

UNIVERSITY OF CALIFORNIA
SANTA CRUZ

**LSTM RECURRENT NETWORK FOR STEP COUNTING BASED ON
WEALLWALK DATASET**

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

LSTM Recurrent Network for Step Counting based on WeAllWalk Dataset

Ziyi Chen

Computer Science

University of California Santa Cruz

zchen139@ucsc.edu

Abstract

Smartphones offer various sensors such as accelerometers and gyroscope that can be used for pedometer and environment-related events. This paper trains an LSTM recurrent network for counting the number of steps taken by both blind and sighted users, based on annotated smartphone sensor data from WeAllWork Dataset. The models were built separately for sighted people, blind people with white cane and 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 accelerometers, 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. 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 LSTM models trained by indoor walking sensor data from iPhones and annotated data from a shoes-mounted sensor (cpro) of WeAllWalk dataset as training data and label, to predict left or right steps and calculate the count of steps. The sensor cpro is attached to the user's shoes so it can provide precise times of heel strike. In the dataset, blind volunteers using long canes and guided dogs as well as sighted volunteers have different features of gaits, so we build the models separately for each group and calculate the error rate based on three metrics. We also test the

accuracy of models using leave-one-out validation. Three error metrics splitting intervals differently estimate the overcount and undercount rate of the system. We tried different parameters for the LSTM models such as training steps and input size to compare the difference of output. The model achieved 1% overcount and undercount rate for mixed training data and 5% for 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. It can be embedded in shoes, in smartwatches, in smartphones, and attached to users' ankles or hung on the belt.

(Brajdic and Harle(2013)) evaluates 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) introduces step detection and intelligent detection of cheating based on smartphone sensors. (Naqvib et al.(2012)Naqvib, Kumar, Chauhan, and Sahni) presents 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 various fields including step counting. Since sensors provide time series data, researchers try to use RNN for counting the number of steps.

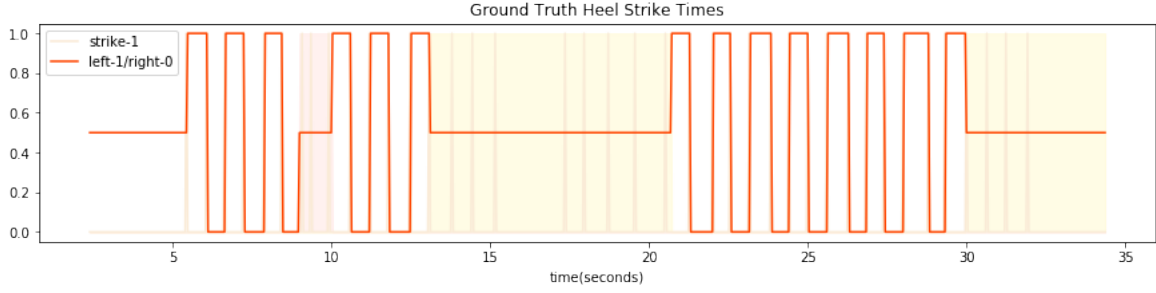


Figure 1: Example heel strike times of T1_ID1_WC.xml from 2.4s to 34.4s. The first step is left feet at 5.4761s. A feature motion (the participant walked into the wall, stopped, and moved to the left) happened from 9.041862s to 10.041862s. Two turn motions, turn east to south from 13.1752s to 20.7419 and turn south to east from 30.0419s to 39.1419s occurred. All data before first step, feature data and turn data are removed.

(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.

Except sighted people, some researchers also pay attention to people with visual impairment. (Haegele and Porretta(2015)) validate 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) validate the same talking pedometer for adults.

2.2 WeAllWalk Dataset

WeAllWalk dataset (Flores and Manduchi(2016)) contains iPhone 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 labels. Each segment of paths is marked as straight segment or turn segment. Special circumstances on the road are also labeled as features. Based on the data from WeAllWalk, they tested six different algorithms for step counting and turn detection.

