Calibrating Recurrent Neural Networks on Smartphone Inertial Sensors for Location Tracking

Citation for published version:
Wei, X & Radu, V 2019, Calibrating Recurrent Neural Networks on Smartphone Inertial Sensors for Location Tracking. In 2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN). Institute of Electrical and Electronics Engineers (IEEE), pp. 1-8, 2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN), Pisa, Italy, 30/09/19. https://doi.org/10.1109/IPIN.2019.8911768

Digital Object Identifier (DOI):
10.1109/IPIN.2019.8911768

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Early version, also known as pre-print

Published In:
2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN)

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
Calibrating Recurrent Neural Networks on Smartphone Inertial Sensors for Location Tracking

Xijia Wei  
University of Edinburgh  
s1574554@sms.ed.ac.uk  

Valentin Radu  
University of Edinburgh  
valentin.radu@ed.ac.uk

Abstract—The need for location tracking in many mobile services has given rise to the broad research topic of indoor positioning we see today. However, the majority of proposed systems in this space is based on traditional approaches of signal processing and simple machine learning solutions. In the age of big data, it is imperative to evolve our techniques to learn the complexity of indoor environments directly from data with modern machine learning approaches inspired from deep learning. We model location tracking from smartphone inertial sensor data with recurrent neural networks. Through our broad experimentation we provide an empirical study of the best model configuration, data preprocessing and training process to achieve improved inference accuracy. Our explored solutions are lightweight to run efficiently under limited computing resources and also achieving accurate estimations, within 5 meters median error from inertial sensors alone.

Index Terms—deep learning, location tracking, indoor localization, inertial sensors, recurrent neural networks, dead reckoning

I. INTRODUCTION

A growing number of location based services has given rise to the research topic of position estimation, proposing innovative solutions for the more difficult cases, such as for indoors where access to GPS is unreliable. Using the inertial sensors available on smartphones (accelerometer, gyroscope and magnetometer) good position estimations are achievable, although not without limitations.

Inertial sensors are commonly used to construct Dead Reckoning systems, taking a confident observation as starting point, followed by consecutive location estimations on top of previous locations by determining direction of movement and traveled distance [1]. However, a severe problem with this approach is that occasional erroneous estimations (due to sensor noise, drift and device calibration) are cumulative in the system, leading to increasing estimation errors very fast [2]. For this reason, Dead Reckoning is often augmented with opportunistic anchoring to the physical space, either by identifying unique signatures of sensors [3], activity recognition [4], ambient conditions [5], collaborative estimation [1] or by radio signal signatures [6].

Exploring the literature, the vast majority of previous solutions to perform location estimation from inertial sensors proposes heavily engineered approaches. These are as good as the quality of human expert observations and their modelling skills. The problem with this manually engineered approach is that edge cases will always exist that are hard to formulate and integrate in these systems, which is also the reason for the wide performance variation we see between such systems. We argue that manual formulation of the location estimation process is limited and so we should rely on automatic learning directly from data instead, without much human intervention.

In this age of data driven systems, adopting modern machine learning techniques, such a deep learning, will help to move our community forward.

In this work we explore a robust modelling solution, Recurrent Neural Networks (RNN) for the task of position tracking on smartphone inertial sensors. RNNs have proven effective in other sequence based tasks, such as in machine translation, speech, and natural language processing. We explore a range of data preprocessing choices and model configurations to assess their impact on location estimation accuracy by training several different models. We find that data down-sampling is beneficial to having a smaller model that can run on mobile devices, while achieving below 5 meters median error, and time window overlapping helps to strengthen observations in the model while also expanding the available training data to benefit training. Transferring models training on data from one device to perform estimation on another device is also explored here, showing the good generalization of RNN models.

Although we move the burden of developing localization systems to generating good labeled training sets, we believe this is more scalable since data collection is easier than human intervention to alter previous systems for new environments and edge cases. Solutions based on infrastructure cameras to extract location estimation [7] for sensor data labeling can be one approach to enhance training data collection at scale.

