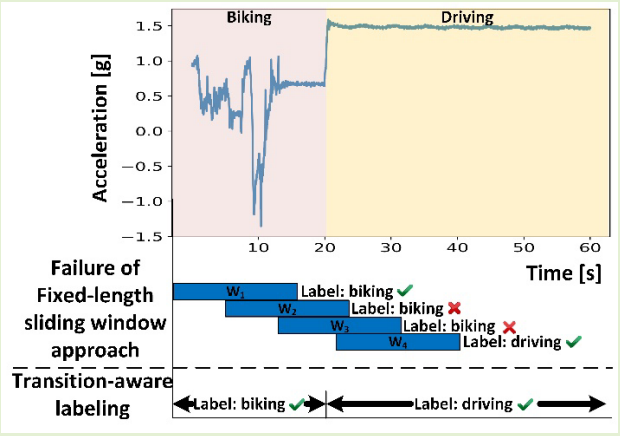
Transition-Aware Detection of Modes of Locomotion and Transportation Through Hierarchical Segmentation

Ali Akbari, *Graduate Student Member, IEEE*, and Roozbeh Jafari, *Senior Member, IEEE*

***Abstract*—Recognizing human daily activities with motion sensors data, specifically, modes of locomotion and transportation provides important contextual information that enhances the effectiveness of mobile applications. For instance, in assisted living or sports monitoring it is essential to log driving or running episodes. Previous studies in this field have utilized a fixed-size windowing technique for segmenting the sequential data of sensors. While segmenting signals into larger windows provides more information about the signal for classifiers, it increases misclassification rate when a transition occurs between the activities (i.e., multi-class windows). This will lead to inaccurate detection and logging of the activities of interest. To identify the exact time of transition from one to another activity as precisely as possible, this article proposes a fast and efficient hierarchical search algorithm that finds the exact sample at which transition occurs. This search algorithm can be applied to any activity recognition model with various lengths of segmentation window. To further improve the performance, we propose a new structure of 2D signal inputs to be used with 2D convolutional neural networks (CNN), which improves the ability of the CNN in capturing patterns underlying in inter-axes correlations. Experimental results show that the proposed transition detection method improves the F1-score by 28% compared to using fixed-size windowing approach for multi-class windows. In addition, the proposed method is 20 times faster than the naïve search.**



***Index Terms*—Activity recognition, deep learning, modes of locomotion and transportation, signal segmentation, transition-aware.**

I. Introduction

RECOGNIZING activities of daily living (ADL) with wearable motion sensors is a convenient way to provide contextual information to many types of tasks. Particularly, detecting modes of locomotion and transportation provides important insight for context-aware services as well as health monitoring applications [1]. For instance, when the user gets off the bus and starts walking, the navigation system needs to switch right away. Equally important, in health monitoring, it is critical to precisely determine the moment of transition from running to walking to determine the amount of calories burned.

The ever increasing advancement in sensing and processing capabilities of wearables along with their pervasiveness and ubiquity makes them an effective and unobtrusive tool for detecting activities including modes of locomotion and transportation [2]. Since the output of the wearable motion sensors, e.g., the ones embedded in smartphones, are continuous time-series, the current approach for recognizing activities is to segment the sequential data into fixed-size windows. Features are then extracted from each segment via domain specific methods to be fed into traditional classification models [3], [4]. Alternatively, segments of raw data may be directly analyzed by deep learning models for automated feature extraction and

Manuscript received June 25, 2020; accepted August 24, 2020. Date of publication September 10, 2020; date of current version January 6, 2021. This work was supported in part by the National Science Foundation under Grant ECCS-1509063. This article was presented in part at the 2018 ACM International Joint Conference and in part at the 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers Conference. The associate editor coordinating the review of this article and approving it for publication was Prof. Nitaigour P. Mahalik. *(Corresponding author: Ali Akbari.)*

Ali Akbari is with the Department of Biomedical Engineering, Texas A&M University, College Station, TX 77843 USA (e-mail: aliakbari@tamu.edu).

Roozbeh Jafari is with the Department of Biomedical Engineering, Texas A&M University, College Station, TX 77843 USA, also with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843 USA, and also with the Department of Computer Science and Engineering, Texas A&M University, College Station, TX 77843 USA (e-mail: rjafari@tamu.edu).

Digital Object Identifier 10.1109/JSEN.2020.3023109

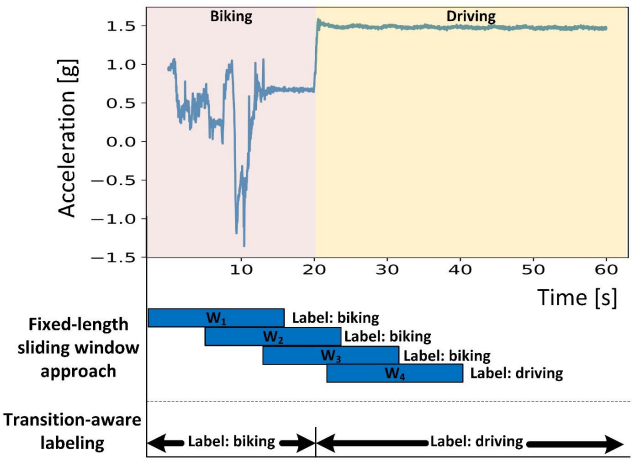


Fig. 1. An example of a multi-class segment that shows the failure of fixed-length windowing approach in class labeling in windows W_2 and W_3 while transition-aware labeling can address this issue.

classification [5], [6]. Using fixed-size signal segmentation (i.e., windowing), however, faces the multi-class problem where the classifier mistakenly labels samples of different classes within the same segment as a single class as shown in Fig. 1, which mainly happens at moments of transition from one to another activity [7].

Given the sequential nature of sensors' data and the inability of a single sample to capture enough information about an activity, sensor data are usually partitioned into relatively large fixed size segments, called windows, for further analysis within machine learning algorithms. Accordingly, the machine learning algorithms will detect and assign an activity label to each individual segment/window [8]. As the window size increases, more information is provided to the classifier. Therefore, intuitively, splitting the dataset into larger segments should achieve greater decision accuracy; however, more than one activity label may exist within one such window, which will cause misclassification – this is known as the multi-class problem as depicted in Fig. 1. In case of such multi-class windows, using larger window size could lead to higher misclassification rate as Fig. 1 illustrates [9]. Specifically, in the application of health monitoring, assisted living, or sports monitoring it is vital to precisely detect and log activities such as driving, walking, running etc. In these applications, detecting the start and stop time, or more generally time of transition between these activities, is important where the aforementioned error in multi-class windows can negatively affect them. On the contrary, using smaller windows will enhance the ability to track such activity label transitions, although, the overall decision accuracy may decline since limited information is provided to the classifier. Hence, there is an obvious tradeoff when selecting optimal window size that depends on the unique characteristic of activity signals. Typically, this will have to be determined empirically or by domain experts.

Extracting features from signal segments requires expert knowledge to design the features carefully and customizing them for recognition systems since the performance of recognition systems is highly dependent on the quality of the features [10]. Furthermore, using such hand-crafted features may miss crucial latent patterns in raw signals. Also, for every potential change in configuration of the system, such as changes in the type of sensors or activity labels, these engineered features may need to be reconsidered and redesigned. To handle these challenges, researchers have proposed deep learning models - specifically convolutional neural networks (CNN) - that are capable of extracting informative features automatically from raw time-series data [11]. The existing approaches still have a few limitations including: (*i*) 1D convolutional filters are usually applied to each axis of a multivariate sensor output separately which ignores inter-axes correlations and patterns; *(ii)* temporal order of human activities is not considered; *(iii)* usually only the raw time series are fed to a CNN, even though the CNN might not be able to distinguish between certain activities as some may have similar signal patterns in time domain. For instance, a smartphone placed in the pocket may measure a similar pattern of acceleration in time domain when the person is riding a car or when he is on a train.

To handle the challenges of selecting a window's fixed size and automated feature extraction with CNNs, we propose 2D signal images along with a hierarchical search framework for detecting the exact moment of transition between activities. The images are in fact 2D matrices of input data that consist of samples of raw timeseries in the rows and different axes of motion sensors placed in the columns in a specific order (Section III-B) to maximize the ability of the CNN to capture inter-axis correlations. The hierarchical segmentation framework incorporates multiple window sizes to analyze motion sensor data and recognize modes of locomotion and transportation while also precisely identifying the exact transition time between activities. Our framework leverages a set of standard, discriminative, hand-crafted features along with a CNN to further extract more complex features that would otherwise be ignored. To distinguish between modes of locomotion and transportation effectively, the deep learning framework receives the input from hand-crafted features, raw time-domain signals, and the power spectral density (PSD). The PSD is essential since it helps to distinguish between challenging modes of motorized transportation that do not have a clear and distinct pattern in the time domain signals. A divide and conquer-based algorithm is proposed to analyze different window sizes, and a searching algorithm is designed to accurately detect the sample at which activity transition occurs. This enables us to take advantage of the strengths of large windows and small windows at the same time.

