Accurate Step Length Estimation for Pedestrian Dead Reckoning Localization Using Stacked Autoencoders

Fuqiang Gu, *Student Member, IEEE*, Kourosh Khoshelham, Chunyang Yu, and Jianga Shang, *Member, IEEE*

***Abstract*— Pedestrian dead reckoning (PDR) is a popular indoor localization method due to its independence of additional infrastructures and the wide availability of smart devices. Step length estimation is a key component of PDR, which has an important influence on the performance of PDR. Existing step length estimation models suffer from various limitations such as requiring knowledge of user's height, lack of consideration of varying phone carrying ways, and dependence on spatial constraints. To solve these problems, we propose a deep learning-based step length estimation model, which can adapt to different phone carrying ways and does not require individual stature information and spatial constraints. Experimental results show that the proposed method outperforms existing popular step length estimation methods.**

***Index Terms*— Autoencoder, deep learning, neural networks, positioning, smartphone sensors, step length.**

I. Introduction

INDOOR localization has applications in a variety of domains such as museum guide, shopping guide, search and rescue, mobile advertising, and location-enabled social networks [1]. The fundamental task of indoor localization is to determine the location of an entity (e.g., a person) in indoor spaces where the widely-used and well-established global positioning system does not work. A lot of indoor localization methods have been proposed and developed in [2], which differ from each other in terms of the localization techniques used, coverage, accuracy, cost of deployment, and maintenance.

Among various indoor localization methods, pedestrian dead reckoning (PDR) [3]–[6] has become one of the mainstream methods due to the advent of smart devices such as smartphones, smart watches, smart bands, and smart glasses. Compared to other indoor localization methods, PDR has several advantages. First, unlike WiFi-based methods [7] or Bluetooth-based methods [8], which depend on an infrastructure of access points or beacons, PDR does not require any infrastructures. Second, it has no need for a laborious pretraining process, whereas WiFi-based or Bluetooth-based methods usually need to collect fingerprints before localization, which is time-consuming and labor-intensive. Third, it has wider availability than other methods because of the popularity of smart devices. Although WiFi is also accessible in many public places, it is still challenging to provide continuous localization service by using only WiFi access points since their coverage is limited. By contrast, PDR has no coverage limitation. Given an initial location, it can infer the location of the user in real time by using the readings from inertial sensors (e.g., accelerometers, gyroscopes, and magnetometers) built in most modern smart devices.

Step length estimation is one of the key components of PDR, and its accuracy will directly affect the accuracy of PDR localization. Many methods have been proposed for estimating the step length, mainly including human gait-based [9]–[12], step frequency-based [13], [14], and step counting (SC)-based methods [15], [16]. However, these step length estimation methods suffer from various limitations such as unsuitability for smartphone-based applications [9]–[11], lack of consideration of different phone poses [17], [18], being user dependent [13], [14], and relying on spatial constraints [15], [16], [19], [20].

The purpose of this paper is to design a step model that considers varying phone poses and walking speeds, works for different users, and does not require spatial information assistance. This is a challenging and complex task due to three reasons. First, the step length varies from person to person, resulting in the generic model being less accurate. Second, the accelerometer readings, which are used to estimate the step length, are affected by different phone poses and user's walking speeds. This leads to the difficulty in accurately estimating the step length using accelerometer readings. Third, the spatial constraints such as landmarks, which can be used to calibrate the user's step length, are not always available.

On the other hand, the recently-developed deep learning is suitable for dealing with complex tasks, which has been used in many domains such as image classification [21], natural language processing and speech recognition [22], [23],

Manuscript received April 29, 2018; revised September 4, 2018; accepted September 9, 2018. This work was supported in part by the National Key Research and Development Program of China under Grant 2016YFB0502200 and in part by the China Scholarship Council—University of Melbourne Research Scholarship under Grant CSC 201408420117. The Associate Editor coordinating the review process was Subhas Mukhopadhyay. *(Corresponding author: Fuqiang Gu.)*

F. Gu and K. Khoshelham are with the Department of Infrastructure Engineering, University of Melbourne, Melbourne, VIC 3000, Australia (e-mail: fuqiangg@student.unimelb.edu.au; k.khoshelham@unimelb.edu.au).

C. Yu is with the Department of Geomatics Engineering, University of Calgary, Calgary, AB T2N 1N4, Canada (e-mail: chunyang.yu@ucalgary.ca).

J. Shang is with the Faculty of Information Engineering, China University of Geosciences, Wuhan 430074, China, and also with the National Engineering Research Center for Geographic Information System, Wuhan 430074, China (e-mail: jgshang@cug.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIM.2018.2871808

activity recognition [24], and WiFi fingerprinting [25]. This paper is especially motivated by the success of deep learning for activity recognition that uses the same accelerometer signals as the step length estimation does. However, activity recognition using deep learning is based on classification, while step length estimation is based on regression. In [26], a bidirectional long short-term memory recurrent neural network is used to achieve more robust step detection and counting for PDR, after which the step length is estimated by a linear model based on accelerometer data. In this paper, we estimate the step length by directly using the stacked autoencoders (SAs) [27], [28] based on both accelerometer data and gyroscope data. The reason for fusing gyroscope data is its usefulness for recognizing different phone poses, making our step length estimation more robust. To our knowledge, this paper is the first to directly estimate the step length using deep learning.

The main contributions of this paper are as follows.

1) We propose a deep learning-based step length estimation method that considers different walking speeds, phone carrying ways, and can adapt to characteristics of different users.
2) We analyze the influence of different network configurations on the accuracy of step length estimation. The impacts of different number of layers, number of neurons, and noise level are analyzed.
3) We compare our method with conventional step length estimation methods and demonstrate that our method outperforms the existing commonly-used methods.

The remainder of this paper is organized as follows. In Section II, we review the related works. Section III describes the proposed step length estimation method. The experiments and results are presented in Section IV. Finally, this paper is concluded in Section V.

II. Related Work

PDR consists of two components: step length estimation and heading estimation. The heading estimation can be obtained from the compass readings (derived from the magnetometer readings and accelerometer readings) or the gyroscope readings. However, the compass readings are susceptible to the ferromagnetic materials and the gyroscope has the drift problem. One solution to achieve an accurate heading estimation is to use a Kalman filter to combine the compass readings with the gyroscope readings [16]. A more complex heading estimation method is proposed in [29], which considers different device poses.

