LSTM Recurrent Network for Step Counting based on WeAllWork Dataset

Ziyi Chen
Computer Science
Unversity of California Santa Cruz
zchen139@ucsc.edu

Abstract

Smartphone offers various sensors including accelerometers, gyroscope, the magnetometer that can be used for pedometer and environment-related events. This paper train an LSTM recurrent network for counting the number of steps taken by blind/sighted users, based on WeAllWork Dataset. The model is built separately for blind volunteers using long canes and guided dog as well as sighted volunteer.

1 Introduction

Originally used by sports and physical activity tracking, pedometers are now becoming popular as daily exercise counter and motivator. It counts each step a person takes by detecting the motion of the person's hands or hips. Step counters can give encouragement to compete with oneself in getting fit and losing weight. Step counters are being integrated into an increasing number of portable consumer electronic devices such as music players, smartphones, and mobile phones.

With the increasing ubiquity of smartphones, users are now carrying around a plenty of sensors like accelerometers, gyroscope, magnetometer with them wherever they go. There are various of step counting apps in smartphones. Step counters can also be used for estimating the distance and the position in indoor pedestrian navigation systems, which is especially helpful not only for blind people, but also for sighted people who need directional information in unfamiliar places.

This paper proposes an LSTM model trained by indoor walking sensor data of iPhone to predict left/right steps and calculate the count of steps. Since Blind volunteers using long canes and guided dogs as well as sighted volunteers have different features of gaits, we separately build the model and calculate the error rate of three metrics.

2 Background and Related Work

2.1 Step Counting

Automatic step counting has received substantial attention from both research and commercial. There is a wealth of studies on the use of inertial sensors for detecting and characterising walk-related activities. Pedometer is usually portable and electronic or electromechanical. When walking, the center of gravity should be slightly moved up and down, with the most obvious upward and downward displacement of the waist. Therefore, it is most appropriate for the pedometer to be hung on the belt. It can be embedded in shoes, in a smartwatch, in a smartphone, and attached to ones ankles or hung on the belt.

A variety of algorithms have been proposed for stride event detection from inertial sensor time series. Some of these algorithms operate on the accelerometer data, while others use data from the gyros. For example, (WPD[7]) The Window Peak Detection algorithm runs a moving average window on the smoothed accelerometer magnitude to find peaks associated with a heel strike; (AMPD [44]) Automatic multiscale-based peak detection generic dectect peak detection in a signal; UPTIME [1]: UPTIME (Ubiquitous Pedestrian Tracking usIng Mobile phonEs) uses the de-trended magnitude of acceleration in a finite state machine (FSM), with six states to identify the peaks associated with heel strikes; HMM-acc [7][32] trains a Hidden Markov Model (HMM) to discern the different phases during a gait period; ZC-gyro [26] searches for zero crossings (ZC) within a moving window of the data from the gyro aligned with medial-lateral axis. All of the above algorithms have some parameters that must be learned from training data.

Recent approaches to step detection include the use of recurrent neural networks. Researchs use RNN for evaluating and optimising accelerometer-

