An Energy-Aware Method for the Joint Recognition of Activities and Gestures Using Wearable Sensors

Joseph Korpela[1], **Kazuyuki Takase**[1], **Takahiro Hirashima**[1], **Takuya Maekawa** [1],
Julien Eberle[2], **Dipanjan Chakraborty**[3], and **Karl Aberer**[2]
[1]Osaka University, {joseph.korpela, takase.kazuyuki, hirashima.takahiro, maekawa}@ist.osaka-u.ac.jp
[2]Ecole Polytechnique Federale de Lausanne, {julien.eberle, karl.aberer}@epfl.ch
[3]IBM Research India, dipanjan.chakraborty@gmail.com

ABSTRACT

This paper presents an energy-aware method for recognizing time series acceleration data containing both activities and gestures using a wearable device coupled with a smartphone. In our method, we use a small wearable device to collect accelerometer data from a user's wrist, recognizing each data segment using a minimal feature set chosen automatically for that segment. For each collected data segment, if our model finds that recognizing the segment requires high-cost features that the wearable device cannot extract, such as dynamic time warping for gesture recognition, then the segment is transmitted to the smartphone where the high-cost features are extracted and recognition is performed. Otherwise, only the minimum required set of low-cost features are extracted from the segment on the wearable device and only the recognition result, i.e., label, is transmitted to the smartphone in place of the raw data, reducing transmission costs. Our method automatically constructs this adaptive processing pipeline solely from training data.

Categories and Subject Descriptors: H.3.4 Information storage and retrieval: Systems and software.

Keywords: Activity recognition; gesture recognition; pattern classification.

INTRODUCTION

Recent smart wearable devices such as smart watches contain a variety of integrated sensors, e.g., accelerometers, and provide a convenient platform for recognizing the actions performed by their users. These actions can represent long-duration activities, such as running or washing dishes, or short hand gestures, such as clockwise or left-to-right gestures. With the proliferation of these devices, the application of their sensor data to gesture and activity recognition has become an active area of research, supporting various applications such as fall detection, physical-activity tracking, medical treatment monitoring, and mobile-device input via wearable sensors [8, 11].

Newer wearable devices, e.g., smart watches, have limited computation power and battery life when compared with larger devices, e.g., smartphones. Because of their limited computation power, many studies use wearable devices as simple collection nodes, with the device transmitting either raw data or extracted features to a larger device that then runs a recognition algorithm. For example, when recognizing a hand gesture, a wrist-worn device may transmit the raw accelerometer data to a smartphone, which then runs a complex algorithm, e.g., dynamic time warping (DTW), to recognize the gesture. Since this study is investigating energy-aware recognition, we focus on wearable devices with low-power CPUs that can achieve long battery lives, such as the Pebble Smartwatch[1]. Such low-power wearable devices have been used in many previous studies on energy-aware techniques for activity and gesture recognition [3, 4, 7, 10, 14].

Previous work on energy-aware recognition methods has generally focused on either long-duration activities or short hand gestures. Studies focused on long-duration activities have focused on techniques such as reducing the sampling rates of sensors [3, 7, 16] and powering down sensors when not in use [5]. Many of these studies assume the use of several sensors, and derive much of their savings by exploiting the redundancy and presence of unnecessary sensors in such situations [4]. Furthermore, many studies assume applications with long-term actions that have many consecutive instances of data for each action, and allow for short segments of data to be missed entirely by the powered-down sensors. Therefore, these techniques are not suited for gesture recognition, where such consecutive instances of the same actions cannot be assumed [16]. On the other hand, studies focused on short hand gestures have mainly focused on reducing costs by filtering out data that are unlikely to contain target actions, i.e., gestures, reducing energy consumption by reducing the amount of data transmitted from the wearable device collecting the data to the main device running the recognition algorithm [10, 14]. However, these energy-aware gesture recognition techniques cannot be applied to long-duration activities, as their cost savings come from exploiting the sparsity of the target actions in the sensor data. This assumption of sparsity is generally not true for long-duration activities, which are often present in much of the sensor data. To the best of our knowledge, there are no studies on energy-aware recognition designed to cope with both activities and gestures.

In this study, we propose a new model for energy-aware recognition that can be applied to time series data containing both activities and gestures. The main feature of our

[1]www.getpebble.com

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ISWC '15, September 7–11, 2015, Osaka, Japan.
Copyright 2015 © ACM 978-1-4503-3578-2/15/09...$15.00.
http://dx.doi.org/10.1145/2802083.2808400

model is its ability to reduce energy consumption by adaptively changing its behavior according to input data segments. Our method reduces computation costs by dynamically extracting only the features needed for classification from the input sensor data, and reduces transmission costs by classifying some classes of actions on board the wearable device, transmitting their estimated labels in place of the raw sensor data. The model is built automatically from training data using a modified version of the random forests algorithm, with the tree construction influenced by computation and transmission costs. In the resulting tree, feature extraction for simple actions that can be recognized using less costly features, e.g., running, is conducted exclusively by the wearable device, while feature extraction for complex actions that require more costly features than can be extracted by the device, e.g., a clockwise gesture, is conducted by both the wearable device and the smartphone. Extracting features on the wearable device allows us to recognize some classes of actions on the device, i.e., without transmitting raw data to the smartphone, while delegating more costly features to the smartphone allows us to compute features that are too complex for the wearable device's limited RAM and CPU. Using this model, we can adaptively select which features to extract from an input sensor-data segment, reducing computation and transmission costs while maintaining high recognition accuracy.

