

UNIVERSITY OF CALIFORNIA
SANTA CRUZ

AN LSTM RECURRENT NETWORK FOR STEP COUNTING

A report submitted in partial satisfaction of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

Ziyi Chen

December 2017

The Report of Ziyi Chen
is approved:

Professor Roberto Manduchi, Chair

Professor Luca de Alfaro, Reader

An LSTM Recurrent Network for Step Counting

Ziyi Chen

Computer Science

University of California Santa Cruz

zchen139@ucsc.edu

Abstract

Smartphones with sensors such as accelerometer and gyroscope can be used as pedometers and navigators. In this paper, we propose to use an LSTM recurrent network for counting the number of steps taken by both blind and sighted users, based on an annotated smartphone sensor dataset, WeAllWork. The models were trained separately for sighted people, blind people with a long cane or a guide dog for Leave-One-Out training modality. It achieved 5% overcount and undercount rate.

1 Introduction

With the increasing ubiquity of smartphones, users are now carrying plenty of sensors with them such as accelerometer, gyroscope, magnetometer, wherever they go. Step counters are being integrated into an increasing number of portable consumer electronic devices such as music players, smartphones, and mobile phones (contributors(2017a)). There are various of step counting apps on 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.

In this paper, we propose an LSTM model trained with indoor walking sensor data from iPhones, and annotated data generated by a shoes-mounted sensor (MetaWear-CPRO) from the WeAllWalk dataset, to predict the number of steps. The MetaWear-CPRO units are attached to the user's shoes so they can provide precise heel strike times. In the dataset, blind volunteers using a long cane or a guide dog and sighted volunteers have different gait patterns, so we built the mod-

els separately for each group of walkers and measured the performances based on three error metrics. We also tested the accuracy of those models by using leave-one-out cross-validation. Three error metrics splitting intervals differently were used for measuring the overcount and undercount rates of the system. We tried different parameters for the LSTM models such as the number of training steps and window size to find the best settings. The model achieved 1% overcount and undercount rate on a mixed dataset and 5% under the Stratified Leave-One-Out training modality.

2 Background and Related Work

2.1 Step Counting

Automatic step counting has received substantial attention in both research and commercial domains. There is a wealth of studies on the use of inertial sensors for detecting and characterizing walk-related activities. Pedometers are usually portable and electronic or electromechanical (contributors(2017a)). They can be embedded in shoes, in smartwatches, in smartphones, and attached to users' ankles or hung on the belt.

(Brajdic and Harle(2013)) evaluated common walk detection (WD) and step counting (SC) algorithms applied to smartphone sensor data. The results favor the use of standard deviation thresholding (WD) and windowed peak detection (SC) with error rates of less than 3%. (Tomlein et al.(2012)Tomlein, Bielik, Krátky, Mitrík, Barla, and Bieliková) introduced step detection and intelligent detection of cheating based on smartphone sensors. (Naqvib et al.(2012)Naqvib, Kumar, Chauhan, and Sahni) presented a method for counting the number of steps taken by a user using the smartphone-based accelerometer while walking at any variable speed.

With the rapid development of deep learning, this advanced technology is used in vari-

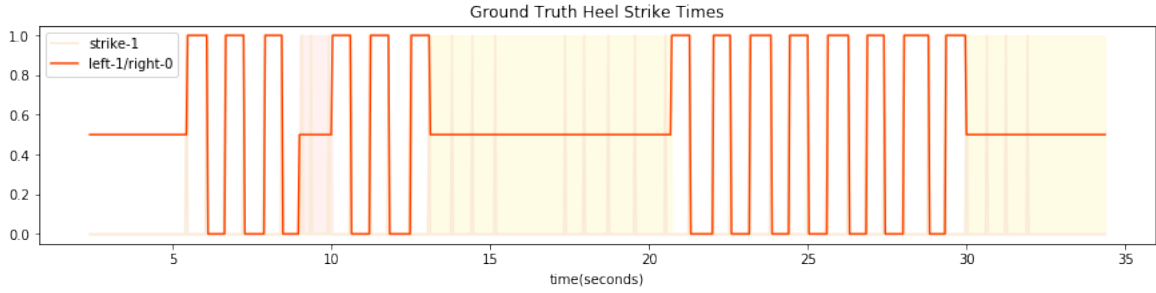


Figure 1: An example of heel strike times in T1_ID1_WC.xml from 2.4s to 34.4s. The first step at 5.4761s is a left step. A feature motion (the participant walked into the wall, stopped, and moved to the left) happened during 9.041862s to 10.041862s. Then one turn from east to south and another turn from south to east occurred during 13.1752s to 20.7419s, and 30.0419s to 39.1419s respectively. Feature data, turn data, and all data recorded before the first step are removed.

ous fields including step counting. Since sensors provide time series data, researchers have tried to use RNN for counting the number of steps. (Edel and Köppe(2015)) uses Bidirectional Long Short-Term Memory Recurrent Neural Networks (BLSTM-RNNs) for step detection, step length approximation as well as heading estimation.