The contributions of this work are as follows:

•We propose a framework to analyze data gathered by smartphone embedded sensors and detect transitions between modes of locomotion or transportation. In the experiments, our system precisely detects such transitions.

•We propose an activity signal image (2D input matrices of raw data) that allows the typical 2D CNN, to capture inter-axes correlations more effectively. Our proposed model also takes into account the temporal order of human activities. Moreover, by mixing the expert knowledge with automatically extracted features from the CNN, we achieve a higher accuracy

compared to the conventional deep learning based systems that use merely the raw data.

•We demonstrate the effectiveness of our system on a publicly available dataset typically used for locomotion and transportation detection via smartphones. Unlike most existing work in this area, we evaluate our model with sample-based classification metrics.

II. Related Works

Detecting modes of locomotion and transportation is a particular case of activity recognition that has been widely studied within the wearable and mobile computing communities [12]. Different pedestrian modalities such as walking or running, non-motorized transportation, such as biking, and motorized transportation, such as bus, train or car are different type of locomotion and transportations that have been studied by researchers [1]. Many researchers have used accelerometers for detecting different pedestrian and non-motorized locomotion including walking, running or biking [13], [14]. A novel accelerometer-based technique was presented for accurate and fine-grained detection of transportation modes using smartphones [1]. To segment the sensors data, they used a sliding window with a duration of 1.2 seconds in an effort to promptly detection transitions. Consequently, accuracy suffered when attempting to distinguish between bus, train, metro, and tram modes.

Adaptive signal segmentation techniques have been proposed to overcome the aforementioned issues with a fixed size windowing approach. A novel approach to adaptively adjust the window size was designed to achieve the most effective segmentation based on the probability that the signal belongs to a particular activity class [5]. They started with 3-second windows, and if they detected that a window contains a transitional activity, they expanded the window size until they get the most confident decision from the classifier. Another adaptive time window method was proposed to extract features from quasi-periodic signals for activity recognition [15]. They detect changes in activity and take the beginning and end time as segmentation boundaries. However, their technique worked only for signals that have periodic patterns.

A dense labeling paradigm based on fully connected neural network was designed for predicting the label of each sample in a time-series instead of segmenting and finding the label for each segment [7]. This approach overcomes the problems posed by multi-class and fixed size sequence. Another study proposed a human activity recognition algorithm based on a U-Net to perform activity labeling and prediction at each sampling point [16]. The activity data of the triaxial accelerometer is mapped into an image with the single pixel column and multi-channel as input into the U-Net network which can perform pixel level activity recognition. These models, however, are computationally expensive as they need to run classifier on every single sample to generate per-sample labels. Moreover, to work with larger window sizes, which is required for detection of certain complex activities, these models become arbitrarily huge with a large number of trainable parameters, which increases their memory and computational requirements. To address the memory challenges, the dense models typically run with smaller window size [16], which reduces the ability of the models in detecting more complex activities in which analyzing longer windows would be helpful. Our proposed method, however, handles both large and small window sizes without an increase in required memory or trainable parameters.

Multi-labeling methods have been proposed to detect more than one label in a single instance of input. The idea of multi-label activity recognition from video frames was proposed in [17]. In this article, the authors were able to detect multiple activities occurring in one scene by leveraging clustering and classification methods at the same time based on the assumption that instances from the same class usually organize themselves into clusters. This idea, however, needs extracting informative features that guarantee the aforementioned assumption, which requires carefully crafted features. Therefore, applying such a technique to deep neural networks is challenging and optimizing such a model will be computationally expensive. A multi-label classification approach was also proposed to detect activity and accelerometer sensor location simultaneously [18]. In this article, every combination of the activity and sensor on-body locations was considered as a separate class and a single classifier was trained to detect them. Although this idea is applicable to our problem, its shortcoming is to train a single classifier for all the possible outcomes, which introduces challenges in distinguishing complex labels that may have similar patterns of motion signal. Our proposed approach can be seen similar to [18] in the sense that we train separate classifiers for different possible combination of two activities. In our model, in the first stage we detect one out of all possible activities. However, in contrast to the multi-label approach proposed in [18], in the next levels of the hierarchical model, we only utilize two-class classifiers which improves the likelihood of detecting the correct class due to using a limited search space. Limiting the search space becomes more important as we move towards detecting activities from smaller windows since lesser amount of information would be provided to the classifier.

To address the challenge of extracting informative features for complex activities, deep learning has been used in recent works. Several prior investigations use data from inertial measurement units (IMU) or motion sensors to automatically extract features and detect human gestures and activities using CNN. In one study, axes of the IMUs were treated like different channels of an image provided to a three layer CNN, outperforming traditional machine learning algorithms with respect to the accuracy [11]. A CNN based approach that captures temporal dependency of a signal was proposed to perform activity recognition with a cell phone's accelerometer [19]. These investigations used 1D convolutional units which can consider temporal patterns of the signal but not the relationship between different axes of a sensor.

To overcome this problem, data from an accelerometer and a gyroscope were converted into images to be used in a CNN for detecting basic human activities, which outperformed state-of-the-art in terms of both recognition accuracy and

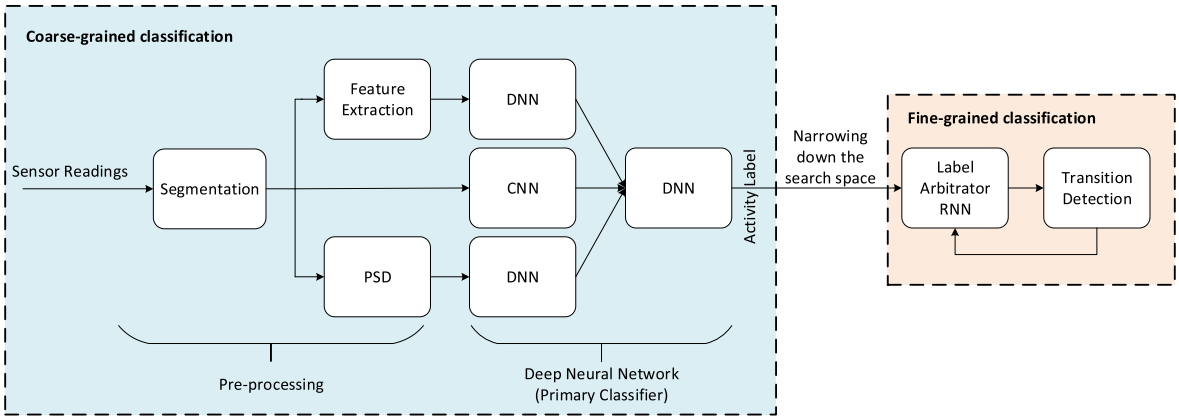


Fig. 2. The overview of the proposed framework for transition-aware activity recognition.

computational efficiency [20]. However, this study did not take into account the correlation between all combinations of sensor axes. Deep learning methods that use more than just CNN for ADL recognition have also been developed. A combination of CNN and recurrent neural network (RNN) was used to detect modes of locomotion as well as hand gestures [21]. In this article, four convolutional layers were used to extract features of the signal and two recurrent layers were used to model temporal behavior of the features. The article shows that using RNN in addition to CNN can slightly increase accuracy. This works uses RNN to understand the temporal features within a window of signal. However, the authors ignore the temporal order of human activities between consecutive segments of data, which can also be learnt by RNNs.

III. Methods

Fig. 2 shows our proposed framework for transition-aware activity recognition. We start by segmenting the sensors' data into relatively large windows (one minute in this study) and recognize one out of N possible activities called coarse-grained classification. This consists of three components. First, a dense neural network (DNN) that accepts the hand-crafted features and maps them to higher level features. Next, a CNN that accepts raw time domain signal and performs automated feature extraction. Last, a DNN processes signal's PSD. One final DNN combines the outputs of the three layers and applies Softmax activation to produce a one-hot vector decision of length N. Since this may be considered as a distribution, the index with the greatest probability corresponds to the appropriate label. A label arbitrator module makes use of an RNN to modify the outputs of the coarse-grained classifier with respect to the temporal order of activities. Lastly, if an activity transition is detected for two consecutive windows, they are then divided into smaller sub-windows. The sub-windows are processed by a constrained version of the classifier to find the exact location of transition.