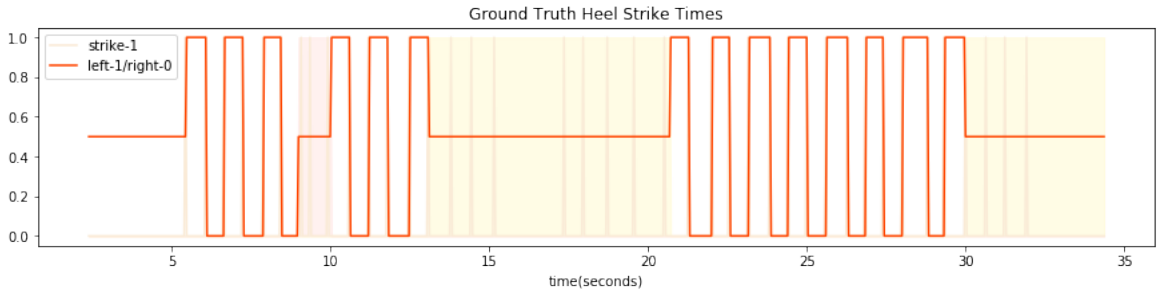

Figure 1: Example heel strike times of T1_ID1_WC.xml from 2.4s to 34.4s. First step is left feet at 5.4761s. A feature motion (the participant walked into the wall, stopped, and moved to the left) happened from 9.041862s to 10.041862s. Two turn motion, Turn east to south from 13.1752s to 20.7419 and Turn south to east from 30.0419 happened. All data before first step as well as feature and turn data are removed.

based gesture recognition, self-calibration, pedestrian dead reckoning. Whereas the vast majority of step counting algorithms have been developed for able-bodied ambulators, some researchs have addressed the performances of these algorithms with sensors carried by people with some level of mobility impairment.

2.2 WeAllWalk Dataset

WeAllWalk data set contains inertial sensor time series collected from ten blind walkers using a long cane or a guide dog and five sighted walkers. The participants walked through fairly long and complex indoor routes that included obstacles to be avoided and doors to be opened. Inertial data was recorded by iPhone 6s carried by participants in their pockets. Ground truth heel strike times were measured by two small inertial sensor units attached to the participants shoes. The data set contains a mobility impairment that may result in a gait pattern that is quite different than for sighted walkers. The data is subdivided into straight paths and turns and carefully annotated, with special events (or features, such as bumping into an obstacle) individually identified and marked.

3 Implementation

We proposes an LSTM model trained by indoor walking sensor data of iPhone to predict left/right steps and calculate the count of steps. Since Blind volunteers using long canes and guided dogs as well as sighted volunteers have different features of gaits, we separately build the model and calculate the error rate of three metrics.

3.1 Data Preprocess

WeAllWalk dataset contains detailed maps of indoor paths, scripts to facilitate visualization of all the information in this data set and recorded sensor data for all users and all paths.

We use two kinds of file from the dataset. First is the CSV file of iPhone sensor data as input. It contains records of all iPhone sensors. There are total 39 columns of sensor data and some of them are more useful such as rotation rate and user acceleration.

Second is the XML files that contain annotated ground truth data for all the paths walked by all the participants as labels. Each file contains start and end times for each segment the user walked through and feature information like the left or right foot as well as direction. We only train and test the step detection algorithms on data acquired while traversing straight segments in the paths, where gait is assumed to be regular, and avoided segments labeled as features. This is because the notion of step is not well defined in such situations. In general, we argue that step counting only really matters during regular ambulation, as an indirect way to measure distances traversed.

Since it only contains heel strike times, we cannot use it as labels directly. It is unreasonable to predict next strikes times by previous records. If we regard heel strike times as 1 and other times as 0, then the labels are unbalanced. So we apply the same sampling rate as CSV files and transform the XML file to 0-1 in which left steps as 1 and right steps as 0.

As shown in picture 1, the light color line of strike time is the origin data from XML file. It is transform to red line which regard left as 1 and

