Joint Pedestrian Motion State and Device Pose Classification

Parinaz Kasebzadeh, Kamiar Radnosrati, Gustaf Hendeby and Fredrik Gustafsson

Kasebzadeh, P., Radnosrati, K., Hendeby, G., Gustafsson, F., (2020), Joint Pedestrian Motion State and Device Pose Classification, *IEEE Transactions on Instrumentation and Measurement*, 69(8), 5862-5874. https://doi.org/10.1109/TIM.2019.2958005

Joint Pedestrian Motion State and Device Pose Classification

Parinaz Kasebzadeh, Kamiar Radnosrati, Gustaf Hendeby, Fredrik Gustafsson
Department of Electrical Engineering, Linköping University, Linköping, Sweden
Email: {firstname.lastname}@liu.se

***Abstract*—Novel features for joint classification of gait and device modes are proposed and multiple machine learning methods are adopted to jointly classify the modes. The classification accuracy as well as the $\mathbf{F}_1$ score of two standard classification algorithms, K-nearest neighbor (KNN) and Gaussian process (GP), are evaluated and compared against a proposed neural network (NN)-based classifier. The proposed features are the correlation scores of a detected gait cycle relative to a set of unique gait signatures as well as the gait cycle time, all extracted from hand-held inertial measurement units (IMUs). Each gait signature is defined such that it contains one full cycle of the human gait. In order to take the temporal correlation between classes into account, the initial classifiers' estimates are fed into a hidden Markov model (HMM) unit to obtain the final class estimates. The performance of the proposed method is evaluated on a large dataset including two classes of gait modes (walking and running) and four classes of device modes (fixed and face-up in the hand, swinging in the hand, in the pocket and in the backpack). The experimental results validate the reliability of the considered features and effectiveness of the HMM unit. The initial classification accuracy of the NN-based approach is 91%, which is further improved to 99% after the smoothing stage on the validation set and 98% on the test set.**

I. Introduction

Gait analysis and activity monitoring are among the most informative features to be considered in location analysis such as collapse detection, monitoring athletics activity, balance control evaluation, animal activity tracking, pedestrian navigation and deadreckoning, etc. Typically, all these applications rely on inertial measurement unit (IMU) sensors to log the activity information [1–4].

IMUs are embedded in many devices such as smartphones, smartwatches, virtual reality headsets, etc. and have been considered as the dominating motion detection enabler in many applications. For instance, IMUs are widely used in pedestrian navigation systems for human motion tracking, resulting in inertial navigation systems (INS)s [5–8].

The quantities measured by IMUs include angular velocity and acceleration of the sensor. Given this information, the orientation and position information can be estimated by integrating the angular velocity and double integrating the acceleration, respectively. The process of attaining orientation and position information from inertial sensors is called dead reckoning. Pedestrian dead reckoning (PDR) uses the dead reckoning principle in pedestrian navigation system to locate the mobile user in outdoor and/or indoor environments. A crucial part of PDR algorithms are accurate step detection, step length estimation and heading estimation. For instance, a lack of accuracy in gait detection will lead to inaccurate counting of the number of steps and inaccurate estimated step length which in turn will provide a pedestrian position estimate with large uncertainty. One source of information that is beneficial for accurate position estimation or activity analysis is the knowledge of motion mode and device pose that are typically acquired from accelerometer and gyroscope sensor readings [5, 6, 9, 10].

Besides solving dedicated tasks, the IMU signals also contain what will be referred to as the *gait signature* created by the steps we take when moving. The gait signature, as observed by the IMU, depends on both the gait mode (e.g. running, walking, strolling) and the device mode (for instance, a smartphone can be held in the hand, stored in a pocket or backpack, etc.), and as such reveals a rich source of information that is suitable for a variety of applications.

There is a rich literature on gait mode classification using body-mounted or wearable IMUs [11–13]. The authors in [14] propose a continuous HMM method using chest-mounted IMUs for gait analysis and activity classification. Portable devices already equipped with IMUs such as smartphones, tablets, or smartwatches, however, attract more interest for activity recognition [4, 15–17]. The authors in [18] use an artificial neural network (ANN) approach for step detection which is further used in estimating the step length. A pedestrian movement direction recognition approach using convolution neural network (CNN) is proposed in [19] to detect pedestrians movements. The method achieved 94% accuracy in the validation set and 79% in the test set. For a thorough survey of the existing results on recognition of various motion modes, see [17].

In addition to gait mode, device pose recognition also plays an important role in accurate gait parameters estimation [20, 21]. The measured accelerometer and gyroscope magnitudes are considered for classifying four different smartphone modes in [20], resulting in 86% classification accuracy. Accurate step detection requires joint gait mode (the pedestrian movement profile) and device mode classification [22, 23].

To the best of our knowledge the joint motion mode and device pose classification problem, is rather an unexplored area with the exception of [24] in which machine learning approaches such as multilayer perceptron (MLP) and support vector machine (SVM) have been considered to identify the activity mode and device placement simultaneously. Various frequency and time domain features are extracted to create the model that fits the data. The classification rate is evaluated

based on the joint device pose and the motion activities.

It should be declared that this paper reuses some content from thesis [25] with permission. The remainder of this paper is organized as follows: Sec. II presents motivation and contributions of this study as well as the high-level description of the methodology used in this paper. The gait segmentation and feature extraction stages are formulated in Sec. III. Three classification methods and their structures are given in Sec. IV. The HMM block used to include the temporal coherence between two consecutive gait modes is introduced in Sec. V. The performance of the proposed methods is evaluated in Sec. VI, followed by concluding remarks on the work presented in Sec. VII.

II. Motivation and Contribution

Accurate joint motion and gait mode classification are important aspect for pedestrian navigation and also for improving pedestrian dead reckoning (PDR) algorithms. In this paper we propose new features to address a joint gait and motion mode classification for analyzing the human gaits behavior. The benefits of the mode classification for PDR can be explained as follows. The key design parameters in PDR contain the step length and the step detection threshold estimation when the magnitude of the measured acceleration is considered to be caused by a step. Both of these step length and step detection depend on the motion mode [26]. Typically, the smaller the step length the smaller the threshold needs to be. Additionally, the device mode can simplify the model further. For example, if the device is hand held flat, the heading relating to the projection (rotation) to the horizontal plane (heading) can be computed by just integrating the angular rate around the gravity vector.

In addition to PDR, there are numerous other applications that can benefit from this work. The most nearby illustration is the kind of step counters you can find in your smartphones and smart watches, which can differ a factor of two compared in the same situation. Also, the motion mode provided in Android and iOS often fails and can, for instance, decide cycling when walking. Certainly, there is room for improvement in the commercial solutions today.

Major research within the instrumentation and measurement community have been dedicated to the applications of inertial sensors in different areas such as inertial navigation. [8, 11, 14, 15, 27] and behavioral analysis [28, 29]. The main contribution of this work is to shed light on joint device mode and activity mode classification using standard machine learning methods and more advanced neural networks. To be more specific, the contributions of this study can be listed as follows,

- Use accelerometer measurements collected from sensors embedded in almost all recent smartphones. This extends the range of applications that can benefit from the proposed method compared to the similar algorithms tailored for body-mounted sensory data.
- Extract a low dimensional feature vector using raw accelerometer measurements.
- Propose a machine learning framework that can be trained and tested relatively fast and considers the inherent temporal correlation in the human gait in the class estimates.

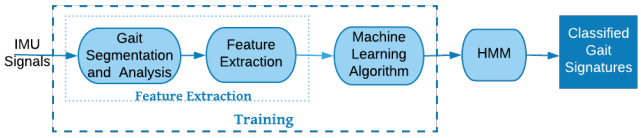


Fig. 1: Overview of the classification process for joint activity and smartphone mode recognition.

Noting that the gaits over time are consistent, we propose a hierarchical algorithm for the joint classification problem. In the first step, we treat each gait independently and identify its corresponding class. Then, a second stage is performed in which we use the HMM to impose the consistency. It is worth noting that recurrent neural network (RNN) also considers the temporal correlation and can be an alternative solution to the considered classification problem. The advantage of the proposed hierarchical model to RNNs is two fold. We note that every second step represents one full gait, and that all gaits over time are ”consistent” This nicely lends itself to the hierarchical solution proposed in this paper where we try to identify each step independently, and then use the HMM to impose the consistency. An RNN cannot capture this hierarchal structure. Additionally, tuning RNNs is more difficult, especially when the dataset is small relative to the length of the sequences as in this case. The methodology used in this paper consists of four main building blocks as presented in Fig. 1, with the following outline:

- The feature extraction phase consists of two building blocks. In the “Gait Segmentation and Analysis” block, the measured IMU signals are pre-processed and fed into the “Feature Extraction” block where unique signatures for each gait mode and device mode are extracted using a linear search optimization algorithm. This phase is the result of our previous work [30] which sets the stage for this contribution.
- The extracted signatures are then used in the “Machine Learning Algorithm” block to train classifiers for multi-class classification. This completes the training phase of the proposed method. The performance at this stage is evaluated using the trained classifier applied to the development and test data.
- Finally, an additional “HMM” block is used and applied to the initial estimated classes. In this application, the hidden states in the HMM block are the activity modes and the device poses, *i.e.*, the classes.