The step length estimation is usually based on the accelerometer readings. It involves the detection of step events, which can be done by detecting the step cycle of a user's walking [15], [16], [34]. After this, different models can be used to compute the step length. Since the low-cost smartphone sensors are not very reliable and accurate, it is inaccurate to estimate the step length by double integrating the acceleration. Weinberg proposed a step length estimation approach based on the maximum vertical displacement of the hip, which can be approximated as a function of the maximum and minimum of vertical accelerations [17]. Kim *et al.* [18] also introduced a similar model that uses the acceleration samples to estimate the step length. The disadvantage of these acceleration samples-based models is that they do not consider different phone carry ways and varying walking speeds, which have an important effect on the estimation accuracy. A linear model that considers walking speeds was used in [13] and [14], but it requires knowledge of the user's height, which may limit its applicability since some users are not willing to provide their individual information. There are other frequency-based step length estimation methods [33], which consider different walking speeds, but they also require user's height information. An adaptive step model is proposed in [19], which uses a personalization algorithm to learn a personal model from a generic step model. However, this personalization process is based on spatial constraints from the floor plan, which are not always available. In [12], a neural network-based method is introduced, which considers walking frequency, variance of the accelerometer signals, and the ground inclination. However, it is based on the shoe-mounted accelerometer and, hence, is unsuitable for smartphone-based applications. A knowledge-based step length estimation method is proposed in [30], which is based on fuzzy logic and multisensor fusion. This method assumes that the device is mounted on the user's waist, which is a limiting assumption in practical applications. Park *et al*. [31] proposed a walking speed estimation method independent of device poses, which uses regularized kernel methods. However, the method proposed by Park *et al*. [31] requires to design features manually, which involves expert knowledge. Hu *et al*. [32] developed a speed estimation method by using a kinematic human-walking model based on a waist-mounted accelerometer. The step length can be estimated by combining SC with spatial information such as landmarks or floor plans [15], [16]. Although these methods eliminate the requirement for individual height information and are independent of phone carrying ways, their assumption that the user walks at a consistent speed is not always practical.

Recently, deep learning has become a hot research topic since it can learn features of data automatically and has shown excellent performance in different application domains such as image classification [21], natural language processing and speech recognition [22], [23], and playing games [35]. The commonly-used deep learning methods include SAs [36], deep belief networks [37], convolutional neural networks [38], and recurrent neural networks [39]. These methods are originally proposed for image classification and natural language processing and speech recognition, but they have also been using in human activity recognition [24], indoor localization [25], and other domains. However, deep learning has not been used for estimating the step length. To our knowledge, this paper is the first to use deep learning for step length estimation.

III. Proposed Method

A. Architecture

The architecture of the proposed step length estimation method is illustrated in Fig. 1, mainly including segmentation, feature learning, and step length estimation modules.

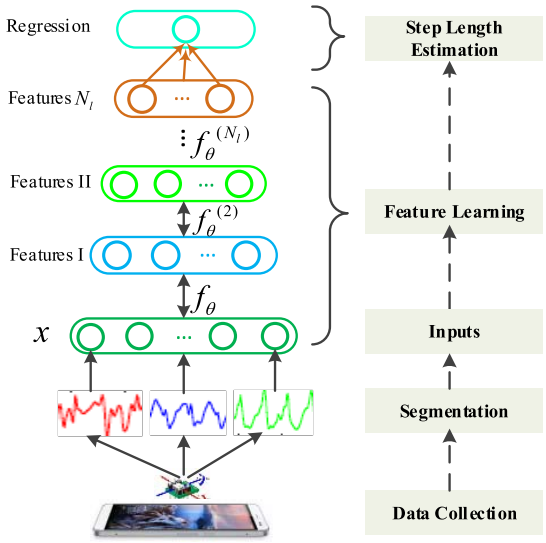

Fig. 1. Architecture of the step length estimation using SA.

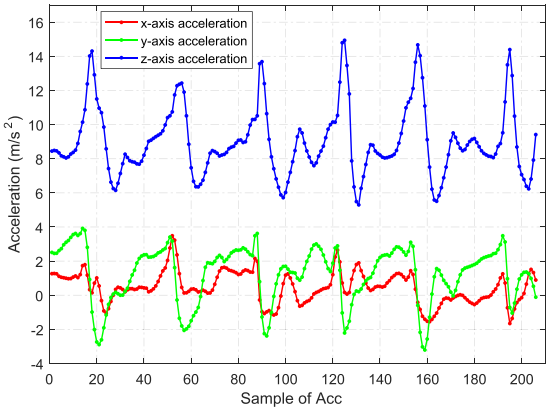

Fig. 2. Periodicity and repetitiveness of walking (the user walks six steps with the phone in hand).

The smartphone is used to collect the accelerometer data and gyroscope data, which are fed to a low-pass filter to remove random noise. Next, both the smoothened accelerometer readings and gyroscope readings are divided into segments with each segment representing the data for one step. Then, these segments are fed to the SA to learn useful features, which is a training process. On the top layer is an affine regression layer, which estimates the step length. In the following, we will elaborate the key steps of our method.

B. Segmentation

Before computing the step length, we need to divide the sensor readings into segments with each segment corresponding to one step. This is done by detecting when a step event happens.

The acceleration readings present a periodical and repetitive pattern when the user walks, as shown in Fig. 2. To make the detection method independent of the smartphone's orientation, the amplitude of the acceleration is utilized to detect the step

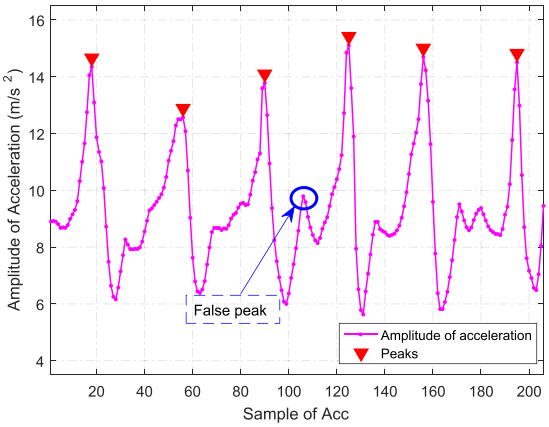

Fig. 3. Peak detection (the user walks six steps with the phone in hand).

event, namely

$$\text{acc}_t = \sqrt{\text{acc}_{x_t}^2 + \text{acc}_{y_t}^2 + \text{acc}_{z_t}^2} \tag{1}$$

where acc_{x_t}, acc_{y_t}, and acc_{z_t} are the acceleration at time t along the x-, y-, and z-axes, respectively. A low-pass filter is used to improve the accuracy of peak detection.

