Sect. 7, a conclusion from the current research is presented.

2 The wireless traffic dataset

2.1 Data sources

The data of this paper comes from the 2020 MathorCup university mathematical modeling challenge big data competition. Because the original data set is large, this paper randomly extracts part of the cell data to form a sampling data set. The data set consists of 1200 h of hourly up-down-traffic of 20 cells from 0:00 on March 1, 2018 to 23:00 on April 19, 2018. The missing value accounts for about 7.49% of the total data. An example of the dataset is shown in Table 1. The up-traffic volume and down-traffic volume represent the up-traffic volume and down-traffic volume of a cell in a specific hour, both in Gigabyte (GB).

Figure 1 shows the up-down-traffic changes of the 16th cell in the dataset in the first five days. The usage of up-traffic is obviously smaller than that of down-traffic. Although the upstream and downstream traffic are different in size, they have similar change rules: the upstream and downstream traffic are at the lowest point from 24:00 to 6:00, and the traffic usage gradually rises from 6:00, there is a small trough from 12:00 to 18:00, and reaches the peak from 18:00 to 24:00. This is consistent with the work and living habits of the residents. From 24:00 to 6:00, most people have fallen asleep, so the use of traffic is the least. After 6:00, people get up to start the day's activities, and the use of traffic begins to increase. In the afternoon, some people will take a nap, when traffic usage will decline. After 6 p.m., it is basically time for residents to rest and entertainment, so the traffic usage reaches the peak at this time.

The upper (lower) line traffic data missing rate of an hour is defined as the proportion of the number of cells missing data in the hour to all cells (20). Calculate the data missing rate of up-down-traffic for each hour, as shown in Fig. 2. The loss of up-traffic data is almost the same as that of down-traffic data. The data of up-down-traffic from 11:00 to 17:00 on March 27 and the whole day on April

Fig. 1 Time variation of upstream and downstream traffic

14 are completely missing. The missing rate of other periods is basically below 0.1, and there is almost no missing before March 22.

Fig. 2 Time variation of upstream and downstream traffic loss rate

Table 1 Partial data of cell up-down-traffic

Cell number	Date	Time	Up-traffic	Down-traffic
1	2018-3-1	0:00:00	0.324709606	3.560743589
1	2018-3-1	1:00:00	0.070473995	1.154026413
1	2018-3-1	2:00:00	0.065063486	0.571824083
1	2018-3-1	3:00:00	0.032563429	0.435221386
2	2018-3-1	0:00:00	0.072757406	0.830949554
2	2018-3-1	1:00:00	0.018987942	0.287364187
20	2018-4-19	22:00:00	0.00360497	0.02369873
20	2018-4-19	23:00:00	0.00363834	0.06756706

2.2 Data preprocessing

The RNN generally uses gradient descent method to update parameters. In order to avoid gradient explosion in gradient calculation, it is necessary to normalize the original data. One of the most common normalization methods is min–max normalization, as shown in Eq. (1).

$$\mathbf{x}^* = \frac{\mathbf{x} - \mathbf{x}_{\min}}{\mathbf{x}_{\max} - \mathbf{x}_{\min}} \quad (1)$$

In Eq. (1), $\mathbf{x}$ represents the original vector or matrix, $\mathbf{x}_{\max}$ and $\mathbf{x}_{\min}$ represent the maximum and minimum values in the original data respectively, and $\mathbf{x}^*$ is the normalized vector or matrix. The min max normalization of the upstream and downstream traffic is performed respectively, and the original data is linearly mapped to the range of [0, 1]. Taking the time step of 48 h and predicting the flow of the next hour, a total of 1152 samples were obtained. Training set, verification set and test set are divided according to the proportion of 6:2:2.

3 Tensor decomposition for imputation

3.1 CP decomposition

CP decomposition is to decompose a high-dimensional tensor into a single-rank tensor to obtain the sum. Let $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times \dots \times n_d}$ denote the d -order tensor, $i = (i_1, i_2, \dots, i_d)$ denote the index of the tensor input value, and x_i the observation value, then the CP decomposition can be expressed as Eq. (2):

$$\hat{\mathbf{X}} = \sum_j^r \mathbf{u}_j^{(1)} \circ \mathbf{u}_j^{(2)} \circ \dots \circ \mathbf{u}_j^{(d)} \quad (2)$$

In Eq. (2), $\mathbf{u}_j^{(k)} \in \mathbb{R}^{n_k}$ is the column vector of the j -th single-rank tensor before the outer product operation on the k -th dimension, r is the total number of single-rank tensors, and $\circ$ is the outer product operator of the vector. In a special case, the principle of CP decomposition when the tensor order is 3 is shown in Fig. 3.

3.2 Bayesian Gaussian CP decomposition

Figure 4 is the schematic diagram characterizing the overall data generation process of the BGCP method. The detailed generation process is described below.