The classified gait cycles reveal which signature the current gait cycle belongs to. This additional information can further be used for gait cycle tuning. Consequently, step lengths can be estimated accurately, resulting in high precision gait cycles. The final result, for example, can be a more accurate PDR algorithm for pedestrian navigation purposes or activity analysis.

The dataset that has been used in this work is publicly available [31]. For more detailed information about the measurement campaign, see [32]. The performance of the proposed classification techniques will be compared to the algorithm suggested in [24].

TABLE I: Notation.

$\boldsymbol{y}(s)$	Measured data with noise
τ	Normalized time $\tau \in [0, 1)$
$\hat{\boldsymbol{g}}_m(\tau)$	m:th gait cycle
$\bar{\boldsymbol{g}}_c(\tau)$	Gait signature for c:th class
T_m	Duration time for m:th gait cycle
$\boldsymbol{X}_m$	Features matrix for m:th gait cycle
$\hat{\mathcal{C}}_m$	Snap-shot class estimate
$\hat{\mathcal{C}}_{m\|m}$	Filtered class estimate
$\hat{\mathcal{C}}_{m\|M}$	Smoothened class estimate
s	Number of IMU samples
M	Number of gait cycles
n_c	Number of classes
n_r	Size of training set
n_d	Size of development set
n_e	Size of test set

TABLE II: Gait cycle classification overview

	Input	Output
Gait Segmentation (Sec. III-A)	$\boldsymbol{y}(t_1{:}t_i)$	$\hat{\boldsymbol{g}}_m(\tau), \bar{\boldsymbol{g}}_c(\tau), T_m$
Feature Extraction (Sec. III-B)	$< \hat{\boldsymbol{g}}_m(\tau), \bar{\boldsymbol{g}}_c(\tau) >$ T_m	$\boldsymbol{X}_m$
Classification (Sec. IV)	$\boldsymbol{X}_m$	$\hat{\mathcal{C}}_m$
Filtering and Smoothing (Sec. V)	$\hat{\mathcal{C}}_m$	$\hat{\mathcal{C}}_{m\|m}, \hat{\mathcal{C}}_{m\|M}$

III. Gait Segmentation and Feature Extraction

The most frequently used notations in this paper are summarized in Table I.

The gait cycle classification procedure, from collecting data using IMUs to detection and classification of gait cycles, is performed in four steps. An overview of all steps is summarized in Table II.

Given N measurements, we first find a segmentation of the signal, each segment to one full gait cycle. It is worth mentioning that a human gait cycle consists of two consecutive steps. Each step is defined as the period it takes for the right/left foot to toe off, while the other leg is on the ground and stationary, until the other foot toes off. This periodic behavior implies that the measured IMU signals from human activities should have a periodic component.

An IMU, in general terms, consists of a tri-axis accelerometer, a tri-axis gyroscope and a tri-axis magnetometer. In the rest of this work, we only use the norm of the accelerometer signals.

A. Gait Segmentation

In order to extract the pattern of the gait cycle with better quality, a pre-processing stage is applied to the raw measured data. In the pre-processing stage, the signal is filtered through a fourth-order Butterworth band pass filter with cut-off frequency $[0.1, 10]$ Hz to attenuate all frequencies outside the band-pass and to obtain a *clean* signal. The cut-off frequencies are selected by noting that, in the considered problem, each gait takes around 1 second, see Fig. 4. Additionally, given our observations, the values above 10 Hz are high frequency measurement noise and not of interest. Since we have the periodicity of around 1 s, that translates into 1 Hz, all values below 0.1 Hz could also be safely removed and not considered. We keep a factor 10 each side to include all important, and drop the rest. Depending on the application, different attributes of the clean signal can be used for classification purposes.

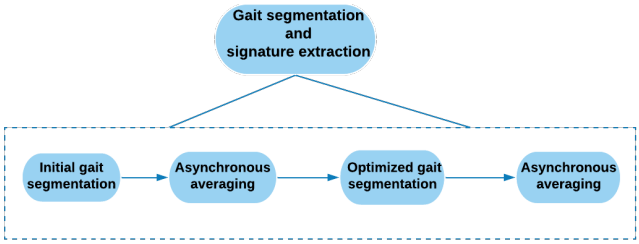


Fig. 2: The flow diagram of the gait signature extraction using norm of pre-processed IMU signals.

TABLE III: Experimental scenarios. For more information, see [32]. (Class label c=1–8 within parenthesis.)

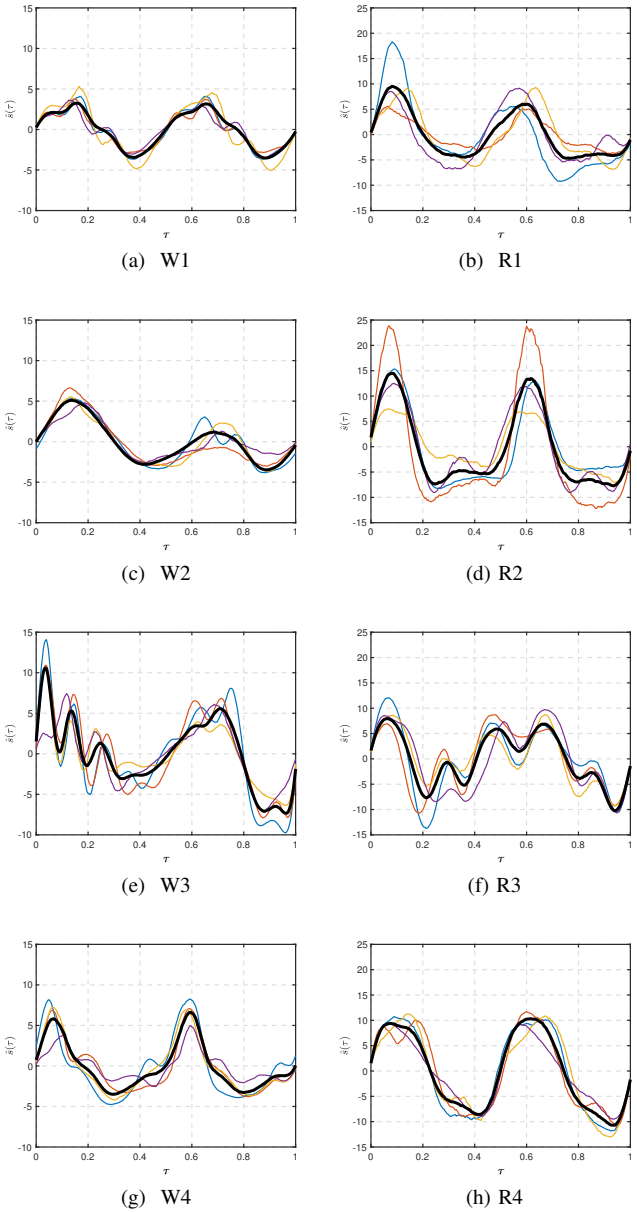
Motion mode / Device mode	Walking (W)	Running (R)
Fixed hand (1)	W1 /(1)	R1 /(5)
Swinging hand (2)	W2 /(2)	R2 /(6)
Pocket (3)	W3 /(3)	R3 /(7)
Backpack (4)	W4 /(4)	R4 /(8)

The gait segmentation and analysis block in Fig. 1 is based on the approach introduced in our previous work [30] and uses the norm of the accelerometer signal to compute gait segmentations. The whole approach is summarized and depicted in Fig. 2.

In the initial gait segmentation block, we first find segmentations of the signal, each corresponding to one full gait cycle using a classical, thresholding-based algorithm as introduced in [30]. For each combination of gait and device mode, a unique signature can be estimated based on all detected gait cycles, $\hat{\boldsymbol{g}}_m(\tau)$ where $m \in \{1, ..., M\}$. In this work, eight different scenarios are considered corresponding to two gait motion patterns and four device modes. Table III summarizes all the considered scenarios, in which $n_c = 8$ classes are defined from the combination of gait and device modes. In each scenario, the motion mode is mentioned first where "W" stands for walking and "R" stands for running. In order to distinguish between different device modes, numbers 1 to 4 are used. For instance, $W1$ means walking with mobile phone fixed in hand while R3 means running with mobile phone in the pocket.

The asynchronous averaging stage is first performed to find an initial guess of the gait signatures using the detected gait cycles. The underlying assumption is that there is an average gait cycle of the form $\bar{\boldsymbol{g}}_c(\tau)$ in normalized time $\tau \in [0, 1)$ for class c. All the detected gait cycles together with initial signatures are then fed into an optimization block where the gait cycles are fine-tuned.