The contributions of this study are that (1) we propose a method for reducing the computation and transmission costs for the joint recognition of activities and gestures using a wearable sensor by using a tree-structured feature extraction model, (2) we propose a method for automatically determining the structure for the recognition model that balances accuracy with computation and transmission costs, and (3) we evaluate our method using 60 sessions of real-life sensor data.

RELATED WORK

Activity Recognition with Decision Trees

The tree-structured model used in this study is based on the C4.5 decision tree algorithm [13]. The C4.5 algorithm learns decision trees from the top down, determining the optimal node to create at each point in the tree by first finding the optimal split of training instances to use for each feature at that point, and then choosing the feature whose split has the highest gain ratio among all possible splits. This gain ratio is based on the Shannon entropy and indicates how well a given split distinguishes between the various classes in the data. The standard C4.5 decision tree algorithm is designed to optimize the tree to give high classification accuracy. In our proposed method, we keep much of the basic structure of the C4.5 algorithm, but we also incorporate energy consumption into the algorithm in order to address our desired balance between cost and accuracy.

Energy-aware Activity Recognition

Several existing studies have examined energy-aware activity recognition. French et al. investigated trade-offs between costs and accuracy when determining an application's overall architecture [3], while Lukowicz et al. proposed reducing transmission costs by shifting the feature extraction process from a main processing node to sensor nodes [9]. Studies such as these focus on general techniques to use when selecting the sampling rates, feature types, and/or recognition algorithms to use in energy-aware applications, unlike our research which proposes a recognition algorithm that dynamically adjusts the processes executed based on the sensor data collected. Ghasemzadeh et al. went beyond general design guidelines by developing a recognition algorithm based on the minimum cost dominating set problem, which can be used to determine a subset of features that can achieve sufficient recognition accuracy at reduced computation cost [4]. While our research also focuses on reducing costs by computing only subsets of features, we propose a dynamic model that allows the subset to vary for each activity recognized, whereas Ghasemzadeh et al. generated a static feature set used for all activities.

Another technique commonly used in energy-aware activity recognition involves predicting the most likely subsequent activity based on the most recently recognized activity [5, 7, 16], deactivating sensors and adjusting sampling rates based on those predictions. However, this technique does not work when conducting gesture recognition, as it assumes that any sensor data missed due to an incorrect prediction will not significantly affect recognition accuracy, relying on the fact that the subsequently collected data in long-duration activities would be for the same activity as that which was missed.

Energy-aware Gesture Recognition

Existing work on energy-aware gesture recognition has focused on techniques for filtering sensor data to reduce the amount of processing done on segments that are unlikely to contain gestures [10, 14]. These studies try to find potential gestures using light-weight sequence analysis, achieved by using manually constructed processing pipelines. In contrast, our method automatically constructs a tree-structured processing model (i.e., pipeline) based on the training data. Furthermore, these existing studies implement the gesture recognition pipeline independently from activity recognition, achieving cost reductions by discarding all data that do not correspond to gestures. Our work instead investigates the joint recognition of activities and gestures, and so the light-weight sequence analysis in our method is used both to identify potential gestures, and to recognize activities as well.

METHOD

Overview

In this study, we propose an energy-aware method for the joint recognition of activities and gestures from accelerometer data using a tree-structured classifier. Our tree-structured classifier is automatically constructed so that low-cost features are located at shallower levels of the tree, allowing a subset of actions to be recognized by the tree without the need to extract high-cost features for those actions.

Fig. 1 shows an example of such a classifier, with *DTW-Y* representing a high-cost feature and all other features representing low-cost features. Using this classifier, the wearable device will first extract the feature *Mean-X* from each segment

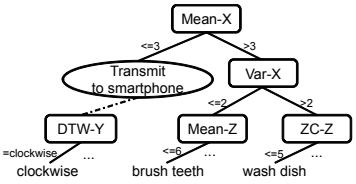

Figure 1. Example tree-structured classifier.

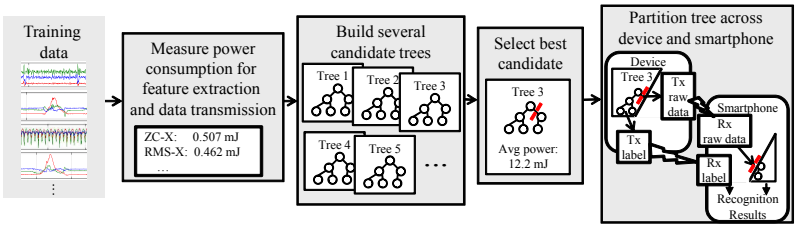

Figure 2. Overview of proposed method for constructing an energy-aware recognition model from labeled training data.

of sensor data. If the *Mean-X* value is less than or equal to 3, then the wearable device will transmit the raw sensor data to the smartphone where *DTW-Y* will be extracted to complete the recognition process. On the other hand, if *Mean-X* is greater than 3, the wearable device will extract *Var-X* followed by *Mean-Z* or *ZC-Z*, and then transmit the corresponding recognition result, e.g., "wash dish," to the smartphone.

