Received March 26, 2020, accepted May 5, 2020, date of publication May 8, 2020, date of current version May 19, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2993534

StepNet–Deep Learning Approaches for Step Length Estimation

ITZIK KLEIN[1,2], (Member, IEEE), AND OMRI ASRAF[1]
[1]Huawei Tel-Aviv Research Center, Hod HaSharon 4524075, Israel
[2]Department of Marine Technologies, University of Haifa, Haifa 3498838, Israel

Corresponding author: Itzik Klein (kitzik@univ.haifa.ac.il)

ABSTRACT The case of a user walking with a smartphone in an indoor environment is considered. Instead of using traditional pedestrian dead reckoning approaches to estimate the user step-length, we define a deep learning based framework with an activity recognition model to regress the user change in distance and step-length. We propose StepNet - a family of deep-learning based approaches to regress the step-length or change in distance. In addition, we propose regressing a time-varying gain instead of a constant one used for traditional step-length estimation. A comparison is made between the proposed approaches and different network architectures. Experimental results show that the proposed deep-learning approaches outperform traditional ones for the examined trajectories.

INDEX TERMS Deep Learning, indoor navigation, pedestrian dead reckoning.

I. INTRODUCTION

Smartphones are a vital instrument in our daily lives for many reasons such as safety, information sharing and accessing, sports, positioning and more. Focusing on user latter, determination of the user position outdoors is usually made by using one of the global navigation satellite system (GNSS). However, indoors GNSS is not available and the positioning is based on other approaches such as wi-fi [1], [2], vision [3], [4] or inertial sensors based solutions [5], [6]. Several approaches are available to handle the inertial sensors measurements: 1) inertial navigation system (INS) [7], [8] 2) shoe mounted INS [9], [10] and 3) pedestrian dead reckoning (PDR) [11], [12]. Since we consider smartphone based navigation, the shoe mounted INS approach is not relevant. The smartphone inertial measurement unit (IMU) sensors typically contain large error terms allowing only low-performance navigation accuracy. Thus, using the IMU measurements in a classical INS algorithm (requires three integrations on the measured data), results in a large position and heading errors, making this approach inappropriate. Therefore, only PDR is applicable for smartphone based navigation using inertial sensors.

In traditional PDR algorithms, user steps are detected (for example by using a peaks approach [34]) using the accelerometers measurements. Step length is then estimated by empirical or bio-mechanical approaches [14]–[16]. In parallel, using the gyroscopes measurements the change in heading during the user step is calculated [17]. Given the user initial conditions at the beginning of the step, together with the step length and heading estimation, the current user two-dimensional position can be found. The empirical or bio-mechanical approaches used to estimate the user step length, requires a determination of the approach gain (or other parameters) before PDR can be applied. This gain remains constant throughout PDR application. Yet, such gain was shown to be very sensitive to the user dynamics [18], [19] and smartphone location [20], [23] and thus an activity recognition (AR) model was suggested to identify the user dynamics.For the latter, in [22] time-series based deep learning approaches were applied. A different method was suggested in [21]. Rather than exploring handcrafted features or a time series problem, a signal sequences of accelerometers and gyroscopes was assembled into a activity image for deep learning.

Later on, the smartphone location on the user was also shown to effect the PDR performance [23]. To that end, the authors proposed a smartphone location recognition (SLR) feature based model. In [33], a deep-learning based robust (SLR) approach, which was evaluated on 107 people using four smartphone location - talk, text, swing and pocket, was suggested.

Recently, machine learning and deep learning based PDR and INS approaches are proposed in an attempt to improve traditional PDR positioning accuracy. For example, in [24]

The associate editor coordinating the review of this manuscript and approving it for publication was Ehsan Asadi.

a deep learning approach was suggested to estimate the pedestrian velocity instead of integrating the acceleration. Another approach for velocity estimation was suggested in [25]. There, a complementing non-linear state estimation by a deep-learning model was applied.

Among them, the robust IMU double integration (RIDI) approach [26] focuses on regressing velocity vectors in the smartphone coordinate frame, while relying on traditional sensor fusion methods to estimate device orientations. They use a support vector machine to identify four smartphone locations - pocket, bag, text and waist followed by a support vector regression model to regress the 2D velocity vector. In turn, it is used to correct the accelerometer measurements prior to double integration to obtain the user position. The first deep-learning based PDR approach, IONet [27], uses only accelerometers and gyroscopes measurements to regress the change in distance and heading in a predefined time window. To that they adopt a long short-term memory (LSTM) network and distinguish between four smartphone locations: pocket, bag, handheld and in a trolley. In the same manner, [28] used also accelerometers and gyroscopes measurements as inputs, but in addition, also used the device orientation. They examined several deep learning architectures to regress the user velocity in 2D. The velocity is then integrated to obtain the user position. In contrast ot RIDI and IONet, no AR model was applied and the users participating in the train and test dataset moved freely equipped with an unconstrained smartphone location.

Instead of addressing the whole PDR problem in a single network, as in the previous approaches, another possibility is to divide the heading determination and the step length estimation into two separate tasks. In [29], a network architecture based on spatial transformer networks and LSTM was suggested for heading estimation. Focusing on step-length estimation [30] proposed learning the individual user patters while outdoors using GPS data as ground truth to train a deep learning model. While indoors, the trained model is used to regress the user velocity and traveled distance. There, three smartphone locations were addressed: handheld, swing and pocket with several deep-learning architectures. In [31], an LSTM model with denoising autoencoders were used to regress the user change in distance in each predefined time-window. In addition to the accelerometers and gyroscopes measurements, they added PDR features to the input of the network. Their approach, Tapeline, achieved a walking distance error rate of 1.43%. Yet, this approach suffers from a significant drawback since it requires the pedestrians to hold their phone horizontally with their hand in front of their chest. The same authors follow up paper [32], focused on estimating the stride length while employing an AR model to identify the smartphone location before the regression process. To that end, decision-tree feature based approaches were used to achieve a walking distance error rate of 2.62%.