This paper makes the following contributions:

- We formulate location tracking as a recurrent neural network problem, incorporating all the complexity of mobility model generation into an automatic learning process from location labeled sensor data.
- Training and testing of recurrent neural networks is done on a sizable dataset we collect for this exploration.
- We offer insights into the best options to calibrate recurrent neural networks to achieve improvements in location estimation with these recurrent neural network models.
term memory cannot accumulate indefinitely, so a forget gate is used to keep the size tractable. Figure 2 shows the internal structure of one LSTM unit. In each unit, there are not only input and output gates but also a forget gate that controls the amount of information propagated to the next block and what is dropped in the current stage [10]. The input to a block for us is a concatenation of acceleration, gyro scope and magnetometer values over a time window.

The value in the current state is controlled by the forget gate $f$ signal. Specifically, this saves the value when the signal is set to $I$ and forgets when the gate is set to $0$. The activation of receiving a new input and propagating this are determined by signals to the input gate and to the output gate respectively [11]. Equations 1 to 6 show the formulation of transformations performed inside the block, where $W$ are weights learnt in training.

$$i_t = \sigma(W_{ix}x_t + W_{im}m_{t-1})$$  \hspace{1cm} (1)

$$f_t = \sigma(W_{fx}x_t + W_{fm}m_{t-1})$$  \hspace{1cm} (2)

$$\alpha_t = \sigma(W_{ox}x_t + W_{om}m_{t-1})$$  \hspace{1cm} (3)

$$c_t = f_t \odot c_{t-1} + i_t \odot h(W_{cx}x_t + W_{cm}m_{t-1})$$  \hspace{1cm} (4)

$$m_t = \alpha_t \odot c_t$$  \hspace{1cm} (5)

$$p_{t+1} = \text{Softmax}(m_t)$$  \hspace{1cm} (6)

As inertial sensor data is streamed in time sequences, the LSTM model is ideal for location estimations on this type of data. The size of one sample is $\text{time} \_\text{step} \times \text{no} \_\text{features}$, where features are the magnitude of measured values on the three axes of each sensor, accelerometer, gyroscope and magnetic sensor. A time window is formed of a number of sensor signal samples collected over such interval of time and regularised to a fixed sampling rate. Each data instance has a target position $(X_t, Y_t)$ as label. On this data representation, the LSTM model produces position estimations in coordinates $(X_{est}, Y_{est})$. This is formulated by equations 7-9.

$$x_{-1} = \text{Sensor Data}(I)$$  \hspace{1cm} (7)

$$x_t = W_s S_t, \quad t \in \{0...N - 1\}$$  \hspace{1cm} (8)

$$p_{t+1} = \text{LSTM}(x_t), \quad t \in \{0...N - 1\}$$  \hspace{1cm} (9)

Because LSTMs use long-term memory, this has an advantage over traditional Dead Reckoning in tolerating more local noise, benefiting from long-term memory as an superimposed global filter. Also, long-term memory is important to avoid vanishing gradients when propagating information over longer sequences. Through this, distant events like unique signatures on the path [3] and specific activities [4] are used as anchor points automatically in the model, specializing on the most distinctive observations and their order in training sequences.

### III. Evaluation

This section presents our data collection process, model training and validation of different data preprocessing options and LSTM configurations.
to differentiate between various walking conditions.

stringent condition to experiment the ability of LSTM models
one run. This variation in walking speed was enforced as a
the speed of walking, between 2.5 minutes to 4 minutes for
trajectory, each taking different amount of time exercised by
a single participant who performed 14 runs on the same
to variations in walking styles [4], we collected data from
phone. To avoid calibration across many participants due
track, to input ground truth locations with the foreground
interface displaying the building map by user inputs
in the foreground mode (as shown in Figure 3(a)). A long
tap on the map triggers an event to store the latitude and
coordinates as provided by Google Maps API at
the location indicated by the user as ground truth coordinates.
This application can be configured to operate in tandem with a
second phone operating in background mode to collect inertial
sensor data only (Figure 3(b) shows the options available to
configure the application for data collection in background
mode). This permits the second phone to be placed in any
position, in a bag, in pocket or anywhere else without the need
for user interactions with the device during data collection,
which resembles the perspective of sensors in natural motion.