Fig. 2 shows an overview of our method for constructing and partitioning the tree-structured classifier. First, we use labeled training data to measure in advance the energy consumed by the sensor device and the smartphone when extracting features and when transmitting data and labels from the wearable device to smartphone. We next construct a set of candidate decision trees using a technique based on the random forests algorithm to generate a variety of trees in which we incorporate energy consumption to generate lower-cost trees. Several candidate trees are generated because we are seeking to balance two objective functions, energy-consumption costs and recognition accuracy, and by generating several trees we are able to obtain trees with different balances between these two functions. Next, we select a single tree from these candidate trees that has both low energy consumption and acceptable recognition accuracy. We then partition this tree across the wearable device and the smartphone, with nodes representing features that can be computed by the wearable device included in the device's partition of the tree, and nodes that must be computed by the smartphone included in the smartphone's partition.

Hardware

We evaluate our method using an off-the-shelf smartphone and a wearable device developed for this study. Fig. 3 shows the internal components of the wearable device, which has a KXR94-2050 3-axis accelerometer (sampled at 100Hz), an Atmel ATmega328P 20 MHz MCU with 2 KB RAM, a connectBlue OLS425 BLE module, and a 3.7V 400mAh battery. These low-energy components were selected to achieve a two-week battery life. The smartphone used in this study was a Nexus 5 smartphone with a 2.26 GHz quad-core CPU, 2 GB of RAM, and a 3.8V 2300mAh battery.

Recognition Features

The decision tree used in our method is trained using several features extracted from 1-sec windows of accelerometer data. We use several simple features, selected due to their common use in activity and gesture recognition studies, along with a feature based on the output of a DTW-based k-nearest-neighbor (k-NN) classifier, selected due to its good performance at gesture recognition.

[Zero-crossings]: A count of the number of the zero-crossings (ZC) in a segment of accelerometer data, with a value computed for each axis (ZC-X, ZC-Y, ZC-Z).

[Root mean square]: The root mean square (RMS) for each axis's data (RMS-X, RMS-Y, RMS-Z).

[Mean]: The mean for each axis's data (Mean-X, Mean-Y, Mean-Z).

[Variance]: The variance for each axis's data (Var-X, Var-Y, Var-Z).

[FFT-based energy]: The Fast Fourier Transform (FFT) energy, which is calculated by summing the magnitudes of the squared discrete FFT components, excluding the DC component, for each axis's data (Energy-X, Energy-Y, Energy-Z).

[DTW]: A nominal value, e.g., "run," corresponding to the single nearest neighbor returned by a k-NN classifier that uses DTW [15] as the distance metric, with k-NN classifiers built for each axis. Since running a DTW-based k-NN classifier requires more RAM than is available in our wearable device, this feature is extracted solely by our study's smartphone. Furthermore, in order to reduce the costs of running the k-NN classifiers, we also run them using piecewise aggregate approximation [6] to reduce the amount of sensor data used in the DTW comparisons down to 50%, 25%, or 12.5% of the raw sensor data, resulting in 12 DTW-based classifiers in total, one for each axis for each amount of sensor data (e.g., DTW-X-100, DTW-X-50, DTW-Y-25, DTW-Z-12.5, etc.).

Measuring Energy Consumption

Using the wearable device and smartphone described above, we first measure the energy-consumption costs for feature extraction using the features listed above, along with the cost of transmitting data from the device to the smartphone. Table 1 shows the average feature-extraction and transmission costs in mJ per 1-sec window of data, with the *Label* row referring to the transmission of the recognition results from the wearable device to the smartphone, i.e., no raw data, and the following data transmission rows referring to the transmission of 12.5%, 25%, 50%, and 100% of the raw sensor data. These costs do not include the baseline energy consumption of the devices, as measured when no feature extraction or data transmission is executed, and all transmission costs assume an active connection.

Tree Nodes

Each node in our decision tree corresponds to a single feature, e.g., ZC-Y or DTW-X-50, with the nodes for DTW-based features using nominal values and the nodes for all other features

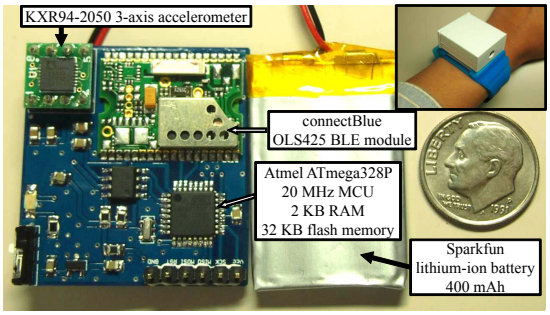

Figure 3. Prototype wearable device used in this study, including a view of the internal components of the device.

Table 1. Average energy consumption for feature extraction and data transmission from 1-sec windows of data on the Nexus 5 smartphone and wearable device.