A. Problem Formulation

Let, X_i, be a d-dimensional feature vector at time-step, i. The features consist of x, y, and z direction values recorded by accelerometer, gyroscope, and magnetometer sensors. If we segment the data into lengths of T, as shown in Fig. 3,

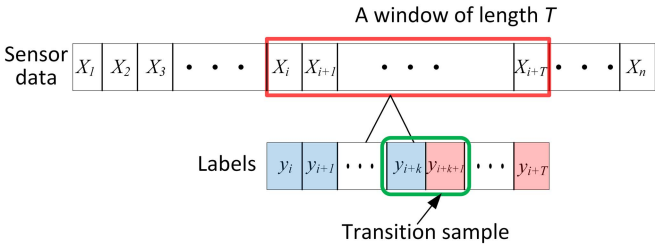


Fig. 3. A multi-class window of data.

the label assigned to the window as a whole may correspond to that of the first, last, or any other sample within that window. However, labels are not always consistent through the segment which leads to misclassification - such as for the k^{th} transition sample of the window in Fig. 3. We call such a window as a multi-class window and the k^{th} sample at which transition happens as the transition sample. To handle this, we propose an efficient search algorithm to detect such transition samples, k, where $y_k \neq y_{k+1}$, so we may accurately classify activity labels to samples occurring before and after such transitions.

B. Coarse-Grained Classification

The first stage of our framework classifies the appropriate activity label for each segment of sensor readings. This requires preprocessing the data and a deep neural network referred to as primary classifier.

1) Preprocessing: Input data is segmented into windows of one minute. Such a large window size is chosen to: 1) cover enough information to detect activities and 2) reduce computational time-complexity significantly.

A set of conventional features are extracted from each segment of sensor data. Table I lists the features used in this study, including the name and dimension. These are all consistently used time-domain features for activity recognition [10], [22]. The features are collected from each axis of the sensor modalities (i.e., accelerometer, gyroscope, pressure sensor). The same set of features are extracted from the magnitude of each signal as well in order to remove the effect of sensor orientation.

The raw data of each segment is also fed to a CNN for automated feature extraction to compensate for the possibility

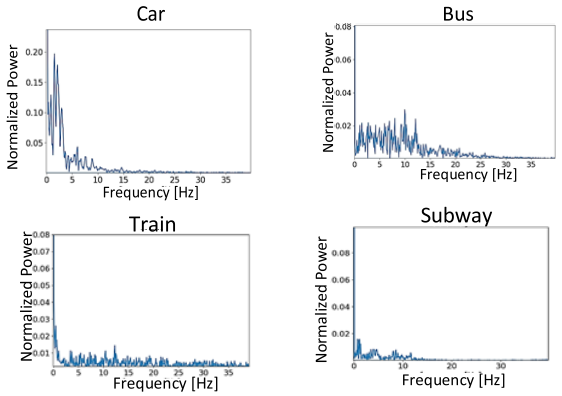


Fig. 4. Example to show PSD of magnitude of acceleration varies among different modes of motorized transportation.

TABLE I
HAND-CRAFTED FEATURES

Feature	Dimension (per axis)
Mean	1
Standard deviation	1
Root Mean Square Error	1
Mean Cross Rate	1
Skewness	1
Kurtosis	1
Zero Cross Rate	1
Entropy	1
AR Coefficients	10
Integration	1
Signal Magnitude Area	1
Total	**189**

of missing significant underlying patterns when relying on hand-crafted features. For example, bus, car, train, and subway modes have similar time-domain signal patterns, which cannot be distinguished by either those hand-crafted features or the CNN that works on raw time-domain signals. However, we observed that the amount of vibration varies among different vehicles. This is captured by frequency components of the acceleration signal as shown in Fig. 4. This figure illustrates an example of how the power spectrum density of the magnitude of the acceleration is different among different modes of transportation. To distinguish between those activities, we calculate the PSD for each segment of the sensor readings and feed it to a DNN. All the inputs, that is, hand-crafted features, raw data, and PSD, are normalized to 0 mean and unit standard deviation.

2) Primary Classifier: The primary classifier consists of three independent deep networks that work on three different types of input as shown in Fig. 2. The first dense neural network (DNN), which is fed with hand-crafted features, contains three fully connected layers where the details are represented in Table II. It has been shown that increasing the number of layers (known as depth) can significantly increase learnability and decrease the number of required neuron units in each layer [23]. In the current study, we use a DNN with three fully connected layers as we did not gain significant improvement by increasing this beyond three.

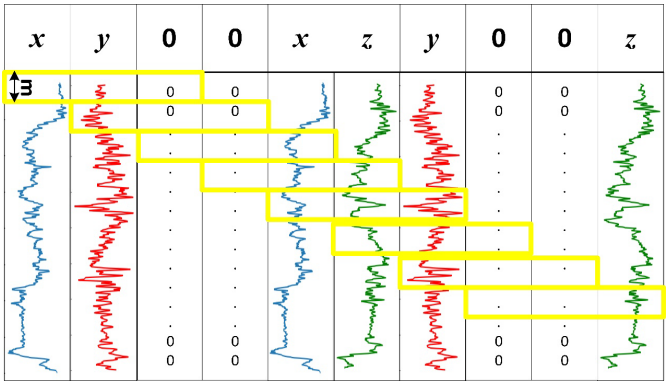


Fig. 5. Our proposed 2D signal image (i.e., 2D matrices composed of raw timeseries data) for considering the correlation between all the axes of a sensor. The yellow rectangles show m∗3 kernels where m is the kernel length. By putting sensor axes in this order, the 2D kernels with width of 3 can capture the correlation between all the axes.

TABLE II
DETAILS OF DNN WORKING WITH HAND-CRAFTED FEATURES

Layer	# of neurons	Activation function
Fully connected_1	200	ReLU
Fully connected_2	64	ReLU
Fully connected_3	16	ReLU

The second neural network is a CNN that extracts features from raw time domain signals segmented into one-minute windows. Previous studies have shown the ability of CNN in capturing short-term temporal patterns in time-series motion signals through applying convolution operation between trainable filters and the timeseries along the time axis [11]. Existing works that use CNNs for automated feature extraction for this task usually apply 1D kernels to each axis of the sensor data ignoring possible correlations between the axes. To prevent this, we propose sorting the axes of a single sensor to a specific order creating signal images so that we can take advantage of 2D convolutional. Fig. 5 shows how sensor axes should be ordered to create the signal image so that an $m \times 3$ convolutional kernel (where m is the kernel length) can capture the patterns from all combinations of sensor axes. In Fig. 5, the yellow rectangles show the $m \times 3$ kernels where starting from top to the bottom of the kernels extract features from *xy, y, x, xz, xzy, zy, y*, and *z* axes respectively, where *xy* indicates the correlation between x and y. The CNN in this study is composed of three convolutional layers where the details are represented in Table III. The stride length of 1 and ReLU activation function is used in all the layers. In an effort to distinguish between the challenging modes of motorized transportation, we feed the PSD of the signals to a DNN consisting of an architecture similar to that expressed in Table II. The intuition behind using a DNN is because CNNs capture patterns of signals regardless of the location of their occurrence, while in PSD the location of the patterns are important as they correspond to different frequencies. In this case, using a DNN, which assigns different weights to different regions of PSD signal, is a better choice.

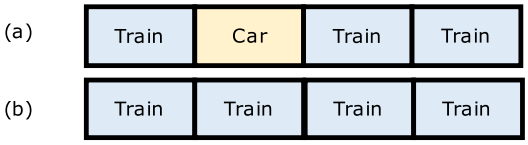


Fig. 6. (a) An example of a misclassification that can be fixed by label arbitrator; (b) the output of RNN as label arbitrator.

TABLE III
DETAILS OF CNN FOR AUTOMATED FEATURE EXTRACTION FROM RAW TIME-SERIES

Layer	Kernel Size	Number of Neurons
Convolution 1	500*3	64
Convolution 2	300*1	100
Convolution 3	100*1	100

Finally, a shared DNN is responsible for combining the output of all three networks and mapping them to the final class labels. The output of each network, before feeding to this DNN, is first passed through a batch normalization layer so that they all be in the same range. Then a fully connected layer with 128 neurons is used to combine all the features together with ReLU activation function. The last layer of this DNN works with Softmax activation function as shown in Equation 1. The number of the neurons in the last layer is equal to the number of the classes to be classified.

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}} \tag{1}$$

Equation 1 shows the Softmax where x_i is i^{th} input to the function and N is the total number of function inputs, which, in the case of classification, is equal to the total number of classes. In this study, in the coarse grained classifier the value of N is equal to the total number of modes of locomotion and transportation to be recognized. For the fine-grained classifier, however, this value is equal to two (see Section III-C-2). The fine-grained classifier works with a narrower search space since it is expected to detect the activities with smaller window sizes. The outputs of the Softmax function are values between zero and one, and the total sum is equal to one. The output of this function is equivalent to a categorical probability distribution.

For each of these classifiers, the dropout technique is utilized to reduce chance of overfitting. The key idea of dropout is to randomly deactivate some units along with their connections in the neural network during training [24]. This deactivation prevents the units from co-adapting too much and improves generalization because it forces the layers to learn the same concepts with different neurons. However, the dropout layer is deactivated during the prediction phase.