Given that the d -order tensor $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times \dots \times n_d}$ contains missing values, it is assumed that each observation value obeys the distribution shown in Eq. (3):

Fig. 3 Schematic diagram of CP decomposition of the third-order tensor

$$x_i \sim N\left(\sum_{j=1}^r u_{i,j}^{(1)} u_{i,j}^{(2)} \dots u_{i,j}^{(d)}, \tau^{-1}\right), \forall i = i_1, i_2, \dots, i_d \quad (3)$$

x_i in Eq. (3) obeys a Gaussian distribution. τ represents accuracy, and the obeyed gamma distribution is shown in Eq. (5). Gaussian distribution is the most common statistical distribution, and this assumption is also convenient for the subsequent calculation process such as Gibbs sampling. Although the subsequent modeling is based on Gaussian assumption, the experiments in Sect. 6.2 show that this assumption can be extended to other continuous distributions. Under the assumption that Eq. (3) obeys Gaussian distribution, the precision parameter τ is used to capture the noise level in the data, which cannot be captured by the inverse of all observation variances. Setting τ as a flexible Gamma conjugate prior distribution can improve the robustness of the model [8]. Combined with Bayesian probability theory, assuming that the variable x obeys $P(X|\theta)$, its observation sample is $X = \{x_1, x_2, \dots, x_m\}$, and the parameter θ obeys the prior distribution $\Pi(\theta)$, then the posterior distribution is Eq. (4).

Fig. 4 Schematic diagram of Bayesian model data generation

$$P(\theta|X) = \frac{\Pi(\theta)P(X|\theta)}{P(X)} \quad (4)$$

If posterior distribution $P(\theta|X)$ and prior distribution $\Pi(\theta)$ belong to the same distribution family, then prior distribution $\Pi(\theta)$ is called conjugate distribution of likelihood distribution $P(X|\theta)$. The conjugate prior distribution of Gaussian likelihood distribution whose mean value is known is gamma distribution. Therefore, when the likelihood distribution x_i is assumed to be Gaussian distribution, the mean value is calculated by using the elements in the corresponding factor matrix in Eq. (3), and τ still obeys Gamma distribution after continuous iteration and updating of Gibbs sampling.

$$\tau \sim \text{Gamma}(a_0, b_0) \quad (5)$$

In Eq. (5), a_0 is the shape parameter, and b_0 is the velocity parameter.

Based on the Gaussian hypothesis, in the model of tensor elements formed, the parameters and hyperparameters obey the distributions shown in Eq. (6) and Eq. (7), respectively:

$$\mathbf{u}_{i_k}^{(k)} \sim N(\boldsymbol{\mu}_{\mathbf{u}^{(k)}}, \boldsymbol{\Lambda}_{\mathbf{u}^{(k)}}^{-1}), \forall i_k, \quad (6)$$

$$\boldsymbol{\mu}_{\mathbf{u}^{(k)}} \sim N(\boldsymbol{\mu}_0, (\beta_0 \boldsymbol{\Lambda}_{\mathbf{u}^{(k)}})^{-1}), \boldsymbol{\Lambda}_{\mathbf{u}^{(k)}} \sim W(\mathbf{W}_0, \nu_0) \quad (7)$$

In Eq. (7), $W(\cdot)$ is the Wishart distribution, and $\boldsymbol{\mu}_0, \beta_0, \mathbf{W}_0, \nu_0$ are hyperparameters. In Eq. (6), $\mathbf{u}_{i_k}^{(k)} \in \mathbb{R}^r$ represents the vector of the i_k row in the factor matrix of the k -th dimension, assuming that it obeys the multivariate normal distribution. $\boldsymbol{\mu}_{\mathbf{u}^{(k)}}$ represents its mean vector, and $\boldsymbol{\Lambda}_{\mathbf{u}^{(k)}}^{-1}$ represents its covariance matrix. In particular, if a positive definite matrix $\boldsymbol{\Lambda}$ obeys Wishart distribution, then the inverse matrix $\boldsymbol{\Lambda}^{-1}$ of matrix $\boldsymbol{\Lambda}$ obeys inverse Wishart distribution. Inverse Wishart distribution is often used as conjugate prior distribution of covariance matrix of multivariate normal distribution in Bayesian probability theory. This setting can enhance the robustness of the

update all variables in each iteration, and each variable is sampled from its hypothetical distribution (the hypothetical distribution has been given in the previous section), and the condition is the current value of all other variables. In this paper, the parameter $\mathbf{W}_0^{(k)}, \nu_0^{(k)}, \boldsymbol{\mu}_0^{(k)}, \boldsymbol{\Lambda}_0^{(k)}$ are updated by Gibbs sampling. The process is shown in Eq. (8)–(11). In the case of three-dimensional tensor, k is 1, 2 and 3 in turn.

$$\hat{\mathbf{W}}_0^{(k)} = (n_k \mathbf{S}^{(k)} + \frac{n_k \beta_0}{n_k + \beta_0} (\bar{\mathbf{u}}^{(k)} - \boldsymbol{\mu}_0)(\bar{\mathbf{u}}^{(k)} - \boldsymbol{\mu}_0)^T + \mathbf{W}_0^{-1})^{-1} \quad (8)$$

$$\hat{\nu}_0^{(k)} = n_k + \nu_0, \quad (9)$$

$$\hat{\boldsymbol{\mu}}^{(k)} = \frac{1}{n_k + \beta_0} n_k (\mathbf{u}^{(k)} + \beta_0 \boldsymbol{\mu}_0), \quad (10)$$

$$\hat{\boldsymbol{\Lambda}}^{(k)} = (n_k + \beta_0) \boldsymbol{\Lambda}^{(1)} \quad (11)$$

In Eq. (8), $\mathbf{S}^{(k)}$ and $\bar{\mathbf{u}}^{(k)}$ are defined as $\mathbf{S}^{(k)} = \frac{1}{n_k} \sum_{i_k=1}^{n_k} (\mathbf{u}_{i_k}^{(k)} - \bar{\mathbf{u}}^{(k)})(\mathbf{u}_{i_k}^{(k)} - \bar{\mathbf{u}}^{(k)})^T$ and $\bar{\mathbf{u}}^{(k)} = \sum_{i_k=1}^{n_k} \mathbf{u}_{i_k}^{(k)}$.