Feature Extraction

	Cost (mJ)	
	Device	Smartphone
Mean	0.000329	0.000359
Var	0.000575	0.001034
ZC	0.000530	0.000291
RMS	0.000497	0.000312
Energy	0.001951	0.006345
DTW-12.5		0.101
DTW-25		0.276
DTW-50		0.823
DTW-100		2.940

Data Transmission

	Cost (mJ)	
	Device	Smartphone
Label	0.109	8.485
Data-12.5	0.148	8.564
Data-25	0.186	8.642
Data-50	0.264	8.800
Data-100	0.418	9.115

using numeric values. In the case of nodes for numeric feature values, the nodes represent binary splits on the features' values. These split thresholds are set using the training data to choose a threshold that maximizes the gain ratio for the split. Fig. 1 shows an example of such a split where the Mean-X node represents a split on Mean-X values with data segments with Mean-X values greater than 3 going to the Var-X node, and all other segments going to the DTW-Y-50 node.

The nodes for nominal features, i.e., DTW-based features, also represent splits on the features' values, but have two different split types possible.
- $Binary$**:** One branch for a single nominal value and a second branch for all remaining values, e.g., two branches: "run" and "all others."
- $Multiway$**:** A separate branch for each nominal value, e.g. four branches: "run," "brush teeth," "clockwise," and "left-to-right."

We investigate the results for the proposed method using each of these split types in the *Evaluation* section.

Generating Energy-aware Decision Trees

Normally, the C4.5 decision tree algorithm would construct trees with high recognition accuracy by generating splits that maximize the gain ratio at each node. However, in our method, we want to construct trees that have low energy consumption as well as high recognition accuracy. In this study, we generate trees that balance these two requirements by introducing elements of the random forests algorithm [1] into our tree construction. Using this modified version of random forests, we build a forest of varied lower-cost trees and then pick from those trees a single tree that gives an optimal combination of cost and accuracy.

Forest Construction

Our method starts by generating a forest of trees from our training data using our modified version of the random forests algorithm. The random forests algorithm is an ensemble learning method in which a large number of trees are constructed from a single set of training data by introducing random variation into each tree's construction. This random variation is introduced by choosing the optimal split to use for each node in the tree from only a randomly selected subset of the total set of features available [1]. Like in the random forests algorithm, each tree in our forest is built independently, with each node choosing the optimal feature to use for its split from only a subset of the features. Unlike in the random forests algorithm, our method biases the probability for including each feature in a subset based on the feature's estimated energy consumption.

When estimating the energy-consumption cost for each feature, if the feature is a DTW-based feature or if a node for a DTW-based feature lies on the path from the current node to the root node, then the feature's cost is estimated as the energy consumed when extracting the feature on the smartphone, given in the *Smartphone* columns in Table 1. Additionally, if the feature is a DTW-based feature and no node on the path from the current node to the root node uses a DTW-based feature, then the feature's cost also includes the energy consumed when transmitting the raw data from the wearable device to smartphone, given in Table 1. In all other cases, the feature's cost is estimated as the energy consumed when extracting the feature on the wearable device, given in the *Device* column in Table 1.

Using these costs, the probability with which feature f is chosen is expressed by $Pr(f) = 1 - c\frac{cost_f}{\max_{i \in F}(cost_i)}$ where $cost_f$ is the cost for feature f, F is the set of all features, and c is a constant value that controls the importance of energy-consumption costs. This bias in the probabilities causes the tree to make use of costly features less often, generally reducing the costs of trees. Using these biased subsets of features, we can then generate forests of decision trees which attempt to maximize classification accuracy while incorporating energy-consumption costs.

Optimal Tree Selection

After using the method described above to generate a forest of trees, we then choose from that forest a tree that has an acceptable cost along with good accuracy. First, we estimate the cost C_p of each path through the tree using

$$C_p = wd_p + (1 - w)s_p \quad (1)$$

where d_p is the sum of transmission and feature extraction costs for path p for the wearable device, s_p is the sum of transmission and feature extraction costs for path p for the smartphone, and w is a constant weight ($0 < w < 1$) that sets the relative importance of the wearable device's energy consumption when compared to that of the smartphone. We then estimate the cost C_t of each tree as $C_t = \sum_p \frac{|I_p|C_p}{|I|}$ where

$|I_p|$ is the number of training instances using path p through the tree and $|I|$ is the count of all training instances.

We estimate the accuracy for each tree as being inversely proportional to the tree's size (i.e., # tree nodes), since larger trees are more likely to overfit the training data while smaller trees are more general and more able to correctly classify test instances, as has been observed in previous research on decision trees [12] and was again observed in this study's examination of a series of 210 trees generated using the method of randomized feature selection, in which there was a Pearson correlation coefficient of -0.861 between average F-measure and tree size. Using these estimates of cost and accuracy, we then choose the optimal tree from a forest as the smallest tree with an acceptable cost that contains at least one DTW-based feature node. Note also that the threshold to use for the acceptable cost is dependent on the end application, and therefore we investigate a range of such costs in the *Evaluation* section.

Partitioning the Tree

The final step in our method for constructing an energy-aware tree is to partition the tree across the wearable device and smartphone. To do so, we start by setting the wearable device's partition as being the whole tree and from this partition remove all subtrees that have DTW as a root node. Each of these subtrees becomes a partition for use on the smartphone, and each is replaced in the wearable device's partition by a node signaling the transmission of raw sensor data from the wearable device to the smartphone.