Then, the peak detection method can be used to identify a step event, which is based on the fact that the acceleration will periodically present peaks when a user is walking, as shown in Fig. 3. The peaks can be extracted by checking whether the peak detection condition is met as follows:

$$\begin{aligned}\text{peak}_t = (\text{acc}_t | \text{acc}_t >= (\text{acc}_{t-K} : \text{acc}_{t-1}) \\ \&\&\text{acc}_t >= (\text{acc}_{t+1} : \text{acc}_{t+K}))\end{aligned} \tag{2}$$

where K is a threshold used to help detect the right peaks, the value of which is determined by both the sampling rate of the accelerometer and the user's walking speed. Note that false peaks (e.g., as marked by the blue circle in Fig. 3) are avoided by considering the user's step periodicity. If the step periodicity is beyond a certain interval, it will be considered as a false peak. More details about false peak detection can be found in [40]. After the peak detection, we can divide the accelerometer readings and gyroscope readings into segments that are used to compute the step length at different speeds and phone poses. The step events can also be detected by utilizing zero crossings, autocorrelation, and spectral analysis [3].

Once step events are detected, we are able to partition the accelerometer readings and gyroscope readings along each axis into segments. These segments are created using a sliding window as follows:

$$\boldsymbol{s}_i^{\text{acc}_x} = [\text{acc}_t^x, \text{acc}_{t+1}^x, \cdots, \text{acc}_{t+m-1}^x] \tag{3}$$
$$\boldsymbol{s}_i^{\text{acc}_y} = [\text{acc}_t^y, \text{acc}_{t+1}^y, \cdots, \text{acc}_{t+m-1}^y] \tag{4}$$
$$\boldsymbol{s}_i^{\text{acc}_z} = [\text{acc}_t^z, \text{acc}_{t+1}^z, \cdots, \text{acc}_{t+m-1}^z] \tag{5}$$
$$\boldsymbol{s}_i^{\text{gyro}_x} = [\text{gyro}_t^x, \text{gyro}_{t+1}^x, \cdots, \text{gyro}_{t+m-1}^x] \tag{6}$$
$$\boldsymbol{s}_i^{\text{gyro}_y} = [\text{gyro}_t^y, \text{gyro}_{t+1}^y, \cdots, \text{gyro}_{t+m-1}^y] \tag{7}$$
$$\boldsymbol{s}_i^{\text{gyro}_z} = [\text{gyro}_t^z, \text{gyro}_{t+1}^z, \cdots, \text{gyro}_{t+m-1}^z] \tag{8}$$

where m is the segment size, which corresponds to the number of sensor reading samples for one step. Since the

sampling frequency of the low-cost smartphone accelerometer and gyroscope is not very stable or the user may walk in different speeds, we use the *spline interpolation* to generate accelerometer reading and gyroscope reading samples of the same size for each step, which is a prerequisite to use deep neural networks.

C. Deep Model for Step Length Estimation

In this section, we present the proposed model for step length estimation, which integrates the SA with a linear regression model. The SA learns useful features for step length estimation from accelerometer data and gyroscope data, which are then fed to the regression layer to compute the step length.

We first introduce the feature learning of step length using the SA, which encompasses multiple layers of autoencoders. An autoencoder learns features automatically by minimizing the error of reconstructing the input [27], [28]. Let $\boldsymbol{x}_i$ be the input vector at step i, consisting of acceleration segments along three axes, gyroscope reading segments along three axes, and the time interval T_i between two neighboring peaks reflecting the step frequency, namely

$$\boldsymbol{x}_i = \left[\boldsymbol{s}_i^{\text{acc}_x}, \boldsymbol{s}_i^{\text{acc}_y}, \boldsymbol{s}_i^{\text{acc}_z}, \boldsymbol{s}_i^{\text{gyro}_x}, \boldsymbol{s}_i^{\text{gyro}_y}, \boldsymbol{s}_i^{\text{gyro}_z}, T_i\right]^T \tag{9}$$

where $\boldsymbol{x}_i$ is a $M \times 1$ vector and $M = 6m + 1$ (m is the segment size). The encoding process of an autoencoder is done by applying a sigmoid function f to the input vector

$$\boldsymbol{a} = f(\boldsymbol{W}_1 \boldsymbol{x}_i + \boldsymbol{b}_1) \tag{10}$$

where $\boldsymbol{W}_1$ is a $N \times M$ encoding matrix, and N is the number of input segments or features. $\boldsymbol{a}$ and $\boldsymbol{b}_1$ are the N-dimensional activation vector and bias vector, respectively. The decoding is done by performing a similar process

$$\widehat{\boldsymbol{x}}_i = g(\boldsymbol{W}_2 \boldsymbol{a} + \boldsymbol{b}_2) \tag{11}$$

where g is the decoding mapping (a sigmoid function), $\boldsymbol{W}_2$ is a $M \times N$ decoding matrix, and $\boldsymbol{b}_2$ is a M-dimensional bias vector. The goal of feature learning is to minimize the reconstruction error, which is done by minimizing the square error loss function $J(\boldsymbol{x}_i, \widehat{\boldsymbol{x}}_i)$

$$J(\boldsymbol{x}_i, \widehat{\boldsymbol{x}}_i) = \frac{1}{2} \sum_{j=1}^{M} (x_j - \widehat{x}_j)^2. \tag{12}$$

To enable the SA to work even when the number of hidden units is larger than the input dimension, we add a sparsity term to the objective function. The resulting cost function J_{ae} is described as

$$J_{\text{ae}} = J(\boldsymbol{x}_i, \widehat{\boldsymbol{x}}_i) + \beta \sum_{j=1}^{N} \text{KL}(\rho || \hat{\rho}_j) \tag{13}$$

where KL is the Kullback–Leibler divergence [41] between the sparsity parameter ρ and the average activation $\hat{\rho}_j$ of hidden unit j. β is the sparsity penalty.