In this paper, step-length estimation of unconstrained smartphones using deep-learning approaches is considered. The present paper contributions are as follows:

1) We define a deep learning based framework with an activity recognition model to regress the user change in distance both for fixed time windows (regardless of the step duration) and for varying time windows (aligned with the step duration).
2) We present StepNet, a family of deep-learning based PDR approaches able to regress the step-length or change in distance.
3) A through comparison between the regression approaches: step length or change in distance? and also a comparison of deep learning architectures including with and without predefined features at the network input.
4) Examine the possibility to regress the step-length using only accelerometers readings. The motivation stems form the fact that in traditional PDR, only the accelerometers readings help to determine the step-length.
5) A novel approach to regress a time-varying PDR gain instead of a constant one used for the step-length estimation. In that manner, traditional PDR approaches can be applied using the regressed time-varying gain.

The rest of the paper is organized as follows: Section II presents the traditional PDR approach. Section III, present our StepNet approach and network architectures. Section IV show the results and Section V gives the conclusions.

II. TRADITIONAL PEDESTRIAN DEAD RECKONING

In general, a traditional PDR algorithm is usually defined in an horizontal plane (2D). It has four parts: 1) Step detection: the pedestrian steps are detected using accelerometer measurements 2) Step length estimation: the pedestrian step-length can be estimated using several approaches such as regression-based, biomechanical models or empirical relationships 3) User heading determination: the walking direction is obtained from the gyroscope and/or magnetometer measurements (in some approaches the accelerometers readings are also needed) 4) Position update: the current pedestrian position is determined giving initial conditions, heading estimation (from part 3) and step length estimation (from part 2). Recently, an activity recognition model was added as a fifth part to the traditional PDR algorithm. It's purpose is to perform user activity recognition and SLR to select a proper gain for the step length estimation. This procedure is illustrated in Figure 1

Given the pedestrian initial position x_k, y_k at step k, the current step length s_k and heading ψ_k, the current user position is given by

$$\begin{aligned} x_{k+1} &= x_k + s_k \cos \psi_k \\ y_{k+1} &= y_k + s_k \sin \psi_k \end{aligned} \tag{1}$$

Before the step-length can be estimation a step event needs to be determined. One of the common ways to detect a step is by using a peak detection algorithm [34], commonly based on the magnitude acceleration value threshold and

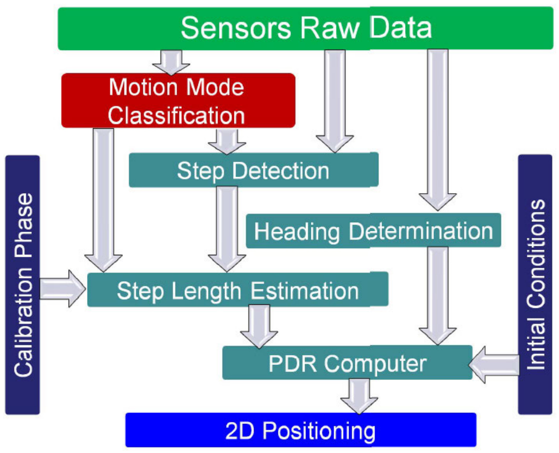

FIGURE 1. Overview of pedestrian dead reckoning algorithm [23].

minimum step period. A step is then defined between two successive peaks. To estimate the step-length two approaches are employed in this paper - Weinberg's [15] and Kim's [14]. The Weinberg's approach is based on the amplitude of the vertical acceleration during the step (we use the measured specific force magnitude)and is given by

$$s_W = G_W\,(\max(f) - \min(f))^{1/4} \tag{2}$$

where G_W is a gain which needs to be calibrated prior to PDR application, f is the set of specific force magnitudes during the step and s_W is the Weinberg-based step-length. Kim's approach is based on the average acceleration during the current step and defined by

$$s_K = G_K \left(\frac{\sum_{j=1}^{N} |f_j|}{N} \right)^{1/3} \tag{3}$$

where N is the number of acceleration samples during the current pedestrian step, G_K is a gain which needs to be calibrated prior to PDR application and s_K is the Weinberg-based step-length.

In order to calculate each of the approaches gains (prior to PDR application), the user needs to walk across a known distance which is equal to the step-lengths sum. Given the accelerometers measurements and known distance, the gain can be calculated by using (2) or (3). For accurate gain estimation this procedure must be repeated several times for any pedestrian dynamics. For example, there will be a different gain value for normal walking speed and a gain for fast walking. In that manner for any user dynamics and smartphone location. The use of inaccurate gain will result in an erroneous step-length estimation which in turn will cause an error in the user position estimation. There are many approaches dealing with the heading estimation, yet in this paper we are only focus on step-length estimation and therefore those approaches are not presented here. The interested reader is referred to a recently published paper comparing several heading estimation approaches [35].

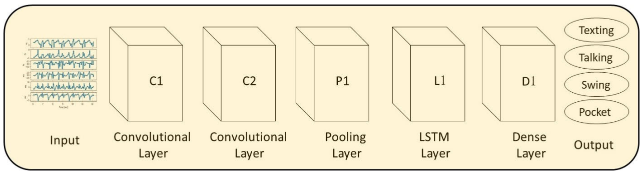

FIGURE 2. CNN/LSTM network architecture used for the SLR model [33] in the StepNet framework.