Recognizing Test Data

Using a decision tree output by the above method, we can then classify test instances into activity and gesture classes. For each test instance, if we are able to recognize the instance using only the wearable device's partition of the tree, i.e., without reaching a node that signals the transmission of raw sensor data to the smartphone, we then transmit the label corresponding to the recognized activity/gesture class to the smartphone. If, on the other hand, we reach a node that signals the transmission of raw data, we then transmit the raw data and complete the recognition process using the corresponding partition on the smartphone.

In our method, we also examine an additional variation on the C4.5 decision tree which relates to how feature values are used when classifying an instance of data. In the original C4.5 decision tree algorithm, the feature values are only used for local decisions in the tree, with the final decision for the class of an instance determined by the class distribution of the training instances assigned to a leaf node. However, since in our research we use the output of a k-NN classifier as a feature in our decision trees, we also examine the effect of increasing the weight given to the output of the k-NN classifier beyond its use in determining a single split within the decision tree. During classification, any test instance that does not pass through a DTW-based node is classified using the same method as that of the C4.5 decision tree. On the other hand, when a test instance passes through DTW-based nodes, the results from those k-NN classifiers are combined with the results from the decision tree.

Table 2. Activities/gestures performed in the 50 sessions of our primary data set, along with proportions of instances included in the sensor data. This table does not reflect the additional 20 sessions used to evaluate simultaneous gestures and activities.

	activity		gesture
A	run (15.4%)	G	left to right (2.2%)
B	draw on whiteboard (16.3%)	H	right to left (2.2%)
C	wash dishes (16.0%)	I	clockwise (3.0%)
D	write in notebook (16.1%)	J	counter-clockwise (3.0%)
E	brush teeth (15.9%)	K	down to up (2.4%)
F	none (7.6%)		

In the original C4.5 decision tree algorithm, the probability of each class c given a test instance t is expressed by $Pr(c|t) = \frac{|Ic|}{|I|}$ where $|Ic|$ is the count of training instances assigned to the leaf node with class c and $|I|$ is the count of all training instances assigned to the leaf node. In our method, we increase the effect of a k-NN classifier k for an instance t by adjusting the probabilities of classes in this distribution to

$$Pr(c|t) = \frac{\frac{|Ic|}{|I|} + \sum_{k \in K} Pr(c|k(t))}{\sum_{k \in K} R_k + 1} \quad (2)$$

with

$$Pr(c|k(t)) = \begin{cases} R_k, & \text{if } k(t) = c \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

where K is the set of k-NN classifiers used to make decisions in the tree for test instance t, $k(t)$ is the output of each k for test instance t, and R_k is the reliability of each k, set as the average F-measure for k when classifying training data.

EVALUATION

Data Set

The primary data set used in this study consists of 50 sessions of accelerometer data collected from five research participants. All data were collected using a 3-axis accelerometer sampled at 100Hz mounted on each participant's right wrist. Feature extraction was conducted using 1-sec sliding windows with 50% overlap, with the window size chosen based on the average size of gestures in our data set (mean 1016 ms, standard deviation 306 ms). Each participant conducted 10 sessions of data collection, performing the eleven activities and gestures listed in Table 2 in an arbitrary order, with each session containing instances of each activity and gesture. These activities and gestures were selected from among those used in previous accelerometer-based activity and gesture recognition studies. Participants were instructed on the expected motion for each activity and gesture; however, they were allowed to perform the motions at their own pace. In the case of write on whiteboard and write in notebook, the participants were allowed to write arbitrary text during their activity, with three participants writing in Japanese, one writing in Chinese, and one writing in English.

Because the data set suffers from a large class imbalance (see Table 2 for the proportion of instances belonging to each class), we also oversampled the training data prior to building our decision trees using the synthetic minority over-sampling technique (SMOTE) [2]. Using SMOTE, we oversampled all but the top majority class so that the proportion of training instances from each class was equal.

Table 3. Energy-consumption costs and recognition accuracies for Act, DTW and $Tree$. Energy estimates are per 1-sec data segment.

	cost (mJ)		avg. F-measure		
	device	smartphone	overall	activities	gestures
Act	0.119	8.485	0.914	0.949	0.845
DTW	0.418	17.935	0.935	0.935	0.934
$Tree$	0.345	11.168	0.956	0.974	0.936

Table 4. Energy-consumption costs and recognition accuracies for $Proposed$-B and $Proposed$-M. Energy estimates are per 1-sec data segment.

$Split$-$type$ (threshold)	cost (mJ)		avg F-measure: C4.5 estimates w/ k-NN estimates			avg F-measure: C4.5 estimates only
	device	smartphone	overall	activities	gestures	overall
$Proposed$-M (11 mJ)	0.237	9.741	0.956	0.969	0.941	0.950
$Proposed$-M (9.05 mJ)	0.151	8.768	0.951	0.969	0.929	0.944
$Proposed$-M (8.75 mJ)	0.127	8.575	0.943	0.964	0.918	0.927
$Proposed$-B (11 mJ)	0.227	9.965	0.965	0.974	0.954	0.955
$Proposed$-B (9.05 mJ)	0.147	8.792	0.960	0.967	0.951	0.949
$Proposed$-B (8.75 mJ)	0.126	8.583	0.919	0.949	0.883	0.782

Along with the primary 50 sessions of data, an additional 20 sessions of data were collected to evaluate recognition accuracy when gestures are conducted simultaneously with activities, including 10 sessions containing only *walking* activity and 10 sessions containing both *walking* activity and gestures conducted simultaneously with *walking* activity. The results for these additional 20 sessions of data are reported in the *Simultaneously Conducting Activities and Gestures* section.