3 Implementation

We propose an LSTM model trained by indoor walking sensor data of iPhone to predict left or right steps for step counting. Since blind participants using a long cane or a guide dog and sighted volunteers have different features of gaits, we separately build the model and calculate the error rate using three metrics. Also, we only considered straight segments in the paths, since the gait is

more likely to be regular rather than turn segments, and avoided segments labeled as features such as a door that need to be opened on and an obstacle needs to be eluded. We try different parameters for the LSTM model and add a dropout layer to make the result more robust.

3.1 Data Preprocess

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

The second type 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, removing segments marked as features.

Since the XML files only contain heel strike times, we cannot use it as labels directly. 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 (25Hz) and transform the XML file to binary signal in which left steps trigger the signal from 0 to 1 and right steps trigger the signal from 1 to 0.

As shown in picture 1, the light color line of strike time is the original data from XML file. It is transformed to redline which is binary signal. We also remove the turn data(light yellow back-

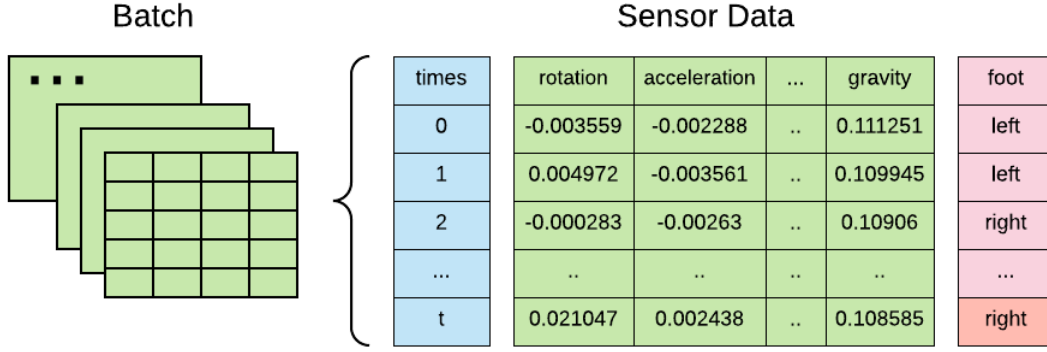


Figure 2: Input (green) and output (pink) example for LSTM. Use previous 0 to t number records of sensor data (green blocks) to predict the last t signal (last pink block). The intermediate outputs (light pink) would be regards as output for first (t-1) records at beginning of each segment since they don't have enough previous record.

ground part) and feature data(light red background part). Then we windowed the data for corresponding sensor data and labels.

3.2 LSTM Model

Long short-term memory (LSTM) network is a recurrent neural network. It is a time-recursive neural network, suitable for processing and prediction the important events of time series in the relatively long delay . LSTM has many applications in science and technology domains. The LSTM-based system can learn tasks such as translating languages, image analysis, speech recognition, controlling chat robots and so on.

The main difference between LSTM and RNN lies in that it adds a "processor" to judge whether the information is useful or not in the algorithm. The structure of this processor is called cell. Three multiplicative units called the input door, forgotten door and output door are placed in the cell. When a message enters the LSTM's network, it would determine whether it is useful. Only information that complies with algorithmic certification will remian, and misleading information will be forgotten through forgotten doors.

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 data flow graph to represent computation in terms of the dependencies between individual operations. We first define the data flow graph and then create a session to run the graph. The Saver class of TensorFlow can add op-

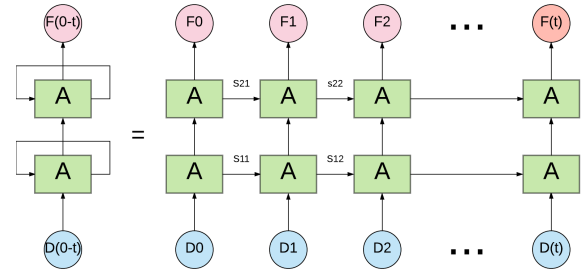


Figure 3: Two layer LSTM recurrent network. Each green block is a LSTM block. Blue circle $D(t)$ represent for sensor data recorded at time t , s is inner lstm state, pink circle F is the output signal.

erations to save and restore variables to and from checkpoints, which map variable names to tensor values.