In addition to sighted people, some researchers also pay attention to people with visual impairment. (Haegele and Porretta(2015)) validated Centrios talking pedometer for adolescents with different level of visual impairment under daily-living condition, while (Holbrook et al.(2011))Holbrook, Stevens, Kang, and Morgan) validated the same talking pedometer for adults.

2.2 The WeAllWalk Dataset

The WeAllWalk dataset (Flores and Manduchi(2016)) contains sensor data gathered from ten blind participants with a long cane or a guide dog and five sighted participants. The smartphone (iPhone 6s) carried by participants recorded the sensor data. Two small sensors attached to the participants shoes recorded heel strike times as the labels. Each segment of paths is marked as either a straight segment or a turn segment. Special circumstances on the road are also labeled as features. (Flores and Manduchi(2016)) have tested six different algorithms for step counting and turn detection.

3 Implementation

We propose an LSTM model trained with indoor walking sensor data from iPhone for step counting. Since blind participants using a long cane

or a guide dog and sighted volunteers have different gait patterns, we build the models separately and calculate the error rates by using three metrics. Also, we only consider straight segments in the paths, since the gait pattern is more likely to be regular than the gait pattern of turn segments, and remove segments marked as features such as opening a closed door and avoiding an obstacle. We try different parameters for the LSTM model and add a dropout layer to make the result more robust.

3.1 Data Preprocessing

We use two kinds of files from the dataset. The first type is CSV format iPhone sensor data. Each CSV file contains the sensor data recorded with a path and a person. There are 39 columns of sensor data and some of them are more useful than others. So we select 6 columns (rotationRateX, rotationRateY, rotationRateZ, userAccelerationX, userAccelerationY, userAccelerationZ) from the sensor data to building our model.

The second type is XML file containing annotated ground truth data for paths walked by each participant. Each file contains tags of start time, end time and direction for each segment the participant walked through. For each segment, the time and foot of each step is recorded, and special situation such as moving to the wall is also marked as feature with start time, end time and detailed description. We only train and test the step counting system with the segments recorded when participants traverse straight in each path, where gait patterns are assumed to be regular. We also remove time slots that have been marked as feature.

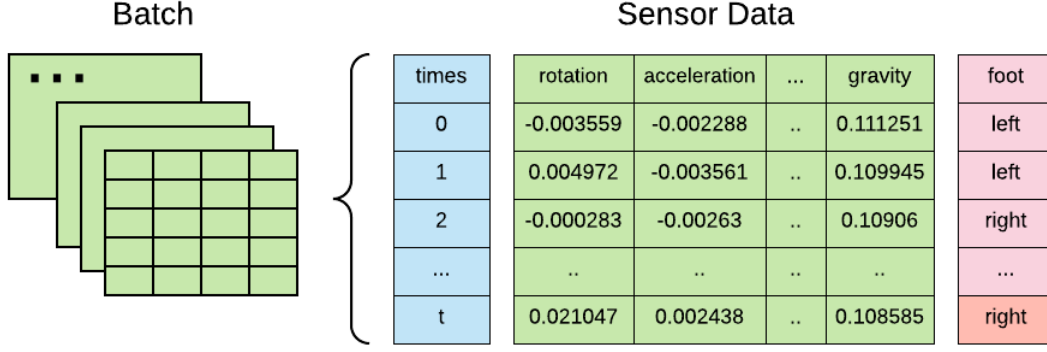


Figure 2: Input (green) and output (pink) samples for our model. The model uses the latest t records of the sensor data (green blocks) to predict the current status (the last pink block). The first $t-1$ predictions of each segment are from the intermediate outputs (light pink) of the recurrent neural network, since they do not have t records for prediction as the rest do.

Since the XML files only contain heel strike times and it's very difficult to predict heel strike times with the sensor data, we can not use them as the labels directly. If we regard the labels at heel strike times as 1s and the labels at other points in time as 0s, then the labels are too unbalanced to train a model to predict the labels at each point in time. So we transform the XML files to binary signals in which left steps toggle the signal from 0 to 1 and right steps toggle the signal from 1 to 0 as shown in Figure 1.

Finally, we windowed the data (window size is equal to the number of timesteps) for corresponding sensor data and ground truth signals to generate the formatted training data and labels.

3.2 The LSTM Model

Long short-term memory (LSTM) network is a time recursive neural network, suitable for processing and predicting important properties of time series data. LSTM has many applications in the science and technology domains. LSTM-based systems can learn tasks such as translating natural languages, image analysis, speech recognition, controlling chat robots and so on.

LSTM network is a special kind of recurrent neural network (RNN). The main difference between LSTM and RNN lies in that LSTM adds a "processor" to judge whether the information is useful or not in the algorithm. The structure of this processor is called cell.