III. DEEP-LEARNING BASED STEP LENGTH ESTIMATION

We define the deep-learning based step-length estimation framework to include the IMU raw data, deep-learning based activity recognition model and deep-learning based regressor to output the change in distance of the user.

The IMU raw data consists of the measured specific force and angular velocity vectors. The AR model can hold a classifier to identify the user dynamics and another one to determine the location of the smartphone on the user. In this research, it is assumed the user is walking and thus we employ only a SLR AR model. The smartphone location and IMU raw data are then passed to our proposed StepNet to yield the change in distance of the user.

We consider two approaches to regress the user change in distance in our proposed StepNet: 1) fixed time windows (regardless of the step duration), were the raw IMU data is passed directly to the network, and 2) varying time windows aligned with the step duration, were features are calculated on the raw IMU measurements and are feed to the network. Notice, that in the second approach although deep-learning approaches perform automatic feature extraction, handcraft features are computed and used as input to the network. The motivation for this combination of features and deep learning enables a simple way of working with a varying window size aligned with the varying step duration. As a result, the dataset (features) is much smaller than a dataset containing the sensors raw data which leads to a much faster training period and requires less network parameters. Both of the approaches are described in the following sections.

A. SMARTPHONE LOCATION RECOGNITION

We follow [33] and employ a one-dimensional convolutional neural network (CNN) and a Long- and short-term memory (LSTM) in an architecture combines both of them; CNN as the first layer followed by an LSTM layer. The motivation for such architecture lies in the fact CNN has the ability to extract features from the data while LSTM can explore temporal dependencies in the time series problem. The CNN/LSTM architecture that was used for the evaluation of SLR is presented in Figure 2. This architecture obtained the best performance as described in [33]. The input to the first CNN layer, C1, is the specific force and angular velocity vectors. C1 has 32 units with an ReLU activation. The next layer is also a 1D-CNN, C2, with the same parameters as C1. After dropout of 0.6, the next layer is a polling layer of size 2 followed by LSTM layer with 32 units. The last layer D1 has 4 units with Softmax activation. The loss function was

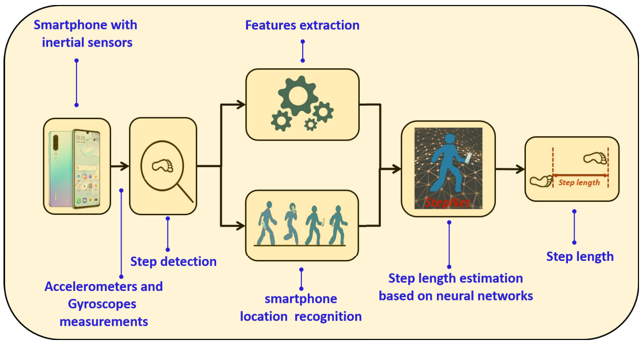
Smartphone with inertial sensors
Features extraction
Accelerometers and Gyroscopes measurements
Step detection
smartphone location recognition
Step length estimation based on neural networks
Step length
Step length

FIGURE 3. Proposed StepNet architecture to output the step length.

categorical cross entropy (CCE) for a single label categorization. The optimization was performed with the RMS propagation algorithm (RMSProp), which divides the gradient by a running average of its recent magnitude.

B. STEP-LENGTH ESTIMATION

As in traditional PDR, we design a network to regress the step length as illustrated in Figure 3. We employ a peak based step detection algorithm as in [34]. A step is then defined between two successive peaks. The raw IMU data within the step duration is passed to the SLR model and to the feature extraction model. The classified smartphone location and calculated features are plugged to the StepNet network to output the step length.

In [32], a similar feature approach was suggested. There, features were calculated both on accelerometer and gyroscopes outputs and the regression was on the stride length using classical machine learning (CML) approaches like Gradient Boosting or K-Nearest Neighbor. In our approach, deep-learning (not CML) is employed for the regression of the step length (instead of stride) and we examine the possibility of using only the accelerometers features.

1) FEATURE EXTRACTION

The specific force vector $\mathbf{f}_j$ at time index j be defined as

$$\mathbf{f}_j = [f_{x,j}\, f_{y,j}\, f_{z,j}]^T \tag{4}$$

and the angular velocity vector ω_j at time index j be defined as

$$\omega_j = [\omega_{x,j}\ \omega_{y,j}\ \omega_{z,j}]^T \tag{5}$$

Features were calculated on each specific force and angular velocity vector components and also on the magnitude of the specific force vector

$$f_m(j) = \sqrt{f_x^2(j) + f_y^2(j) + f_z^2(j)} \tag{6}$$

and the magnitude of the angular rate vector

$$\omega_m(j) = \sqrt{\omega_x^2(j) + \omega_y^2(j) + \omega_z^2(j)} \tag{7}$$

That is, features are calculated on the dataset $\mathbf{X} \in \mathbb{R}^8$ where

$$\mathbf{X} = \left\{x \,|\, f_x\ f_y\ f_z\ f_m\ \omega_x\ \omega_y\ \omega_z\ \omega_m\right\} \tag{8}$$

The following statistical features were calculated for all components in (8)

- **Mean**. The mean of a signal $\mathbf{x}$ of length n is defined as

$$\bar{x} = \frac{1}{n}\sum_{j=1}^{n} x_j \tag{9}$$

- **Standard deviation**. The square root of the variance (measure of the spread of data around the mean) is defined as

$$\sigma_x = \left[\frac{1}{n-1}\sum_{j=1}^{n}(x_j - \bar{x})^2\right]^{1/2} \tag{10}$$

- **Average Absolute Difference**. Measure of the spread of data around its mean, taking the absolute difference between values and the mean.

$$AAD_x = \frac{1}{n}\sum_{j=1}^{n}|x_j - \bar{x}| \tag{11}$$

- **Maximum Value**. The maximum value in the window of the signal.
- **Minimum Value**. The minimum value in the window of the signal.
- **Amplitude**. The absolute difference between the maximum value and minimum value.

$$AMP_x = |\max \mathbf{x} - \min \mathbf{x}| \tag{12}$$

In addition, for the specific force components and magnitude also the gravity crossing rate is calculated

- **g-crossing rate**. A count of how many times within a window the accelerometer signal crosses the gravity value.