After updating each parameter, $\mathbf{u}_{i_k}^{(k)}$ is sampled according to the distribution of Eq. (6). The factor matrix $\mathbf{U}^{(k)}$ is denoted as $\mathbf{U}^{(k)} = (\mathbf{u}_1^{(k)}, \mathbf{u}_2^{(k)}, \dots, \mathbf{u}_{n_k}^{(k)})$, which is composed of all $\mathbf{u}_{i_k}^{(k)} \in \mathbb{R}^r (1 \leq i_k \leq n_k)$ in the k -th dimension. After the recombination of all factor matrices $\mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \dots, \mathbf{U}^{(d)}$ is completed, $\hat{\mathbf{X}}$ is calculated from Eq. (2) to obtain the estimation result. In fact, this operation corresponds to the Einstein summation convention in linear algebra.

For the need of sampling iteration, the accuracy τ_ϵ is also updated, and its likelihood estimation is obtained from all observations.

$$L(\mathbf{X}|\mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \mathbf{U}^{(3)}, \tau_\epsilon) \propto \prod_{i \in \Omega} (\tau_\epsilon)^{1/2} \exp\{-\frac{\tau_\epsilon}{2} (x_i - \hat{x}_i)^2\} \quad (12)$$

By combining the likelihood probability with the conjugate prior probability in Eq. (5), the posterior probability with $\hat{a}_0, \hat{b}_0$ as the parameter still obeys gamma distribution,

$$p(\tau_\epsilon|\mathbf{X}, \mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \mathbf{U}^{(3)}, a_0, b_0) \propto L(\mathbf{X}|\mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \mathbf{U}^{(3)}, \tau_\epsilon) \times \text{Gamma}(\tau_\epsilon|a_0, b_0) \propto \text{Gamma}(\tau_\epsilon|\hat{a}_0, \hat{b}_0) \quad (13)$$

model and accelerate the convergence when using the sampling algorithm [8].

After initializing the hyperparameters in the model, the Gibbs sampling method is used to iteratively update all the parameters and the factor matrix $\mathbf{U}^{(k)}$. The basic idea of the Gibbs sampling method is to sequentially

where $\hat{a}_0 = \frac{1}{2} \sum_{i \in \Omega} b_i + a_0, \hat{b}_0 = \frac{1}{2} \sum_{i \in \Omega} (x_i - \hat{x}_i)^2 + b_0$. After multiple sampling, the value of the parameter will converge and tend to be stable with the characteristics of the sampling and the original data set. A suitable threshold is set here. After iterating from the initial value to the threshold number, the small-scale sampling number is recycled,

Table 2 Pseudo code of BGCP algorithm

BGCP: Bayesian Gaussian tensor decomposition	
Input: tensor $\mathbf{X} \in \square^{n_1 \times n_2 \times \dots \times n_d}$, Factor matrix $\mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \dots, \mathbf{U}^{(d)}$	
Initialization: $\boldsymbol{\mu}_0 = \mathbf{0}$, $\beta_0 = 1$, $a_0 = 1$, $b_0 = 1$, $v_0 = r$, $\mathbf{W}_0 = \mathbf{I}$ (r order unit matrix), factor matrix $\mathbf{U}^{(k)}$ ($\forall k = 1, 2, \dots, d$) with random values	
Repeat	
For $k = 1, 2, \dots, d$:	
Calculate $\widehat{\mathbf{W}}_0^{(k)}$ and $\widehat{v}_0^{(k)}$	
Sample $\boldsymbol{\Lambda}^{(k)}$, $\boldsymbol{\Lambda}^{(k)} \sim \text{Wishart}(\widehat{\mathbf{W}}_0^{(k)}, \widehat{v}_0^{(k)})$	
Calculate $\widehat{\boldsymbol{\mu}}_0^{(k)}$ and $\widehat{\boldsymbol{\Lambda}}_0^{(k)}$	
Sample $\boldsymbol{\mu}^{(k)}$, $\boldsymbol{\mu}^{(k)} \sim N(\widehat{\boldsymbol{\mu}}_0^{(k)}, (\widehat{\boldsymbol{\Lambda}}_0^{(k)})^{-1})$	
For $i_k = 1, 2, 3 \dots n_k$:	
Calculate $\widehat{\boldsymbol{\Lambda}}_{ik}^{(k)}$ and $\widehat{\boldsymbol{\mu}}_{ik}^{(k)}$	
Sample $\boldsymbol{\mu}_{ik}^{(k)}$, $\boldsymbol{\mu}_{ik}^{(k)} \sim N(\widehat{\boldsymbol{\mu}}_{ik}^{(k)}, (\widehat{\boldsymbol{\Lambda}}_{ik}^{(k)})^{-1})$	
Update $\mathbf{U}_{ik}^{(k)}$	
End	
End	
Calculate $\widehat{a}_0, \widehat{b}_0$	
Sample $\tau_\varepsilon \sim \text{Gamma}(\widehat{a}_0, \widehat{b}_0)$	
Until Maximum sampling times	
Calculate the $\widehat{\mathbf{X}}$	
Output: $\widehat{\mathbf{X}}$	

and the average value of $\widehat{\mathbf{X}}$ is calculated within this range as the final output result.