Three non-linear units (a input gate, a forget gate and a output gate) are placed in the cell. When a message enters the LSTM network, the system would determine whether the message is

useful or not. Only information that complies with algorithmic requirements will remain, and misleading information will be forgotten through the forget gate.

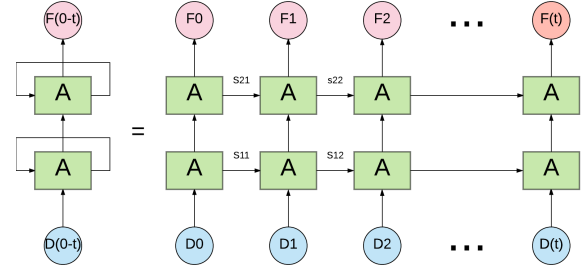


Figure 3: A two-layer LSTM recurrent network. Each green block is an LSTM cell. Blue circle $D(t)$ represents the sensor data recorded at time t while s is the inner LSTM state, and pink circle F is the output. Each LSTM cell receives the input of current sensor data and state from previous cell (the initial state is 0), and outputs a state and an output.

We use TensorFlow to implement the LSTM network. TensorFlow is an open-source software library that can be used for machine learning applications such as neural networks (contributions(2017b)). It supports both CPU and GPU and it provides an interface in Python. TensorFlow uses a data flow graph to represent computation in terms of the dependencies between individual operations (TensorFlow(2017)). We first define the data flow graph and then create a session to run the graph. The saver class can save and restore the values of graph variable to and from checkpoints by mapping variable names to tensor values.

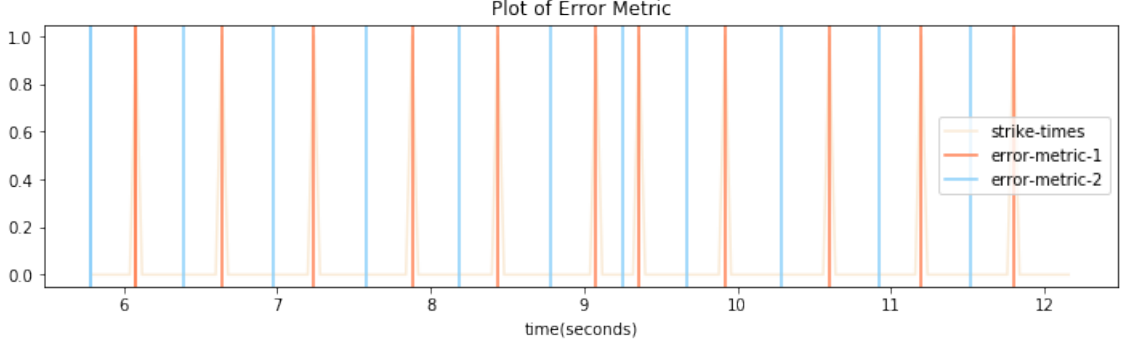


Figure 4: An example of three error metrics. Assume above plot is an entire segment. Error metric 1 (red line) divides segments into intervals exactly at each heel strike time. Error metric 2 (blue line) divides intervals at the center point of every pair of adjacent heel strike times. Both error metric 1 and 2 compute undercount and overcount rates for each interval independently while error metric 3 calculates undercount and overcount rates within the entire segment.

As shown in Figure 3, a two-layer RNN network with 2 basic LSTM cells is built for training with mean squared error loss function. As shown in Figure 2, the input batch is a list of matrices. Each row of the matrices is a list of sensor data at a point in time. The length of the list is called the number of timesteps. For example, if we want to use the sensor data recorded at the latest 50 points in time to predict the current label, each sensor data contains input size (6 in our model) values of rotation rate and user acceleration in 3-dimension. So the matrix size is the number of timesteps multiply input size which is 50×6 . All such matrices form the input list and are transformed to tensor as the training data. Since the dataset is big and there are more than one hundred thousand elements in the input list which would make the training process very slow, we shuffle and divide training data into small batches (batch size = 256) and feed the batches to the optimizer one step by one step.

There are two ways to get the outputs. The first one is to use the latest 50 inputs to predict the current output. The second way is to use the intermediate outputs of RNN to generate predictions. The first one can not predict the first 50-1 labels, but it is more precise since more information is used to produce one output. We use the first way to predict all other labels.

3.3 Error Metrics

The original result signal is a list of float numbers around 0 to 1. Then we transform the signal into a binary signal where the value is either 0 or 1. And the predicted heel strike times is the signal change

times. Now we can calculate the accuracy rate of the result, which is the proportion of correct predictions (1 is predicted as 1 or 0 is predicted as 0) in total predictions. The error rate of the step counting model was measured using three different metrics as shown in Figure 4. The error metric 1 and 3 are as same as the error metrics proposed in (Flores and Manduchi(2016)).