C. Fine-Grained Classification

1) Label Arbitrator: To further modify the labels created by the coarse-grained classifier, a label arbitrator module is proposed to model the temporal order of human activities. Fig. 6-a shows an example of a misclassification due to labeling windows as a whole. The second segment, which belongs to the class "car" is misclassified because it may have a signal similar to the class "train". In general, one mode of locomotion or transportation cannot occur within another. Moreover, certain orderings of activities are not typical for a given user. For example, in the SHL dataset [25], the studied subject never rides a bike right after leaving the train or subway. To best capture this, we train an RNN which receives the latest five segments' labels as input (i.e., last five minutes) and outputs the most likely sequence of labels for those five segments. An example of the output of this RNN, which may be described as a sequence-to-sequence model, is depicted in Fig. 6-b. This RNN is trained on the whole training dataset and learns the temporal order of activities. During the testing phase, the labels created by the coarse-grained classifier are passed through this RNN to modify the labels. This RNN contains one layer with 25 long-short-term-memory (LSTM) neurons.

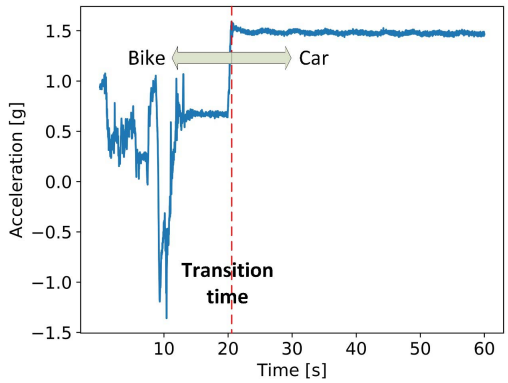


Fig. 7. An example of a 60-second segment of the data that contains more than one activity. The activity on the right and left hand side of the transition sample is different.

2) Transition Detection: Although segmenting the data into large windows (60-second in this study) achieves better accuracy in recognizing activities, especially for detecting modes of transportation, it may lead to a higher error for multi-class windows that contain more than one activity (i.e., transition from one activity to another). In such cases, detecting the sample at which the transition between activities occurs is desired to improve the performance.

We know that when a transition occurs at time stamp $t_{transition}$, the activity occurring at $t > t_{transition}$ should be different from that at $t < t_{transition}$ as shown in Fig. 7. Therefore, at the transition sample if we look at a segment to the right and a segment to the left, we expect the corresponding windows to express different probability distributions [26]. So, by monitoring such characteristics, we may identify transition points. In other words, it is the problem of identifying the sample where the probability distribution of a time series changes [27]. This means that by going through every sample and estimating the probability density of the activities for a segment to the right and left, we can find the sample at which the transition happens. However, this naïve approach is very computationally expensive as it depends on detecting labels for the prior and post segments of every sample of data. For instance, working with 60-second windows with sampling rate of 100 Hz (as in SHL dataset in this study), we need to run the

classifier 6000∗2 times (i.e., for all the samples within the two windows). To address this problem, we propose a divide and conquer algorithm to narrow the windows in which a transition is more likely to occur as much as possible before searching for the exact sample of transition. In Section IV-C we prove that this approach has a substantial improvement with respect to computational cost.

Algorithm 1 Divide and Conquer for Narrowing Down Multi-Class Windows

Input: Two consecutive 60-second window W_{t-1} and W_t and their labels A and B, two-class classifier f_{AB}
Output: Two 5-second windows W_1, W_2containing transition
1: $W_1 = W_{t-1}$
2: $W_2 = W_t$
3: **for** i from 1 to 2: // dividing 60-second window first in 30 seconds and then 15 seconds
4: Split W_1 in two same length windows $W_{1,1}$and $W_{1,2}$ and feed them to f_{AB} to get $L_{1,1}$ and $L_{1,2}$
5: Split W_2 in two same length windows $W_{2,1}$and $W_{2,2}$and feed them to $f_{AB}$to get $L_{2,1}$ and $L_{2,2}$
6: $L = [L_{1,1}, L_{1,2}, L_{2,1}, L_{2,2}]$ // $L_{i,j} \in$
7: **if** there is any A to B transition in L:
8: $W_1 =$ *the last segment with label A in L*
9: $W_2 =$ *the first segment with label B in L*
10: **else if** L only contains A:
11: $W_1 = W_{2,2}$
12: W_2= [ ]
13: **else:**
14: $W_1 = W_{1,1}$
15: W_2= [ ]
16: **end if**
17: **end for**
18: Split W_1 in three same length windows $W_{1,1}, W_{1,2}, W_{1,3}$ and feed them to f_{AB} to get $L_{1,1}, L_{1,2}, L_{1,3}$
19: Split W_2 in three same length windows $W_{2,1}, W_{2,2}, W_{2,3}$ and feed them to f_{AB} to get $L_{2,1}, L_{2,2}, L_{2,3}$
20: $L = [L_{1,1}, L_{1,2}, L_{1,3}, L_{2,1}, L_{2,2}, L_{2,3}]$ // $L_{i,j} \in \{A,B\}$
21: **if**there is any A to B transition in L:
22: W_1= *the last segment with label A in L*
23: W_2= *the first segment with label AB in L*
24: **else if** L only contains A:
25: $W_1 = W_{2,3}$
26: W_2= [ ]
27: **else:**
28: $W_1 = W_{1,1}$
29: W_2 = [ ]
30: **end if**

The proposed divide and conquer algorithm is shown in Algorithm 1. We first detect the windows that are susceptible to contain a transition. This is executed by comparing the activity labels of two consecutive windows. The labels of t^{th} and $t\text{-}1^{th}$ windows may be denoted as L_t and L_{t-1} respectively. If $L_t = A$, $L_{t-1} = B$, and $A \neq B$, then we identify the two windows *t-1* and t susceptible to contain a transition. We then divide each of the one-minute segments (t and *t-1*) into two non-overlapping 30-second sub-windows.

Algorithm 2 Search for Transition Sample

Input: Two consecutive 5-second window W_1 and W_2 and their labels L_1 and L_2, two-class classifier f_{AB}
Output: The *transition sample*
1: S= [ ]
2: **for** i in all the samples within W_1 and W_2:
3: *W1 = window(i,i+500) // window(start,end)* segments the data from index *start* to *end*
4: *W2 = window(i-500,i)*
5: feed *W1* and *W2* to f_{AB} to get *P(A|W1)*, *P(B|W1)*, *P(A|W2)*, and *P(B|W2)*
6: $S_i = P(B \mid W_{i+}) * P(A \mid W_{i-}) - P(A \mid W_{i+}) * P(B|W_{i-})$
7: **end for**
8: *transition sample = index(max(S))*
9: **for** j in *W1*, *W2*:
10: **if** $j <$ *transition sample*:
11: assign label A to sample j
12: **else**
13: assign label B to sample j
14: **end if**
15: **end for**

This process is depicted in Fig. 8. For each of the four 30-second sub-windows, we use a fine-grained classier to identify the appropriate activity label. Although its architecture is similar to the coarse-grained classifier, it is only trained for two-class predictions to distinguish between labels A and B (which were the labels assigned to the original windows by the coarse-grained classifier). In this way, the search space is limited for the fine-grained classifier.

As the coarse-grained classifier works with a larger window size, it accesses more information describing each segment, so it can detect one out of N different activities with higher confidence. The fine-grained classifier, however, runs on a smaller window size and analyzes much less information.

Despite this, limiting the search space enables the fine-grained classifier to sustain accuracy [28]. Again, for each of the four 30-second sub-windows we compare the activity labels and further divide those consecutive windows that differ into two 15-seconds sub-windows and pass them into the fine-grained classifier. This whole process is repeated once more by dividing the two 15-second sub-windows with distinct labels into three 5-second sub-windows. Finally, the two remaining 5-second windows with distinct labels are chosen as the candidates for containing a transition. It should be noted that we have trained a separate fine-grained classifier for every possible combination of two activity labels and for every window size.