The SA is composed of multiple layers of autoencoders where the outputs of each layer are used as the inputs of the next layer. The training of the SA is done by the greedy layerwise training method. Once the SA is built, a supervised regression layer is placed on its highest layer to compute the step length. The global objective is to minimize the following cost function, namely:

$$J = \frac{1}{2N_L} \sum_{i=1}^{N_L} (\boldsymbol{\theta} \boldsymbol{a}_i - y_i)^2 + \frac{\lambda}{2} \boldsymbol{\theta} \boldsymbol{\theta}^T \tag{14}$$

where N_L is the number of units on the last layer of the SA, y_i is the ground-truth step length corresponding to the input $\boldsymbol{x}_i$, and $\boldsymbol{a}_i$ is the output from the last layer of the SA. $\boldsymbol{\theta}$ is a $1 \times N_L$ weight vector connecting the units on the last layer of the SA and the unit on the regression layer, and λ is a weight decay parameter. The first term of (14) is the error between the ground-truth step length and the estimated value, while the second term is a weight decay term to avoid overfitting.

Algorithm 1: Proposed Step Length Estimation Model

Input : labeled training data set $D_{labeled} = \{X^{tr}, Y\}$, unlabeled testing data set $D_{test} = \{X^{te}\}$
Output: Step length sequence L of the unlabeled testing data

// Initialization:
Initialize the network parameters
Segment the accelerometer data and gyroscope data by detecting the peaks of the amplitude of accelerometer readings
Stabilize the number of sensor samples for each step by the spline interpolation
Form a sequence of segments with the same number of samples $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N\}$
// Training from the first layer ($l = 1$):
Set the layer index l to 1;
repeat
  Train the l-th layer of the SA using the data sequences, and obtain the encoding function $f^{(l)}$
  Compute the outputs of the l-th layer by using the learned function $f^{(l)}$ on the input $\{\boldsymbol{x}_1^{l-1}, \boldsymbol{x}_2^{l-1}, \cdots, \boldsymbol{x}_N^{l-1}\}$, which will feed to the $l+1$-th layer as inputs
until $l++ == L$;
Use labeled data set $D_{labeled}$ to train the top layer (regression layer)
Fine-tune the entire network through backpropagation
// Testing:
Use the trained network to predict the step length sequence L of data set D_{test}

The complete procedures of the proposed step length estimation model are shown in Algorithm 1. It takes as input a set of training samples X^{tr} with the corresponding ground-truth step length Y to train the network. This algorithm starts by initializing the network parameters. Specifically, we adopt the weight initialization strategy in [42], which involves initializing the weights W_{ij}^l to values that are randomly drawn from the interval $[-((6/(n_{\text{in}} + n_{\text{out}} + 1)))^{1/2}], ((6/(n_{\text{in}} + n_{\text{out}} + 1)))^{1/2}]$, where n_{in} is the number of inputs feeding into a node and n_{out} is the number

Fig. 4. Phone poses in the experiments.

of units that a node feeds into. The biases b_i^l are set to zero. Then, the accelerometer readings and gyroscope readings are divided into segments by conducting peak detection on the amplitude of accelerometer readings. The *spline interpolation* is applied to make these segments have an equal number of samples. Then, the network is trained in a layerwise way. The labeled data set is used to train the linear regression layer on the top. A fine-tuning operation is then followed to optimize the parameters of all layers through backpropagation. Once the training is done, the network can be used to compute the step length of given samples.

IV. Experiments and Results

A. Experimental Setup

The proposed step length estimation method was evaluated by a series of experiments. Twelve participants were asked to collect data using two phones (Samsung Galaxy S III and S IV). During the data collection, the participants were required to count the number of steps they took, which was used to calibrate the peak detection to make the ground-truth step length more accurate. Data collection includes training data collection and testing data collection. In the process of collecting training data, participants were asked to walk along a path of 50 m in four motion modes (slow walking, normal walking, fast walking, and jogging) and two phone carrying ways (swinging with the arm, and in the pocket, as shown in Fig. 4), respectively. Each trajectory of collecting training data corresponds to one mode and one phone carrying way, which means that the participant walked at a constant pace and carried the phone in a fixed way. This is to guarantee the accuracy of the training data. When a user walks at a constant pace, his/her step length for each step is approximately the same. The ground-truth step length for training data can be then obtained by dividing the length of the path by the number of steps walked. In the testing data collection, the participants were asked to take 100 m for four times in two motion modes (fixed speed mode and variable speed mode) and two phone carrying ways, respectively. During the process of variable speed mode, the users were asked to change their walking speeds to include data of different walking speeds. The motion speed of users varies from 3.4 to 13.5 km/h, which is computed by dividing the length of the test path by the time consumed to travel the given path. Table I shows the height and gender of the participants.

TABLE I
User Profile

User No.	Height (cm)	Gender	User No.	Height (cm)	Gender
1	170	male	7	176	male
2	166	male	8	174	male
3	170	male	9	175	male
4	162	female	10	166	female
5	173	male	11	178	male
6	162	female	12	173	male

TABLE II
Experiment Configuration

Parameter	Value
Number of users	12
Number of training trajectories	76
Number of testing trajectories	38
Number of training data segments	4834
Number of testing data segments	4784
Path length for training data collection	50 m
Path length for testing data collection	100 m

TABLE III
List of Hyperparameters for Deep Networks

Hyperparameter	Description	Considered values
M	input dimension	193
L	number of hidden layers	{**2**,3,4}
N_h	number of units per hidden layer (same for all layers)	{50, 100, 200, **500**, 1000}
α	learning rate	1×10^{-3}
β	weight of sparsity penalty term	1

Table II gives the experiment configuration. In total, we collected training data of 76 valid trajectories (consisting of 4834 data segments) and testing data of 38 valid trajectories (comprising 4784 data segments). Each segment is a vector of 193 elements, including 96 acceleration samples (32 samples from each axis), 96 gyroscope samples (32 samples from each axis), and one time interval representing the step frequency between two neighboring peaks.