As shown in picture 3, a two-layer RNN network with basic LSTM cell is built for training with squared difference loss function. As shown in picture 2, the input batch is a list of metrics. Each row of metrics is a list of sensor data at a time. The length of times is called timesteps. For Example, we want to use previous timesteps ($= 50$) record of sensor data to predict the result, each sensor data contains input number ($= 6$) of values of rotation rate and user acceleration in 3-dimension. So the matrix size is timesteps multiply input number(50×6). All such metrics form the input list and is transformed to tensor as train data. 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

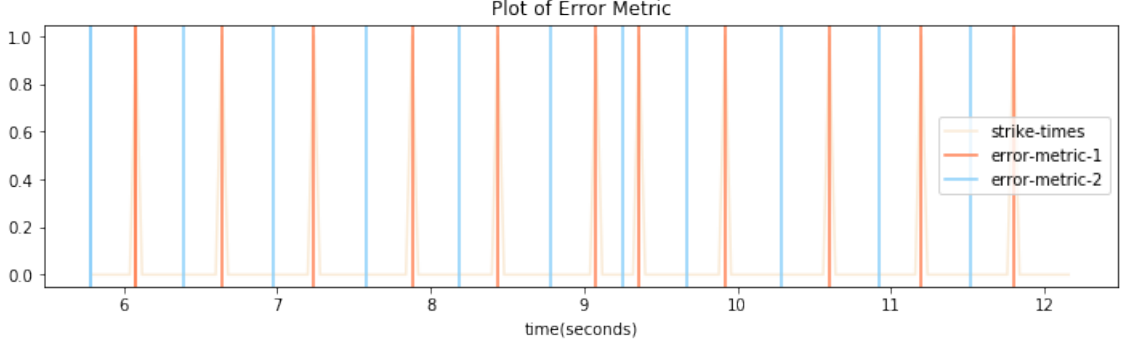


Figure 4: Example of three error metrics. Assume above plot is a entire segment. Error metric 1 (red line) split intervals exactly at each heel strike times. Error metric 2 (blue line) split intervals exactly at middle of every two continues heel strike times. Error metric 3 calculate undercount and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

batch (batch size = 256) and feed the batches to model one step by one step.

There are two ways of output. The first one is to use previous timesteps data to predict only the last step signal value. The second way 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. Outputs that have enough previous records (timesteps previous record) are predicted output using first way. But the first timestep outputs of each segment don't have enough previous records, so they are predicted using the second way.

3.3 Error Metrics

The origin result signal is a float function around 0 to 1. Then we use 0.5 as the border to turn the signal into 0 or 1. And the predicted heel strike times is the signal change times. We can calculate the accuracy rate of the result, which is the proportion of correct numbers (1 is predicted to be 1 and 0 is predicted to be 0) and total numbers. The error rate of the step counting model was measured using three different metrics as shown in picture 4. The first two metric is same as error metric in (Flores and Manduchi(2016)).

The first metric split intervals at each ground-truth heel strike times, and counts the number of predicted steps within each time interval $[T_i, T_{i+1}]$. If exactly one step is detected at each interval, the count of step is correct. If no step is detected during the interval, then one undercount event happens. If multiple steps are detected during the interval, then number of count-1 overcount

events happen. 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 continues 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 picture 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 segment. 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