After narrowing down to the 5-second sub-windows, we start sample-wise search to identify the exact transition point. Algorithm 2 shows this procedure. We process each sample of the two 5-second sub-windows and segment two more 5-second windows to the right and to the left of it. They are then again fed to the fine-grained classifier to detect the activity label of A versus B. The probability of each activity

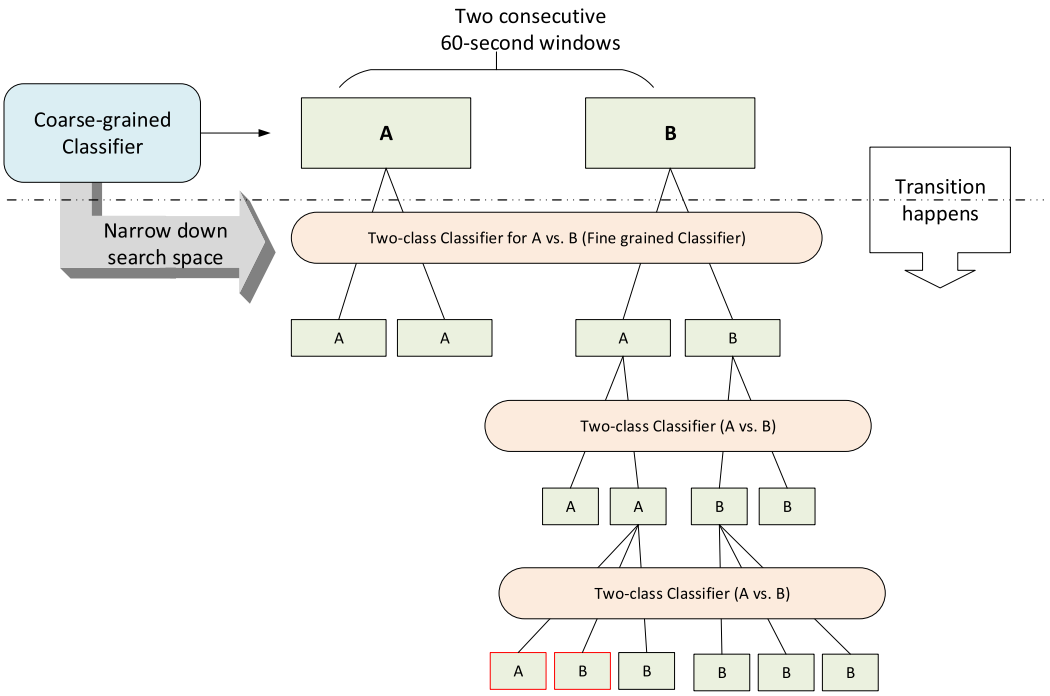


Fig. 8. Divide and conquer-based hierarchical segmentation to find small windows in which the activity transition happens. The two class-classifiers only look for the activities that were initially determined by the primary classifier.

for each sample is put into a matrix as shown in Equation 2, and the transition score is calculated based on Equation 3 for each sample. In Equation 2, M_i is the matrix of transition between activities for i^{th} sample, W_{i-} is the 5-second sub-window containing the samples before sample i, W_{i+} is the 5-second sub-window containing the samples after sample i, and $P(A|W)$ is the output probability of the neural network from Equation 1 which classifies activity A for the data of window W. In Equation 3, S_i is the transition score for sample i^{th} with respect to what we detect as the transition point.

$$M_i = \begin{bmatrix} \mathrm{P(A}|W_{i-}) & \mathrm{P(A}|W_{i+}) \\ \mathrm{P(B}|W_{i-}) & \mathrm{P(B}|W_{i+}) \end{bmatrix} \tag{2}$$

$$S_i = \mathrm{P\,(B}\mid W_{i+}) * P\,(\mathrm{A}\mid W_{i-}) - \mathrm{P\,(A}\mid W_{i+}) * P(B|W_{i-}) \tag{3}$$

The transition score calculated by Equation 3 is greatest at the sample where the window to the right and left have two different labels (i.e., A and B) with highest confidence (i.e., highest probability) [26]. These scores are then sorted and the sample with the greatest score is chosen as the transition point. All samples occurring before the transition sample receive label A while those occurring after receive label B. In Section IV-C we show that our proposed framework is competitive with the naïve search method while improving its computational cost by a power of 20.

The two components of the fine-grained classifier including label arbitrator and transition detection could introduce some latency in the classification process. Given the largest window size in our study is one minute and the transition detection module analyzes two consecutive windows, the minimum latency required for transition detection would be two minutes. The other component, label arbitrator, acts as a post-processing module to further refine the labels based on the temporal order of activities. This module looks at labels over last five minutes, which can translate into a latency of five minutes. For real-time applications, however, by using a moving window over last five minutes and refining only the label of the most recent window, one can ignore this latency.

IV. Results

To demonstrate the performance of our system when recognizing modes of locomotion and transportation and detecting the exact moment of transition between modes, we conduct four experiments: 1) we show effectiveness of our method using sample-based F1 score and compare it to the conventional fixed-window based approaches (Section IV-B); 2) we compare the accuracy and computational time of our algorithm to the naïve search method (Section IV-C); 3) we provide activity-based misalignment measures (Section IV-D) to show an in-depth assessment of our proposed transition detection approach for multi-class windows; and 4) we analyze the effect of each component of the proposed framework on the performance of the system (Section IV-E).

We evaluate our approach using the SHL dataset explained in Section IV-A. The activities to detect are still, walk, run, bike, car, bus, train, and subway. They are further divided into three different sub-tasks that mimic different application interests, and we report the performance for each of them. The sub-task scenarios are defined as follows:

- Scenario1: Five classes as still, walk, run, bike, and vehicle. The vehicle class contains all car, bus, train, and subway activities.
- Scenario2: Six classes including still, walk, run, bike, road, and rail. The road class contains bus and car activities while the rail class contains train and subway activities.

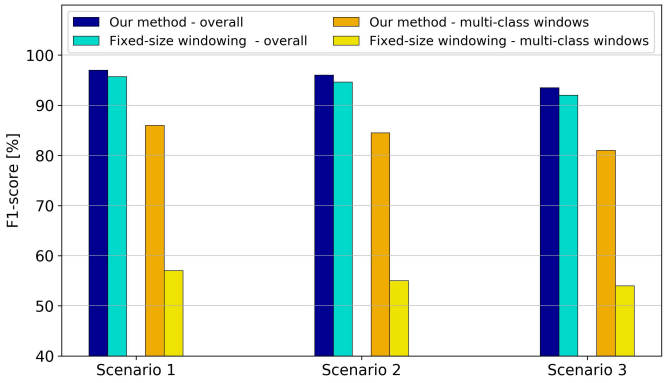


Fig. 9. Comparing the F1-score of our proposed method with traditional methods who use fixed-size windowing approach for signal segmentation shows the superiority of our method and importance of considering multi-class windows in activity recognition with timeseries.

- Scenario3: Eight classes considering each activity as a separate class.

A. Dataset and Task Description

The objective of this article is to recognize modes of locomotion and transportation activities and precisely detect transition points between the activities via data recorded with motion sensors. The activities that have to be recognized are still, walk, run, bike, car, bus, train, and subway from the publicly available SHL dataset. The SHL dataset used for this study comprises 271 hours of training data and 95 hours of test data [25]. The data is recorded by a Huawei Mate 9 smartphone attached to the right front pocket of participants over a period of seven months. The orientation of the smartphone is not necessarily fixed. The data includes readings from a 3D accelerometer, gyroscope, magnetometer, and ambient pressure sensor as well as linear acceleration, gravity, and orientation (represented in quaternion form). Data is collected from all sensors at the frequency of 100 Hz and all samples are labeled. We aim to precisely identify transition points between modes of locomotion and transportation in time-series sensor data. Therefore, we evaluate the performance of our system with sample-based classification metrics.

B. Sample-Based Classification Results

To demonstrate the overall effectiveness of our activity recognition framework we compare it to the conventional approach of using fixed size windows. The results of this comparison are shown in Fig. 9. The overall performance, blue and cyan bars, report the F1-score for all the samples in the testing dataset, and the multi-class windows, orange and yellow bars, show the F1-score for only the 60-second windows that contain a transition between the activities. As the figure shows, in all three scenarios, our method (blue and orange bar) outperforms the fixed size windowing approach by 28% in multi-class windows that contain transitions between modes of locomotion and transportation. The overall accuracy of our method for all test samples is 1.5% better than the fixed-windowing approach since most of the windows, 96%, do not contain a transition in this dataset. To mimic the conventional

	S	W	R	Bi	C	Bu	T	Su
Still	94	0.5	0	0.1	0.5	0.9	1.9	2.1
Walk	0.3	98.8	0.3	0.2	0.3	0.2	0.2	0.1
Run	0.1	0	99.5	0.1	0	0.2	0	0
Bike	0.2	0.2	0.1	99.6	0	0.1	0	0
Car	0.1	0	0	0	93.2	3.5	1.1	2.7
Bus	1	0.1	0.1	0	3.2	91.4	1.5	2.2
Train	2.4	0	0	0	1.8	3.4	85.2	13.6
Subway	1.9	0.3	0	0	1	0.3	10.1	79.3

Fig. 10. The overall confusion matrix for detecting eight modes of locomotion and transportation with our method (numbers are percentage).

fixed windowing approach we used the same classifier as Section III-B-2 and removed the transition detection module (Section III-C-2), so the classifier analyzes only 60-second windows and assigns one label to all the samples within a window. This approach performs poor in multi-class windows. The lower accuracy of fixed-size windowing approach, specifically for transitional windows, stresses the importance of multi-class windows for accurate recognition. Moreover, from Fig. 9 it can be seen that when we combine some modes of transportation together as one class (i.e., scenarios 1 and 2) the accuracy is higher compared to the case of detecting them separately (i.e., scenario 3).