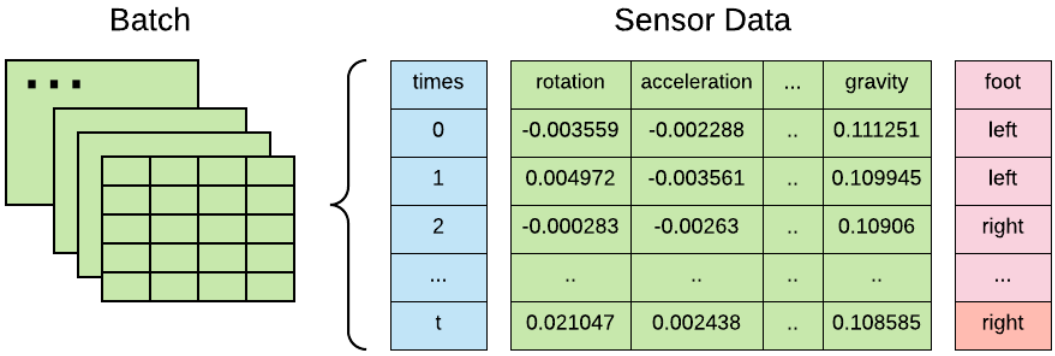

Figure 2: Input (green) and output (pink) example for LSTM. Use previous timesteps(t) records of sensor data (green blocks) to predict the last foot step (last pink foot). The fisrt timesteps(eg. t=50) outputs of each segement are determined by intermediate outputs (light pink) since they don't have enough previous record. Shuffle and divide train data into the small batchs as inputs.

right as 0. We also remove the turn data(light yellow background part) and feature data(light red backgound part).

3.2 LSTM Model

Long short-term memory (LSTM) network is a recurrent neural network. It can be used as blocks of recurrent neural network(RNN) network. There are different types of LSTMs, which differ in the components or connections that they have. An LSTM layer consists of a number of recurrently connected memory blocks. Each block contains one or more recurrently connected memory cells and three multiplicative units, the input, output, and forget gates, which control the information flow inside the memory block. The surrounding network can only interact with the memory cells via the gates. In other words, these gates and the memory cell allow an LSTM unit to adaptively forget, memorize and expose the memory content. Due to above characteristic, LSTM is suitable to predict time series such as step counting.

We use TensorFlow to implement LSTM network. TensorFlow is an open-source software library that can be used for machine learning applications such as neural networks. It supports both CPU and GPU that can be imported as a python library. TensorFlow uses a dataflow graph to represent computation in terms of the dependencies between individual operations. We first define the dataflow graph and then create a session to run the graph. The Saver class of TensorFlow can easily add ops to save and restore variables to and from checkpoints, which map variable names to tensor values.

A two-layer RNN network with basic LSTM cell and dropout wrapper is built for training with squared difference loss function. The input is a list of metrics with the row of timesteps and col of sensor data. For Example, we want to use previous timesteps ($= 50$) record of sensor data to predict the result, each sensor data contains input number ($= 6$) values of rotation rate and user acceleration. So the matrix size is timesteps multiply input number(50×6). All such metrics form the input list and is transformed to tensor. Since the dataset is big and there are more than one hundred thousand elements in input list which would make the training process very slow, we shuffle and divide train data into the small batch (batch size = 256) and feed the batch to model.

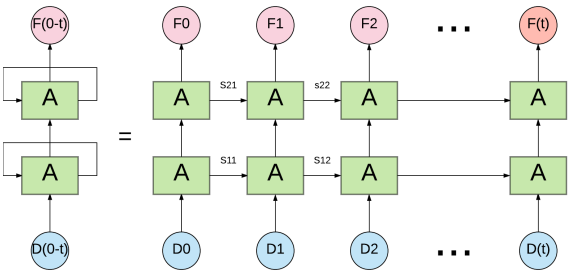

Figure 3: Two layer LSTM recurrent network. Each green block is a LSTM Block with number of hidden nodes. Blue circle D represent for sensor data, s is the lstm state, pink circle F is the output feet, which is a float value around 0 to 1.

There are two ways of output. One is to use previous timesteps data to predict only the last step result, which is a float value around 0 to 1. We use 0.5 as the border to classify left and right steps.