Evaluation Methodology

We conducted our evaluation using "leave-one-session-out" cross validation, using the following four methods:

- Act: This method conducts recognition using only the low-cost features that can be extracted on the wearable device, i.e., non-DTW-based features. All actions are recognized on board the device and only the recognition results are transmitted to the smartphone.
- DTW: This method conducts recognition using only DTW-based k-NN classifiers. The wearable device transmits 100% of the raw sensor data to the smartphone where a k-NN classifier is run for each axis's data, with the final result determined based on a majority vote.
- $Tree$: This method conducts recognition using a C4.5 decision tree that includes all of the features, i.e., including the DTW-based features. This is an unconstrained version of the proposed method with energy-consumption costs ignored when selecting features during tree construction, producing trees that attempt to maximize recognition accuracy.
- $Proposed$: This is the proposed method, with $Proposed$-M referring to the use of $Multiway$ splits and $Proposed$-B referring to the use of $Binary$ splits. The value for w in Equation 1 is set to 0.5.

Each of these methods assumes that all recognition results are needed on the smartphone immediately after data collection, with all methods transmitting either raw sensor data or recognition results for each window of sensor data collected. Recognition accuracy for each of the above methods is evaluated using the macro-averaged F-measure, calculated based on the recognition results per window of data. In the case of windows that overlap multiple actions, the true class is set using the true class at the center of the window.

Results

Baseline Methods

Table 3 shows the results for Act, DTW, and $Tree$. In terms of recognition accuracy, Act has the worst overall recognition accuracy. Its overall F-measure of 0.914 is due to the inability of the activity recognition features (i.e., low-cost features) to recognize gestures accurately, with Act's F-measure for gestures at only 0.845. On the other hand, Act's energy-consumption costs for both the wearable sensor and the smartphone were much lower than those for DTW or $Tree$. The DTW method achieved a higher overall accuracy than Act, but had the worst results in regards to activity recognition, with an F-measure for activities of only 0.935. DTW also had the highest costs of the three methods, due to the high cost of transmitting raw data from the wearable device to the smartphone coupled with the high cost of running DTW-based k-NN classifiers. With an overall F-measure of 0.956, the $Tree$ method had the best overall accuracy of the three methods, benefiting from its ability to use the activity recognition features together with the DTW-based k-NN classifiers when conducting recognition. While $Tree$ had lower energy-consumption costs when compared to DTW, the costs were still higher than those of the Act method, with $Tree$'s costs for the device almost three times higher than those for Act.

Proposed Method

In the proposed method, we are able to set a threshold cost when running our algorithm, with the trees output by the algorithm limited to a cost below the given threshold. Fig. 4 shows the changes in the average F-measure and energy consumption costs as we vary this cost threshold when using the $Proposed$-M method. At higher thresholds (e.g., 13 mJ), the results are approximately the same as with the $Tree$ method, since high thresholds do not require the algorithm to give much importance to energy-consumption costs when generating trees. As the threshold is reduced, recognition accuracy stays relatively high while energy-consumption costs are significantly reduced. For example, at a threshold of 9.5 mJ, the recognition accuracy is still 0.955, while the energy consumption for the wearable device is reduced by 50%.

Comparing the results for the proposed method in Table 4 ($Proposed$-M) to those for the baseline methods in Table 3, we see that when optimizing for high recognition accuracy (threshold 11 mJ), the proposed methods accuracy of 0.956 is equal to that of the most accurate of the baseline methods ($Tree$), while consumption costs are reduced by 31.5% for the wearable device and by 12.8% for the smartphone. Additionally, at that threshold (11 mJ), $Proposed$-M outperforms DTW in both activity and gesture recognition while using only about 55% as much energy for both the wearable device and the smartphone. When optimizing for low energy consumption (threshold 8.75 mJ), the costs for the proposed method relative to the costs of the lowest costing of the baseline methods (Act) are only 6.5% higher for the wearable de-

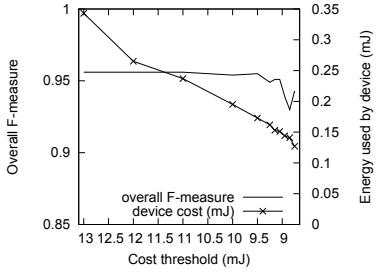

Figure 4. Transition in the overall F-measure and energy-consumption costs for the wearable device when the cost threshold is varied for $Proposed$-M.

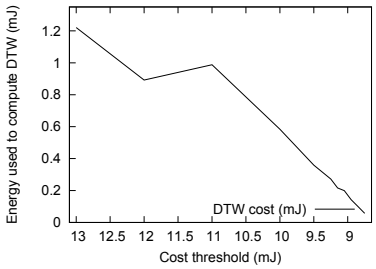

Figure 5. Transition in the energy consumed to compute DTW when the cost threshold is varied for $Proposed$-M.