The first metric splits intervals at each real heel strike times, and counts the number of predicted steps within each time interval $[T_i, T_{i+1}]$. If exact one step is detected within each interval, the count of step is correct. If no step is detected during the interval, then one undercount event happens. If n steps are detected during the interval, then the measured model overcounts $n - 1$ times. The undercount and overcount rate is number of undercount and overcount event divided by the total count of ground-truth steps.

The second metric is similar to the first one, but it split intervals at middle of two continuous ground-truth heel strike times, and counts the number of predicted steps within each time 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. As shown in Figure 15 at end of the paper, the metric can decrease error rate when one step is detected slightly 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 counts number of predicted steps during each segment and compare to the number of ground-truth steps during the same seg-

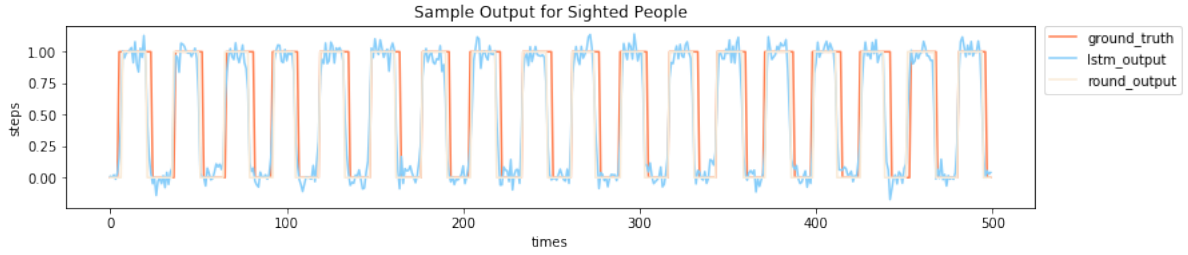


Figure 5: Sample output from mixed data for all sighted people. The origin output signal (blue line) swings near 0 or 1. The round signal (yellow line) is quite similar to the ground truth data (red line).

ment. If the ground-truth number is larger, then undercount events happen. If the predicted number is larger, then overcount events happen. The undercount and overcount rate is proportion between sum of undercount and overcount numbers over all segment and the total count of ground-truth steps. The undercount and overcount rate obtained by the error metric 3 metrics is always smaller than or equal to the corresponding values computed by the metric 1 and 2.

4 Experience

We preprocessed the WeAllWalk data to train LSTM model for the different communities using two ways of splitting train and test data. First way is simply mixed all data and split into train and test set. In this way, both train and test data contain records from all participant and all segment. The second way is leave one person out, which means test set contains and only contains all records from one participant and train set contains all other records from remain participants.

The model is trained with different sensor data, 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 0.01 can make it convergence, and then fix the two parameters.

4.1 Sighted people

There are more than 120,000 valid records of sighted people, we split it into 100,000 train data and 25,000 test data. Each input data is a 3-dimensional tensor, with a shape of batch size \times timesteps \times number of sensor data.

For mixed data, we apply 10 folds cross-validation. A good result sample is shown in Figure 5. The output signal swings near 0 or 1. So it is reasonable to round the signal (> 0.5 turn to

1, ≤ 0.5 turn to 0). The round signal is similar to the ground truth data. However, not all result is as good as this sample, some bad samples are shown in end of the paper. Some result has a certain offset between ground truth data and predicted output, some result function shakes more fiercely so that not all values are close to 0 and 1. There tend to be more strikes if the output value shakes around 0.5. So in the experience the overcount rate is always higher than the undercount rate.

Since there are 6 error values, it difficult to determine a weighted format over each error rate and tell which model is better. All ten results are shown in Figure 6. The average undercount rate of metric 3 is around 0.6%, and the average overcount rate of metric 3 is around 1.4%. The average undercount rate of metric 2 is around 0.9%, and the average overcount rate of metric 2 is around 1.7%. The value of error rate is shown in table 2. The error rate of metric 1 is much higher than error rate of metric 2 and 3, the undercount rate of metric 1 is similar to overcount rate of metric 1 (around 20%) as shown in Figure 16. The reason that caused such situation is described in Figure 15. So the metric 1 error rate of undercount and overcount is similar since when a undercount happened it is likely to cause a overcount.

For leave one person out method, the cross validation result is shown in Figure 7. The average undercount rate of metric 3 is around 2.5%. The average overcount rate of metric 3 is around 7%. In some case, metric 2 has bad performance, this is because there is an offset around half of each interval as shown in the Figure 9. In the experience with different parameters of LSTM model, some validation sets (cv3 perform best in most of experience) have overcount and undercount rate less than 0.1% of metric 3, while less than 1% for metric 1 or 2. But for other validation set, the overcount rate is extremely high likely 30% to 60%.