For this purpose, in [30], we propose a nonlinear least squares framework where we optimize the step times in order to minimize the variance of the gait segmentations and the signature. The least squares are formed on a normalized time scale, so small variations in step cycle times are handled by



(a) W1 (b) R1 (c) W2 (d) R2 (e) W3 (f) R3 (g) W4 (h) R4

Fig. 3: Reference signals for four different individuals for all scenarios in Table III. The bold solid line indicates the mean of all four reference signals which is considered as the signature for each case.

resampling techniques as explained in [30]. The optimization problem is solved using a linear search method. The signature will then be updated before finding the next, fine-tuned, gait segment. More details are provided in [30].

The gait signatures obtained by applying this method to the experimental data of four individuals, explained in detail in Sec. VI, are given in Fig. 3. Different colors in this figure are used to represent different individuals. For instance, in Fig. 3a, four different individuals performed the class "W1" (walking while phone is fixed in hand). The signal marked with thick black solid line is the average of all individuals. As the figures suggest, each class can be represented using a unique signature.

TABLE IV: Correlation matrix for different signatures.

Classes	W1	W2	W3	W4	R1	R2	R3	R4
W1	1.00	0.72	0.78	0.67	0.80	0.78	0.62	0.95
W2	0.72	1.00	0.57	0.27	0.52	0.48	0.30	0.65
W3	0.78	0.57	1.00	0.55	0.62	0.62	0.71	0.77
W4	0.67	0.27	0.55	1.00	0.92	0.92	0.59	0.81
R1	0.80	0.52	0.62	0.92	1.00	0.91	0.66	0.85
R2	0.78	0.48	0.62	0.92	0.91	1.00	0.70	0.89
R3	0.62	0.30	0.71	0.59	0.66	0.70	1.00	0.66
R4	0.95	0.65	0.77	0.81	0.85	0.89	0.66	1.00

B. Feature Extraction

In order to train the classifiers in this work, we use two novel characteristics of the detected gait cycles; the 8-dimensional correlation scores of each detected gait cycle relative to the gait signatures and the 1-dimensional gait cycle duration time. All in all, in this work, 9 features, $n_f = 9$, are considered for classification purposes.

1) Correlation Score: The correlation scores are defined based on eight unique signatures of the reference signals corresponding to the scenarios introduced in Table III. Define $\hat{\boldsymbol{g}}_m(\tau)$, as the m:th detected gait cycle and $\bar{\boldsymbol{g}}_c(\tau), c \in \{1, \ldots, n_c\}$, as the signature of the c:th class. The correlation score can be computed by

$$f_{\text{corr}}(\hat{\boldsymbol{g}}_m, \bar{\boldsymbol{g}}_c) = \frac{\hat{\boldsymbol{g}}_m^\top \bar{\boldsymbol{g}}_c}{\sqrt{(\hat{\boldsymbol{g}}_m^\top \hat{\boldsymbol{g}}_m)(\bar{\boldsymbol{g}}_c^\top \bar{\boldsymbol{g}}_c)}}. \tag{1}$$

In order to classify the gait cycles, we compute the correlation between each gait cycle m, and the 8 signatures. The obtained correlation value, corresponding to the m:th gait cycle and the signature in the c:th class, is then used as a feature in multiple classification algorithms. Given that, in this work, 8 different classes are considered, the correlation score results in an 8-dimensional feature vector, $n_f^{corr} = 8$. One way to evaluate the uniqueness of the gait signatures is to compute their cross-correlations with each other. The correlation matrix corresponding to the correlation scores of the gait signatures are given in Table IV. As the table indicates, the cross-correlation values typically lie in the range $[0.6, 0.85]$ while there are some classes with more similar signatures. Hence, the correlation score has a great potential in forming feature vectors to be used in developing classifiers.

For example, as the simplest solution, one can solve the classification problem solely based on the highest correlation score. Let $\boldsymbol{X}_m^\gamma = f_{\text{corr}}(\hat{\boldsymbol{g}}_m, \bar{\boldsymbol{g}}_c) \in \mathbb{R}^{n_f^{corr}}$ denote the feature vector containing the correlation scores computed by (1) for the m:th gait cycle and the signature of the c:th class. Using the highest correlation score approach, the class of the m:th gait cycle is estimated as

$$\hat{\mathcal{C}}_m = \underset{\gamma \in \{1, \ldots, n_f^{corr}\}}{\arg\max} \boldsymbol{X}_m^\gamma, \tag{2}$$

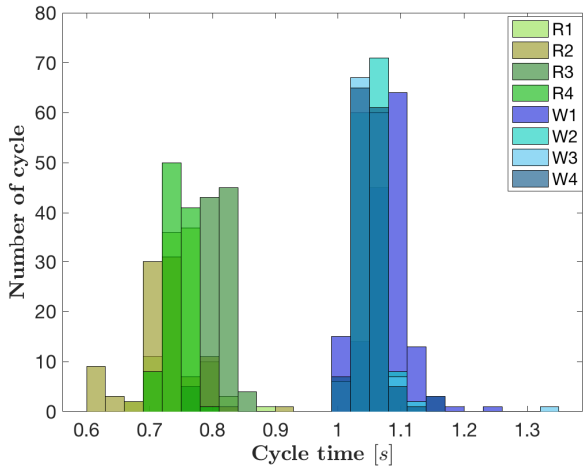


Fig. 4: Histogram of gait cycle time for running and walking modes.

where $\hat{\mathcal{C}}_m \in \{1, \ldots, n_c\}$ is the index indicating that the m:th gait cycle belongs to the c:th class.

2) Gait cycle time: Different gait modes, walking and running, result in different gait cycle times. In order to consider the extra information obtained from the user's velocity, the gait cycle time is also considered as a feature, $\boldsymbol{X}_m^T$, in the classifiers. Although the gait cycle time is not of any use for device mode identification, it gives a clear decision boundary for the two motion modes.

Fig. 4 shows histograms of all 8 signatures proving that the gait modes can be trivially identified by considering the gait cycle time. In the case of walking mode, the gait cycle time is typically between 1 and 1.2 s, while for the running mode the cycle time interval mostly lies between 0.6 and 0.85 s. Hence, this feature is very beneficial to minimize the misclassification between gait modes.

C. Generating the dataset

The introduced 9-dimensional feature vector containing the correlation scores for all gait cycles, $\boldsymbol{X}^\gamma \in \mathbb{R}^{M \times n_f^{corr}}$, and the gait cycle times, $\boldsymbol{X}^T \in \mathbb{R}^{M \times n_f^T}$, are stacked together to make the input matrix $\boldsymbol{X} \in \mathbb{R}^{M \times n_f}$. In this work, we divide the data into *training set* of size n_r, *development set* n_d, and *test set* n_e with $n_d = n_e = \frac{M - n_r}{2}$.

It is worth noting that the development set is used to cross-validate the trained algorithms using classifiers introduced in Sec. IV in order to avoid overfitting, underfitting or any other penalties that might increase the classification error of the test set. The hyperparameters of the algorithms are all selected such that the development set error is minimized. Then, as the final step, the trained classifiers are applied to the test set (the part of the dataset which has not been seen during the training procedure) and the performance of the proposed algorithms is reported.

IV. Classification Methods

In the classification methods proposed in this section, the n_r training examples together with the development set are used to train multiple classifiers. The final result is then obtained by feeding the output of the classifier into an HMM block to exploit the temporal correlation between two consecutive time instances.

A. Weighted K-Nearest Neighbor

The K-nearest neighbor is a simple, non-parametric classifier which computes the distance between the test (unseen) input and the training data points. As distance to compute neighbors we use the classical Euclidean distance. Selecting the K training points with the closest distance to the input, the algorithm counts how many members of each class are in this set.

Let $\boldsymbol{X}_r$ be the r:th training example in the whole training set $\mathcal{T}$. For a new gait cycle in the development set, $\hat{\boldsymbol{g}}_d$, with feature vector $\boldsymbol{X}_d$, the KNN classifier estimates the membership probability as the empirical fraction given by

$$\begin{aligned}\hat{\mathcal{C}}_d =& p(\hat{\boldsymbol{g}}_d \in c | \boldsymbol{X}_d, \mathcal{T}, K) \\ =& \frac{1}{K} \sum_{r \in N_K(\boldsymbol{X}_d, \mathcal{T})} \mathbb{I}(\boldsymbol{X}_r \in c),\end{aligned} \tag{3}$$

where $N_K(\boldsymbol{X}_d, \mathcal{T})$ are the indices of the K nearest points to $\boldsymbol{X}_d$ in the training set $\mathcal{T}$. $\mathbb{I}(\cdot)$ is the indicator function defined as

$$\mathbb{I}(v) = \begin{cases} 1 & \text{if } v \text{ is true} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The weighted-KNN is an extension of the KNN which gives higher weights to the classes in the training set which are particularly close to the new input $\boldsymbol{X}_d$ compared to the rest of the neighbors further away from $\boldsymbol{X}_d$. Let $w_{d,r}$ denote the weight assigned to each training example r, the membership probability can be computed as

$$\hat{\mathcal{C}}_d = \frac{\sum_{r \in N_K(\boldsymbol{X}_d, \mathcal{T})} w_{d,r} \times \mathbb{I}(\boldsymbol{X}_r \in c)}{\sum_{r \in N_K(\boldsymbol{X}_d, \mathcal{T})} w_{d,r}}. \tag{5}$$

The weighting scheme used here is the squared inverse Euclidean distance which gives each neighbor a weight of $w_{d,r} = 1/{D_r}^2$, where D_r is the Euclidean distance to the r:th neighbor.