Finally, two PDR features are also calculated

- **Weinberg**. The measured part from Weinberg's step-length formula (2)

$$W_f = (\max(f) - \min(f))^{1/4} \tag{13}$$

- **Kim**. The measured part from Kim's step-length formula (3)

$$K_f = \left(\frac{\sum_{j=1}^{N}|f_j|}{N}\right)^{1/3} \tag{14}$$

To summarize, six features are calculated for each member in the dataset (8) (specific force and angular velocity magnitudes and components) and an additional three can be calculated only on the specific force magnitude and components.

2) NETWORK ARCHITECTURES

Two CNN based architectures are considered for the step-length regression task CNN1 and CNN2. The base architecture, CNN1, is presented in Figure 4. The input to the first layer, C1, is the features calculated on the sensors measurements. C1 has 32 units with an ReLU activation. The next layer, C2, is identical to C1. Next, is a polling layer

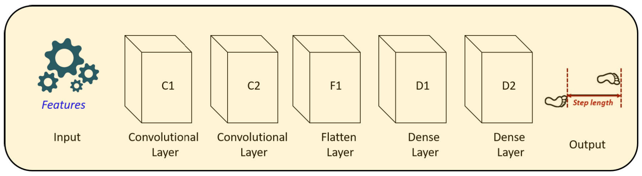

FIGURE 4. StepNet CNN1 architecture.

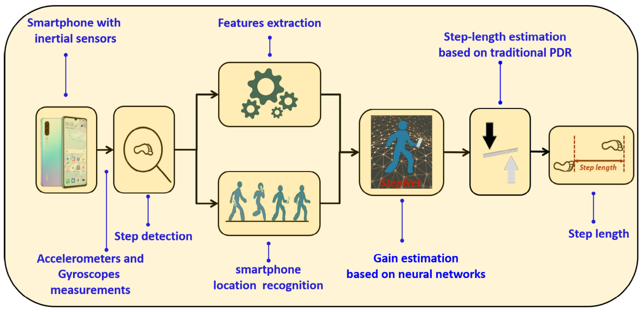
Smartphone with inertial sensors
Features extraction
Step-length estimation based on traditional PDR
StepNet
Step length
Step detection
Accelerometers and Gyroscopes measurements
smartphone location recognition
Gain estimation based on neural networks
Step length

FIGURE 5. Proposed StepNet architecture to output the PDR gain to use in the PDR step length estimation.

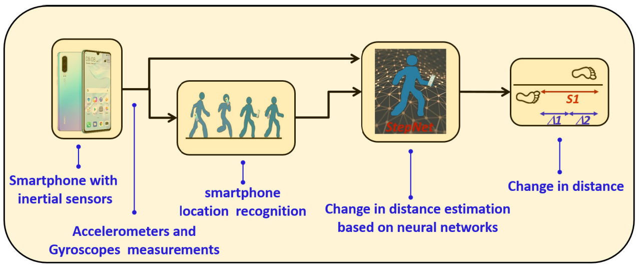
StepNet
S1
Δ1
Δ2
Smartphone with inertial sensors
smartphone location recognition
Change in distance estimation based on neural networks
Change in distance
Accelerometers and Gyroscopes measurements

FIGURE 6. Proposed StepNet architecture to output the change in distance.

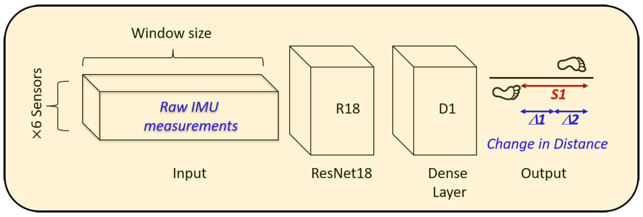

FIGURE 7. StepNet modified ResNet-18 architecture.

of size 2 followed by a dense layer D1 with 128 units. The regression layer D1 has 1 units with linear activation. The loss function used was mean square error and the optimization was performed with the Adam stochastic algorithm. The second CNN architecture, CNN2, has an additional layer, C3, with 64 units after C2. Also, instead of 128 units in D1, it has only 64. All other network parameters are identical to CNN1.

3) PDR GAIN ESTIMATION

Similar to the direct step length estimation approach described in III-B, in this section we propose a deep learning network to estimate the Weinberg PDR gain, G_W. The Setp-Net architecture to output the gain is presented in Figure 5. After a step is detected (Section III-B), the raw IMU data within the step is passed to the SLR and feature extraction models. The classified smartphone location and calculated features are plugged to the StepNet network to output the Weinberg gain, which in turn is substituted into (2) to calculate the step-length. In that manner, the only difference to traditional PDR is an online regressed gain instead of a constant gain.