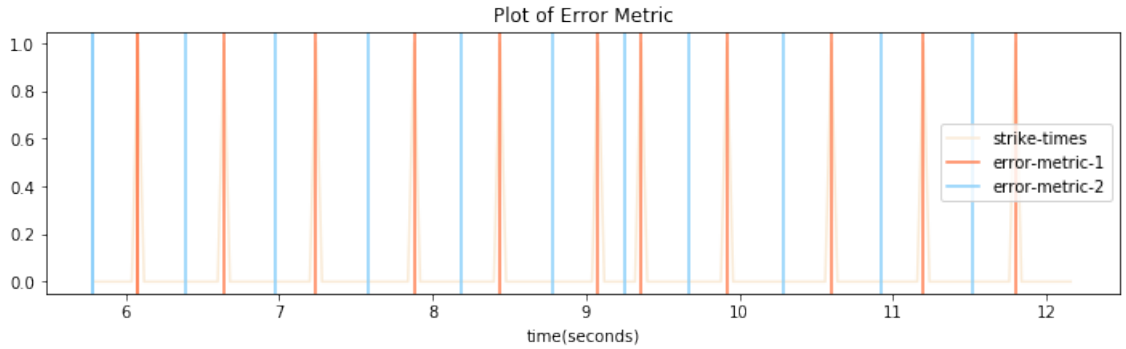

Figure 4: Example of three error metrics. Assume above plot is a entire segement. Error metric 1 (read line) split intervals exactly at each heel strike times. Error metric 2 (blue line) split intervals exactly at middle of every two continues heel strike times. Error metric 3 calculate undercount and overcount for the entire segment.

And the heel strike times is when 0 change to 1 or 1 change to 0. The other is to use previous timesteps data to predict corresponding timesteps step results. The first one cannot predict the first timesteps result, but it is more precise since all result is predicted by previous timesteps record. Most outputs that have enough previous records use first way to calculate.

3.3 Error Metrics

The round of result is a list of left/right step corresponding to origin step. Then calculate the accuracy rate of the result, which is the proportion of correct numbers (left is predicted to be left and right is predicted to be right) and total numbers.

The quality of the step counting model was measured using three different metrics. The first metric looks at the number of steps detected within each time interval $[T_i, T_{i+1}]$ separating two consecutive ground-truth heel strikes. Ideally, exactly one step should be detected within one interval $[T_i, T_{i+1}]$. If no steps are detected in $[T_i, T_{i+1}]$, then an undercount event is recorded. If steps are detected within that interval, the overcount events are recorded. The cumulative number of undercount and overcount events are computed and normalized (divided) by the number of ground-truth steps.

The second metric is similar to the first, but it looks at the number of steps detected within each time interval $[\frac{T_{i-1}+T_i}{2}, \frac{T_i+T_{i+1}}{2}]$ separating two consecutive ground-truth heel strikes. Ideally, exactly one step should be detected within one interval $[\frac{T_{i-1}+T_i}{2}, \frac{T_i+T_{i+1}}{2}]$. The undercount and overcount is calculated same as first metric. The metric can decrease error when one step is detected sightly earlier ($t_i < T_i$) and the next step is detected sighted later ($t_{i+1} < T_{i+1}$). For this case, interval $[T_i, T_{i+1}]$ has one undercount, and interval $[T_{i-1}, T_i]$ and $[T_{i+1}, T_{i+2}]$ have one overcount.

The third metric simply computes the difference between the number of detected steps within each segment and the number of ground-truth steps within the same segment. The difference between the two is recorded as an undercount value if negative, as an overcount if positive. Undercount and overcount values are then summed together over all segments, and normalized by the total number of ground-truth steps.

Note that the normalized undercount and overcount values obtained by the error metric 1 and 2 metric is always larger than or equal to the corresponding values computed by the metric 3, as missed counts or double counts between a time interval $[T_i, T_{i+1}]$ may even out over multiple steps. While the more lenient metric 5 may be appropriate when measuring step counts over long hauls, the more conservative metric 1 may be useful when fine-grained tracking is desired. When training and benchmarking the algorithms, the sum of overcount and undercount rates is used for both error metrics.

4 Experience

The model is trained with different input numbers, timesteps, output number, hidden layer number, optimizer, learning rate and training steps. We first try different optimizers with various learning rate and find that AdamOptimizer with learning rate around 0.001 can make it convergence, and then

fix the two parameters.