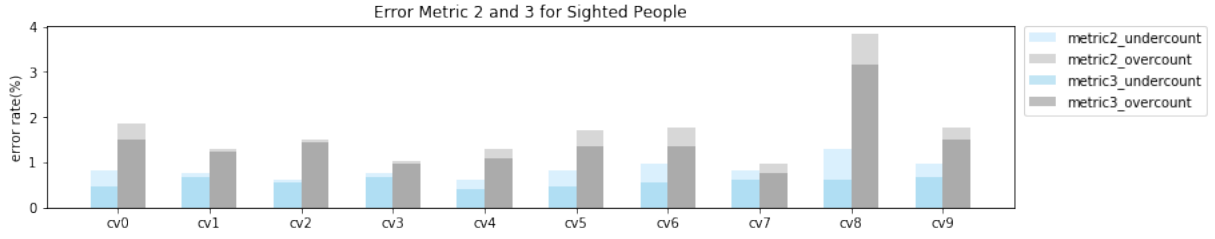


Figure 6: Error metric 2 and 3 for 10 fold cross validation of mixed data for sighted people. The error rate of metric 2 is always higher than metric 3. The undercount error (blue blocks) is less than overcount error (grey blocks). Detailed values is shown in table 2.



Figure 7: Error metric 1, 2 and 3 for leave one out of sighted people.

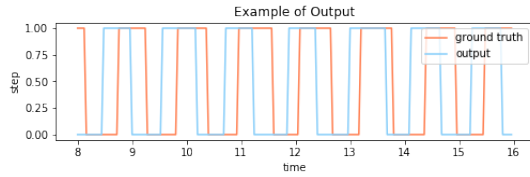


Figure 8: Sample output of offset. In this case, error metric 1 works better than error metric 2.

Each validation set has better performance with models of different parameters. It may be because different people have different gaits. So I choose the model where each validation sets have average performance.

When the train and test data both contains the sensor data from certain people, the predicted signal is quite accurate, and the error (metric 2 and 3) of step counting is small. However, for leave one out method, there would be offsets for both train and test data, and the accuracy of predicted signal is low. But the error 1 could be small if the offset is similar for each heel strikes as shown in Figure 8.

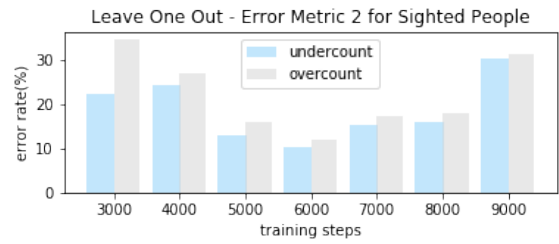


Figure 9: Error metric 1 and 3 for 10 fold cross validation of sighted people. The error rate of metric 1 is always higher than metric 3 with regard to training steps.

4.2 Blind People with a long cane

There are more than 120,000 records from blind people with a long cane, participants 1, 2, 3, 5, 6, 7, 8 have sensor data for six paths. So we use participant 8 as test data, and records from remain six people as train and validation data for 6 fold cross validation for leave one out. The valid and test result is shown in Figure 10 and 11. All error rate is higher than mix data. The validation set has good performance don't means that the test data also have good performance. The training data only contains six different people, which is not a big number. If we have annotated data from more peo-