B. Gaussian Process

A random process $q(\cdot)$ is Gaussian if

$$\forall n \in \mathbb{N}, \quad \forall \delta_1, \ldots, \delta_n \in \mathbb{R}^n:$$
$$\begin{pmatrix} q(\delta_1) \\ q(\delta_2) \\ \vdots \\ q(\delta_n) \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mu(\delta_1) \\ \mu(\delta_2) \\ \vdots \\ \mu(\delta_n) \end{pmatrix}, \begin{pmatrix} \kappa(\delta_1, \delta_1) & \ldots & \kappa(\delta_1, \delta_n) \\ \kappa(\delta_2, \delta_1) & \ldots & \kappa(\delta_2, \delta_n) \\ \vdots & \ddots & \vdots \\ \kappa(\delta_n, \delta_1) & \ldots & \kappa(\delta_n, \delta_n) \end{pmatrix} \right), \tag{6}$$

where for two random variables δ and δ', $\mu(\delta) = E[q(\delta)]$ is the mean function $\kappa(\delta, \delta') = E\left[(q(\delta) - \mu(\delta))\,(q(\delta') - \mu(\delta'))\right]$ is the positive definite covariance function. Let $\boldsymbol{\delta} = [\delta_1, \ldots, \delta_n] \in \mathbb{R}^n$, we get

$$q(\boldsymbol{\delta}) \sim \mathcal{GP}(\mu(\boldsymbol{\delta}), \kappa(\delta, \delta')), \tag{7}$$

where the notation $\mathcal{GP}(\cdot,\cdot)$ denotes a Gaussian process. Given the training set $\mathcal{T}$ containing input vectors and their corresponding class labels, $\{\boldsymbol{X}_r, c_r\}_{r=1}^{n_r}$, the Gaussian process classifier predicts the class membership probability for a new, unseen development point $\boldsymbol{X}_d$. If the function values $q^{(r)} \triangleq q(\boldsymbol{X}_r)$ were known, it would then be possible to use a GP for regression to find $q(\boldsymbol{X}_d)$ and then map it into the interval $[0 \quad 1]$ by means of the softmax function. However, the training data only includes the class labels c_r.

To solve this problem, as discussed in [33], we introduce the latent function values at all n_r training examples, for all classes

$$q^{(r)} \triangleq \boldsymbol{q} = (q_1^{(1)}, \ldots, q_1^{(n_r)}, q_2^{(1)}, \ldots, q_2^{(n_r)}, \ldots, q_8^{(1)}, \ldots, q_8^{(n_r)})^\top, \quad r = 1, \ldots, n_r. \tag{8}$$

Define $\mathcal{C}$ as a vector of the same length as $\boldsymbol{q}$. For each training example, the value of $\mathcal{C}$ for the true class label is 1 and for the rest of the classes is 0. The distribution of the latent variable q_d, corresponding to the development point $\boldsymbol{X}_d$, can then be computed by

$$p(q_d \mid \boldsymbol{X}_d, \mathcal{T}, \mathcal{C}) = \int p(q_d, \boldsymbol{q} | \boldsymbol{X}_d, \mathcal{T}, \boldsymbol{q}) = \int p(q_d | \boldsymbol{X}_d, \mathcal{T}, \boldsymbol{q}) p(\boldsymbol{q} | \mathcal{T}, \mathcal{C})\, d\boldsymbol{q}, \tag{9a}$$

which is further used to compute the class membership distribution

$$\hat{\mathcal{C}}_d = p(\hat{\boldsymbol{g}}_d = c | \boldsymbol{X}_d, \mathcal{T}, \mathcal{C}) = \int \sigma(q_d) p(q_d \mid \boldsymbol{X}_d, \mathcal{T}, \mathcal{C})\, dq_d, \tag{9b}$$

where $\sigma(q_d)$ is the softmax function, which for any vector $\varphi \in \mathbb{R}^{n_\varphi, 1}$ is given by

$$\sigma(\varphi)_j = \frac{\exp(\varphi_j)}{\sum_{i=1}^{n_\varphi} \exp(\varphi_i)}. \tag{10}$$

The non-Gaussian likelihood in (9a) makes the integral analytically intractable. One approach is to use Laplace approximation [33, 34] to form $p(\hat{\boldsymbol{q}}|\mathcal{T}, \mathcal{C}) \approx \mathcal{N}(\hat{\boldsymbol{q}}, \boldsymbol{A}^{-1})$ with the statistics estimated as

$$\hat{\boldsymbol{q}} = \arg\max_{\boldsymbol{q}} p(\boldsymbol{q}|\mathcal{T}, \mathcal{C}), \tag{11a}$$

$$\boldsymbol{A} = -\nabla\nabla \log p(\boldsymbol{q}|\mathcal{T}, \mathcal{C}) \mid_{\boldsymbol{q}=\hat{\boldsymbol{q}}}. \tag{11b}$$

In this work, the mean function $\mu(\cdot)$ is assumed to be constant, corresponding to stationary GP, and set to zero. The covariance function is the squared exponential (SE) kernel, $\mathcal{K}_{SE}$, which for two arbitrary points, δ and δ', is given by

$$\mathcal{K}_{\mathrm{SE}}(\delta, \delta') = \sigma_q^2 \exp\{-\frac{D_\delta^2}{2\Gamma^2}\}, \tag{12}$$

where $D_\delta = \|\delta - \delta'\|$ is a function of input training points and Γ and σ_q are the hyper parameters. While σ_q controls the amount of deviation around the posterior mean at each point, Γ specifies the smoothness of the kernel function.

C. Feed Forward Neural Network

The last classifier considered in this work is based on feed forward neural networks, trained using the n_r training examples. A rather shallow, fully connected neural network is developed which outputs probabilities of class memberships given the 9-dimensional input vectors computed using the softmax function.

Let $\ell \in [1, \ldots, L]$ denote the number of layers, including hidden layers and the output layer, and $n^{[\ell]}$ denote the number of hidden units, or neurons, in the ℓ:th layer of the neural network. The fully connected NN structure is given in Fig. 5 in which the output layer, corresponding to the classes defined in Table III, is a vector of size $n_c = 8$.

For consistency in the derivations that follows, assume that $\ell = 0$ corresponds to the input layer, *i.e.* $\mathbf{X} = a^{[0]}$. Then, the output of each hidden layer, denoted by $a^{[\ell]}$, can be computed by, a linear transformation of the previous layer's output $a^{[\ell-1]}$ followed by a nonlinear transformation using an activation function denoted by $u^{[\ell]}(\cdot)$

$$z^{[\ell]} = w^{[\ell]} a^{[\ell-1]} + b^{[\ell]}, \tag{13a}$$

$$a^{[\ell]} = u^{[\ell]}(z^{[\ell]}), \tag{13b}$$

where $w^{[\ell]} \in \mathbb{R}^{n^{[\ell]} \times n^{[\ell-1]}}$ is the weight matrix and $b^{[\ell]} \in \mathbb{R}^{n^{[\ell]} \times 1}$ is the bias vector of the ℓ:th layer. To better explain the procedure, Fig. 6 illustrates the linear and nonlinear transformations of a single unit of the ℓ:th layer in which $w_{n^{[\ell]}}^{[\ell]}$ denotes the weight vector and $b_{n^{[\ell]}}^{[\ell]}$ denotes the bias scalar corresponding to the $n^{[\ell]}$:th hidden unit. In this work, we use the rectified linear unit (ReLU) as the activation functions for all the hidden layers. For any value $\phi \in \mathbb{R}$, the ReLU activation function is given by

$$u_{\mathrm{relu}}(\phi) = \max(0, \phi). \tag{14}$$

Upon taking the path through all hidden layers $\ell \in [1, \ldots L-1]$, the output of the NN classifier is computed by first passing $a^{[L-1]}$ through the same linear function (13a) to find $z^{[L]}$. A softmax regression layer is then governed to estimate the class membership probabilities $a^{[L]} = \hat{\mathcal{C}} = \sigma(z^{[L]})$. The unknown weight and bias parameters are learned using the back propagation technique.