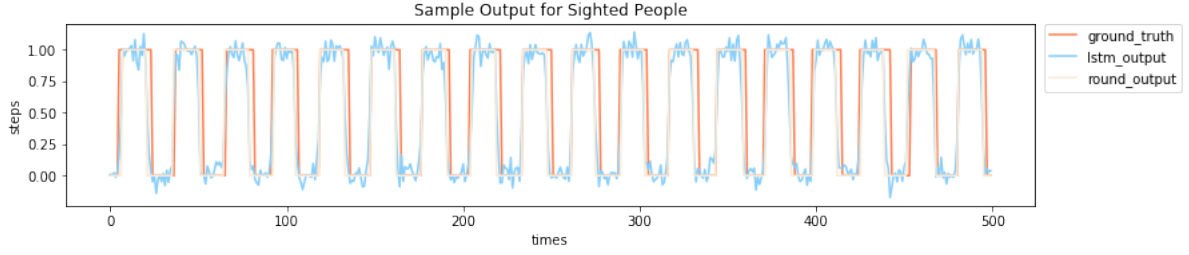


Figure 5: Sample output of mixed data for sighted people. The origin output signal (blue line) swings near 0 or 1. The round signal (yellow line) is quite similar to the ground truth data (red line).

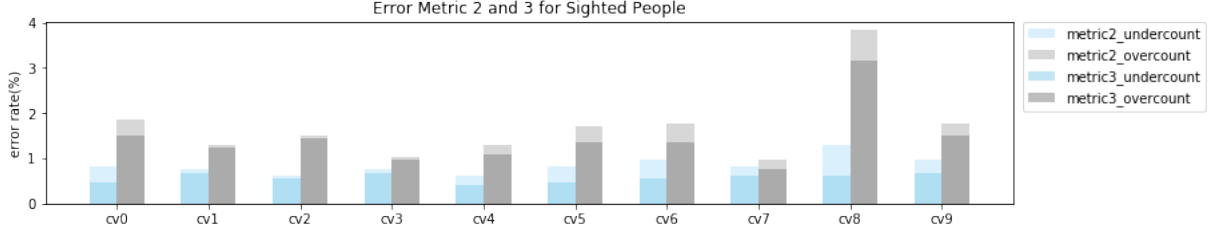


Figure 6: Error metric 2 and 3 for 10 fold cross validation of mixed data for sighted people. The error rate of metric 2 is always higher than metric 3. The undercount error (blue blocks) is less than overcount error (grey blocks). Detailed values is shown in table 2.

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 picture 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 overcount 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 picture 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 picture 16. The reason that caused such situation is described in picture 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 picture 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 in-

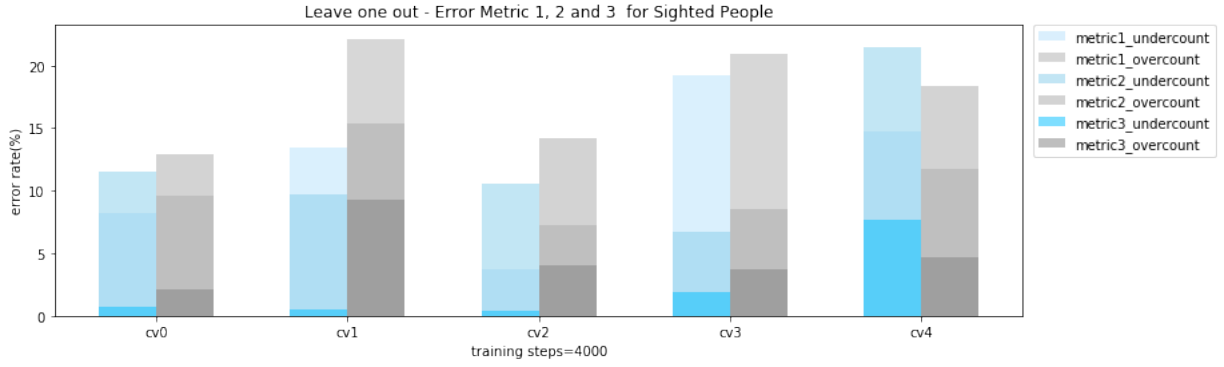


Figure 7: Error metric 1, 2 and 3 for leave one out of sighted people.