Ground truth labels are transferred from the phone operating
in foreground mode and accepting manual location inputs from
our human annotator to the phone operating in background
mode, which collects sensor data. We conducted a long
data collection campaign following this collection approach.
Ground truth positions were provided sporadically by an
external observer, by following participants on the experiment
track, to input ground truth locations with the foreground
Phone. To avoid calibration across many participants due
to variations in walking styles [4], we collected data from
a single participant who performed 14 runs on the same
trajectory, each taking different amount of time exercised by
the speed of walking, between 2.5 minutes to 4 minutes for
one run. This variation in walking speed was enforced as a
stringent condition to experiment the ability of LSTM models
to differentiate between various walking conditions.

A. Data Collection

Sensor data is collected with an Android application de-
digned and built specifically for this task. This application can
be configured to collect inertial sensor data (accelerometer, magnetometer, gyroscope) continuously in foreground ben-
fitting from a visual interface to accept user inputs, or in
background mode when carrying the phone in pocket with the
screen off. Ground truth information is provided through the
visual interface displaying the building map by user inputs
in the foreground mode. We experiment training LSTMs under different parameter
conditions to identify the best configurations on inertial sensor
data (tuples of acceleration, gyroscope and magnetic field). We
also impose a position invariant condition by working
with the magnitude value on the three orthogonal axes of
measurement:

$$sensor_{magnitude} = \sqrt{sensor_{x}^2 + sensor_{y}^2 + sensor_{z}^2}$$

where $sensor_{\{x, y, z\}}$ are the values measured on each of
the three Cartesian axes.

Several time window sizes were explored based on the
following considerations. Firstly, a small time window prompts
location updates at higher frequency over a small time
windows. The second consideration was inference time, a large
time window although might yield better estimation, it requires
more computation resources for one inference, because it
needs to connect more artificial neurons (more connections)
to a large input size, compared with fewer needed for smaller
time windows. At the other extreme is estimation accuracy,
which benefits from a larger time window to capture more
relevant information from signal. In this trade-off we chose 4
time windows of 10ms, 100ms, 1000ms and 2000ms, which
are presented further in the experiments section. We also
consider the case of time window overlapping (with 10%,
50%, 90%) to increase the frequency of location updates for
a more responsive systems. Overlapping windows are also
useful for LSTM models since information from previous
time windows are reinforced over several instances for better
predictions.

To improve forward-pass speed, we also explore down-
sampling, or compressing the input over the same time
window. Figure 4 shows the magnitude value of an accelerometer
with 1000 data points (1000ms) on the left side and down-
sampled by 90% linear compression to the right, where we can
observe that signal trend is retained at this high compression
rate, as also observed from a larger time window of 7s in
Figure 4(b). The other sensors have a similar performance
with down-sampling.

B. Data Preprocessing

The Android API provides sensor samples on event basis,
updating only on value change, which leads to irregular
sampling frequency. We normalise the input frequency by
interpolating at a rate of 1 kHz. These are grouped in a time
window, which we discussed later, and associated one position
(latitude, longitude) to each time window by interpolating
available ground truth locations (which are already dense
enough, about 0.5 Hz).

C. Calibration of Recurrent Neural Networks

We experiment training LSTMs under different parameter
conditions to identify the best configurations on inertial sensor
data (tuples of acceleration, gyroscope and magnetic field). We
collect a dataset of 9366 instances (split with a ration 8:5:1
between training, validation and test).