After one full forward propagation, in the backward path, an optimization algorithm is employed to find the optimal value of the unknown parameters $w^{[\ell]}$ and $b^{[\ell]}$, $\ell \in [1, \ldots, L]$ for a given loss function. A softmax classifier is typically trained using the cross-entropy loss function. Given the true classes vector $\mathcal{C}_r \in \mathbb{R}^8$, one-hot encoded vector, with 1 at the true class and zeros elsewhere, and the estimated class vector $\hat{\mathcal{C}}_r$, the loss function, for each training example is given by

$$\mathcal{L}(\hat{\mathcal{C}}_r, \mathcal{C}_r) = -\sum_{c=1}^{8} \mathcal{C}_c \log(\hat{\mathcal{C}}_c). \tag{15a}$$

Let $\mathcal{L}(\hat{\mathcal{C}}_{\mathrm{r}}, \mathcal{C}_r)$ denote the loss function for all the unknown parameters $\theta = \{w_{n^\ell}^{[\ell]}, b_{n^{[\ell]}}^{[\ell]}\}_{\ell=1}^{L}$. The cost function, given all

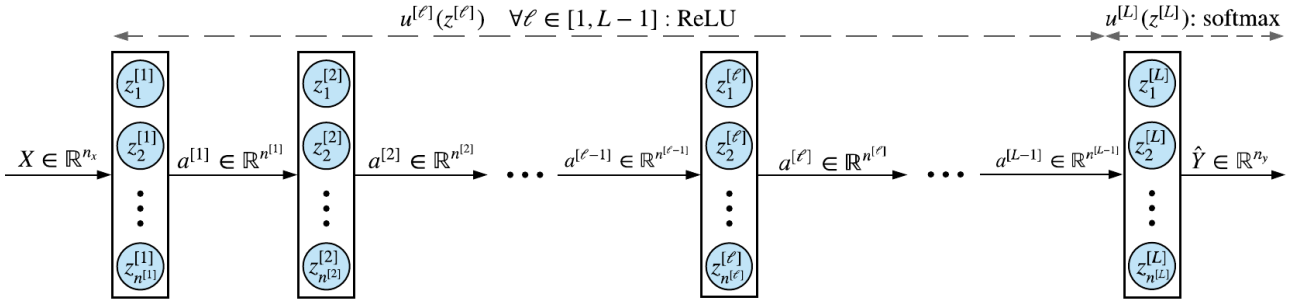


Fig. 5: The L-layer feed forward neural network structure with $n^{[\ell]}$ number of hidden units in each hidden layer $\ell, \ell \in 1, \ldots, L-1$ and $n_x = 8$ input and $n_c = 8$ output layers. The activation functions $u^{[\ell]}(z^{[\ell]})$ in all of the hidden layers are ReLU and the activation function in the output layer $u^{[L]}(z^{[L]})$ is softmax.

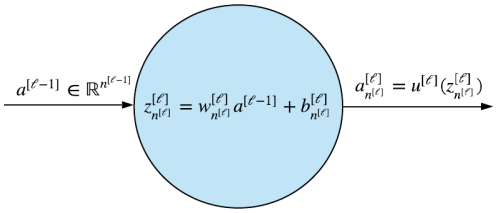


Fig. 6: Input-output relationship in the $n^{[\ell]}$:th hidden unit of layer ℓ.

of the m training examples, can then be obtained by

$$J(\theta) = \frac{1}{m} \sum_{\tau=1}^{m} \mathcal{L}(\hat{\mathcal{C}}_r, \mathcal{C}_r) + \frac{\lambda}{2m} \sum_{l=1}^{L} \|w^{[l]}\|_F^2, \tag{15b}$$

where $\| \cdot \|_F$ operator denotes the Frobenius norm of the matrix. The second term on the right hand side of (15b) is the regularization term added to deal with the overfitting problem.

In this work, we use adaptive learning rate optimization (Adam) first introduced in [35] to optimize the cost (15b). The Adam optimization combines the benefits of two optimization methods: gradient descent with momentum [36] and root mean squared prop (RMSprop). The RMSprop is an unpublished adaptive learning rate method proposed by Geoff Hinton. The Adam method, in addition to storing an exponentially decaying average of past squared gradients as in RMSprop, also keeps an exponentially decaying average of past gradients, similar to momentum.

In addition to the regularization procedure, the proposed method also employs dropout [37] to avoid overfitting the data. For this purpose, during the training phase, 5% of the hidden units are ignored. The hidden units to be omitted are randomly selected at each epoch.

V. Hidden Markov Models

For the classification problem in hand, we note that there is a high probability that the activity profile and the device mode will remain constant between two consecutive gait cycles. This implies that there is a high temporal correlation between the classes. This additional information can improve the classification rate noticeably and can be modeled using a discrete time, discrete state hidden Markov model.

Let $\boldsymbol{X}_e$, $e \in \{0, \ldots, n_e\}$, denote the correlation vector corresponding to the e:th test data where $n_e = \frac{M-n_r}{2}$. Similarly, $\hat{\mathcal{C}}_e$ contains the estimated class membership of the test data samples estimated by any of the classifiers introduced in Sec. IV. It is worth noting that HMM is only applied to the test set. For the sake of simplicity in the notation, we remove the subscript e; hence, in what follows, by the m:th gait cycle, we mean the m:th gait cycle in the test set data.

In order to capture the temporal correlation between the classes, as shown in Fig. 7, the estimated class membership vectors $\hat{\mathcal{C}}_m$ for gait cycle m are fed into an HMM block whose output is the final class membership estimates $\hat{\mathcal{C}}_{m|m}$ for filtering and $\hat{\mathcal{C}}_{m|M}$ for smoothing approaches.

Relying on the first-order Markov assumption, HMM indicates that the current state of the system depends on its previous states. We let the classification vector $\hat{\mathcal{C}}_{m|M}$ denote the *hidden states* of the HMM and define the *observation* vector at each time instance based on the classifier output $\hat{\mathcal{C}}_m$. The joint distribution of the observation and hidden states takes the form

$$\begin{aligned} p(\hat{\mathcal{C}}_{0:M|0:M}, \hat{\mathcal{C}}_{0:M}) &= p(\hat{\mathcal{C}}_{0:M|0:M}) p(\hat{\mathcal{C}}_{0:M} \mid \hat{\mathcal{C}}_{0:M|0:M}) \\ &= \left[p(\hat{\mathcal{C}}_{0|0}) \prod_{m=1}^{M-1} p(\hat{\mathcal{C}}_{m+1|m} | \hat{\mathcal{C}}_{m|m}) \right] \left[\prod_{m=0}^{M} p(\hat{\mathcal{C}}_m | \hat{\mathcal{C}}_{m|m}) \right], \end{aligned} \tag{16}$$

where $p(\hat{\mathcal{C}}_m | \hat{\mathcal{C}}_{m|m})$ is the observation (emission) model and $p(\hat{\mathcal{C}}_{m+1|m} | \hat{\mathcal{C}}_{m|m})$ is the transition model. The filtering, $p(\hat{\mathcal{C}}_{m|m} | \hat{\mathcal{C}}_{0:m})$, and smoothing, $p(\hat{\mathcal{C}}_{m|m} | \hat{\mathcal{C}}_{0:M})$, distributions can be computed using the Baum-Welch forward-backward (FB) algorithm [38]. Using the FB algorithm, the filtering and smoothing distributions can be computed separately by defining two independent values $\alpha(\hat{\mathcal{C}}_{m|m})$ and $\beta(\hat{\mathcal{C}}_{m|m})$. Here, we consider a discrete-time, first-order HMM model in which the hidden state $\hat{\mathcal{C}}_{m|m}$ depends only on its one step predecessor. Applying the recursive forward-backward algorithm [38] to the considered discrete-time, first-order, HMM model, the marginal distributions can be computed by

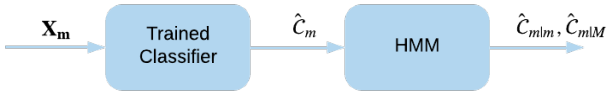


Fig. 7: The output of the trained neural network, $\hat{\mathcal{C}}_m$, is smoothed using an additional HMM block providing a final estimate of the classes, $\hat{\mathcal{C}}_{m|M}$.

- Filtering:

$$p(\hat{\mathcal{C}}_{m|m}|\hat{\mathcal{C}}_{0:m}) = \frac{\alpha(\hat{\mathcal{C}}_{m|m})}{\sum_{\hat{\mathcal{C}}_{m|m}} \alpha(\hat{\mathcal{C}}_{m|m})}, \tag{17}$$

where $\alpha(\hat{\mathcal{C}}_{m|m})$ is defined as

$$\alpha(\hat{\mathcal{C}}_{m|m}) = p(\hat{\mathcal{C}}_m \mid \hat{\mathcal{C}}_{m|m}) \times \sum_{\hat{\mathcal{C}}_{m-1|m}} \alpha(\hat{\mathcal{C}}_{m-1|m}) p(\hat{\mathcal{C}}_{m|m} \mid \hat{\mathcal{C}}_{m-1|m}), \tag{18}$$

and initialized at $\alpha(\hat{\mathcal{C}}_{0|0}) = p(\hat{\mathcal{C}}_0 \mid \hat{\mathcal{C}}_{0|0}) p(\hat{\mathcal{C}}_{0|0})$.