B. Hyperparameter Setting

Table III gives a list of the hyperparameters we considered in this paper. To reduce the selection space, we let all the hidden layers share the same number of units and the same learning rate. It should be noted that the bold value for each hyperparameter is used in the following analysis when there is no mention specifically.

C. Step Length Estimation Accuracy

We use the relative error to measure the performance of our step length estimation model, namely

$$e = \frac{\left| y_g - \sum_{i=1}^{N} \hat{y}_i \right|}{y_g} \times 100\% \tag{15}$$

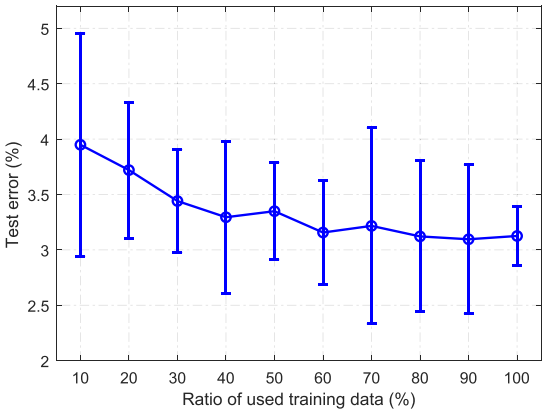

Fig. 5. Training curve.

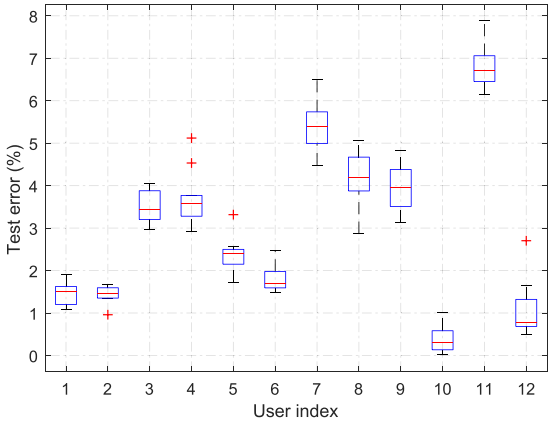

Fig. 7. Test error of the proposed method for different users.

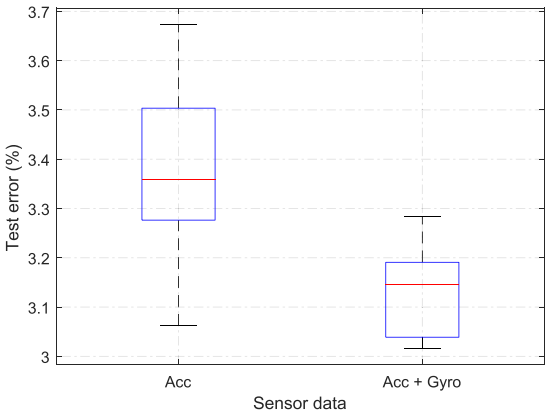

Fig. 6. Test error of the proposed method using different sensors.

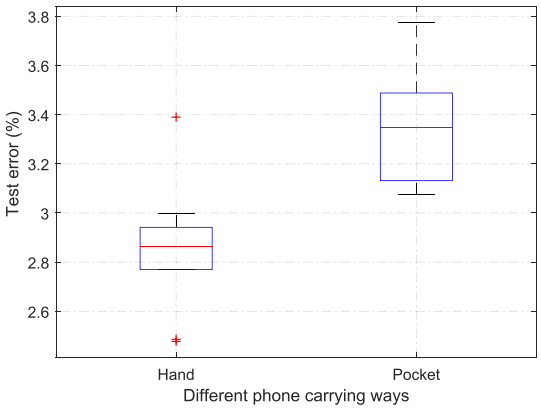

Fig. 8. Test error of the proposed method in different phone poses.

where $\hat{y}_i$ is the estimated step length for the ith step, and y_g is the length of the testing path. As the performance of SAs is affected by the initial values of network parameters, we ran the program 10 $\times$ for each parameter setting and used the average performance to analyze the effect of different parameters and variables.

We first give the training curve as shown in Fig. 5, which implies that the network is sufficiently trained with the available samples since it converges toward the end and the test error rate shows little improvement with more samples. The average training error using 10-fold cross validation on the training data set is about 0.3%, showing the sufficiency of network training.

Then, we compare the performance of the proposed step length estimation method using accelerometer readings only and that using the combination of accelerometer readings and gyroscope readings. As demonstrated in Fig. 6, the test error of using the combination of accelerometer data and gyroscope data (3.13%) is lower than that of using accelerometer data only (3.36%), though both use the same network structure (two layers, 500 units per layer). This is attributed to that the gyroscope readings are helpful in determining different phone carrying ways. Therefore, in the following, we use the combination of both sensor data to analyze the effect of other parameters and variables.

Fig. 7 shows the test error of the proposed method for different users. Note that although both training data set and testing data set were collected by the 12 users, they are from different trajectories and, hence, are independent. It can be seen that the step length estimation error varies from user to user since different users have varying walking characteristics. The user 11 witnesses a large error, and this might be due to the walking characteristics he/she behaved in collecting testing data are different from these characteristics in the training data set (including from herself and other users). The average test error for the 12 users is about 3.1%.

Next, we analyze the influence of different phone poses on the proposed step length estimation method. It is interesting to see from Fig. 8 that the error for the Swing phone pose (2.85%) is much smaller than that for the pocket case (3.35%). This is because when the user walks naturally with the phone swinging with the arm, the pace of swinging arm is consistent with the pace of taking steps. On the other hand, there may be certain noisy movement between the phone and the trouser's pocket when the phone is put in the trouser's pocket, which contributes to a larger error in the step length estimation.