It should be noted that our proposed algorithm functions based on the assumption that one window of size 60 seconds could contain up two activity classes. Although in some cases there might be more than two activities occurring within a 60 seconds window (e.g., walk to train platform, wait few seconds, and take train), in the current application we are mainly interested in detecting long-term transportation and locomotion activities. Hence, very short instances of walking or standing, which may happen in between two transportation and locomotion activities are not of interest. In this case, our proposed algorithm may consider such a short-term activity as one of the two longer term classes. In most of context-aware applications and services detecting long-term activities and the moment of change between the two is more important than short term transitional activities. In the SHL dataset, 60-second windows with more than two activity label were not observed as the labels are provided for long-term modes of locomotion and transportation. In fact, in SHL, short-term transitional activities that may happen in between two modes of transportation and locomotion have not been considered. Expanding the proposed algorithm for detecting cases with short-term transitional activities inspires an excellent future direction.

Fig. 10 shows the overall confusion matrix of our model for scenario 3 where all the class labels are considered independently. In the confusion matrix, each column corresponds to the true label and each row corresponds to the estimated label. As seen in Fig. 10, most of the misclassifications occur between the train and subway classes because they have very similar signal behaviors. This also explains why the performance is better in scenario 1 and 2 compared to scenario 3 as seen in Fig. 9. The activity ‘still’ is also confused with some of transportation activities when the vehicle moves steadily. Walk, run and bike are the easiest to be recognized as they possess unique and specific patterns in both time and

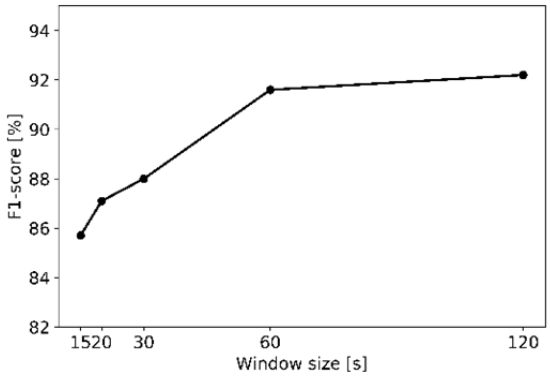


Fig. 11. The performance versus window size which shows that 60-second window is the optimum window size.

frequency domains. We also assess the effect of the size of the window on the performance to find the most optimal window size to start with in our algorithm, as shown in Fig. 11. The values correspond to scenario 3 where all activity labels are treated separately. As the figure represents, a 60-second window size is the smallest choice that yields an acceptable accuracy before a significant decline occurs. This proves that 60-second window is the best option to begin analysis with for our proposed framework. Furthermore, this justifies the fairness of our comparison in Fig. 9 as 60 seconds is the best choice for the fixed-size windowing approach.

Lastly, we compare the performance of our method to the existing state-of-the-art methods with respect to recognition accuracy for multi-class windows. As aforementioned, we make use of the SHL dataset which has been used in the HASCA Sussex-Huawei Locomotion Challenge 2018 [29]. The F1-Score of the winner of the competition on multi-class windows is approximately 65%, which is 17% less than our method, while their overall accuracy is 93.9% [30]. Our framework achieved overall 0.5% higher F1-Score than [30] which is mainly due to better detection of multi-class windows although less than 4% of all data windows contain such multi labels. Another study [31] achieved the best overall F1-score of 92.9% using CNN with frequency-domain sensor inputs and post-processing based on majority voting, which is 1.5% less than ours. They do not report the performance over multi-class labels but it is believed that the main difference is related to those challenging segments of data. It must be noted that the results demonstrate without frequency domain inputs, the F1-score drops to 86.6% and without the postprocessing, it drops to 82.5%. Finally, the best accuracy that was provided with traditional machine learning models was 87% through SVM. There are only a few studies that share the results specifically over multi-class (i.e., transition) windows. Such results were reported for the participants of the HASCA2018 challenge [29]. The two best performance obtained on multi-class windows were approximately 75% [32] and 74% [28] in this competition, which are 7% and 8% less than our method.

C. Computational Time

To show the effectiveness of our divide and conquer algorithm which narrows down the window that is susceptible to have multiple, distinct labels, we compare it to the case of searching for the transition sample within a 60-second window called "naive search" in Table IV. In this approach, when we suspect a transition from one window to the next, we go through all the samples within those two windows (of size 60 seconds) and at each sample we apply the transition detection algorithm. In fact, in this approach, the divide and conquer method is not utilized and transition search algorithm is directly applied to the samples of 60-seconds windows.

TABLE IV
COMPARING COMPUTATIONAL TIME AND ACCURACY OF OUR METHOD WITH NAÏVE SEARCH APPROACH WHO PERFORMS THE SEARCH ON ALL THE SAMPLES WITHIN A 60-SECOND WINDOW

	Computational time* [s]	F1-Score [%]
Naive search	3669	94.49
Our algorithm	180	94.41

* time for processing the whole testing dataset

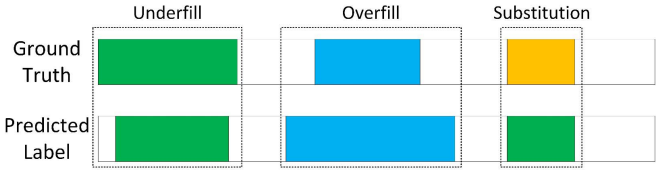


Fig. 12. Illustration of different misalignment performance metrics. Different colors show different activities.

As Table IV shows, our algorithm achieves the same accuracy as the naive search, yet it performs the task 20 times faster. This improvement is achieved due to reducing the window size for sample-wise search through changing the window size hierarchically with divide and conquer algorithm. In fact, the naive search approach requires the classifier to be executed for 12000 samples within the two windows. Meanwhile, our approach only requires runtime execution for: four 30-second windows, four 15-seconds, six 5-second windows, and finally sample-wise search for 600 samples within the two 5-second windows. This is 614 runs of classifier which is significantly less than 12000 runs in the naïve, brute-force search.

D. Activity-Based Misalignment Performance

To demonstrate the effectiveness of our activity recognition and transition detection method, we evaluate performance regarding activity misalignment measures proposed in [33]. These activity-based metrics capture the failures of standard sample-based classification evaluation methods by measuring artifacts such as overfill, underfill, and substitution as shown in Fig. 12.

For each activity A the underfill is the ratio of the samples that are predicted as not A while there are instances of A at the beginning or end of a period of activity A. The overfill is the ratio of samples that are predicted as A while they are not A at the beginning or end of a period of activity A. These two metrics are not related to classifier's mistakes but they happen due to inappropriate signal segmentation. These two metrics are of interest of this study, where our method tries to minimize them. Substitution is the typical misclassification

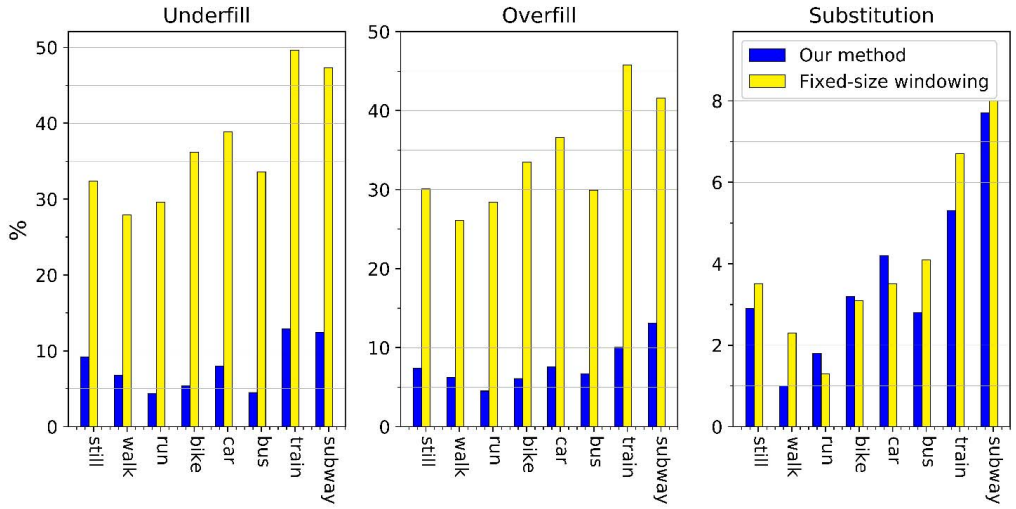


Fig. 13. Sample-wise misalignment performance metrics. The figure depicts these metrics for only multi-class windows. It shows that a huge amount of misclassifications happened in fixed-size windowing approach is due to underfill and overfill but not substitution.

which is the actual mistake of the classifier. These measures provide deeper insight to the sample-wise misclassifications within a class as they can provide an in-depth evaluation of the effectiveness of the proposed approach for the problem of transition detection within multi-class windows. Specifically, the two metrics underfill and overfill demonstrate the weakness of inappropriate signal segmentation. These measures are independent of typical classifier's mistakes that happen due to insufficient training. We report these three activity misalignment measures with respect to the ground truth for each activity.