- Smoothing:

$$p(\hat{\mathcal{C}}_{m|m}|\hat{\mathcal{C}}_{0:M}) = \frac{\alpha(\hat{\mathcal{C}}_m)\beta(\hat{\mathcal{C}}_{m|m})}{\sum_{\hat{\mathcal{C}}_{m|m}} \alpha(\hat{\mathcal{C}}_{m|m})\beta(\hat{\mathcal{C}}_{m|m})}, \tag{19}$$

where $\beta(\hat{\mathcal{C}}_{m|m})$ is

$$\beta(\hat{\mathcal{C}}_{m|m}) = \sum_{\hat{\mathcal{C}}_{m+1|m}} \beta(\hat{\mathcal{C}}_{m+1|m}) p(\hat{\mathcal{C}}_{m+1} \mid \hat{\mathcal{C}}_{m+1|m}) \times p(\hat{\mathcal{C}}_{m+1|m} \mid \hat{\mathcal{C}}_{m|m}), \tag{20}$$

and initialized at $\beta(\hat{\mathcal{C}}_{M|M}) = 1$.

The emission model is given by the confusion matrix presented in Table IV. Additionally, we assume that the states are highly likely to remain in the same regime between the consecutive gait cycles $m-1$ and m. This can be captured in the *transition matrix* $p(\hat{\mathcal{C}}_{m|m} \mid \hat{\mathcal{C}}_{m-1|m}) = \pi_{ij}$ that gives the probability of remaining in the same regime or switching to another. The elements of the transition matrix are given by

$$\pi_{ij} = \begin{cases} 0.99 & \text{if } i{=}j \\ \frac{0.01}{n_c - 1} & \text{otherwise.} \end{cases} \tag{21}$$

VI. Experimental results

In order to evaluate the performance of the proposed method, an extensive dataset with different human motion modes and device poses is considered [32].

A total number of 8900 gait cycles were collected, in two different campaigns, 80% of which were chosen for training. Since all the measurements were collected using the same device, they belong to the same underlying distribution. This allowed us to use the remaining 20% to extract equally-sized development and test sets. The first set of measurements was collected in a building at Twente University. During the experiments the subjects walked three different paths, with a mixture of different motion modes. For more details, see [39].

TABLE V: Initial classification confusion matrix corresponding to the KNN classifier.

Classes	W1	W2	W3	W4	R1	R2	R3	R4
W1	68	9	3	16	1	0	0	0
W2	6	81	1	3	0	0	0	0
W3	6	3	94	2	0	0	0	0
W4	20	6	2	79	0	0	0	0
R1	0	0	0	0	74	10	0	12
R2	0	1	0	0	8	79	1	7
R3	0	0	0	0	0	1	96	1
R4	0	0	0	0	17	10	3	80

The second campaign was performed over the same trajectory, which was 249 m in length with four sharp corners in a parking lot at Linköping University. Several subjects with different attributes (gender, height and weight) participated in the experiment.

Expectedly, the most time-consuming algorithm, among the considered classifiers, is the one based on neural network. It is worth noting that since the network is rather shallow and the feature vector is low-dimensional, training the neural network classifier took about an hour on a MacBook Pro with 2.3 GHz Intel Core i5 processor. Testing is quite fast and took less than a minute to test the whole test dataset on the same machine. Given a new test input, one forward path of the whole algorithm using any of the classifiers, including the neural network, is fast and can be used in real-time applications.

TABLE VI: Initial classification confusion matrix corresponding to the GP classifier.

Classes	W1	W2	W3	W4	R1	R2	R3	R4
W1	67	7	3	9	2	0	0	0
W2	6	86	0	3	0	0	0	0
W3	2	3	95	3	0	0	0	0
W4	21	4	1	85	0	1	0	0
R1	0	0	0	0	71	7	0	12
R2	0	1	0	0	18	81	1	9
R3	0	0	0	0	0	0	97	3
R4	0	0	0	0	9	11	2	76

TABLE VII: Initial classification confusion matrix corresponding to the proposed neural network classifier.

Classes	W1	W2	W3	W4	R1	R2	R3	R4
W1	88	3	0	12	0	0	0	0
W2	3	95	1	12	0	0	0	0
W3	1	0	99	0	0	0	0	0
W4	7	2	0	86	0	0	0	0
R1	0	0	0	0	84	7	0	11
R2	1	0	0	0	3	92	3	3
R3	0	0	0	0	0	0	97	0
R4	0	0	0	0	13	1	0	86

In the rest of this section, we first evaluate the initial classification performance where the temporal coherence in the consecutive classes are not considered. Then, the final classification performances are provided in which the HMM smoother is adapted to smooth the class probability estimations. Performance is evaluated in terms of the confusion matrix, classification accuracy and the F_1 score.

Let n_{tp} and n_{tn} denote the number of true positives and true negatives, respectively. Assuming that the total number of class memberships in the test set is given by the cardinality $|\mathcal{C}_{\text{test}}|$, the accuracy is given by $n_{tp} + n_{tn}/|\mathcal{C}_{\text{test}}|$. In order to

compute the F_1 score, we first find the per-class scores, F_1^c, and then report their average over the 8 classes.

Define n_{tp}^c and n_{fp}^c as the number of true positives and false positives corresponding to the class $c \in [1, \ldots, 8]$, respectively. The per-class precision, p^c, which is the ratio of true positives to those predicted positive, and the recall, ν^c, which is the ratio of true positives to all positives, are given by

$$p^c = \frac{n_{tp}^c}{n_{tp}^c + n_{fp}^c}, \tag{22a}$$

$$\nu^c = \frac{n_{tp}^c}{n_{tp}^c + n_{fn}^c}, \tag{22b}$$

where n_{fn}^c is the number of false negatives corresponding to class c. The per-class F_1 score that weights precision and recall equally is given by

$$F_1^c = \frac{2p^c\nu^c}{p^c + \nu^c}. \tag{22c}$$

A. Initial Classification

The KNN and GP classifiers were trained using 5-fold cross validation using the training and development datasets and their hyperparameters are hand-tuned. For each classifier, different hyperparameters, selected over a grid, were evaluated and their performances were compared. The best results using the KNN classifier was obtained when 10 nearest neighbors were considered. Table V presents the confusion matrix computed for the initial KNN classifier relative to the true class labels $\mathcal{C}_e$. The classification accuracy obtained using the KNN method is 82% and the F_1 score is 0.79.

The trained GP classifier is based on the squared exponential kernel whose hyperparameters are $\sigma_f = 1$ and $l = 1$. The initial classification confusion matrix using GP is provided in Table VI. As expected, GP improves the classification performance and reaches a classification accuracy of 83% and F_1 score of 0.80.

In order to tune the neural network, we first ignore the regularization term in the cost function (15b) as well as the dropout technique. The main objective, at this stage, is to design a neural network with low bias, hence low training error rate, without considering the overfitting problem, denoted by the variance. The hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$ of the Adam optimization algorithm were chosen as recommended by [35]. Then, the dropout and regularization methods are activated to reduce the variance between the train and test error rates.

In order to find the number of hidden layers and hidden units in each layer, a range of different candidates was evaluated. The best results were achieved using a 4-layer neural network that can be defined by a $9-450-250-150-8$ structure. All the hyperparameters of the developed NN are summarized in Table VIII.

The confusion matrix corresponding to the proposed NN initial classification is reported in Table VII. The performance metrics are improved compared to both KNN and GP classifiers with a classification accuracy of 91% and F_1 score of 0.89. Using mini-batches of size 256, the cost values during the whole training phase are also presented in Fig. 8. Although the cost function has a decreasing trend over the whole training phase, as a result of using mini-batch optimization, during each epoch, the cost values might even increase for some mini-batches.

TABLE VIII: Hyperparameters of the proposed 4-layer neural network, with 8 units in the output layer.

Hyper parameter	β_1	β_2	ϵ	λ	α	Mini-batch size	Epochs	Units per hidden layer
Values	0.9	0.999	10^{-8}	0.1	$2e^{-3}$	256	500	$n^{[1]} = 450$ $n^{[2]} = 250$ $n^{[3]} = 150$

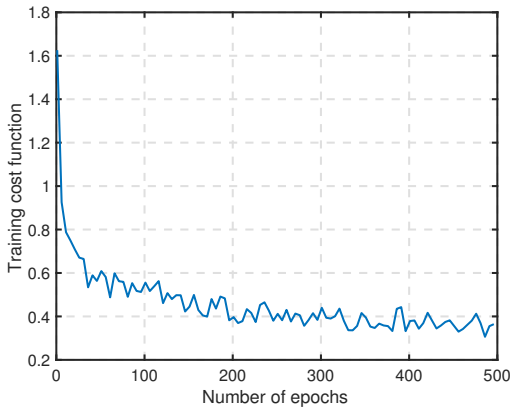


Fig. 8: The training costs of the proposed neural network, obtained during training using Adam optimizer and back propagation technique.