The effect of different testing speed modes is shown in Fig. 9. The case of fixed testing speed mode witnesses a smaller error (2.91%) than that of the variable speed mode (3.22%). This is because the users were free to change their

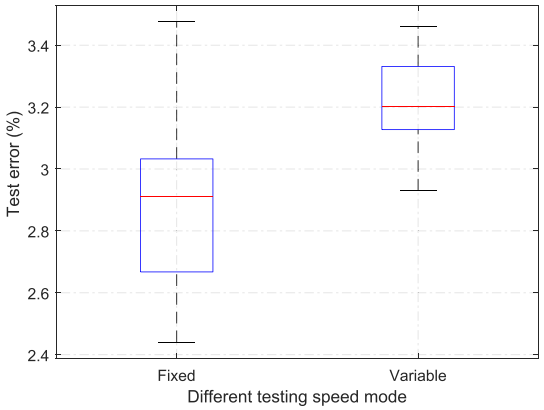

Fig. 9. Test error of the proposed method in different speed modes.

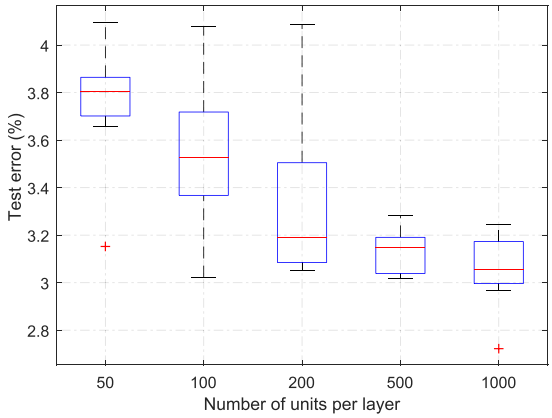

Fig. 11. Test error of using different units per layer.

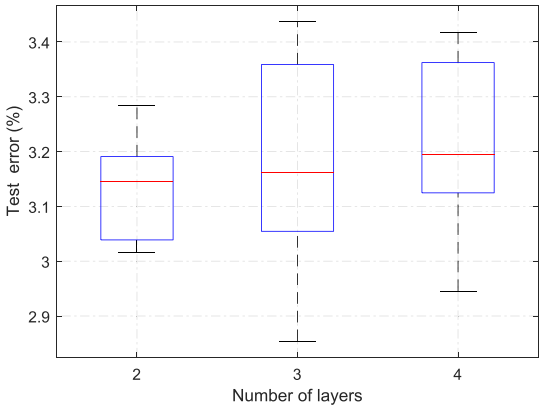

Fig. 10. Test error of using different layers.

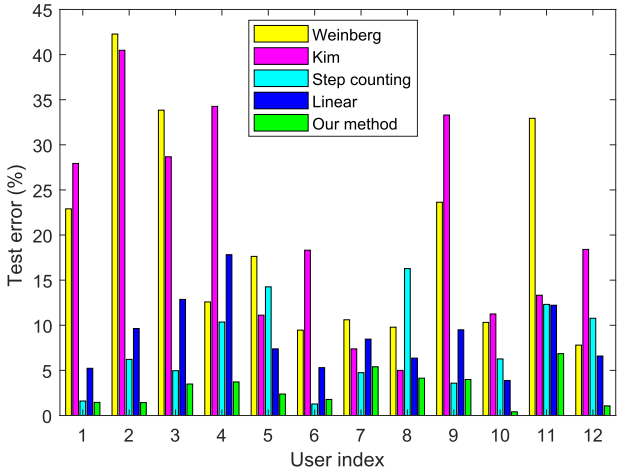

Fig. 12. Performance comparison with commonly-used methods.

walking speeds in the variable speed mode, and therefore, it is more likely to introduce more uncertainty in the testing data.

D. Effect of Network Structure

We analyze the effect of different number of layers and number of units on the step length estimation. Other network parameters such as learning rate α and weight of sparsity penalty term β are simply set to the default values as shown in Table III, which are empirically determined.

Fig. 10 shows the step length estimation error of using different number of layers with 500 units per layer, from which we can see that the best performance is achieved by the network with two layers and increasing the number of layers does not improve the step length estimation. This is because there are no sufficient data segments to well train a complex network with many layers.

Fig. 11 shows the performance of the proposed step length estimation model with different number of neuron units. It is clear that the general trend in the error is that using more neurons per layer will decrease the estimation error. This is especially obvious when increasing the number of units from 50 to 100, and further to 200, the corresponding error decreases from 3.80% to 3.52% and further to 3.19%. After the number of units reaches 500 per units, the further increase of units does not significantly reduce the error but will considerably increase the computational cost. Therefore, there is usually a tradeoff between the performance and the cost of computation and storage.

E. Comparison With Popular Methods

We compare the proposed step length estimation model with the commonly-used methods, including the Weinberg model [17], Kim model [18], linear model [14], and SC-based method [15]. The parameters of these methods are calculated in a way that minimizes the training error by using the training data set. The comparison results are shown in Fig. 12 and Table IV.

Generally, our method outperforms these commonly-used step length estimation methods. For all the users, our method can achieve a good estimation accuracy with an average error of 3.01%. Among these commonly-used methods, the SC-based method and the linear step length model perform much better than the model-based methods (Weinberg model and Kim model). This is because the linear step length model considers the user's height and step frequency, which is more robust against different walking speeds and phone poses than model-based methods. The reason why the SC-based method performs the best among conventional methods might be that

TABLE IV
PERFORMANCE COMPARISON WITH COMMONLY-USED METHODS

Method	Average error (%)
Weinberg	19.48
Kim	20.79
Linear model	8.77
Step counting-based	7.72
Our method	3.01

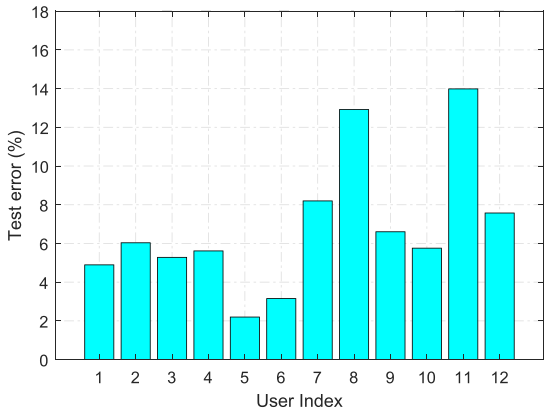

Fig. 13. Estimation error for new users.

the average step length for each step in the testing data set is close to that in the training data set. The performance of the Weinberg model (19.48%) and the Kim model (20.79%) is similar since both of them take as input the acceleration and a parameter that is related to user's height, but they do neither consider step frequency nor different phone poses.