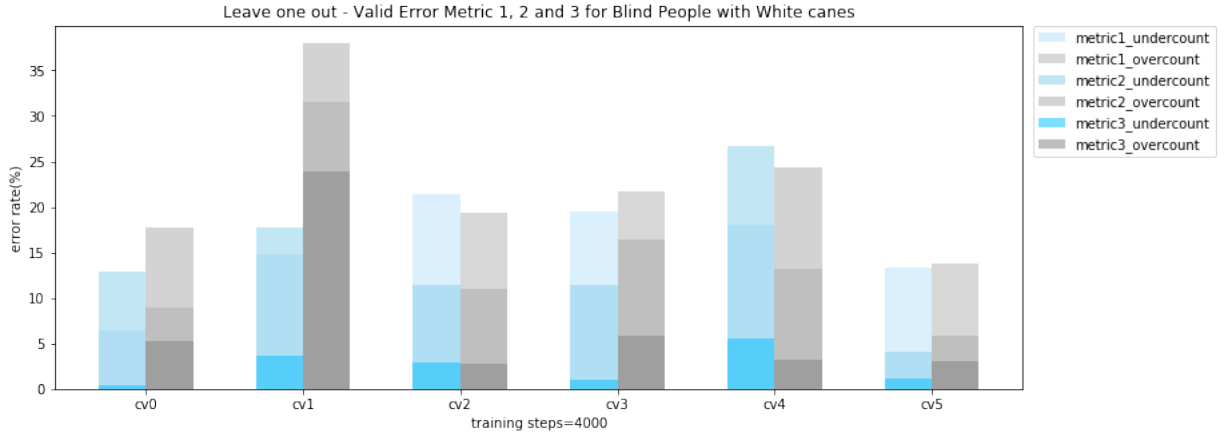


Figure 10: Leave one out - Valid Error for Blind People with a long cane

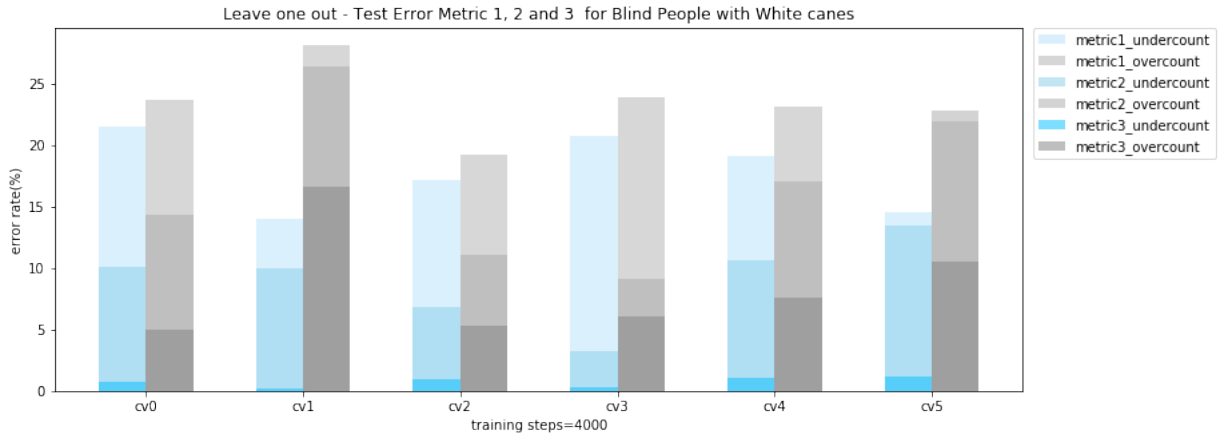


Figure 11: Leave one out - Test Error for Blind People with a long cane

ple, LSTM model may detect common features of blind people and provide a better result.

4.3 Blind walkers using a guide dog

There are only 3 people walked with the guide dog, each has 30,000 data. So we regard two people as train data, and the left one as test data. The result is bad as shown in 12. Although mixed data shows good performance around 5% for metric 2 and 2% for metric 3, it is not likely to predict the strike heel times of a person from only one different person.

5 Conclusion and Future Work

We train an LSTM model using annotated smartphone sensor data from the WeAllWalk dataset, to predict output signal. In the dataset, blind volunteers using a long cane or a guide dog, sighted volunteers have different features of gaits, so we separately build the model and calculate the error rate of three metrics. We also apply leave one person

Table 1: Error of blind people with guide dog

error(%)		valid	test
metric1	undercount	23.90	10.18
	overcount	26.14	25.75
metric2	undercount	22.56	12.27
	overcount	29.03	19.49
metric3	undercount	4.52	0.53
	overcount	10.98	17.75

out pedometers to test the accuracy of our mode using three error metrics, whihc splitting intervals differently to estimate the overcount and undercount of steps. The model achieved 1% overcount and undercount rate for mixed training data and 5% for Leave-One-Out training modality.

As shown in Figure 14 and Figure 13, simply round the origin output signal may cause the overcount or undercount. So we can use a better way to turn the signal into binary signal. In this paper, we

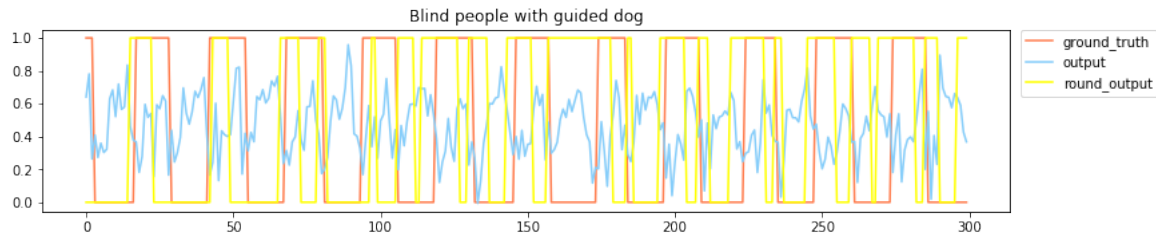


Figure 12: Sample output of blind people with guide dog.

only use the data of straight segments, removing turn segment and other features. We could consider about turn segments and feature motions. We now only use rotation rate and user acceleration as input sensor data, we can also try more sensor data in future.

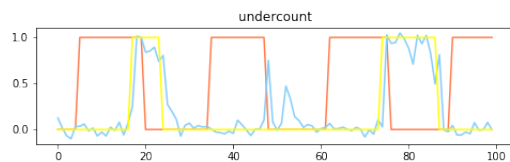


Figure 13: Sample of undercount.