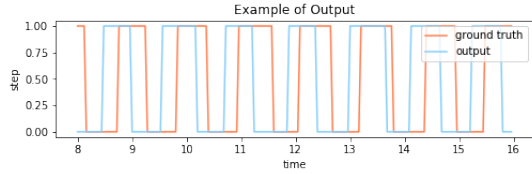


Figure 8: Sample output of offset. In this case, error metric 1 works better than error metric 2.

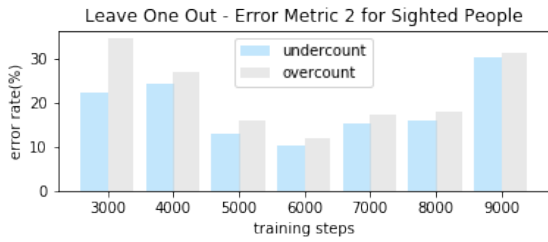


Figure 9: Error metric 1 and 3 for 10 fold cross validation of sighted people. The error rate of metric 1 is always higher than metric 3 with regard to training steps.

interval as shown in the picture 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%. 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 picture 8.

4.2 Blind People with White Canes

There are more than 120,000 records from blind people with white canes, 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 picture 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 people, LSTM model may detect common features of blind people and provide a better result.

4.3 Blind People with Guide Dogs

There are only 3 people walked with the guided 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 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