Overall, the commonly-used step length estimation methods are user specific, which means that the model trained by a user does not work well for another. Also, they usually have the need for users' information such as height. However, our method can adapt to characteristics of different users, varying walking speeds, and does not require individual information.

F. Estimation Error for New Users

To analyze the performance of the proposed method with data from a new user, we select in turn a user from the 12 users. The data from the remaining 11 users are used as training data while the data from the selected user as test data. Fig. 13 demonstrates the estimation error of the proposed method for a new user. Overall, the average error of estimating the step length of a random new user is about 6.85%, which is higher than the 3.01% achieved by using data from all users. The relatively high average error of step length is mainly caused by user 8 and user 11, who experience an error of about 13% and 14%, respectively. This is due to the two participants (user 8 and user 11) share less common walking characteristics with other participants. Another possible explanation is that these participants have less uniform walking characteristics and their step lengths tend to vary between different walking modes resulting in large testing errors. It is expected that the estimation error will be reduced by using more data from users of different heights and walking characteristics.

TABLE V
TRAINING AND TEST TIME

Number of layers	Number of neurons	Training time (s)	Test time (s)
2	500	458.1	0.17
	1000	1999.1	0.57
3	500	793.4	0.27
	1000	4672.7	1.03
4	500	1107.5	0.37
	1000	21543.1	1.35

G. Computational Cost

The computational complexity of the proposed method is $\mathcal{O}(N_L \cdot M + L \cdot N_L^2)$, where L is the number of layers, N_L is the number of neurons per layer, and M is the dimension of input data. Table V shows the training and test time of conducting the proposed method with different network parameters on the whole training data and test data. The proposed method was implemented in MATLAB and conducted on a PC equipped with an Intel Core i5-8400 CPU at 2.80 GHz and a memory of Ramaxel DDR4 8G. It can be seen that both training time and test time increase as the number of layers or the number of neurons per layer increases. Note that these computation times are indicative. We expect that more optimized implementations will be able to run in real time on modern smartphones and other smart devices.

V. CONCLUSION

This paper presents a deep learning-based method for accurately estimating the step length of a user, which is important for the PDR indoor localization. The proposed method can adapt to characteristics of different users, varying walking speeds, and phone poses and has no need for spatial constraints. The influence of different values of network parameters is analyzed, including the number of layers and number of neurons. By comparing with existing commonly used step length estimation methods, we show the superiority of our method.

In the future, we will investigate how to obtain training data automatically by crowdsourcing, which will significantly increase the volume of training data. This will undoubtedly further improve the performance of the proposed method.

REFERENCES

[1] J. Shang, X. Hu, F. Gu, D. Wang, and S. Yu, "Improvement schemes for indoor mobile location estimation: A survey," *Math. Problems Eng.*, vol. 2015, Mar. 2015, Art. no. 397298.

[2] P. Davidson and R. Piché, "A survey of selected indoor positioning methods for smartphones," *IEEE Commun. Surveys Tuts.*, vol. 9, no. 2, pp. 1347–1370, 2nd Quart., 2016.

[3] R. Harle, "A survey of indoor inertial positioning systems for pedestrians," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 3, pp. 1281–1293, 3rd Quart., 2013.

[4] Y. Li, P. Zhuang, X. Niu, Y. Zhang, H. Lan, and N. El-Sheimy, "Real-time indoor navigation using smartphone sensors," in *Proc. IEEE Int. Conf. Indoor Positioning Indoor Navigat. (IPIN)*, Oct. 2015, pp. 1–10.

[5] A. Perttula, H. Leppäkoski, M. Kirkko-Jaakkola, P. Davidson, J. Collin, and J. Takala, "Distributed indoor positioning system with inertial measurements and map matching," *IEEE Trans. Instrum. Meas.*, vol. 63, no. 11, pp. 2682–2695, Nov. 2014.

[6] B. Zhou, Q. Li, Q. Mao, W. Tu, and X. Zhang, "Activity sequence-based indoor pedestrian localization using smartphones," *IEEE Trans. Human-Mach. Syst.*, vol. 45, no. 5, pp. 562–574, Oct. 2015.

[7] M. Raspopoulos, "Multidevice map-constrained fingerprint-based indoor positioning using 3-D ray tracing," *IEEE Trans. Instrum. Meas.*, vol. 67, no. 2, pp. 466–476, Feb. 2018.

[8] P. Kriz, F. Maly, and T. Kozel, "Improving indoor localization using bluetooth low energy beacons," *Mobile Inf. Syst.*, vol. 2016, Apr. 2016, Art. no. 2083094.

[9] I. Tien, S. D. Glaser, R. Bajcsy, D. S. Goodin, and M. J. Aminoff, "Results of using a wireless inertial measuring system to quantify gait motions in control subjects," *IEEE Trans. Inf. Technol. Biomed.*, vol. 14, no. 4, pp. 904–915, Jul. 2010.

[10] J. Jahn, U. Batzer, J. Seitz, L. Patino-Studencka, and J. G. Boronat, "Comparison and evaluation of acceleration based step length estimators for handheld devices," in *Proc. IEEE Int. Conf. Indoor Positioning Indoor Navigat. (IPIN)*, Sep. 2010, pp. 1–6.

[11] D. Alvarez, R. C. González, A. López, and J. C. Alvarez, "Comparison of step length estimators from weareable accelerometer devices," in *Proc. IEEE. Conf. Eng. Med. Biol. Soc.*, Aug. 2006, pp. 5964–5967.

[12] S. Y. Cho and C. G. Park, "MEMS based pedestrian navigation system," *J. Navigat.*, vol. 59, no. 1, pp. 135–153, Jan. 2006.

[13] R. Chen, L. Pei, and Y. Chen, "A smart phone based PDR solution for indoor navigation," in *Proc. 24th Int. Tech. Meeting Satell. Division Inst. Navigat. (ION GNSS+)*, Sep. 2011, pp. 1404–1408.

[14] V. Renaudin, M. Susi, and G. Lachapelle, "Step length estimation using handheld inertial sensors," *Sensors*, vol. 12, no. 7, pp. 8507–8525, 2012.