The algorithm flow of BGCP is shown in Table 2.

4 Recurrent neural network for prediction

Compared with the traditional fully connected neural network, RNNs can better mine the timing information and semantic information in the data. Using this ability of RNN, deep learning models have made breakthroughs in the fields of speech recognition, language models, machine translation, and timing analysis.

Assuming that $\mathbf{X}_t \in \mathbb{R}^{n \times d}$ is the small batch input at time step t in the sequence, $\mathbf{H}_t \in \mathbb{R}^{n \times h}$ is the hidden variable at this time step, and $\mathbf{O}_t \in \mathbb{R}^{n \times q}$ is the output at time t , the update rule of the standard RNN can be expressed as Eqs. (14) and (15).

$$\mathbf{H}_t = \phi(\mathbf{X}_t \mathbf{W}_{xh} + \mathbf{H}_{t-1} \mathbf{W}_{hh} + \mathbf{b}_h), \quad (14)$$

$$\mathbf{O}_t = \mathbf{H}_t \mathbf{W}_{hq} + \mathbf{b}_q \quad (15)$$

In Eq. (14), ϕ is the activation function, $\mathbf{W}_{xh} \in \mathbb{R}^{d \times h}$, $\mathbf{W}_{hh} \in \mathbb{R}^{h \times h}$ are the hidden layer weights, and $\mathbf{b}_h$ is the hidden layer deviation. In Eq. (14), $\mathbf{W}_{hq} \in \mathbb{R}^{h \times q}$ is

the weight of the output layer, and $\mathbf{b}_q \in \mathbb{R}^{1 \times q}$ is the deviation of the output layer.

Many variants of RNN have appeared afterwards, such as LSTM, Gated Recurrent Unit (GRU), etc., but their basic ideas are the same as standard RNNs, namely The definition of the hidden state of the current time step relates to the hidden state of the previous time step.

The activation function of RNN has an important influence on the calculation of hidden state. Common activation functions include Sigmoid function, Tanh function, ReLU function and so on. After the complete sample is obtained by filling in the missing values, this paper chooses the RNN with the activation function of ReLU as the prediction model. The ReLU activation function realizes the nonlinear transformation in a concise form, and at the same time makes the output of some neurons become 0, resulting in the sparsity of the network, and reducing the interdependence of parameters, and alleviating the occurrence of over-fitting problems. The combination of ReLU and standard RNN greatly reduces the amount of calculation. Compared with complex models such as LSTM and GRU, RNN with ReLU activation function obtains faster training and prediction speed on cell wireless network traffic data, and the experimental results show the prediction accuracy Also better.

Fig. 5 Application process of BGCP-RNN method

5 The proposed BGCP-RNN model

The process of BGCP-RNN prediction method proposed in this paper is shown in Fig. 5. Compared with two-dimensional tensor and four-dimensional tensor, the result of tensor decomposition is more stable when three-dimensional tensor is used to represent the input data [13]. Therefore, in the data conversion stage, this paper transforms the data set into three-dimensional tensor form. The first dimension represents the time in hours, the second dimension represents different residential areas, and the third dimension represents the up or down-traffic of a specific area. In the data division stage, the data set is divided into training set, verification set and test set in chronological order.

In the missing value filling stage, BGCP method is used to fill the missing value. BGCP method needs to specify the number of single rank tensors in advance, so a variety of different values are pre-tested on the training set and the best value that makes the error smallest is used to fill missing values. One of the key requirements of time series prediction task is that the prediction at time t cannot use any information from after time t . Since the training data does not involve this problem, when filling the missing values in the training set, this study only does one tensor decomposition for the data in all known periods. In the process of prediction, in order to avoid using the future information, this study chooses to decompose the tensor sample by sample. After the tensor decomposition, the values of parameters are updated by Gibbs sampling iteratively, and the factor matrix of the prediction tensor is reconstructed. Finally, the estimated tensor of the original data is obtained according to Einstein summation convention. The estimation tensor

includes the estimation of the observed value and the estimation of the missing value, and only the estimated value of the missing part is filled into the data set.

The RNN-ReLU neural network is trained, verified and tested using the training set, verification set and test set that have been filled in, and the error of the RNN-ReLU neural network on the test set is finally output.

6 Experimental analysis

6.1 Performance index

One of the commonly used predictive model performance evaluation indicators is the mean squared error (MSE). If there are no missing values in the data set, y_t^d represents the true upper (lower) traffic of the d -th cell at time t , and $\hat{y}_t^d$ represents the neural network prediction value of the d -th cell at time t , then the definition of MSE is shown in Eq. (16):

$$MSE = \frac{\sum_{t=1}^T \sum_{d=1}^D (y_t^d - \hat{y}_t^d)^2}{T \times D} \quad (16)$$

The smaller the mean square error, it means that the prediction result of the neural network is closer to the real observation. However, for time series with missing values, it is not reasonable to use MSE to evaluate the model after filling, because the filling value is only an estimate of the missing value. The closer the prediction result is to the filling value, the actual prediction effect may not be better. Therefore, a more reasonable evaluation method is to only calculate the mean squared error