C. CHANGE IN DISTANCE ESTIMATION

In contrast to the step-length estimation, the change in distance calculation occurs in a predefined time interval - the window size. Thus, a step detection algorithm is not required. In this approach, the raw IMU measurements within the window size are passed to the SLR model to determine the smartphone location which in turn is passed combined with the IMU raw data to a StepNet architecture to regress the change in distance estimation. The proposed approach as shown in Figure 6.

In the literature, two leading approaches were suggested to regress the change in distance: 1) [27] using accelerometer and gyroscope readings as input to a LSTM network. This was the first deep-learning PDR approach, however they do not use an activity recognition model and provide the change in distance every 2s which is much larger than a typical step duration. In our approach, we use 0.5s window size to allow more the network to be more robust to the user time varying step-length 2) [28] uses in addition the device attitude as input to the network in order to regress the velocity, which requires an iteration to obtain the change in distance. Also, they do not distinguish between different user dynamics and smartphone locations. In our approach, we assume the user is walking and apply activity recognition to identify the smartphone location. Also, the device attitude is not used at all in our approach.

1) NETWORK ARCHITECTURE

The network architecture, shown in Fig (7), is based on a modified 2D ResNet-18 [36] network. Since our input data is constructed from stacked 1D IMU readings, we have replaced the Conv1 layer to receive single channel input instead the original published version. The structure of our network is similar to ResNet-18 truncated before the last average pooling layer. We add a single fully-connected layer to regress the change in distance.

IV. ANALYSIS AND RESULTS

The RIDI [26] dataset is employed for the evolution of the StepNet architectures. It contains recordings from eight people for a time duration of about 56 minutes. For the pocket mode, 10 trajectories exists, while for the texting mode 19. For each mode three trajectories were used for the test set and the rest for train set. The distance, time duration and number of steps for each trajectory in test set are presented in Table 1. The three pocket trajectories are denoted as P1, P2 and P3 while the texting trajectories are T1, T2 and T3. All of the trajectories contain turns and most of them have a square shape. Since we are addressing only the estimation of the step-length, we assume the heading is known perfectly and the trajectories are addressed as if they were a straight line.

TABLE 1. Test trajectories parameters.

	P1	P2	P3	T1	T2	T3
Distance [m]	61.9	147.8	67.4	50.8	61.2	130.5
Time [s]	68.4	125.9	71.9	65.0	67.4	113.6
# Steps	70	172	91	77	93	175

To evaluate the performance of the proposed approaches we examine the distance error at the end of the trajectory. For the direct step-length and PDR gain PDRNet architectures the error is defined as

$$e_j = \frac{\left|\sum_{i=1}^{m} \hat{S}_i - \sum_{i=1}^{n} S_{GT,i}\right|}{\sum_{i=1}^{n} S_{GT,i}} \tag{15}$$

where e_j is the error of trajectory j, n is the number of the actual steps taken, m is the number of steps identified, $\hat{S}_i$ is the estimated ith step length and $S_{GT,i}$ is the nominal ith step length. Similar to (15), for the change in distance PDRNet architecture is

$$e_j = \frac{\left|\sum_{i=1}^{k} \Delta\hat{L}_i - \sum_{i=1}^{n} S_{GT,i}\right|}{\sum_{i=1}^{n} S_{GT,i}} \tag{16}$$

where k is the number of times the change in distance was calculated during the trajectory and $\Delta\hat{L}_i$ is the estimated ith distance.

A. STEP LENGTH

Employing the two networks architectures described in Section III-B.2, we examine the influence of the feature dimension input (8) by comparing three feature-set possibilities. Two of them, are based only on the accelerometers data. In the first set (FS1), the statistical features (9)-(12) and g-crossing feature are calculated only on the specific force vector magnitude (6) (8 features) while in the second, (FS2), also on the specific force components (4) (28 features). In the third set, (FS3), the gyroscopes readings are also employed and also the PDR features (13)-(14) are used (54 features).

For each set, 50 Monte-Carlo (MC) runs were made and mean error at the end of the trajectory, calculated using (15), is used as a measure to evaluate the accuracy performance.

For the Weinberg gain estimation, only CNN2 architecture results are presented, since CNN1 produces poor performance.

1) SPECIFIC FORCE MAGNITUDE FEATURES

The mean error, as calculated from 50 MC runs, at the end of the trajectory is presented in Table 2. For the direct step-length estimation, in pocket mode, CNN2 architecture obtained slightly better results than CNN1 but still the mean error is above 10% for two out of three trajectories. The accuracy for the texting mode is much better than the pocket mode for the two networks, where the maximum error is 5.8%. The Weinberg gain estimation (WG) approach, gave the poorest results with two trajectories obtained more than 23% error.

TABLE 2. Mean error for feature set FS1 (8 features).

FS1	P1	P2	P3	T1	T2	T3
CNN1	4.2	10.5	13.5	4.3	3.4	5.8
CNN2	3.1	10.2	12.8	3.4	3.4	4.7
WG	7.6	12.3	19.5	23.1	24.7	16.3

TABLE 3. Mean error for feature set FS2 (28 features).

FS2	P1	P2	P3	T1	T2	T3
CNN1	3.4	3.9	13.6	3.2	2.2	5.0
CNN2	2.9	3.3	12.7	3.5	2.6	3.7
WG	7.2	7.8	11.8	6.2	6.8	11.0

TABLE 4. Mean error for feature set FS3 (54 features).