1) Time Window: Inputs to LSTM are sampled over a time
window and a well selected size offers enough information
to the model for location estimation. A larger time window
TABLE I
NEURAL NETWORK TRAINING PARAMETER SETTINGS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epoch</td>
<td>100</td>
</tr>
<tr>
<td>Batch Size</td>
<td>100</td>
</tr>
<tr>
<td>LSTM Hidden Units</td>
<td>128</td>
</tr>
<tr>
<td>LSTM Layer</td>
<td>1 Layer</td>
</tr>
<tr>
<td>Learning Rate</td>
<td>0.005</td>
</tr>
<tr>
<td>Learning Rules</td>
<td>RMSprop</td>
</tr>
<tr>
<td>Training Data</td>
<td>One Round</td>
</tr>
</tbody>
</table>

exploits larger scale observations, which could prove more reveling for some movement patterns, although being computationally demanding to perform inferences on a mobile device, with less frequently location updates. In contrast, a smaller time window captures limited information causing difficulty in discriminating between similar activities like moving on a flat surface and climbing stairs, although requiring less computations due to a smaller input layer.

We evaluate different time windows by trained the model with instances capturing 10ms, 100ms, 1000ms and 2000ms of sensor samples, using the training configuration indicated in Table I. Figure 5 presents the training with different sizes of time window. The 10ms input size performs the best on test, with a median error lower than for the other three sizes (error computed as euclidean distance between estimation and ground truth). This is because a smaller time window produces more estimations which falls closer to the ground truth than the other time windows, although differences between time windows are minimal as observed in the Cumulative Distribution Function (CDF) plots in Figure 5(b).

The 1000ms based model indicates a good performance on our evaluation data set and given this larger time window captures more variations and different activities relevant when transitioning between floors (climbing stairs), we select this time window size to use in the following experiments.

2) Overlapping Time Windows: This experiment presents the impact of using samples with overlapping time windows. The overlapping ratio experimented with are 30%, 50% and 90%, which increase the amount of training data subsequently by 1.3x, 2x and 9x respectively. Figure 6 shows time windows overlapping each other by 50% (each sample containing half of data points from previous sample and half new ones).

There are two main reasons for using window overlapping. The first is to enhance dependency between consecutive instances by exposing repeated information in the overlapping parts. For LSTM models, this has the role of strengthening through memory adjacent actions and features over consecutive inputs. Secondly, a higher overlap allows us to generates
more unique instances for training, which is beneficial when starting from a limited dataset.

Figure 7 presents the models trained on data generated with overlapping time windows using the three ratios. In training, the model with data overlapping at 90% performs consistently better than the other two trained with 30% and 50% overlapping data, reaching 90% training accuracy (within one meter to the ground truth), while in CDF performance is also the best with median error just above 5 meters (Figure 7(b)). This is an impressive result considering that it is based on nothing more than inertial sensor data, with relevant calibration points extracted from sequence of unique signal characteristics. From this experiment we observe that a higher overlap in consecutive samples is beneficial to strengthening adjacent patterns, not neglecting that it produces more samples to train on. An overlapping of 90% has the best performance, so we adopt this data preprocessing in following experiments.

3) Reducing Input Size: As observed from Figure 4, moderate down-sampling of data points has minimal impact on preserving information and signal trends, so a relevant exploration is to observe the impact of input compression. By using Principal Component Analysis (PCA) transformation on input data, instances can be compressed even further, with the benefit of producing a smaller neural network model since LSTM internal networks are proportional to input size. On applying PCA to a vector of sensor samples, new variables in a lower dimension are calculated based on eigenvalues and eigenvectors, which retain the relevant information needed by a model to perform efficient estimations. We compare the performance of LSTM models taking as input instances after down-sampling (by superimposing a lower sampling frequency on the available data) and with PCA for sample compression. With down-sampling, the size of input is reduced to $10 \times 3(\text{axis}) \times \text{no}_\text{sensors}$. For fairness, we constrain the PCA to use the same number of samples after compression.