Fig. 6 An example comparing MSE and MSOE

of the observed data and its predicted value, which is defined as the mean squared observed error (MSOE), as shown in Eqs. (17) and (18):

$$m_t^d = \begin{cases} 1, & \text{If } y_t^d \text{ is an observed value} \\ 0, & \text{If } y_t^d \text{ is a missing value} \end{cases} \quad (17)$$

$$MSOE = \frac{\sum_{t=1}^T \sum_{d=1}^D (y_t^d - \hat{y}_t^d)^2 m_t^d}{\sum_{t=1}^T \sum_{d=1}^D m_t^d} \quad (18)$$

MSOE measures how close the predicted value is to the observed data. A simple example is shown in Fig. 6. y represents the down-traffic records of 4 cells in a certain hour, NA represents the missing value, $\tilde{y}$ represents the data after filling in the missing value, and $\hat{y}_1$ and $\hat{y}_2$ represent the prediction results of neural network 1 and neural network 2, respectively. It is easy to see that if the missing values are not considered, the prediction result of network 1 is better than that of network 2. However, the predicted result of network 1 and the MSE of $\tilde{y}$ are 0.055, and the predicted result of network 2 and the MSE of $\tilde{y}$ are only 0.045, which is obviously inconsistent with reality. The MSOE of network 1 is 0.02, and the MSOE of network 2 is about 0.057. The MSOE proposed in this paper can more accurately measure the prediction accuracy of the model for time series with missing values.

Let $a = \sum_{t=1}^T \sum_{d=1}^D m_t^d$, $b = \sum_{t=1}^T \sum_{d=1}^D (y_t^d - \hat{y}_t^d)^2 m_t^d$. $\forall t, d, m_t^d = 0$ may appear in the process of calculating MSOE, at this time the denominator term of MSOE. In order to avoid this situation from causing program operation errors, you can add a very small number λ ($\lambda > 0$) in front of the denominator term of MSOE to make the fraction meaningful. The adjusted MSOE is shown in Eq. (19):

$$MSOE = \frac{b}{\lambda + a} = \frac{\sum_{t=1}^T \sum_{d=1}^D (y_t^d - \hat{y}_t^d)^2 m_t^d}{\lambda + \sum_{t=1}^T \sum_{d=1}^D m_t^d} \quad (19)$$

In Eq. (19), if $\forall t, d, m_t^d = 0$, it can be known that $a = 0$ and $b = 0$, and the adjusted MSOE value is $\frac{b}{\lambda + a} = 0$. If there is $a = \sum_{t=1}^T \sum_{d=1}^D m_t^d \geq 1$, that is, there is at least one observed value, it is easy to calculate the relative error of MSOE $\left| \frac{b}{\lambda + a} - \frac{b}{a} \right| \div \frac{b}{a} = \left| \frac{\lambda}{\lambda + a} \right| = \frac{\lambda}{\lambda + a}$ before and after adding λ . Setting the relative error limits of $\varepsilon_r = 0.01\%$, then λ must satisfy $\frac{\lambda}{\lambda + a} \leq \varepsilon_r \Rightarrow \lambda \leq \frac{\varepsilon_r}{1 - \varepsilon_r} a$. And when $\lambda \leq \varepsilon_r$, $\lambda \leq \varepsilon_r \leq \frac{\varepsilon_r}{1 - \varepsilon_r} \leq \frac{\varepsilon_r}{1 - \varepsilon_r} a$ ($a \geq 1$), that is, as long as $\lambda \leq 1 \times 10^{-4}$, the relative error of MSOE before and after adjustment cannot exceed 0.01%.

6.2 Relaxation of Gauss assumption

The construction of the BGCP model assumes that the observations obey the Gaussian distribution. Although the Gaussian distribution is the most common statistical distribution, in some cases, some random variables may not obey the Gaussian distribution. In this study, three groups of 10,000 data that obey t distribution, F distribution, and uniform distribution were generated by simulation, and set a data missing rate of 30%. The BGCP method with a rank of 40 was used to fill in the missing values, and the mean square error between true values and filled values is shown in Table 3. Table 3 shows that the filling results of the data that obey the t distribution or the uniform distribution are basically consistent with the Gaussian distribution, and the difference between the MSE of the F distribution and the MSE of the Gaussian distribution is only about 18.9% of the MSE of the Gaussian distribution. This experiment shows that the Gaussian distribution assumption of the BGCP method can be relaxed to other continuous distributions without making the BGCP method inoperable or causing drastic performance changes.

6.3 Comparison of tensor decomposition and wavelet decomposition

Wavelet decomposition can also fill in missing values in time series [17–19]. In this study, 3300 uplink traffic data with no missing value were selected as the real data in the cell No. 3–8 within 50 days. The missing rate is set to be 10, 30, 50 and 70%, which means that 10, 30, 50 and 70% of the data are randomly selected as the data to be filled in the input. BGCP and wavelet decomposition are used to output the

Table 3 Filling results of data with different distributions

Distribution	Gaussian normal	t distribution	F distribution	Uniform distribution
MSE	0.0640	0.0663	0.0519	0.0667

Table 4 MSE of wavelet decomposition and BGCP decomposition

Loss rate (%)	Wavelet decomposition		BGCP
	Wavelets bases	MSE(10e-4)	
10	db4	13.1492	2.8308
30	db3	35.9817	9.6421
50	db3	56.5837	16.4336
70	db2	84.5508	31.2454

filling results respectively, and the mean square error is calculated between real data and filled data.