FS3	P1	P2	P3	T1	T2	T3
CNN1	2.8	4.1	16.1	4.5	2.7	4.8
CNN2	2.6	3.5	15.8	4.6	3.5	3.5
WG	6.7	6.1	7.4	0.7	1.9	6.4

2) SPECIFIC FORCE MAGNITUDE AND COMPONENTS FEATURES

In FS2, the same features as in FS1 are calculated also on the specific force components. The mean error, as calculated from 50 MC runs, at the end of the trajectory, using 15, is presented in Table 3. Improvement in both pocket and texting modes was obtained compared to SF1 in five out of six trajectories in the direct step-length estimation approach. CNN2 network has obtained an accuracy less than 5% in those five trajectories while on the sixth performance was not change compered to SF1. The performance of WG approach was improved by more than 50% in the texting mode. Improvement was also in the pocket mode. Yet, except for the P3 trajectory, the direct step-length estimation obtained much better performance.

3) SPECIFIC FORCE, ANGULAR VELOCITY AND PDR FEATURES

In addition to the specific force features, in FS3 angular velocity and PDR features are also added. The mean error, as calculated from 50 MC runs, at the end of the trajectory is presented in Table 4. For the direct step-length estimation, except, for trajectory P1 where a 15% of improvement was achieved and T1 that obtained slight improvement, in all other trajectories $10-25\%$ degradation was achieved.As a consequence, we can state that the addition of the angular velocity features degraded the performance obtained with FS2. Thus, it is better to use FS2 set, which depends only on the accelerometer features.

However, when considering WG approach we observe that FS3 improved FS2. In T1 the error was reduced to 0.7%, the best accuracy obtained so far. In all other trajectories, both pocket and texting, the accuracy was improved.

We observe, based on the results in Tables (2)-(4), that in pocket mode the direct StepNet architecture (using CNN2) which estimates the step-length obtained the best

TABLE 5. Mean error for feature set FS4 (six features) for pocket and FS5 (five features) for texting modes.

FS4/5	P1	P2	P3	T1	T2	T3
CNN2	3.8	8.1	2.7	1.2	1.8	1.6

TABLE 6. End of the trajectory error for the change in distance approach.

MRES18	P1	P2	P3	T1	T2	T3
WS50	4.6	1.6	0.8	1.0	4.3	1.2
WS100	6.4	1.0	1.0	0.3	2.4	1.7
WS200	4.4	0.8	0.7	3.7	5.3	1.3

performance using FS2 while for texting mode, the WG StepNet architecture with FS3 was the best approach.

4) FEATURE SELECTION

To further improve the performance, feature selection is applied on the FS2 set using CNN2 architecture. To that end, the Ridge regression approach [37], which uses the l2-norm for regularization, is applied. New feature sets were found for the pocket and texting modes. For the pocket mode the feature set, FS4, was found to include only statistical features from the specific force components: f_x, f_y, f_z mean using (9), f_y average absolute difference using (11), amplitude f_x and f_y using (12). For the pocket mode the feature set, FS5, includes the amplitude of f_x, f_y, f_z, f_m using (12) and the g-crossing rate for the specific force magnitude f_m. That is, instead of using 28 features (as the original dataset FS2), FS4 has six features while FS5 has five. The mean error, as calculated from 50 MC runs, at the end of the trajectory is presented in Table 5.

Results show that the accuracy was improved in all three texting trajectories between $21 - 65\%$. For the pocket mode, P1 accuracy remains the same while P3 was improved by 78%. Yet, P2 accuracy was degraded by a factor of 2.5. In summary, using feature selection improved the accuracy relative to the original feature set.

B. CHANGE IN DISTANCE

The change in distance estimation is made used the modified ResNet-18 (MRES18) architecture described in Section III-C. The error at the end of the trajectory as calculated using (16) is presented in Table 6. The network inputs the raw specific force and angular velocity vectors within a predefined window size. The accuracy results at the end of the six trajectories are presented in Figure 6 for window sizes of 50 (WS50), 100 (WS100) and 200 (WS200) samples. The sampling rate of the recorded data is 200Hz, thus we address the user change every 0.25, 0.5 and 1 seconds. The error in all trajectories was less than 6.4% and the results of all window sizes are much better compared to those of the step-length approach. In particular, for trajectory P3 the accuracy was lowered from 12.7% to less than 1.0%. The average error of the six trajectories is 2.3% for WS50, 2.1% for WS100 and 2.7% for WS200.

C. COMPARISON

Traditional PDR algorithms versus the three approaches for StepNet architectures, change in distance, step-length and gain regression, are compared. For the PDR, Kim's (3) and Weinberg's (2) approaches are used for the evaluation. The appropriate gains were calculated based on the train dataset, separately for each of the two modes - pocket and texting. For the StepNet step-length architecture, the feature set FS2 calculated only on the specific force vector are used for the compassion, while in the gain approach FS3 is used. In the StepNet change in distance architecture, we compare the case with window size of 100 samples. Although, $WS200$ obtained better accuracy in the pocket mode, one second duration is too long when considering heading changes in practical real-life scenarios. The comparison is presented in Table 7. All three StepNet architectures obtained better accuracy than traditional PDR approaches. Out of three, the change in distance StepNet architecture is the best approach. In five trajectories, its accuracy is less than 2.5%. Taking the average on all six trajectories, it had 2.1% of error while, StepNet step length is the second best architecture with an error of 4.8%. With almost the same performance, StepNet Gain achieved an error of 3.2%. Weinberg PDR error was 12.7% while Kim PDR was 13.1%.

TABLE 7. Comparison of the end of the trajectory error between traditional PDR to the family of StepNet approaches.