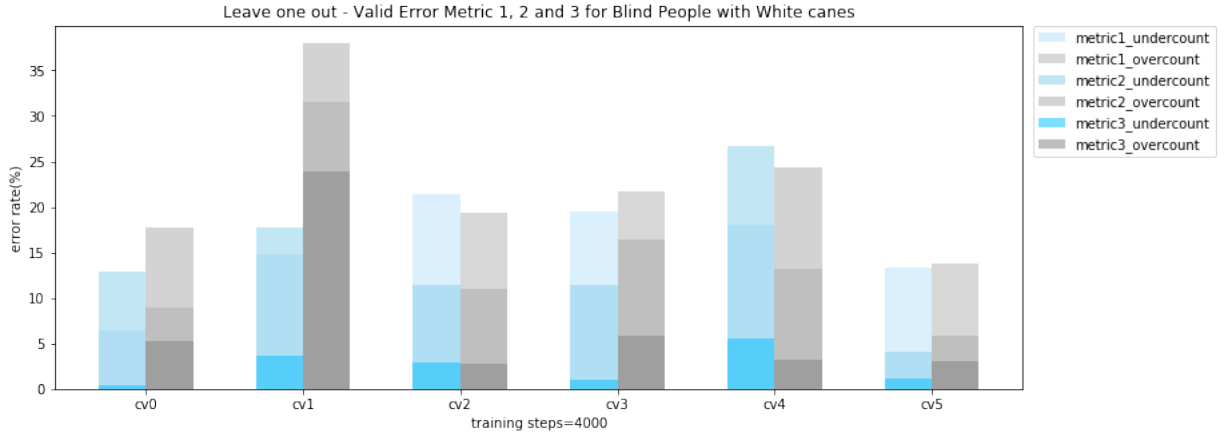


Figure 10: Leave one out - Valid Error for Blind People with White Canes

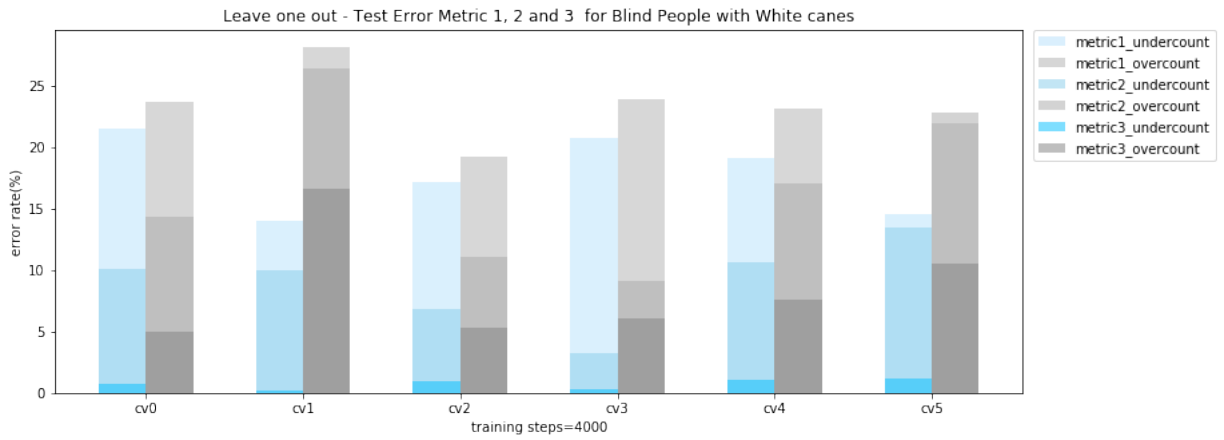


Figure 11: Leave one out - Test Error for Blind People with White Canes

Table 1: Error of blind people with guided 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

separately build the model and calculate the error rate of three metrics. We also apply leave one person 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 picture 14 and picture 13, simply round the origin output signal may cause the over-

count or undercount. So we can use a better way to turn the signal into binary signal. In this paper, we 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 inout sensor data, we can also try more sensor data in future.

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 Mantyjarvi, Elena Vildjiounaite, and Satu-Marja Makela. 2005. Identifying people from gait pattern with accelerometers. In *Proc. SPIE*. volume 5779, pages 7–14.

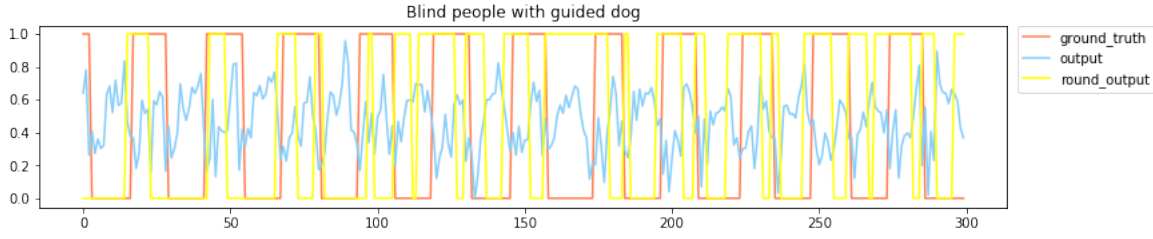


Figure 12: Sample output of blind people with guided dog.

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

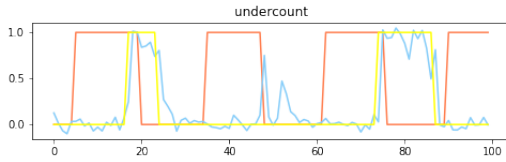


Figure 13: Sample of undercount.

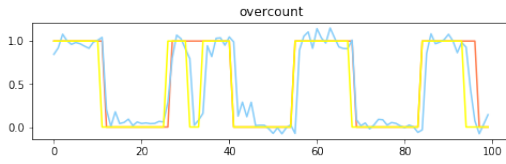


Figure 14: Sample of overcount.

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.

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.

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.

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.