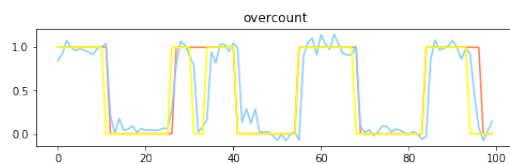


Figure 14: Sample of overcount.

References

- Dragan Ahmetovic, Cole Gleason, Chengxiong Ruan, Kris Kitani, Hironobu Takagi, and Chieko Asakawa. 2016. Navcog: a navigational cognitive assistant for the blind. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services*. ACM, pages 90–99.
- Heikki J Ailisto, Mikko Lindholm, Jani Mantjarvi, Elena Vildjiounaite, and Satu-Marja Makela. 2005. Identifying people from gait pattern with accelerometers. In *Proc. SPIE*. volume 5779, pages 7–14.
- Moustafa Alzantot and Moustafa Youssef. 2012. Up-time: Ubiquitous pedestrian tracking using mobile phones. In *Wireless Communications and Networking Conference (WCNC), 2012 IEEE*. IEEE, pages 3204–3209.
- Agata Brajdic and Robert Harle. 2013. Walk detection and step counting on unconstrained smartphones. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*. ACM, pages 225–234.
- Meredith A Case, 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(6):625–626.
- Wikipedia contributors. 2017a. Pedometer — Wikipedia, the free encyclopedia. <https://en.wikipedia.org/wiki/Pedometer>. [Online; accessed 14-December-2017].
- Wikipedia contributors. 2017b. Tensorflow — Wikipedia, the free encyclopedia. <https://en.wikipedia.org/wiki/TensorFlow>. [Online; accessed 14-December-2017].
- James Coughlan, Roberto Manduchi, and Huiying Shen. 2006. Cell phone-based wayfinding for the visually impaired. In *1st International Workshop on Mobile Vision*. pages 1–15.
- Marcus Edel and Enrico Köppe. 2015. An advanced method for pedestrian dead reckoning using blstm-rnns. In *Indoor Positioning and Indoor Navigation (IPIN), 2015 International Conference on*. IEEE, pages 1–6.
- German H Flores and Roberto Manduchi. 2016. Weall-walk: An annotated data set of inertial sensor time series from blind walkers. In *Proceedings of the 18th International ACM SIGACCESS Conference on Computers and Accessibility*. ACM, pages 141–150.
- Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 1999. Learning to forget: Continual prediction with lstm.
- Justin A Haegele 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(3):219–223.
- Ann Hallemans, Els Ortibus, Francoise Meire, and Peter Aerts. 2010. Low vision affects dynamic stability of gait. *Gait & posture* 32(4):547–551.
- Elizabeth Ackley Holbrook, 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(6):1094–1099.

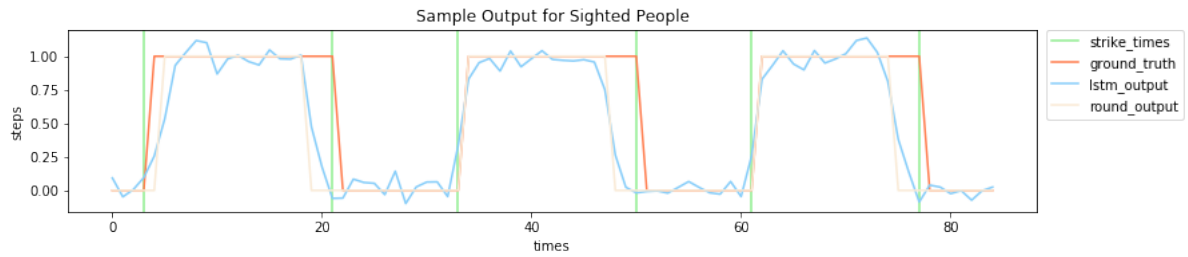


Figure 15: Bad sample for error metric 1. The green line splits the intervals of error metric 1. There are random offset between actual strikes times and predicted strike times. In this case, the first, third and fifth interval has one overcount, second and fourth interval has one undercount.

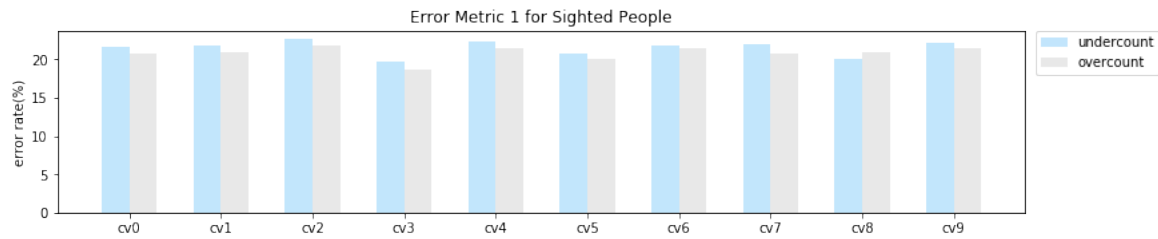


Figure 16: Error metric 1 for 10-fold cross-validation for sighted people. The error rate is much higher than error rate for metric 2 and 3 as shown in this figure. The reason that caused such situation is also described in tge figure.