As a result of governing the gait cycle time feature, even the simplest KNN classifier is able to find a very good decision boundary between the two gait modes. However, this situation does not hold for the device modes within each of the two gait modes and the initial class estimates of the classifiers have different degrees of misclassifications. The W3 and R3 classes, corresponding to walking and running with the device in pocket, are accurately estimated using all three methods. The accurate class estimates can be explained by considering the unique gait signatures of the W3 and R3 classes given in Fig. 3. R1 and R4, on the other hand, are the most problematic classes, for which even the complex NN classifier results in a misclassification rate of around 13%.

The initial classification results validate the correlation scores presented in Table IV. The less correlated classes are identified by all the three classifiers with an acceptable accuracy. The additional gait cycle time feature, results in a clear decision boundary between the two motion modes. Correlated gait signatures within each motion mode, however, affect the classification accuracy negatively.

The W1 class, for instance, is one of the most problematic cases with less than 70% initial classification accuracy with both GP and KNN and less than 90% initial classification accuracy using the proposed neural network classifier. In the case of running motion mode, R1 and R4 are the most challenging cases with the highest missclassification rate over the running experiments. In the next section, we try to mitigate a portion of such misclassifications using the HMM smoothing

TABLE IX: Filtered classification confusion matrix corresponding to the GP classifier.

Classes	W1	W2	W3	W4	R1	R2	R3	R4
W1	76	10	2	10	7	0	0	0
W2	3	84	0	0	0	0	0	1
W3	3	5	93	4	0	0	0	0
W4	17	0	2	87	0	2	0	0
R1	0	0	0	0	84	1	0	14
R2	0	1	0	0	7	95	0	10
R3	0	0	3	0	0	1	97	2
R4	0	0	0	0	3	1	3	73

TABLE X: Smoothened classification confusion matrix corresponding to the GP classifier.

Classes	W1	W2	W3	W4	R1	R2	R3	R4
W1	86	5	2	3	6	0	0	0
W2	0	91	0	0	0	0	0	0
W3	2	4	97	1	0	0	0	0
W4	11	0	0	96	0	1	0	0
R1	0	0	0	0	94	2	0	16
R2	0	0	0	0	0	97	1	8
R3	0	0	1	0	0	0	99	3
R4	0	0	0	0	0	0	0	73

TABLE XI: Filtered classification confusion matrix corresponding to the proposed NN classifier.

Classes	W1	W2	W3	W4	R1	R2	R3	R4
W1	94	1	0	6	0	0	0	0
W2	3	96	0	0	0	0	0	0
W3	0	2	100	0	0	0	0	0
W4	4	0	0	94	0	0	5	0
R1	0	0	0	0	94	2	0	8
R2	0	0	0	0	6	94	1	0
R3	0	0	0	0	0	0	94	0
R4	0	0	0	0	0	4	0	92

stage.

B. Filtering and Smoothing Class Estimates

The outputs $\hat{\mathcal{C}}_m$ of all the classifiers are further processed and the filtering and smoothing estimates are evaluated using the test data. The hyperparameter of the HMM is hand-tuned in this work. The fine-tuned estimated classes $\hat{\mathcal{C}}_m$, provided by the HMM block, are then compared to the true class labels and the same performance metrics as in the previous section are re-calculated. As expected, the KNN method has the poorest performance among the rest with a smoothing classification accuracy of around 86%, which is less than the initial estimates obtained from the NN classifier. The confusion matrices for the filtered and the smoothed class estimates of the KNN method are not provided due to space limitations.

The GP classifier's accuracy improves from 83%, obtained from the initial estimates, to around 85% for filtering and 90% in the case of smoothing estimations. Hence, around 7% classification accuracy can be gained by considering the temporal correlation in the GP classifier. The confusion matrices, for both filtering and smoothing, are given in Tables IX and X. Although the gained accuracy is non-negligible, the final GP-based classifier still has a high rate of misclassification between fixed in hand and backpack device modes for both gait modes, R1-R4 and W1-W4.

The final classification results obtained from the neural network classifier indicate that the uncertainty in highly correlated classes such as W1 and R1 can be resolved fairly well. The worst scenario, in terms of classification performance, corresponds to R4 with 92% percent accuracy for both filtering and smoothing. The W1 scenario, which was the worst class in walking motion mode, can now be identified accurately.

TABLE XII: Smoothened classification confusion matrix corresponding to the proposed NN classifier.

Classes	W1	W2	W3	W4	R1	R2	R3	R4
W1	98	0	0	0	0	0	0	0
W2	2	100	0	1	0	0	0	0
W3	0	0	100	0	0	0	0	0
W4	0	0	0	99	0	0	0	0
R1	0	0	0	0	100	0	0	8
R2	0	0	0	0	0	100	0	0
R3	0	0	0	0	0	0	100	0
R4	0	0	0	0	0	0	0	92

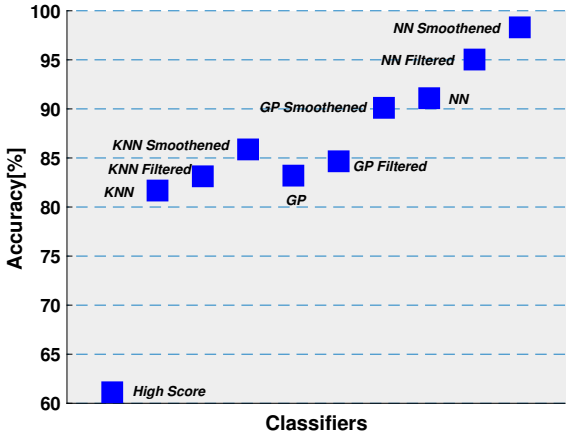


Fig. 9: Comparison of the obtained classification accuracies for all discussed classifiers.

TABLE XIII: Summary of the performance metrics, reported in the form of accuracy (percentage) / F_1 score, achieved for the three estimation methods using the three classifiers.

Classifier \ Estimation method	Initial	Filtered	Smoothened
KNN	82/0.79	83/0.80	86/0.82
GP	83/0.80	85/0.81	90/0.85
NN	91/0.89	95/0.94	98/0.97

The best classification performance is obtained by smoothing the proposed NN initial classifier using the HMM block. As the confusion matrices provided in Tables XI and XII suggest, all the different scenarios of the joint gait mode and device mode classification problem can be distinguished highly accurately. Smoothing class estimates using the NN classifier results in 100% accuracy for five out of eight classes. Although the most problematic case is still finding an appropriate classification boundary between R1-R4, the misclassification error is halved from 16% for GP to 8% for NN.

Fig. 9 compares all the discussed classification methods in terms of the classification accuracy. As a general trend, using the additional smoothing stage improves the classification accuracy by an order of around 4% for KNN, and 7% for GP and NN. Finally, Table XIII summarizes the considered performance metrics achieved for all class estimates using the

three classifiers (no HMM, filtered and smoothened).

There are two main differences between this study and the majority of studies in the literature which either use sensors rigidly attached to he body or use smartphones but only consider the motion mode recognition problem, see for instance [40]. In the former scenario with rigidly attached sensors, the measurements carry more information compared to the measurements collected from IMU sensors embedded in the mobile devices. Since those methods cannot be applied to the mobile device measurements, their applications are limited.

Finally, the performance of the introduced approach in this work is also compared with Table III in [24]. In the reported table, walking and running motion mode together with 4 different device modes (texting, phone, pocket and swinging) are considered. Based on the provided description, the"texting" pose is the same as the fixed in hand device mode in our study. Since the number of classes is the same in both studies, it is fair to compare the performance of the proposed method with the reported results in [24] for six common cases, all except W4 and R4.

The proposed method results in better performance, in terms of classification accuracy, for all of the considered six scenarios in [24] . More specifically, the classification accuracy for all running scenarios is considerably higher than what is achieved in the aforementioned study. Moreover, the highest achievable accuracy of the walking scenarios, as reported in [24], belongs to W1 with 94.8% accuracy. The method proposed in this work, however, results in slightly more accurate results with 98% classification accuracy for the same scenario W1.

VII. Conclusions

The joint classification problem of the user's motion profile and the handheld device's carrying mode using IMU measurements was studied. Eight different classes, based on two different motion modes (walking and running) and four device modes (fixed and face-up in hand, swinging in hand, in the pocket and in the backpack) were considered. K-nearest neighbor, Gaussian process and feed forward neural network classifiers were adopted to jointly classify the aforementioned classes. A set of novel classification features was derived from the pre-processed IMU measurements. Noting that there is a strong correlation between the classes over time, the classifiers' estimated output was further processed using an HMM unit. The best performance between classifiers, before HMM smoothing, was achieved by the neural network classifier with 91% classification accuracy. The filtering stage of HMM enhanced the same classifier's performance achieving 95% classification accuracy. Finally, the best result was achieved by smoothing the neural network classifier's estimates using the HMM unit resulting in 98%. All the reported results are obtained using the test set. Comparing the obtained results with a similar study validated the merit of the proposed method.