Fig. 13 compares overfill, underfill, and substitution metrics between our approach and fixed size windowing approach for all the misclassified samples within only the multi-class windows that contain transition. As the figure shows, our model has less underfill and overfill, 28.98% and 26.28% respectively, and 0.45% less substitution, compared to the fixed size windowing approach on average over eight activities.

This proves its superiority with respect to the precision of transition point detection. As the figure shows, the fixed-size windowing approach performs very poorly; the deficit is more significant in overfill and underfill but not in substitution. This shows that most of the misclassifications come from assigning one label to all the samples of a window. This means that the (in)capability of the classifier is not the source of misclassification; instead, inappropriate signal segmentation causes the problem. The small amount of substitution in both our method and the fixed-size windowing approach shows that the classifier does not make many misclassifications with respect to one activity versus another. However, when fixed-size windowing approach assigns one label to all the samples within a window, it increases the misclassification around the transition moment, which in turn causes more overfill and underfill instances. Since most of the error ($\sim$15%) in multi-class windows is related to overfill and underfill and less is related to the substitution, we can assume that the error of detecting the exact moment of transition could be less than 9 seconds which is 15% of the total length of main windows (i.e., 60 seconds).

	S	W	R	Bi	C	Bu	T	Su
Still	81.3	6.5	0	3.5	4.5	1.6	14.6	11.3
Walk	1	84.0	7.6	8.2	10.2	3.3	7.2	7.8
Run	0.5	0	88.5	1.6	0.2	2.6	0	6.2
Bike	0.7	3.0	2.3	86.6	1.6	2.1	0	0
Car	0.3	0	0	0.1	81.5	2.6	2.0	0.7
Bus	3	1.5	1.5	0	1.2	86.9	2.4	2.2
Train	7.5	0.2	0	0	0.8	0.9	73.8	1.2
Subway	5.7	4.8	0.1	0	0	0	0	70.6

Fig. 14. The confusion matrix of detecting modes of locomotion and transportation over multi-class (i.e., transition) windows with our method (numbers are percentage).

Fig. 14 shows the confusion matrix for multi-class windows that contain transition between activities. This confusion matrix is measured per sample of data. As the figure shows, most of the activities are confused with the class still. The reason for this is to have several multi-class windows containing the still class in which overfill and underfill can cause misclassified samples.

E. Component Evaluation

In this section, we investigate the effectiveness of different parts of our model. First, we assess the effectiveness of our proposed signal images for 2D CNN as described in Section III-B-2 by comparing it to typical 2D CNN and 1D CNN where the results are shown in Fig. 15. The typical 2D CNN includes putting three axes of sensors in a 6000$*$3 image file, where the window size is 6000 corresponding to a minute of data, and apply 2D kernels. As Fig. 15 shows, the signal image proposed in this study (Section III-B-2) along with 2D convolutions improves the accuracy by 2% on average compared to the case of applying 2D convolutions to a simple signal image (typical 2D CNN). The issue with typical 2D CNN approach is that it cannot consider all possible inter-axes correlations. Moreover, our approach outperforms the most commonly used approach in the literature, which is 1D convolutions, by 2.5%. 1D convolution is not capable of capturing any inter-axes correlations, despite that such correlations contain important activity information.

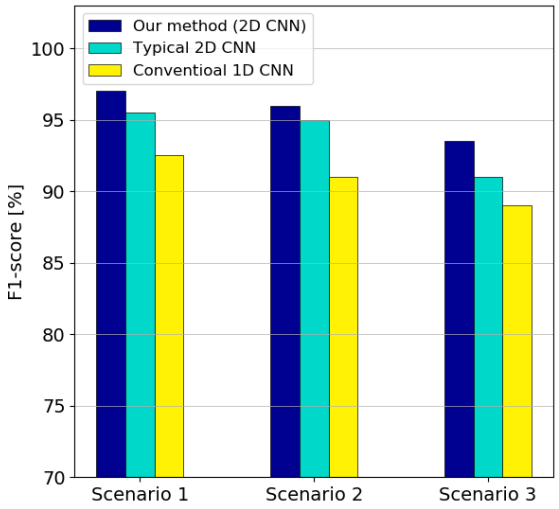


Fig. 15. Comparing the proposed 2D signal image with typical 2D and 1D convolutions.

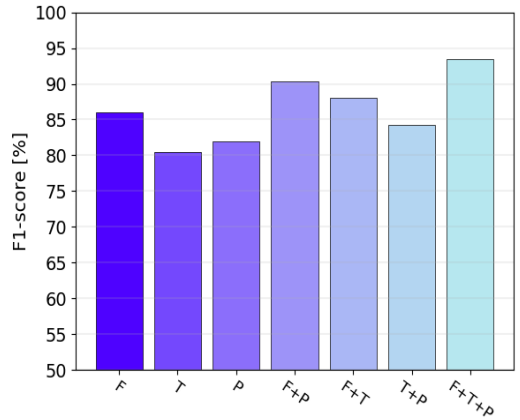


Fig. 16. Evaluation of the effect of each component of coarse-grained classifier. **F**: Hand-crafted Features; **T**: Raw Time-domain Signal; **P**: PSD.

The second investigation is devoted to the components of the coarse-grained classifier. As explained in Section III-B-2, the coarse-grained classifier receives inputs from three different sources, namely the hand-crafted features, raw time-domain signals, and the PSD. Fig. 16 shows the performance of the system with different configurations of inputs in scenario 3 where all the classes are considered separately. Based on this figure, using all three sources together achieves the best results, showing that each part of this network is necessary to obtain good results. Based on Fig. 16, using only raw time-domain data with CNN gives the lowest accuracy. Deeper assessment of this case reveals that the CNN with raw data can detect walk, run, and bike activities very well while it is incapable of distinguishing between motorized transportations as well as "still". The reason behind this is the fact that all the motorized and still activities have a similar time-domain pattern. Moreover, the signal of these activities does not have a specific pattern; instead, it is semi-constant signal dominantly. Another interesting conclusion from this figure is that hand-crafted features perform better than automated features of CNN which is in contrast to [21]. This shows that although CNN performs well in terms of automated feature extraction, it should be used very carefully since it does not work appropriately facing with signals that do not have specific distinguishable patterns. Finally, from this figure we can conclude that utilizing both hand-crafted and extracted features of CNN together is a reasonable solution as they can compensate for shortcomings of each other.

Finally, using the label arbitrator module, which utilizes RNN to model temporal order of human natural activities, improves the performance by 7%. This shows that considering the nature of human activities, which is sequential and follows certain temporal orders, is critical for building accurate activity recognition systems.

V. Conclusion

In this study we proposed a deep learning based framework for detecting activities, specifically, modes of locomotion and transportation with wearable motion sensors, along with the ability to precisely detect transitions between the activities. We improved the performance of a conventional CNN for activity recognition by: 1) proposing a new structure of signal image that allows us to consider all possible multi-axes correlations, 2) incorporating multiple outputs from expert knowledge and automated features, and 3) proposing a label arbitrator based on RNN to model temporal order of human activities. Moreover, the proposed search algorithm for finding the exact sample of transition between activities reduces the computational time significantly while it achieves a reasonable accuracy compared to the case of naïve search. The proposed methodology can be leveraged in different classification tasks where detecting the exact moment of transition in time-series data is of interest. In fact, this technique can be an alternative for fixed-size segmentation approach for analyzing sensors time-series. Our proposed methodology, if deployed on a large scale, can provide important and useful contextual information for the users to mobile applications, and can unlock many new context-aware mobile sensing, computing and application paradigms.

Acknowledgment

Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding organizations.

References

[1] S. Hemminki, P. Nurmi, and S. Tarkoma, "Accelerometer-based transportation mode detection on smartphones," in *Proc. 11th ACM Conf. Embedded Netw. Sensor Syst. (SenSys)*, 2013, p. 13.

[2] G. M. Harari, S. R. Müller, M. S. Aung, and P. J. Rentfrow, "Smartphone sensing methods for studying behavior in everyday life," *Current Opinion Behav. Sci.*, vol. 18, pp. 83–90, Dec. 2017.

[3] H. Guo, L. Chen, L. Peng, and G. Chen, "Wearable sensor based multimodal human activity recognition exploiting the diversity of classifier ensemble," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, Sep. 2016, pp. 1112–1123.

[4] H. Zhao and Z. Liu, "Human action recognition based on non-linear SVM decision tree," *J. Comput. Inf. Syst.*, vol. 7, no. 7, pp. 2461–2468, 2011.