Figure 8 presents down-sampling (filtering data points to a 10 Hz frequency) on the left side, with 90% samples overlapping. By this, one sample is down-sampled from the size of $(1000,3)$ to $(10,3)$. The process of reducing sample size from $(1000,3)$ to $(10,3)$ with PCA is presented on the right. Both of these input reduction methods are advantageous to propose a more efficient model from a computational perspectives, since the input size is $10 \times$ smaller, the size of internal LSTM neural structure is also smaller, leading to more efficient models that can run with lower drain on mobile devices and are also trained faster. The other consideration is prediction accuracy.

Figure 9 shows the comparison between down-sampled input model and PCA input model with an overlapping of 90% between consecutive instances. In Figure 9(a) we observe a faster convergence rate for down-sampling, reaching 95% of the validation accuracy after just eight epochs and then retaining high accuracy. PCA input model has a lower accuracy rate at around 87%. This can be due to better time correlations in the down-sampled input. Figure 9(b) presents the down-sampled input model converging rapidly from a validation loss of 0.06 to 0.005 within 20 epochs. PCA based model has a larger validation loss of 0.08.

Figure 10 presents the performance for these trained models in a CDF format on the validation set and test set. Both experiments demonstrate that an LSTM model using down-sampled input performs better than the unaltered input model, with a median accuracy of 8 metres on the test set. A model using PCA inputs performs similarly to the unaltered input model for both validation and test sets.

In general, the model using down-sampled inputs has a better accuracy than that of a model using PCA inputs and better than a model using the unaltered inputs. In fact, PCA performs least well, which could be due to loss of relevant
information during transformation. On the other side, down-
sampling inputs retains the trend and shape of movement as 
observed in Figure 9. This transformation is also better for 
energy savings due to fewer computations performed on a 
smaller input size.

4) Summary of RNN for Inertial Sensors: We first deter-
mine a suitable time window size from 10ms, 100ms, 1000ms 
and 2000ms. With all time windows performing roughly the 
same, we adopt 1000ms windows because this allows more 
sensor samples to be captured for distinguishing between very 
similar actions. Through samples overlapping we observe an 
increase in performance for a higher overlap (90%), which is 
due to enhancing exposure to relevant events across multiple 
samples and for enlarging the training set by generating more 
overlapping instances from a fixed dataset. To reduce the 
complexity and the training time of models, we compress the 
input size by down-sampling and PCA dimension reduc-
tion, observing that trained models with down-sampled inputs 
achieve the best performance.

Figure 11(b) presents the performance of all trained models, 
with CDF generated on the test set. An LSTM model using 
down-sampled inputs with overlapping of 90% has the best 
performance regarding the convergence rate and prediction 
accuracy, achieving a maximum prediction error of just 6 
metres and median error below 5 meters.

D. Transfer Learning

We evaluate proposed LSTM model using down-sampled 
inputs for robustness across different devices – training with 
data from one and transferring the model to another phone. 
We collected two rounds of sensor data with the entry-level 
phone, Smartisan, and with the flagship phone, OnePlus. Data 
from Smartisan is used for training the model under the 
conditions and parameters identified above. Thus, no sensor 
data from OnePlus is used in the training set. We test both 
models trained for down-sampled inputs and for PCA based 
inputs. Figure 12 shows the results in CDF format for testing 
on the same phone used in data collection, Smartisan and 
also for transferring this model and testing on data from 
OnePlus. We observe that transferring the model between 
devices succeeds, obtaining very similar performance between 
the two test sets. As observed before, models with a down-
sampled input achieve better performance.

