# 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

Step Counting is the automatic determination of the strike heels times in a period. Step counters are becoming popular as a part of indoor navigation systems, as well as an exercise measurer. With the increasing ubiquity of smartphones, users are now carrying around a plenty of sensors like accelerometers, gyroscope, magnetometer with them wherever they go.

This paper uses the indoor walking sensor data of iPhone to train an LSTM model to predict left/right steps and calculate the count of steps. The model can also be used for estimating the distance and the position in pedestrian navigation systems indoor, which is especially helpful not only for blind people, but also for sighted people who need directional information in unfamiliar places.

Since Blind volunteers using long canes and guided dogs as well as sighted volunteers have different features of motion, we separately build the model and calculate the error rate of two metrics.

## 2 Background and Related Work

### 2.1 Step Counting Algorithms

### 2.2 WeAllWalk Dataset

## 3 Method

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 is suitable to predict time series such as step counting.

### 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. 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.

At beginning of each path, we don't the step of the participants, so these data are removed. Since we are interested in straight segments instead of turn segments, we also remove all turn data according to the change of direction recorded in XML files.

### 3.2 LSTM Model

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($50 \times 6$). 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 divide input data into the small batch (batch size = 256) and feed the batch to model.

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. 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.

### 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 two different metrics. The first metric looks at the number of steps detected within each time interval separating two consecutive ground-truth heel strikes. Ideally, exactly one step should be detected within one interval. If no steps are detected in, 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 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.

## 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..* WeAllWalk: An Annotated Data Set of Inertial Sensor Time Series from Blind Walkers. ACM, 2016.