[5] M. H. M. Noor, Z. Salcic, and K. I.-K. Wang, "Adaptive sliding window segmentation for physical activity recognition using a single tri-axial accelerometer," *Pervasive Mobile Comput.*, vol. 38, pp. 41–59, Jul. 2017.

[6] E. Guenterberg, H. Ghasemzadeh, V. Loseu, and R. Jafari, "Distributed continuous action recognition using a hidden Markov model in body sensor networks," in *Proc. Int. Conf. Distrib. Comput. Sensor Syst.*, 2009, pp. 145–158.

[7] R. Yao, G. Lin, Q. Shi, and D. C. Ranasinghe, "Efficient dense labelling of human activity sequences from wearables using fully convolutional networks," *Pattern Recognit.*, vol. 78, pp. 252–266, Jun. 2018.
[8] J. Wu, A. Akbari, R. Grimsley, and R. Jafari, "A decision level fusion and signal analysis technique for activity segmentation and recognition on smart phones," in *Proc. ACM Int. Joint Conf. Int. Symp. Pervasive Ubiquitous Comput. Wearable Comput. (UbiComp)*, 2018, pp. 1571–1578.
[9] O. Banos, J.-M. Galvez, M. Damas, H. Pomares, and I. Rojas, "Window size impact in human activity recognition," *Sensors*, vol. 14, no. 4, pp. 6474–6499, Apr. 2014.
[10] J. Wu and R. Jafari, "Orientation independent activity/gesture recognition using wearable motion sensors," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1427–1437, Apr. 2019.
[11] J. Yang, M. N. Nguyen, P. P. San, X. L. Li, and S. Krishnaswamy, "Deep convolutional neural networks on multichannel time series for human activity recognition," in *Proc. 24th Int. Joint Conf. Artif. Intell.*, 2015, pp. 3995–4001.
[12] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell, "A survey of mobile phone sensing," *IEEE Commun. Mag.*, vol. 48, no. 9, pp. 140–150, Sep. 2010.
[13] A. Brajdic and R. Harle, "Walk detection and step counting on unconstrained smartphones," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput. (UbiComp)*, 2013, pp. 225–234.
[14] R. Subramanian and S. Sarkar, "Evaluation of algorithms for orientation invariant inertial gait matching," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 2, pp. 304–318, Feb. 2019.
[15] Z. Sheng, C. Hailong, J. Chuan, and Z. Shaojun, "An adaptive time window method for human activity recognition," in *Proc. IEEE 28th Can. Conf. Electr. Comput. Eng. (CCECE)*, May 2015, pp. 1188–1192.
[16] Y. Zhang, Y. Zhang, Z. Zhang, J. Bao, and Y. Song, "Human activity recognition based on time series analysis using U-Net," 2018, *arXiv:1809.08113*. [Online]. Available: http://arxiv.org/abs/1809.08113
[17] E. A. Mosabbeb, R. Cabral, F. L. Torre, and M. Fathy, "Multi-label discriminative weakly-supervised human activity recognition and localization," in *Proc. Asian Conf. Comput. Vis.*, 2014, pp. 241–258.
[18] R. Mohamed, M. N. S. Zainudin, M. N. Sulaiman, T. Perumal, and N. Mustapha, "Multi-label classification for physical activity recognition from various accelerometer sensor positions," *J. Inf. Commun. Technol.*, vol. 17, no. 2, pp. 209–231, 2020.
[19] M. Zeng *et al.*, "Convolutional neural networks for human activity recognition using mobile sensors," in *Proc. 6th Int. Conf. Mobile Comput., Appl. Services*, 2014, pp. 197–205.
[20] 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.
[21] F. Ordóñez and D. Roggen, "Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition," *Sensors*, vol. 16, no. 1, p. 115, Jan. 2016.
[22] J. Wu, L. Sun, and R. Jafari, "A wearable system for recognizing American sign language in real-time using IMU and surface EMG sensors," *IEEE J. Biomed. Health Inform.*, vol. 20, no. 5, pp. 1281–1290, Sep. 2016.
[23] H. Mhaskar, Q. Liao, and T. Poggio, "When and why are deep networks better than shallow ones?" in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 1–7.
[24] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.
[25] H. Gjoreski *et al.*, "The university of Sussex-Huawei locomotion and transportation dataset for multimodal analytics with mobile devices," *IEEE Access*, vol. 6, pp. 42592–42604, 2018.
[26] S. Aminikhanghahi and D. J. Cook, "Enhancing activity recognition using CPD-based activity segmentation," *Pervasive Mobile Comput.*, vol. 53, pp. 75–89, Feb. 2019.
[27] S. Aminikhanghahi, T. Wang, and D. J. Cook, "Real-time change point detection with application to smart home time series data," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 5, pp. 1010–1023, May 2019.
[28] A. Akbari, J. Wu, R. Grimsley, and R. Jafari, "Hierarchical signal segmentation and classification for accurate activity recognition," in *Proc. ACM Int. Joint Conf. Int. Symp. Pervasive Ubiquitous Comput. Wearable Comput. (UbiComp)*, 2018, pp. 1596–1605.
[29] L. Wang, H. Gjoreskia, K. Murao, T. Okita, and D. Roggen, "Summary of the Sussex-Huawei locomotion-transportation recognition challenge," in *Proc. ACM Int. Joint Conf. Int. Symp. Pervasive Ubiquitous Comput. Wearable Comput. (UbiComp)*, 2018, pp. 1521–1530.
[30] M. Gjoreski *et al.*, "Applying multiple knowledge to Sussex-Huawei locomotion challenge," in *Proc. ACM Int. Joint Conf. Int. Symp. Pervasive Ubiquitous Comput. Wearable Comput. (UbiComp)*, 2018, pp. 1488–1496.
[31] L. Wang, H. Gjoreski, M. Ciliberto, S. Mekki, S. Valentin, and D. Roggen, "Benchmarking the SHL recognition challenge with classical and deep-learning pipelines," in *Proc. ACM Int. Joint Conf. Int. Symp. Pervasive Ubiquitous Comput. Wearable Comput. (UbiComp)*, 2018, pp. 1626–1635.
[32] P. Widhalm, M. Leodolter, and N. Brändle, "Top in the lab, flop in the field?: Evaluation of a sensor-based travel activity classifier with the SHL dataset," in *Proc. ACM Int. Joint Conf. Int. Symp. Pervasive Ubiquitous Comput. Wearable Comput. (UbiComp)*, 2018, pp. 1479–1487.
[33] J. A. Ward, P. Lukowicz, and H. W. Gellersen, "Performance metrics for activity recognition," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 1, pp. 1–23, Jan. 2011.

Ali Akbari (Graduate Student Member, IEEE) received the B.Sc. and M.Sc. degrees in electrical and bio-electrical engineering from the Sharif University of Technology, Tehran, Iran, in 2013 and 2015, respectively. He is currently pursuing the Ph.D. degree in biomedical engineering with Texas A&M University. His research interests include artificial intelligence, machine learning, deep learning, signal processing, and embedded systems for applications in remote, smart and connected health.

Roozbeh Jafari (Senior Member, IEEE) received the Ph.D. degree in computer science from UCLA. He was a Post-Doctoral Fellowship with UC-Berkeley. He is currently a Professor of Biomedical Engineering, Computer Science and Engineering and Electrical and Computer Engineering with Texas A&M University. He has published over 180 articles in refereed journals and conferences. He has raised more than $77M for research with $18M directed towards his lab. His research has been funded by the NSF, NIH, DoD (TATRC), AFRL, AFOSR, DARPA, SRC, and industries such as Texas Instruments, Tektronix, Samsung, and Telecom Italia. His research interests include wearable computer design and signal processing. Dr. Jafari is also the recipient of the NSF CAREER award (2012), the IEEE Real-Time and Embedded Technology and Applications Symposium best paper award (2011), the Andrew P. Sage best transactions paper award (2014), the ACM Transactions on Embedded Computing Systems best paper award (2019), and the Outstanding Engineering Contribution award from the College of Engineering, Texas A&M (2019). He has been named as Texas A&M Presidential Fellow (2019). He serves on the editorial board for the IEEE TRANSACTIONS ON BIOMEDICAL CIRCUITS AND SYSTEMS, the IEEE SENSORS JOURNAL, the IEEE INTERNET OF THINGS JOURNAL, the IEEE JOURNAL OF BIOMEDICAL AND HEALTH INFORMATICS, the IEEE OPEN JOURNAL OF ENGINEERING IN MEDICINE AND BIOLOGY, and *ACM Transactions on Computing for Healthcare*. He has served as the General Chair and the Technical Program Committee Chair for several flagship conferences in the area of wearable computers. He serves on scientific panels for funding agencies frequently. He serves as a Standing Member of the NIH Biomedical Computing and Health Informatics (BCHI) study section from 2017 to 2021, and the Chair of the NIH Clinical Informatics and Digital Health (CIDH) study section (2020).