The results of our transfer learning experiment are plotted 
on the floor-map. In Figure 13 the estimated trajectory of the 
phone used for training (Smartisan) is presented with an 
orange color. This is very similar to the corridor shape (grey 
path) and across the large room to the right of the track. 
Operating at 90% samples overlap permits a more granular 
location estimation. Figure 13(b) presents the estimated tra-
jectory of the OnePlus phone by using the model trained on 
the Smartisan phone. This transferred model shows a good 
generalization by estimating the trajectory reliably with a few 
exceptions in the open areas, but it realigns with this exact 
path fast on incoming observations.

IV. DISCUSSION

Recurrent Neural Networks have an advantage over simple 
Dead Reckoning approaches since they track the estimation not 
just from the last location as done with dead reckoning, but 
considering longer stances of observations in the past offered 
by the long-term memory mechanism. This compensates for 
local distortions and imperfect observations in sensor data, 
which is not available to dead reckoning approaches easily.

Previous solutions based on signal processing have been 
developed when data was scarce and mathematical modeling 
was the standard. However, with the increasing availability of 
data, which is hard to model entirely with precise mathemat-
ical formulation, deep learning adoption offers the benefit of 
extracting complex features automatically from data. While we 
move the complexity of modeling to generating good quality 
labeled data for training, we believe this is achievable with 
ingenious solution to facilitate data collection and labeling 
such as using camera infrastructure opportunistically.

In future work we will integrate the inertial sensing modality 
explored here, with other type of sensing modalities, like WiFi 
fingerprinting, using modality specific neural networks. The 
advantage of proposed use of LSTM on inertial sensors is that 
gradients can flow similarly in other portions of a network ar-
chitectures, making integration with diverse sensing modalities 
easy an elegant, such as using multi-layer perceptions for WiFi 
fingerprints in a multimodal architecture.
Fig. 10. CDF of down-sampled and PCA transformations and of original data format as inputs to LSTMs.

Fig. 11. Different levels of overlapping showing that training is robust on unseen test set matching in performance with validation set.

Fig. 12. Evaluation on test sets collected with two different devices of a model trained with data from just one and performing transfer learning to the second.

V. RELATED WORK

Without the availability of GPS for tracking as in outdoors [12], indoor tracking is performed by inertial sensors in dead reckoning systems [2]. Inertial sensors achieve this by characterising pedestrian gait cycle [13] and direction of movement [14, 15] to build the trajectory relative to a starting position. As presented by Xiao et al. [16], there are three important aspects to inertial motion sensing: motion mode recognition, orientation tracking and step length estimation. Motion recognition from acceleration data has been modeled with simpler classifiers [4], and its been shown to be body attachment sensitive, which is hard to model accurately [17]. Orientation is commonly performed by combining magnetometer and gyroscope data, as presented by Huyghe et al. [18] using a Kalman filter and using deep learning [19]. Unlike all these systems, we leave the mechanics of motion to be automatically discovered by the LSTM from data. LSTMs for location estimation have been used before by Walch et al. [20] although their input consisted of camera images alone and by Wang et al. [21] using magnetic and light sensors.

To reduce the problems of inertial sensing (drift, device calibration and noisy samples), many solutions choose to rely on periodic anchor points by observing unique characteristics of the signal [4], ambient conditions [3, 5] and in combination with other sensors, most commonly with WiFi [6, 22]. We rely entirely on inertial sensors without such imposed calibrations, noticing that LSTMs identify unique signatures in these signals (similar to reference points in previous methods although extracted automatically here), which helps estimations to recover from occasional bad drifts.
VI. CONCLUSIONS

In this data focused age each research field is adapting to exploit the increasing availability of data. This should also be the case with indoor positioning and navigation suitable to learn directly from data with scalable and robust deep learning models. In this work we demonstrate this to be achievable by adopting a recurrent neural network (LSTM) to track the location of a smartphone based on its inertial sensor data alone. We train several LSTM models using different data preprocessing options and model configurations to offer an insight into how these can be tuned for improved their prediction accuracy. We achieving below 5 meters of median error using LSTM models that are lightweight to run on mobile devices and demonstrate these are transferable across devices.

REFERENCES