For the data with different missing rate, different wavelet bases are used to decompose the data in three layers. At the same time, BGCP method with rank 40 was selected for tensor decomposition. The mean square error under the two methods is shown in Table 4. Figure 7 shows the results of filling the 24 h data in the cell number 3 when the missing rate is set to 30%. It can be seen that BGCP method can fill the missing value more accurately.

Table 4 and Fig. 7 show that the BGCP method is better than the general wavelet decomposition under the four loss rate settings. The third-order tensor is used to represent the network traffic data of different time, different location and different types (uplink or downlink). In this data structure, the dependence or correlation between space and up-down-traffic can be introduced into the process of model parameter estimation and missing value filling. However, wavelet decomposition only considers single location and single type of network traffic data, which may be one of the important reasons why BGCP method is superior to wavelet decomposition. Furthermore, the performance of wavelet decomposition and tensor decomposition in filling non-random missing data (that is, all data in a certain

Fig. 8 Filling results of wavelet decomposition and BGCP decomposition on simulated dataset

period of time is missing) is compared by using simulated data. The flow data of 120 h in 5 days is generated by $y = 0.5\sin(\frac{\pi}{12}t) + 0.3\cos(\frac{\pi}{12}t) + 1 + \epsilon$, where t is the time and y is the flow size, ϵ obeys a Gaussian distribution with a mean value of 0 and a variance of 0.1. Set the data from the 20th hour to the 30th hour as the missing value (represented by 0). The filling results of wavelet decomposition and tensor decomposition are shown in Fig. 8.

Figure 8 shows that from the 20th hour to the 30th hour, wavelet decomposition cannot estimate the missing values, and the filling result of tensor decomposition is more consistent with the real change trend. The BGCP method based on Bayesian theory can determine the parameters from the data of other time periods, and then infer the changes of network traffic in non-random missing periods.

Fig. 7 Filling results of wavelet decomposition and BGCP decomposition on real dataset

The imputation closer to reality can better reflect the change trend of time series, so as to improve the accuracy of the prediction stage. Therefore, this study only chooses the combination of tensor decomposition method and recurrent neural network.

6.4 Experimental setup

6.4.1 Determination of the number of tensors

From Eq. (14), tensor can be divided into the sum of multiple single-rank tensors. When the number of single-rank tensors changes, that is, when r changes, the effect of missing value filling is different. The effect of missing value filling can be determined by the estimated value of the observed value obtained by tensor decomposition and the MSOE value of the observed data, as well as the predicted value of the neural network after filling and the true value of the MSOE. The images of these two indicators changing with different values of r are shown in Fig. 9.

The orange dotted line represents the MSOE between the padded data and the original data obtained by taking different values of BGCP, and the blue solid line represents the MSOE between the predicted value and the verified value of the RNN-ReLU neural network. RNN-ReLU respectively predicts upstream traffic and downstream traffic to produce two MSOE values. This paper takes the average of these two MSOEs to measure different effects on the overall prediction effect of the neural network. It can be seen from Fig. 9 that the MSOE declines faster at $r < 40$, and the MSOE declines slowly at $r > 40$. The point corresponding to $r = 40$ is the inflection point of the image. Taking $r = 40$

can achieve a stable MSOE value with a relatively small amount of calculation.

6.4.2 Comparison model

This paper selects a variety of missing value filling methods and a combination model of LSTM, GRU, RNN-tanh and other neural networks as the comparison of BGCP-RNN. Neural networks such as LSTM and GRU have been introduced in Sect. 2. The following briefly introduces other missing value filling methods.

Mean: The historical mean value is filled. If there is a missing value at a certain time t , the average value of all the observed data before time t is used to fill in the missing value.

Forward: Recently observed and filled. If there is a missing value at a certain time t , the observed data before and closest to time t is used as the missing value.

BTMF: Bayesian time factorization model. Compared with the traditional Bayesian matrix factorization or tensor factorization algorithm, this model adds low-rank matrix (tensor) factorization, and uses vector autoregressive process to model the time factor matrix. The basic assumption of the model is: the time series is highly correlated with the common potential factors.

BPMF: Bayesian probability matrix factorization model. This method automatically controls the model expression space by integrating all model parameters and hyperparameters, and uses the Markov chain Monte Carlo method for approximate inference.

TF-ALS: A tensor decomposition technique based on the Alternating Least Square (ALS) method. The

Fig. 9 Relationship between the number of single rank tensors R and mean square error MSOE

Table 5 Main hyperparameter settings

Main hyperparameter	Set value
Batch size	16
Number of hidden layer units	100
Hidden layers	1
Learning rate	5e-2
Training period	5
Optimizer	Adam
Loss function	MSE

factor matrix can be iteratively updated by using the ALS method.

6.4.3 Neural network settings

The setting of hyperparameters of neural network has a great influence on the final performance of the model. In order to ensure the fairness of the experiment, a unified hyperparameter setting is used, as shown in Table 5:

6.5 Results and analysis

Pytorch (a deep learning library) is used to calculate the results of all models on the test set, as shown in Table 6. Figure 10 is a visual display of the results in Table 6. The horizontal axis represents the MSOE value of the upstream traffic prediction result, and the vertical axis represents the MSOE value of the downstream traffic prediction result. Points with the same color represent the same missing value filling method, and points with the same shape represent the same neural network used for prediction. GRU-D is a non-explicitly filled neural network prediction model. Its processing of missing values is closely integrated with the training process of the neural network. Therefore, it appears in both the filling method and the neural network column. The dots in the red box in Fig. 10 overlap each other, which is not easy to observe. Re-draw the selected part of the red box in Fig. 10, as shown in Fig. 11.