Sampath Jayalath, Nimsiri Abhayasinghe, and Iain Murray. 2013. A gyroscope based accurate pedometer algorithm. In *International Conference on Indoor Positioning and Indoor Navigation*. volume 28, page 31st.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Andrea Mannini and Angelo Maria Sabatini. 2011. A hidden markov model-based technique for gait segmentation using a foot-mounted gyroscope. In *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*. IEEE, pages 4369–4373.

Najme Zehra Naqvib, Ashwani Kumar, Aanchal Chauhan, and Kritka Sahni. 2012. Step counting using smartphone-based accelerometer. *International Journal on Computer Science and Engineering* 4(5):675.

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

TensorFlow. 2017. Programmer's guide - graphs and sessions. https://www.tensorflow.org/programmers_guide/graphs. [Online; accessed 14-December-2017].

Michal Tomlein, Pavol Bielik, Peter Krátky, Stefan Mitrik, Michal Barla, and Mária Bieliková. 2012. Advanced pedometer for smartphone-based activity tracking. In *HEALTHINF*. pages 401–404.

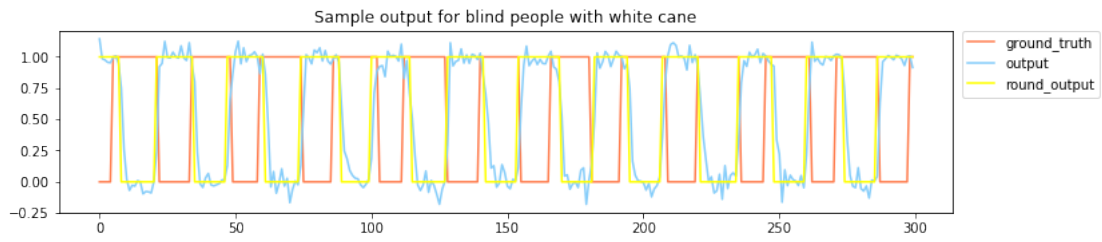


Figure 17: Sample output of blind people with long cane.

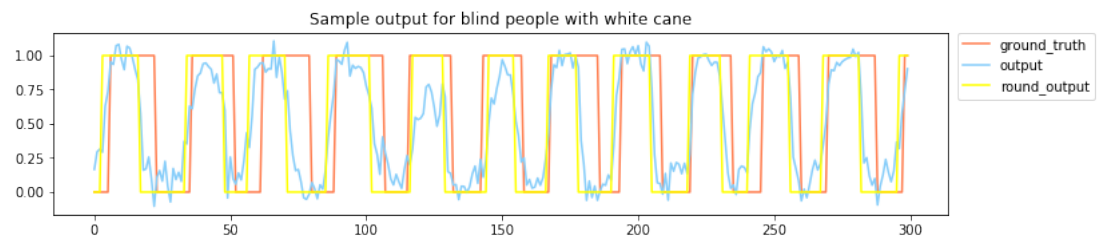


Figure 18: Sample output of blind people with long cane.

Table 2: Error metric 2 and 3 for 10 fold cross validation of sighted people

error(%)		cv0	cv1	cv2	cv3	cv4	cv5	cv6	cv7	cv8	cv9
metric1	undercount	0.82	0.75	0.62	0.75	0.62	0.82	0.96	0.82	1.3	0.96
	overcount	1.85	1.3	1.5	1.03	1.3	1.71	1.78	0.96	3.83	1.78
metric2	undercount	0.48	0.68	0.55	0.68	0.41	0.48	0.55	0.62	0.62	0.68
	overcount	1.5	1.23	1.44	0.96	1.09	1.37	1.37	0.75	3.15	1.5

Table 3: Error mertric 1, 2 and 3 for blind walkers using a long cane

error(%)		cv0		cv1		cv2		cv3		cv4		cv5	
		valid	test	valid	test	valid	test	valid	test	valid	test	valid	test
m1	undercount	6.44	21.49	14.83	14.03	21.45	17.1	19.5	20.67	18.05	19.03	13.35	14.56
	overcount	8.87	23.63	31.5	28.1	19.32	19.2	21.69	23.91	13.17	23.09	13.7	21.94
m2	undercount	12.93	10.09	17.79	9.97	11.36	6.77	11.49	3.28	26.69	10.62	4.09	13.45
	overcount	17.81	14.36	38.01	26.33	10.97	11.07	16.37	9.06	24.35	17.06	5.9	22.76
m3	undercount	0.43	0.7	3.63	0.21	2.94	0.94	1.03	0.29	5.59	1.11	1.2	1.19
	overcount	5.32	4.96	23.85	16.57	2.7	5.25	5.9	6.07	3.25	7.55	3.01	10.5