4.1 Blind People with Long Canes

4.2 Blind People with Guide Dogs

4.3 Sighted people

5 Conclusion and Future Work

References

Flores, German H., and Roberto Manduchi. (2016) *WeAllWalk: An Annotated Data Set of Inertial Sensor Time Series from Blind Walkers.*. Proceedings of the 18th International ACM SIGACCESS Conference on Computers and Accessibility. ACM, 2016.

Brajdic, Agata, and Robert Harle. (2013) *Walk detection and step counting on unconstrained smartphones.*. Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing. ACM, 2013.

Edel, Marcus, and Enrico Kppe. (2015) *An advanced method for pedestrian dead reckoning using BLSTM-RNNs.*. Indoor Positioning and Indoor Navigation (IPIN), 2015 International Conference on. IEEE, 2015.

Naqvib, Najme Zehra, Ashwani Kumar, Aanchal Chauhan, and Kritka Sahni. (2012) *Step counting using smartphone-based accelerometer.*. International Journal on Computer Science and Engineering 4, no. 5 (2012): 675.

Alzantot, Moustafa, and Moustafa Youssef. (2012) *UPTIME: Ubiquitous pedestrian tracking using mobile phones.*. Wireless Communications and Networking Conference (WCNC), 2012 IEEE.

Tomlein, Michal, Pavol Bielik, Peter Krtky, Stefan Mitrk, Michal Barla, and Mria Bielikov. (2012) *Advanced Pedometer for Smartphone-based Activity Tracking.*. In HEALTHINF, pp. 401-404. 2012.

Oshin, Thomas Olutoyin, and Stefan Poslad. (2013) *ERSP: An energy-efficient real-time smartphone pedometer.*. In Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on, pp. 2067-2072. IEEE, 2013.

Case, Meredith A., Holland A. Burwick, Kevin G. Volpp, and Mitesh S. Patel. (2015) *Accuracy of smartphone applications and wearable devices for tracking physical activity data.*. Jama 313, no. 6 (2015): 625-626.

Haegele, Justin A., and David L. Porretta. (2015) *Validation of a Talking Pedometer for Adolescents with Visual Impairments in Free-Living Conditions.*. Journal of Visual Impairment & Blindness 109, no. 3 (2015): 219-223.

Holbrook, Elizabeth Ackley, Sandy L. Stevens, Minsoo Kang, and Don W. Morgan. (2011) *Validation of a talking pedometer for adults with visual impairment.*. Medicine & Science in Sports & Exercise 43, no. 6 (2011): 1094-1099.

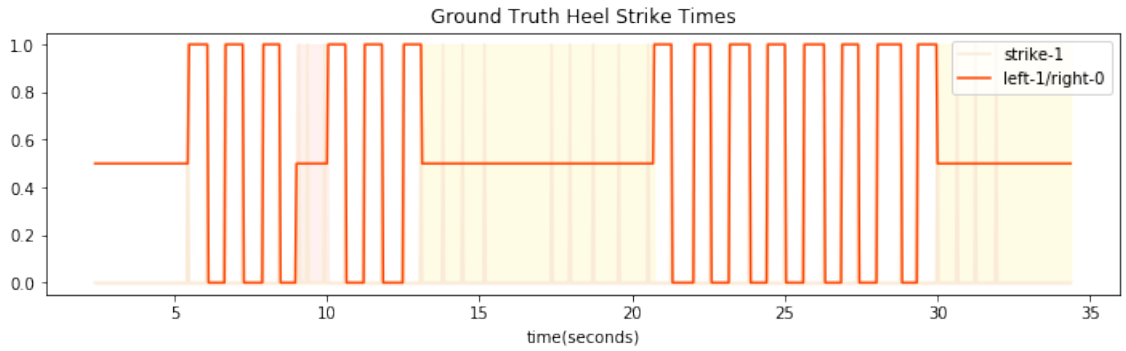

Figure 5: this is a figure demo