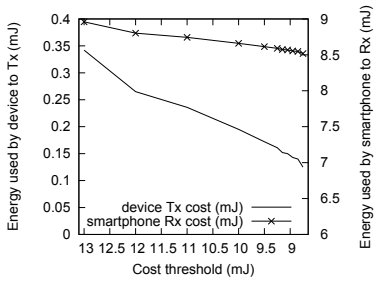

Figure 6. Transition in the energy consumed to transmit data when the cost threshold is varied for $Proposed$-M.

vice and 1.1% higher for the smartphone, while the proposed method achieves a 3.2% higher recognition accuracy.

This study also included an examination of the effect of window size on recognition accuracy when using the proposed method, in which we varied the window size between 500 to 2000 ms. While recognition accuracy for the activities showed little change across this range, i.e., from 0.972 at 500 ms to 0.975 at 2000 ms, recognition accuracy for gestures generally increased across the range, increasing from 0.837 to 0.936 when moving from 500 ms to 1000 ms and from 0.936 to 0.964 when moving from 1000 ms to 2000 ms. However, it should be noted that since most applications of gesture recognition require close to real-time recognition results, the additional recognition delay introduced by 2000 ms windows would negate the benefit gained from the increased recognition accuracy for most applications.

Energy Consumption

Our method achieves lower energy consumption by reducing the energy used to compute DTW and by reducing the energy used to transmit raw sensor data. Fig. 5 shows how the energy used to run the DTW-based k-NN classifiers is reduced by over 90% at lower thresholds. Similarly, Fig. 6 shows how the energy used for data transmission is reduced by about 40% for the wearable device and by about 5% for the smartphone.

Fig. 7 shows how the energy used for recognition is more greatly reduced when recognizing activities (A - F in Table 2) than when recognizing gestures (G - K in Table 2). For example, energy use for the device is reduced by 62.0% for activities when going from a threshold of 13 mJ to a threshold of 8.75 mJ, but by only 45.9% for gestures. This indicates that our model is selectively executing the high-cost features, using low-cost features at shallower nodes to decide which data segments require high-cost features for recognition. Because our method could accurately recognize many activity segments using only low-cost features, it mostly limited the use of high-cost features to recognizing gesture segments. When comparing the average energy used for feature extraction for shallower nodes (i.e., the first two layers of the tree) from $Tree$ with those from $Proposed$-M (threshold 8.75 mJ), we found that $Proposed$-M used as little as 6.8% as much energy for these nodes, with an average of 0.928 mJ used by $Tree$ and 0.063 mJ used by $Proposed$-M, showing that the trees constructed by our method used low-cost features at shallower nodes.

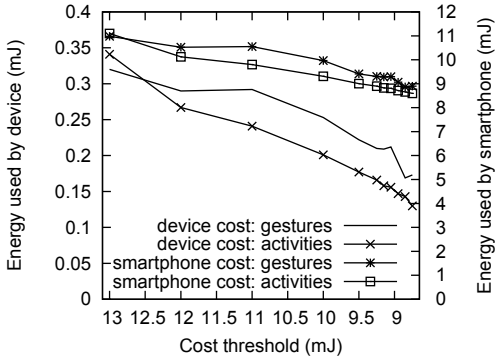

Figure 7. Average energy use by wearable device and smartphone for recognizing activities and gestures when the cost threshold is varied for $Proposed$-M.

Additionally, we estimated the battery life for each device using the figures given in Table 3 and in Table 4 ($Proposed$-M with threshold 8.75 mJ), based on the energy consumed when continuously detecting the instances found in the test data, with the estimates including the baseline energy consumption for each device assuming no other processes are run. We estimate that our method can achieve a battery life of approximately 9.0 days for the Nexus 5 (nominal battery life 12.5 days[2]) and 16.9 days for the wearable device, almost matching Act's estimated 9.1 days and 17.0 days, respectively. Furthermore, our estimates show our method surpassing $Tree$'s battery life by 1.8 days for the device and by 1.0 days for the smartphone, and surpassing DTW's battery life by 2.3 days for the device and by 2.9 days for the smartphone.

Splitting Methods

Comparing the results in Table 4 for $Proposed$-M with those for $Proposed$-B, we can see a significant reduction in recognition accuracy for $Proposed$-B when using lower thresholds. Although $Proposed$-B initially had better recognition accuracy than $Proposed$-M at higher thresholds, $Proposed$-B's F-measure for gestures dropped down to 0.883 at a threshold of 8.75 mJ. This reduced accuracy comes from how $Proposed$-B uses the results of the DTW-based k-NN classifiers. With $Proposed$-M, the splits that are based on the results of the k-NN classifiers can create a separate branch for each output of the classifier, allowing a single node in the

[2]www.google.com/nexus/5

$Proposed$-M tree to use a single k-NN classifier to differentiate between several gestures. However, with $Proposed$-B, all splits on the results of k-NN classifiers can create only two branches, differentiating between one class and all others. We believe that as the threshold value is reduced, this limitation on how well $Proposed$-B can exploit the output of the k-NN classifiers leads to a lower recognition accuracy.