In order to illustrate the effectiveness of the BGCP-RNN-ReLU model, this study uses a variety of explicit filling models and a non-explicit filling model GRU-D as a control.

Table 6 Model comparison experiment results table

Filling method	Forecasting network	Upstream traffic MSOE	Downstream traffic MSOE
<i>Explicit fill model</i>			
Mean	GRU	1.49E-03	1.77E-03
	LSTM	6.92E-03	1.72E-03
	RNN-tanh	3.97E-02	5.89E-03
	RNN-ReLU	1.34E-03	1.87E-03
Forward	GRU	3.96E-03	2.85E-03
	LSTM	1.06E-03	1.33E-03
	RNN-tanh	5.88E-03	3.67E-03
	RNN-ReLU	1.03E-03	1.30E-03
BTMF	GRU	9.65E-04	1.52E-03
	LSTM	9.66E-04	1.45E-03
	RNN-tanh	9.91E-04	1.36E-03
	RNN-ReLU	9.10E-04	1.29E-03
BGCP	GRU	9.68E-04	1.49E-03
	LSTM	1.04E-03	1.38E-03
	RNN-tanh	9.51E-04	1.39E-03
	RNN-ReLU	9.06E-04	1.28E-03
BPMF	GRU	9.34E-04	1.29E-03
	LSTM	9.36E-04	1.46E-03
	RNN-tanh	1.12E-03	1.39E-03
	RNN-ReLU	1.05E-03	1.30E-03
TF-ALS	GRU	1.24E-03	1.84E-03
	LSTM	1.19E-03	1.55E-03
	RNN-tanh	1.30E-03	1.78E-03
	RNN-ReLU	1.14E-03	1.49E-03
<i>Non-explicit filling model</i>			
Model		Upstream traffic MSOE	Downstream traffic MSOE
GRU-D		1.25E-03	1.71E-03

Fig. 10 Overall diagram of model comparison experiment results

Fig. 11 Partial view of model comparison experiment results

First, compare the overall effects of the five missing value filling methods: historical mean filling, recent observation filling, BTMF, BGCP, BPMF and TF-ALS. It can be seen from Fig. 10 that the MSOE of the upstream and downstream traffic forecasts of the four models Mean-RNN-tanh, Mean-LSTM, Forward-RNN-tanh, and Forward-GRU are much higher than those of other models, and the upstream traffic of other models MSOE is concentrated between 0 and 0.002, and downstream flow MSOE is concentrated between 0.001 and 0.002. Figure 11 shows that the upstream and downstream MSOE of the two historical average filling models are also relatively high. The downstream traffic MSOE of the two recently observed filling models is relatively small, but the upstream traffic MSOE is relatively large. Overall,

the historical average filling method has the worst effect. The recent observation filling is slightly better than the historical average filling, but it is also at a general level. The poor performance of these two filling methods may be due to the complete lack of data in two time periods in the data set. If historical mean filling or recent observation filling is used, the filling values of these two time periods will not have obvious periodicity, thus increasing the prediction error. This shows that the simple historical mean or recent observation filling is not suitable for the prediction of periodic time series with whole missing period. The corresponding points of the neural network model using BGCP, BTMF and BPMF are concentrated in the lower left corner of Fig. 11, which shows that compared with simple filling or simple tensor

decomposition, the tensor decomposition algorithm using Bayesian inference for parameter estimation can more effectively repair the missing period data.

In terms of neural networks, although the performance of the LSTM and GRU models is often better than the standard RNNs in actual use, the performance of BGCP-LSTM and BGCP-GRU on the data set used in this article is not as good as that of BGCP-RNN-ReLU. This result can be explained by the No Free Lunch Theorem (NFL Theorem): For all machine learning problems, the expected effect of any algorithm is the same, so LSTM and GRU cannot exceed the standard RNNs on all tasks. The NFL theorem shows that the more complex and commonly used models are not necessarily the better. Specific tasks and data must be analyzed in detail, and the appropriate model must be selected to obtain the desired prediction effect. Another possible reason is that the LSTM and GRU focus more on long-term information contained in time series than standard RNNs. The standard RNNs can respond to the short-term effects of the sequence, but cannot remember the long-term trend of the sequence. In most time series prediction tasks, this characteristic of the standard RNNs is a defect. But the network traffic data is more special: the traffic of an hour is mainly affected by the traffic of the last few hours, and less affected by the traffic of a week or a month ago. Therefore, the characteristic that the standard RNNs only memorizes short-term information may improve its prediction performance.

Among the six RNN-ReLUs using different missing value filling methods, except for Mean-RNN-ReLU which is affected by the historical mean filling method, the prediction accuracy of RNN-ReLU is average. The prediction accuracy of RNN-ReLU using TF-ALS algorithm is average, and the other four The points corresponding to RNN-ReLU are also concentrated in the lower left corner of Fig. 11, indicating that RNN-ReLU has the best overall prediction accuracy among the neural networks involved in this article.

The non-explicitly filled GRU-D has been proven to perform better on many time series classification tasks with missing values, but its performance on the cell wireless network traffic prediction task is more general, even worse than the simple Forward-RNN-ReLU It's slightly worse. This phenomenon also conforms to the NFL theorem.