	P1	P2	P3	T1	T2	T3
PDR: Weinberg	9.5	9.6	10.1	8.9	19.0	18.8
PDR: Kim	6.4	8.3	9.8	14.5	26.9	12.7
StepNet: Step-length	3.8	8.1	2.7	1.2	1.8	1.6
StepNet: Gain	6.7	6.1	7.4	0.7	1.9	6.4
StepNet: Change in distance	6.4	1.0	1.0	0.3	2.4	1.7

V. CONCLUSION

A deep learning based framework with an activity recognition model to regress the user change in distance/step length was suggested. It addresses two scenarios, the change in distance estimation for fixed time windows (regardless of the step duration) and step-length estimation for varying time windows (aligned with the step duration). We proposed a family of three StepNet architectures for the regression task: 1) direct step-length 2) change in distance and 3) Weinberg gain regression for step-length estimation.

A through comparison between the three regression approaches on the same dataset was made with the same data used as input to the networks, i.e., the accelerometers and gyroscopes raw measurements or calculated features.

We show that the StepNet architecture to regress the change in distance obtained the best performance with an average error of 2.1% on the examined six trajectories. When using only the accelerometer measurements, the StepNet architecture which regresses the step length obtained the best performance with an average error of 3.2%. As a consequence, we observe that the gyroscopes measurements help to improve the change in distance accuracy. This conclusion is in contrast to traditional PDR approaches where the step-length is determined only by the accelerometer measurements.

REFERENCES

[1] Y. Zhuang and N. El-Sheimy, "Tightly-coupled integration of WiFi and MEMS sensors on handheld devices for indoor pedestrian navigation," *IEEE Sensors J.*, vol. 16, no. 1, pp. 224–234, Jan. 2016.

[2] L.-H. Chen, E. H.-K. Wu, M.-H. Jin, and G.-H. Chen, "Intelligent fusion of Wi-Fi and inertial sensor-based positioning systems for indoor pedestrian navigation," *IEEE Sensors J.*, vol. 14, no. 11, pp. 4034–4042, Nov. 2014.

[3] H. Abdelnasser, R. Mohamed, A. Elgohary, M. F. Alzantot, H. Wang, S. Sen, R. R. Choudhury, and M. Youssef, "SemanticSLAM: Using environment landmarks for unsupervised indoor localization," *IEEE Trans. Mobile Comput.*, vol. 15, no. 7, pp. 1770–1782, Jul. 2016.

[4] M. Brossard, A. Barrau, and S. Bonnabel, "Exploiting symmetries to design EKFs with consistency properties for navigation and SLAM," *IEEE Sensors J.*, vol. 19, no. 4, pp. 1572–1579, Feb. 2019.

[5] N. Yu, Y. Li, X. Ma, Y. Wu, and R. Feng, "Comparison of pedestrian tracking methods based on foot-and waist-mounted inertial sensors and handheld smartphones," *IEEE Sensors J.*, vol. 19, no. 18, pp. 8160–8173, Sep. 2019.

[6] P. Goyal, V. J. Ribeiro, H. Saran, and A. Kumar, "Strap-down pedestrian dead-reckoning system," in *Proc. Int. Conf. Indoor Positioning Indoor Navigat.*, Sep. 2011, pp. 1–7.

[7] P. D. Groves, *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*, 2nd ed. Norwood, MA, USA: Artech House, 2013.

[8] D. Titterton and J. Weston, *Strapdown Inertial Navigation Technology (Radar, Sonar and Navigation)*, 2nd ed. London, U.K.: The Institution of Engineering and Technology, 2005.

[9] E. Foxlin, "Pedestrian tracking with shoe-mounted inertial sensors," *IEEE Comput. Graph. Appl.*, vol. 25, no. 6, pp. 38–46, Nov. 2005.

[10] I. Skog, J. O. Nilsson, and P. Händel, "Evaluation of zero-velocity detectors for foot-mounted inertial navigation systems," in *Proc. Int. Conf. Indoor Positioning Indoor Navigat.*, 2010, pp. 1–6.

[11] S. Beauregard and H. Haas, "Pedestrian dead reckoning: A basis for personal positioning," in *Proc. 3rd Workshop Positioning, Navigat. Commun.*, Mar. 2006, pp. 27–35.

[12] O. Mezentsev, G. Lachapelle, and J. Collin, "Pedestrian dead reckoning—A solution to navigation in GPS signal degraded areas?" *Geomatica*, vol. 59, no. 2, pp. 175–182, 2005.

[13] D. Pai, I. Sasi, P. S. Mantripragada, M. Malpani, and N. Aggarwal, "Padati: A robust pedestrian dead reckoning system on smartphones," in *Proc. IEEE 11th Int. Conf. Trust, Secur. Privacy Comput. Commun.*, Jun. 2012, pp. 2000–2007.

[14] J. W. Kim, H. J. Jang, D.-H. Hwang, and C. Park, "A step, stride and heading determination for the pedestrian navigation system," *J. Global Positioning Syst.*, vol. 3, nos. 1–2, pp. 273–279, 2004.

[15] H. Weinberg, "Using the ADXL202 in pedometer and personal navigation applicationss," Analog Devices, Norwood, MA, USA, Appl. Note. 602, 2002.

[16] J. Scarlet, "Enhancing the performance of pedometers using a single accelerometer," Analog Devices, Norwood MA, USA, Appl. Note. AN-900, 2005.

[17] A. R. Jimenez, F. Seco, C. Prieto, and J. Guevara, "A comparison of pedestrian dead-reckoning algorithms using a low-cost MEMS IMU," in *Proc. IEEE Int. Symp. Intell. Signal Process.*, Aug. 2009, pp. 37–42.