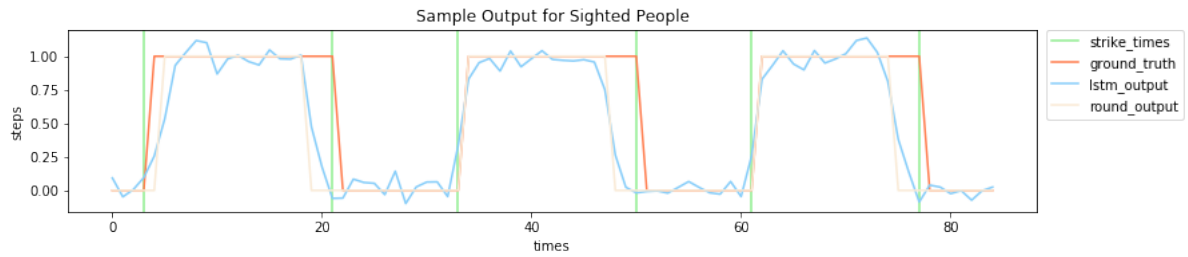


Figure 15: Bad sample for error metric 1. The green line splits the intervals of error metric 1. There are random offset between actual strikes times and predicted strike times. In this case, the first, third and fifth interval has one overcount, second and fourth interval has one undercount.

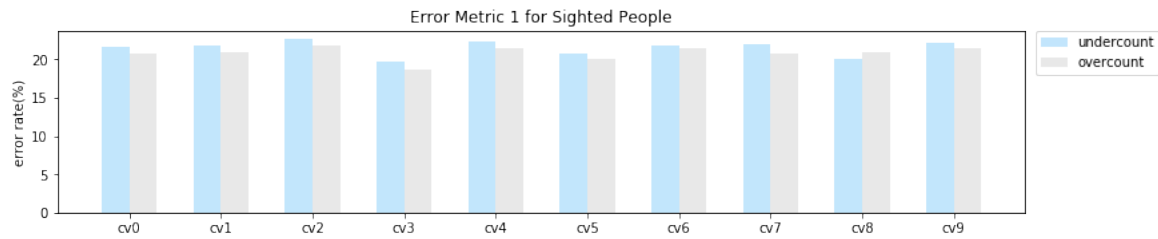


Figure 16: Error metric 1 for 10 fold cross validation for sighted people. The error rate is much higher than error rate for metric 2 and 3 as shown in picture. The reason that caused such situation is described in picture.

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.

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

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

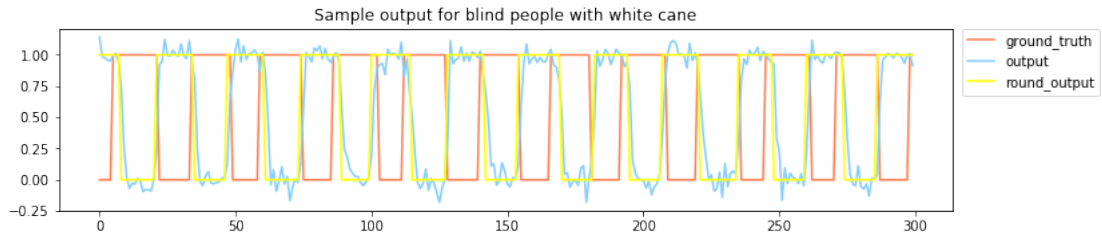


Figure 17: Sample output of blind people with white cane.

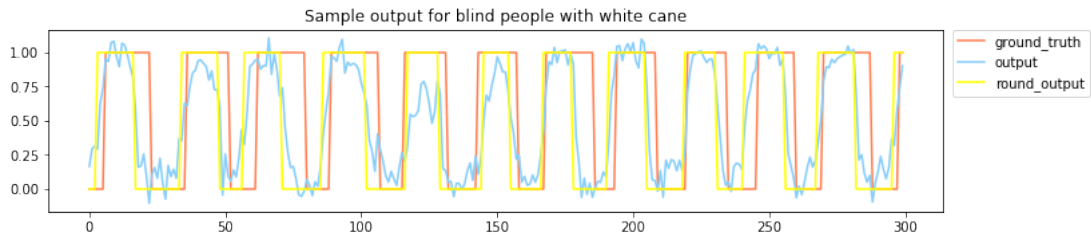


Figure 18: Sample output of blind people with white cane.

Table 3: Error Mertric 1, 2 and 3 for blind people with white canes

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