[15] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury, "No need to war-drive: Unsupervised indoor localization," in *Proc. 10th Int. Conf. Mobile Syst., Appl., Services (MobiSys)*, Jun. 2012, pp. 197–210.

[16] J. Shang, F. Gu, X. Hu, and A. Kealy, "APFiLoc: An infrastructure-free indoor localization method fusing smartphone inertial sensors, landmarks and map information," *Sensors*, vol. 15, no. 10, pp. 27251–27272, 2015.

[17] H. Weinberg, "Using the ADXL202 in pedometer and personal navigation applications," Analog Devices, Norwood, MA, USA, Appl. Note AN-602, 2002, pp. 1–6, vol. 2, no. 2. [Online]. Available: http://www.bdtic.com/DownLoad/ADI/AN-602.pdf

[18] J. W. Kim, H. J. Jang, D.-H. Hwang, and C. Park, "A step, stride and heading determination for the pedestrian navigation system," *Positioning*, vol. 3, nos. 1–2, pp. 273–279, 2004.

[19] F. Li, C. Zhao, G. Ding, J. Gong, C. Liu, and F. Zhao, "A reliable and accurate indoor localization method using phone inertial sensors," in *Proc. ACM Conf. Ubiquitous Comput.*, Sep. 2012, pp. 421–430.

[20] J. Qian, L. Pei, J. Ma, R. Ying, and P. Liu, "Vector graph assisted pedestrian dead reckoning using an unconstrained smartphone," *Sensors*, vol. 15, no. 3, pp. 5032–5057, 2015.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2012, pp. 1097–1105.

[22] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2014, pp. 3104–3112.

[23] R. Socher, E. H. Huang, J. Pennin, C. D. Manning, and A. Y. Ng, "Dynamic pooling and unfolding recursive autoencoders for paraphrase detection," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2011, pp. 801–809.

[24] C. A. Ronao and S.-B. Cho, "Deep convolutional neural networks for human activity recognition with smartphone sensors," in *Proc. Int. Conf. Neural Inf. Process.*, Nov. 2015, pp. 46–53.

[25] X. Wang, L. Gao, S. Mao, and S. Pandey, "CSI-based fingerprinting for indoor localization: A deep learning approach," *IEEE Trans. Veh. Technol.*, vol. 66, no. 1, pp. 763–776, Jan. 2017.

[26] M. Edel and E. Köppe, "An advanced method for pedestrian dead reckoning using BLSTM-RNNs," in *Proc. Int. Conf. Indoor Positioning Indoor Navigat. (IPIN)*, Oct. 2015, pp. 1–6.

[27] H.-C. Shin, M. R. Orton, D. J. Collins, S. J. Doran, and M. O. Leach, "Stacked autoencoders for unsupervised feature learning and multiple organ detection in a pilot study using 4D patient data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1930–1943, Aug. 2013.

[28] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, no. 12, pp. 3371–3408, Dec. 2010.

[29] N. Roy, H. Wang, and R. R. Choudhury, "I am a smartphone and i can tell my user's walking direction," in *Proc. ACM 12th Annu. Int. Conf. Mobile Syst., Appl., Services*, 2014, pp. 329–342.

[30] Y.-C. Lai, C.-C. Chang, C.-M. Tsai, S.-C. Huang, and K.-W. Chiang, "A knowledge-based step length estimation method based on fuzzy logic and multi-sensor fusion algorithms for a pedestrian dead reckoning system," *ISPRS Int. J. Geo-Inf.*, vol. 5, no. 5, p. 70, 2016.

[31] J. G. Park, A. Patel, D. Curtis, S. Teller, and J. Ledlie, "Online pose classification and walking speed estimation using handheld devices," in *Proc. ACM Conf. Ubiquitous Comput.*, 2012, pp. 113–122.

[32] J.-S. Hu, K.-C. Sun, and C.-Y. Cheng, "A kinematic human-walking model for the normal-gait-speed estimation using tri-axial acceleration signals at waist location," *IEEE Trans. Biomed. Eng.*, vol. 60, no. 8, pp. 2271–2279, Aug. 2013.

[33] Q. Tian, Z. Salcic, K. Wang, and Y. Pan, "A multi-mode dead reckoning system for pedestrian tracking using smartphones," *IEEE Sensors J.*, vol. 16, no. 7, pp. 2079–2093, Apr. 2016.

[34] A. Brajdic and R. Harle, "Walk detection and step counting on unconstrained smartphones," in *Proc. ACM Int. Conf. Pervasive Ubiquitous Comput. (UbiComp)*, Sep. 2013, pp. 225–234.

[35] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[36] J. Gehring, Y. Miao, F. Metze, and A. Waibel, "Extracting deep bottleneck features using stacked auto-encoders," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 3377–3381.

[37] X.-L. Zhang and J. Wu, "Deep belief networks based voice activity detection," *IEEE Trans. Audio, Speech Language Process.*, vol. 21, no. 4, pp. 697–710, Apr. 2013.

[38] T. N. Sainath *et al.*, "Deep convolutional neural networks for large-scale speech tasks," *Neural Netw.*, vol. 64, pp. 39–48, Apr. 2015.

[39] F. J. Ordóñez and D. Roggen, "Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition," *Sensors*, vol. 16, no. 1, p. 115, 2016.

[40] F. Gu, K. Khoshelham, J. Shang, F. Yu, and Z. Wei, "Robust and accurate smartphone-based step counting for indoor localization," *IEEE Sensors J.*, vol. 17, no. 11, pp. 3453–3460, Jun. 2017.

[41] S. Kullback and R. A. Leibler, "On information and sufficiency," *Ann. Math. Statist.*, vol. 22, no. 1, pp. 79–86, 1951.

[42] A. Y. Ng, J. Ngiam, C. Y. Foo, Y. Mai, and C. Suen. (Nov. 2017). *Sparse Autoencoder/Preprocessing: PCA and Whitening*. [Online]. Available: http://ufldl.stanford.edu/wiki/index.php/UFLDL_Tutorial

Fuqiang Gu (S'16), photograph and biography not available at the time of publication.

Kourosh Khoshelham, photograph and biography not available at the time of publication.

Chunyang Yu, photograph and biography not available at the time of publication.

Jianga Shang (M'12), photograph and biography not available at the time of publication.