[18] M. Elhoushi, J. Georgy, A. Noureldin, and M. J. Korenberg, "A survey on approaches of motion mode recognition using sensors," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 7, pp. 1662–1686, Jul. 2017.

[19] J. Qian, J. Ma, R. Ying, P. Liu, and L. Pei, "An improved indoor localization method using smartphone inertial sensors," in *Proc. Int. Conf. Indoor Positioning Indoor Navigat.*, Oct. 2013, pp. 1–7.

[20] P. Zhang, X. Chen, X. Ma, Y. Wu, H. Jiang, D. Fang, Z. Tang, and Y. Ma, "SmartMTra: Robust indoor trajectory tracing using smartphones," *IEEE Sensors J.*, vol. 17, no. 12, pp. 3613–3624, Jun. 2017.

[21] W. Jiang and Z. Yin, "Human activity recognition using wearable sensors by deep convolutional neural networks," in *Proc. 23rd ACM Int. Conf. Multimedia (MM)*, 2015, pp. 1307–1310.

[22] M. Zeng, L. T. Nguyen, B. Yu, O. J. Mengshoel, J. Zhu, P. Wu, and J. Zhang, "Convolutional neural networks for human activity recognition using mobile sensors," in *Proc. 6th Int. Conf. Mobile Comput., Appl. Services*, Austin, TX, USA, Nov. 2014, pp. 197–205.

[23] I. Klein, Y. Solaz, and G. Ohayon, "Pedestrian dead reckoning with smartphone mode recognition," *IEEE Sensors J.*, vol. 18, no. 18, pp. 7577–7584, Sep. 2018.

[24] T. Feigl, S. Kram, P. Woller, R. H. Siddiqui, M. Philippsen, and C. Mutschler, "A bidirectional LSTM for estimating dynamic human velocities from a single IMU," in *Proc. Int. Conf. Indoor Positioning Indoor Navigat. (IPIN)*, Sep. 2019, pp. 1–8.

[25] S. Cortes, A. Solin, and J. Kannala, "Deep learning based speed estimation for constraining strapdown inertial navigation on smartphones," in *Proc. IEEE 28th Int. Workshop Mach. Learn. for Signal Process. (MLSP)*, Sep. 2018, pp. 1–6.

[26] H. Yan, Q. Shan, and Y. Furukawa, "RIDI: Robust IMU double integration," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 621–636.

[27] C. Chen, X. Lu, A. Markham, and N. Trigoni, "IONET: Learning to cure the curse of drift in inertial odometry," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 1–9.

[28] H. Yan, S. Herath, and Y. Furukawa, "RoNIN: Robust neural inertial navigation in the wild: Benchmark, evaluations, and new methods," 2019, *arXiv:1905.12853*. [Online]. Available: http://arxiv.org/abs/1905.12853

[29] Q. Wang, H. Luo, L. Ye, A. Men, F. Zhao, Y. Huang, and C. Ou, "Pedestrian heading estimation based on spatial transformer networks and hierarchical LSTM," *IEEE Access*, vol. 7, pp. 162309–162322, 2019.

[30] J. Kang, J. Lee, and D.-S. Eom, "Smartphone-based traveled distance estimation using individual walking patterns for indoor localization," *Sensors*, vol. 18, no. 9, p. 3149, 2018, doi: 10.3390/s18093149

[31] Q. Wang, L. Ye, H. Luo, A. Men, F. Zhao, and Y. Huang, "Pedestrian stride-length estimation based on LSTM and denoising autoencoders," *Sensors*, vol. 19, no. 4, p. 840, 2019.

[32] Q. Wang, L. Ye, H. Luo, A. Men, F. Zhao, and C. Ou, "Pedestrian walking distance estimation based on smartphone mode recognition," *Remote Sens.*, vol. 11, no. 9, p. 1140, May 2019, doi: 10.3390/rs11091140.

[33] I. Klein, "Smartphone location recognition: A deep learning-based approach," *Sensors*, vol. 20, no. 1, p. 214, 2014, doi: 10.3390/s20010214.

[34] Y. Jin, H.-S. Toh, W.-S. Soh, and W.-C. Wong, "A robust dead-reckoning pedestrian tracking system with low cost sensors," in *Proc. IEEE Int. Conf. Pervas. Comput. Commun. (PerCom)*, Mar. 2011, pp. 222–230.

[35] A. Poulose, B. Senouci, and D. S. Han, "Performance analysis of sensor fusion techniques for heading estimation using smartphone sensors," *IEEE Sensors J.*, vol. 19, no. 24, pp. 12369–12380, Dec. 2019.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[37] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. New York, NY, USA: Springer, 2001.

ITZIK KLEIN (Member, IEEE) received the B.Sc. and M.Sc. degrees in aerospace engineering and the Ph.D. degree in civil engineering from the Technion—Israel Institute of Technology, Haifa, Israel, in 2004, 2007, and 2011, respectively. Since 2019, he has been an Assistant Professor, heading the Autonomous Navigation and Sensor Fusion Lab, at the Department of Marine Technologies, University of Haifa. He is currently working with Huawei Tel-Aviv Research Center. His research interests include navigation, unorthodox inertial navigation architectures, autonomous underwater vehicles, sensor fusion, and estimation theory.

OMRI ASRAF received the B.Sc. degree in mechanical engineering from Ben-Gurion University. He is currently pursuing the M.Sc. degree with the Department of Aerospace Engineering, Technion—Israel Institute of Technology. He is also a Navigation Algorithmic Researcher at Huawei Tel-Aviv Research Center. His research interests include active SLAM, planning, and experience-based prediction methods.