There are some possible future research directions. One strong candidate is to consider the intra-mode classification task. Additionally, more possible joint classes could be taken into the account. In this work we consider the smartphone users while in motion (walking or running) in all scenarios. Additionally, a more systematic approach for optimizing the hyper parameters of the considered ML algorithms might be beneficial. For instance, there might be an optimum number of hidden layers and nodes in each layer which can improve the performance of the neural network classifier.

VIII. Acknowledgments

The author would like to thank PhD students from Linköping University who voluntarily participated in the data collection experiment.

This work is funded by the European Union FP7, the Marie Curie training program on *Tracking in Complex Sensor Systems* (TRAX) with grant number 607400, and the Swedish Research Council project *Scalable Kalman Filter*.

References

[1] A. Akl, C. Feng, and S. Valaee, "A novel accelerometer-based gesture recognition system," *IEEE Sensors Journal*, vol. 59, no. 12, p. 6197–6205, Dec. 2011.

[2] C. Yang and Y. Hsu, "A review of accelerometry-based wearable motion detectors for physical activity monitoring," *Sensors*, vol. 10, no. 8, pp. 7772—-7788, 2010.

[3] A. Chowdhury, D. Tjondronegoro, V. Chandran, and S. Trost, "Physical activity recognition using posterior-adapted class-based fusion of multiaccelerometer data," *IEEE J Biomed Health Inf*, vol. 22, no. 3, pp. 678–685, May 2018.

[4] Z. Chen, Q. Zhu, Y. C. Soh, and L. Zhang, "Robust human activity recognition using smartphone sensors via CT-PCA and online SVM," *IEEE Trans Ind Inf*, vol. 13, no. 6, pp. 3070–3080, Dec. 2017.

[5] M. Basso, M. Galanti, G. Innocenti, and D. Miceli, "Pedestrian dead reckoning based on frequency self-synchronization and body kinematics," *IEEE Sensors Journal*, vol. 17, no. 2, pp. 534–545, Jan. 2017.

[6] R. Jirawimut, P. Ptasinski, V. Garaj, F. Cecelja, and W. Balachandran, "A method for dead reckoning parameter correction in pedestrian navigation system," *IEEE Trans. Instrum. Meas.*, vol. 52, pp. 209–215, 2003.

[7] N. Ho, P. Truong, and G. Jeong, "Step-detection and adaptive step-length estimation for pedestrian dead-reckoning at various walking speeds using a smartphone," *Sensors*, vol. 16, no. 9, pp. 1–13, Sep. 2016.

[8] S. Zihajehzadeh, T. Lee, J. Lee, R. Hoskinson, and E. Park, "Integration of MEMS inertial and pressure sensors for vertical trajectory determination," *IEEE Trans. Instrum. Meas.*, vol. 64, no. 3, pp. 804–814, Mar. 2015.

[9] A. Martinelli, H. Gao, P. Groves, and S. Morosi, "Probabilistic context-aware step length estimation for pedestrian dead reckoning," *IEEE Sensors Journal*, vol. 18, no. 4, pp. 1600–1611, Feb. 2018.

[10] J. Li, Q. Wang, X. Liu, and M. Zhang, "An autonomous waist-mounted pedestrian dead reckoning system by coupling low-cost MEMS inertial sensors and GPS receiver

for 3D urban navigation," *Journal of Engineering Science and Technology*, vol. 7, no. 9, pp. 9–14, 2014.

[11] A. R. J. Ruiz, F. S. Granja, J. C. P. Honorato, and J. I. G. Rosas, "Accurate pedestrian indoor navigation by tightly coupling foot-mounted IMU and RFID measurements," *IEEE Trans. Instrum. Meas.*, vol. 61, no. 1, pp. 178–189, Jan. 2012.

[12] C. Ren, T. Fu, M. Zhou, and X. Hu, "Low-cost 3-D positioning system based on SEMG and MIMU," *IEEE Trans Instrum Meas*, vol. 67, no. 4, pp. 876–884, Apr. 2018.

[13] M. Cornacchia, K. Ozcan, Y. Zheng, and S. Velipasalar, "A survey on activity detection and classification using wearable sensors," *IEEE Sensors Journal*, vol. 17, no. 2, pp. 386–403, Jan. 2017.

[14] G. Panahandeh, N. Mohammadiha, A. Leijon, and P. Händel, "Continuous hidden Markov model for pedestrian activity classification and gait analysis," *IEEE Trans. Instrum. Meas.*, vol. 62, no. 5, pp. 1073–1083, May 2013.

[15] M. Elhoushi, J. Georgy, A. Noureldin, and M. J. Korenberg, "Motion mode recognition for indoor pedestrian navigation using portable devices," *IEEE Trans. Instrum. Meas.*, vol. 65, no. 1, pp. 208–221, Jan. 2016.

[16] S. Fang, Y. Fei, Z. Xu, and Y. Tasao, "Learning transportation modes from smartphone sensors based on deep neural network," *IEEE Sensors Journal*, vol. 17, no. 18, pp. 6111–6118, Sep. 2017.

[17] 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.

[18] J. Kupke, T. Willemsen, F. Keller, and H. Sternberg, "Development of a step counter based on artificial neural networks," *Journal of Location Based Services*, vol. 10, no. 3, 2016.

[19] A. Dominguez-Sanchez, M. Cazorla, and S. Orts-Escolano, "Pedestrian movement direction recognition using convolutional neural networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 12, pp. 3540–3548, Dec. 2017.

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

[21] B. Wang, X. Liu, B. Yu, R. Jia, and X. Gan, "Pedestrian dead reckoning based on motion mode recognition using a smartphone," *Sensors Journal*, vol. 18, no. 6, Jun. 2018.

[22] I. P. I. Pappas, M. R. Popovic, T. Keller, V. Dietz, and M. Morari, "A reliable gait phase detection system," *IEEE Trans. Neural Syst. Rehabil*, vol. 9, no. 2, p. 113–125, Jun. 2001.

[23] Z. Zhang, Y. Li, C. Peng, D. Mou, M. Li, and W. Wang, "The height-adaptive parameterized step length measurement method and experiment based on motion parameters," *Journal Sensors*, vol. 18, no. 4, pp. 1039–1050, Mar. 2018.

[24] H. Zhang, W. Yuan, Q. Shen, T. Li, and H. Chang, "A handheld inertial pedestrian navigation system with accurate step modes and device poses recognition," *IEEE Sensors Journal*, vol. 15, no. 3, pp. 1421–1429, 2015.

[25] P. Kasebzadeh, "Learning human gait," Ph.D. dissertation, Linköping University, 2019.

[26] P. Kasebzadeh, C. Fritsche, G. Hendeby, F. Gunnarsson, and F. Gustafsson, "Improved pedestrian dead reckoning positioning with gait parameter learning," in *International Conference on Information Fusion*, Heidelberg, Germany, Jul. 2016.

[27] Y. Tian, W. R. Hamel, and J. Tan, "Accurate human navigation using wearable monocular visual and inertial sensors," *IEEE Trans. Instrum. Meas.*, vol. 63, no. 1, pp. 203–213, Jan. 2014.

[28] P. Händel, "Discounted least-squares gearshift detection using accelerometer data," *IEEE Trans. Instrum. Meas.*, vol. 58, p. 3953–3958, Dec. 2009.

[29] P. Händel, B. Enstedt, and M. Ohlsson, "Combating the effect of chassis squat in vehicle performance calculations by accelerometer measurements," *IEEE Trans. Instrum. Meas.*, vol. 43, no. 4, pp. 483–488,, 2010.

[30] P. Kasebzadeh, G. Hendeby, and F. Gustafsson, "Asynchronous averaging of gait cycles for classification of gait and device modes," in *eprint arXiv: arXiv:1907.02329v2*, 2019.

[31] *IMU Dataset for Device and Motion Mode Classification*. [Online]. Available: http://users.isy.liu.se/rt/parka23/research.html

[32] P. Kasebzadeh, G. Hendeby, C. Fritsche, F. Gunnarsson, and F. Gustafsson, "IMU dataset for motion and device mode classification," in *8th Internation Conference on Indoor Positioning and Indoor Navigation (IPIN2017)*, Sapporo, Japan, Sep. 2017.

[33] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[34] C. Williams and D. Barber, "Bayesian classification with Gaussian processes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 12, 1998.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, Aug. 2014.

[36] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural Networks*, vol. 2, no. 1, pp. 145–151, 1999.

[37] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[38] L. R. Rabiner, "First-hand: The hidden Markov model," *IEEE Global History Network*, Oct. 2013.

[39] P. Kasebzadeh, G. Hendeby, C. Fritsche, F. Gunnarsson, and F. Gustafsson, "Imu dataset for motion and device mode classification," in *Proc. of International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Sapporo, Japan, Sep. 2017, pp. 1–8.

[40] O. D. Lara and M. A. Labrador, "A survey on human activity recognition using wearable sensors," *IEEE Communications Surveys Tutorials*, vol. 15, no. 3, pp. 1192–1209, Mar. 2013.