The upstream MSOE and downstream MSOE of the BGCP-RNN-ReLU proposed in this paper are $9.06\text{E-}04$ and $1.28\text{E-}03$ respectively, which both reach the lowest value. Compared with the best results of historical average filling and recent observation filling (the corresponding model is Forward-RNN-ReLU), the upstream traffic MSOE of BGCP-RNN-ReLU has dropped by about 12.04%, and the downstream traffic MSOE has dropped by about 1.54%, which proves The validity of the model.

7 Conclusion

This paper combines a new tensor decomposition method BGCP with RNN, and first uses BGCP to dynamically adjust the filling strategy according to the state of the neural network: (i) if the neural network is in the training stage, the tensor decomposition is performed on all the data contained in the training set; (ii) if the neural network is in the prediction stage, the tensor decomposition is carried out sample by sample. The filled samples are sent to the standard RNN with the ReLU activation function, and the RNN learns time series information from the data and makes predictions for changes in the short term in the future. The actual results show that the prediction accuracy of BGCP-RNN is higher than other commonly used methods.

The changes in the upstream and downstream traffic of the cell are related to a certain extent. For example, usually low downstream traffic means that there are fewer users using the wireless network currently, and therefore the upstream traffic is also relatively low. However, the forecast of upstream and downstream traffic in this article is carried out separately, and the connection between the two is not considered. Subsequent work can further study the correlation of changes in cell up-down-traffic, and use the correlation to obtain more accurate and stable prediction results.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Parwez MS, Rawat DB, Garuba M (2017) Big data analytics for user-activity analysis and user-anomaly detection in mobile wireless network[J]. *IEEE Trans Industr Inf* 13(4):2058–2065
2. Che Z, Purushotham S, Cho K et al (2018) Recurrent neural networks for multivariate time series with missing values[J]. *Sci Rep* 8(1):1–12

3. Yi X, Zheng Y, Zhang J, et al. ST-MVL: Filling Missing Values in Geo-Sensory Time Series Data Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences[C]. Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2016), New York, NY, USA. 2016: 9–15.
4. Flores A, Tito H, Silva C (2019) Local average of nearest neighbors: univariate time series imputation[J]. *Int J Adv Comput Sci Appl* 10(8):45–50
5. Bashir F, Wei HL (2018) Handling missing data in multivariate time series using a vector autoregressive model-imputation (VAR-IM) algorithm[J]. *Neurocomput* 276:23–30
6. Sidiropoulos ND, De Lathauwer L, Fu X et al (2017) Tensor decomposition for signal processing and machine learning[J]. *IEEE Trans Signal Process* 65(13):3551–3582
7. Zhang Z, Aeron S (2016) Exact tensor completion using t-SVD[J]. *IEEE Trans Signal Process* 65(6):1511–1526
8. Chen X, He Z, Sun L (2019) A Bayesian tensor decomposition approach for spatiotemporal traffic data imputation[J]. *Transp Res Part C: Emerg Technol* 98:73–84
9. Shafiq M, Yu X, Laghari A A, et al. Network traffic classification techniques and comparative analysis using machine learning algorithms[C]. 2016 2nd IEEE International Conference on Computer and Communications (ICCC). IEEE, 2016: 2451–2455.
10. Mozo A, Ordozgoiti B, Gómez-Canaval S (2018) Forecasting short-term data center network traffic load with convolutional neural networks[J]. *PLoS ONE* 13(2):e0191939
11. Li M, Wang Y, Wang Z et al (2020) A deep learning method based on an attention mechanism for wireless network traffic prediction[J]. *Ad Hoc Netw* 107:102258
12. Nie L, Jiang D, Yu S, et al. Network traffic prediction based on deep belief network in wireless mesh backbone networks[C]. 2017 IEEE Wireless Communications and Networking Conference (WCNC). IEEE, 2017: 1–5
13. Vinayakumar R, Soman K P, Poornachandran P. (2017) Applying deep learning approaches for network traffic prediction[C]. 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI). IEEE, 2353–2358.
14. Jinmian Ye, Linnan Wang, Guangxi Li, Di Chen, Shandian Zhe, Xinqi Chu, Zenglin Xu. (2018) Learning Compact Recurrent Neural Networks With Block-Term Tensor Decomposition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 9378–9387.
15. You Y, He Y, Rajbhandari S et al (2020) Fast LSTM by dynamic decomposition on cloud and distributed systems[J]. *Knowl Inf Syst* 62(11):4169–4197
16. Pan Y, Xu J, Wang M, et al. Compressing recurrent neural networks with tensor ring for action recognition[C]. Proceedings of the AAAI Conference on Artificial Intelligence. 2019, 33(01): 4683–4690
17. Qin J, Yang Y, Hong Z, et al. Missing Data Imputation for Operation Data of Transformer Based on Functional Principal Component Analysis and Wavelet Transform[C]//International Conference on Intelligent Computing. Springer, Cham, 2019: 555–567.
18. Wilson RE, Eckley IA, Nunes MA et al (2021) A wavelet-based approach for imputation in nonstationary multivariate time series[J]. *Stat Comput* 31(2):1–18
19. Naber A, Mastinu E, Ortiz-Catalan M (2019) Stationary wavelet processing and data imputing in myoelectric pattern recognition on a low-cost embedded system[J]. *IEEE Trans Med Robot Bionics* 1(4):256–266

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.