Integrating k-NN Classifiers into Recognition Results

Table 4 also shows the comparison of accuracies for $Proposed$-B and $Proposed$-M between integrating the results of the k-NN classifiers into the recognition results (*C4.5 estimates w/ k-NN estimates*) and not integrating the k-NN results (*C4.5 estimates only*). For both methods, there was an average increase in accuracy of about 3%, with each method showing improved accuracy across all threshold values. The most significant increase was for $Proposed$-B when using a threshold of 8.75 mJ. As was discussed earlier, the recognition accuracy for $Proposed$-B drops significantly at lower thresholds, due to its limited ability to make use of the few k-NN classifiers allowed at lower thresholds. Integrating the actual output of these classifiers into the tree's estimates greatly mitigated the impact from this limitation, increasing recognition accuracy by about 18%.

Simultaneously Conducting Activities and Gestures

We also examined the effectiveness of our method when gestures are conducted simultaneously with an activity, using the additional 20 sessions of data described in the *Data Set* section. We trained our recognition models using training data from the primary data set supplemented with the 10 sessions of *walking* activity, and performed recognition on each of the 10 sessions that contain gestures conducted simultaneously with the *walking* activity. Comparing these results to the *gestures* column in Table 4, we found that the recognition accuracy for gestures in these 10 sessions was 1.6% higher (0.957) at a threshold of 11 mJ and 1.7% higher (0.946) at a threshold of 9.05 mJ, but 2.3% lower (0.895) at a threshold of 8.75 mJ. At all thresholds tested, the results were similar, with an average accuracy of 0.944 for gestures and 0.998 for the *walking* activity, indicating that our method is capable of handling such simultaneous recognition.

CONCLUSION

This paper proposes an energy-aware recognition model for the joint recognition of activities and gestures. Our model combines several features that are commonly used for activity and gesture recognition into tree structured classifiers that are built to reduce energy consumption for both feature extraction and data transmission.

REFERENCES

1. Breiman, L. Random forests. *Mach. Learn. 45*, 1 (2001), 5–32.
2. Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. SMOTE: synthetic minority over-sampling technique. *J. Artif. Intell. Res. 16* (2002), 341–378.
3. French, B., Siewiorek, D. P., Smailagic, A., and Deisher, M. Selective sampling strategies to conserve power in context aware devices. In *Int. Symp. on Wearable Comput. 2007*, IEEE (2007), 77–80.
4. Ghasemzadeh, H., Amini, N., Saeedi, R., and Sarrafzadeh, M. Power-aware computing in wearable sensor networks: An optimal feature selection. *Trans. Mobile Comput. 99*, PrePrints (2014).
5. Gordon, D., Czerny, J., Miyaki, T., and Beigl, M. Energy-efficient activity recognition using prediction. In *Int. Symp. on Wearable Comput. 2012*, IEEE (2012), 29–36.
6. Keogh, E., Chakrabarti, K., Pazzani, M., and Mehrotra, S. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems 3*, 3 (2001), 263–286.
7. Krause, A., Ihmig, M., Rankin, E., Leong, D., Gupta, S., Siewiorek, D., Smailagic, A., Deisher, M., and Sengupta, U. Trading off prediction accuracy and power consumption for context-aware wearable computing. In *Int. Symp. on Wearable Comput. 2005*, IEEE (2005), 20–26.
8. Lu, Z., Chen, X., Li, Q., Zhang, X., and Zhou, P. A hand gesture recognition framework and wearable gesture-based interaction prototype for mobile devices. *Trans. Human Mach. Syst. 44*, 2 (2014), 293–299.
9. Lukowicz, P., Junker, H., Stager, M., Buren, T. V., and Tröster, G. Wearnet: A distributed multi-sensor system for context aware wearables. In *UbiComp 2002* (2002), 361–370.
10. Park, T., Lee, J., Hwang, I., Yoo, C., Nachman, L., and Song, J. E-gesture: a collaborative architecture for energy-efficient gesture recognition with hand-worn sensor and mobile devices. In *SenSys 2011* (2011), 260–273.
11. Preece, S. J., Goulermas, J. Y., Kenney, L. P., Howard, D., Meijer, K., and Crompton, R. Activity identification using body-mounted sensors-a review of classification techniques. *Physiol. Meas. 30*, 4 (2009), R1.
12. Quinlan, J. Improved use of continuous attributes in c4.5. *J. Artif. Intell. Res. 4* (1996), 77–90.
13. Quinlan, J. R. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
14. Raffa, G., Lee, J., Nachman, L., and Song, J. Don't slow me down: Bringing energy efficiency to continuous gesture recognition. In *Int. Symp. on Wearable Comput. 2010* (2010), 1–8.
15. Sakoe, H., and Chiba, S. Dynamic programming algorithm optimization for spoken word recognition. *Trans. Acoust., Speech, Signal Process. 26*, 1 (1978), 43–49.
16. Yan, Z., Subbaraju, V., Chakraborty, D., Misra, A., and Aberer, K. Energy-efficient continuous activity recognition on mobile phones: An activity-adaptive approach. In *Int. Symp. on Wearable Comput. 2012*, IEEE (2012), 17–24.