An Iterative Dimensionality-Scaling System for Real-Time Health Monitoring Applications

Haik Kalantarian, Majid Sarrafzadeh
Center for SMART Health
University of California, Los Angeles
Los Angeles, CA 90024
e-mails:{kalantarian, majid}@cs.ucla.edu

Shibo Zhang, Nabil Alshurafa
Department of Preventive Medicine
Northwestern University, Chicago
Chicago, Illinois 60611
e-mails:{shibozhang, nabil}@northwestern.edu

***Abstract*—Wearable health-monitoring systems must achieve a balance between the often opposing goals of hardware overhead and classification accuracy. Prior works have presented various approaches to dynamically scale the accuracy of these systems as a function of available resources. In this paper, we present a framework which retroactively improves the accuracy of prior estimates when resources become available, using a novel global cost minimization function. We benchmark our algorithm on an audio-based nutrition monitoring dataset. Results confirm the efficacy of our technique.**

I. Introduction

Various wearable health monitoring gadgets have been proposed in recent years. These devices have broad applications ranging from physical activity monitoring [1], [2], to more experimental applications such as diet tracking [3]–[5], mental stress detection [6], and smoking [7]. These lightweight, portable devices consist of one or more sensors, an on-board microcontroller, and an RF module which transmits the data to a mobile application using WiFi, ZigBee, or Bluetooth. On the mobile device, various algorithms are used to identify actions of interest from the continuous stream of real-time data. One of the often conflicting challenges associated with processing these signals is to improve the accuracy of activity recognition, while minimizing the use of hardware resources and, in effect, power.

To address these concerns, various systems have been proposed in which accuracy of computation is dynamically adjusted as a function of available hardware resources. When the device is idle, it is possible to employ complex algorithms to obtain highly accurate results at the expense of CPU resources and battery life. When resources are limited, alternative techniques can obtain a general estimate at a much lower cost. Dynamic optimization of classification accuracy for power reduction is particularly useful in scenarios with sparse, short duration events distributed across an entire day or week's worth of data. For example, a heart-rate monitor could be optimized to enter low-power mode when a coupled accelerometer shows little physical activity. Moreover, adjusting the sample rate of an accelerometer upon movement can be used to maintain high classification accuracy, while reducing power when more simple motions are performed [8].

A major challenge associated with these techniques is the fact that the accuracy of recognition for a particular sample (or window) of data is constant throughout the device lifetime. For example, it is possible that a particular event is recognized using a weak classifier because there are insufficient hardware resources available for real-time processing using a more complex, but accurate algorithm. However, the device may subsequently become idle and existing schemes do not have a mechanism to retroactively improve the prior predictions. In this paper, we propose a scheme in which the accuracy of prior estimates is improved as a function of available hardware resources, in an iterative manner. That is, all previous calculations are retained as accuracy is gradually improved while hardware resources are available.

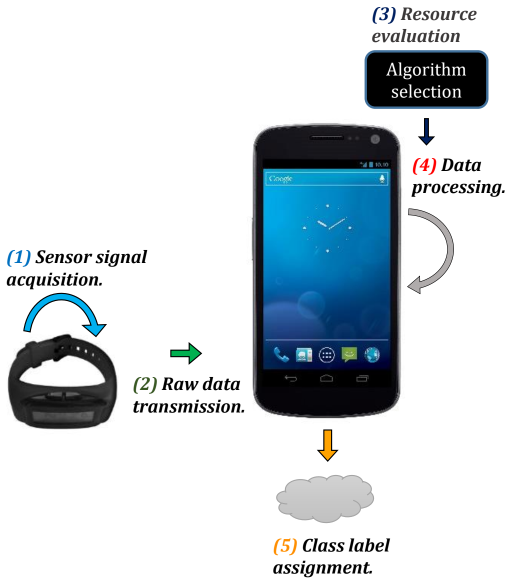
(3) Resource evaluation
Algorithm selection
(4) Data processing.
(1) Sensor signal acquisition.
(2) Raw data transmission.
(5) Class label assignment.

Fig. 1. This figure shows the flow of data in a typical wearable health monitoring system. Data is acquired from a wearable device such as a smartwatch (pictured), and is transmitted to a mobile phone for detailed analysis using Bluetooth. Various dynamic schemes are capable of identifying a model to provide an appropriate balance between classification accuracy and resource usage based on system resources and external factors. Subsequently, the data is processed using a statistical classifier or other algorithms, and a final class label is uploaded to cloud services.

This paper is organized as follows:

- In Section II, we describe related work in this field.

- In Section III, we present a typical wearable system flow.
- In Section IV, we describe our classification mechanism.
- In Section V, we describe our experimental setup.
- In Section VI, we present our experimental results.
- In Section VII, we provide concluding remarks.

II. Related Works

Several prior works have proposed modifying system parameters during runtime to save power. One approach is described by Benbasat et al. [9], in which the authors propose a power-efficient activity recognition framework in which sampling rate is varied to optimize detection of system state. Results show that lower sample rates can save power while maintaining similar classification accuracies to the baseline in selected situations.

In [10], Ghasemzadeh et al. propose low-power digital signal processing modules called *screening blocks*. These mechanisms allow the sensor signal to be pre-screened for relevance. Consequently, the microcontroller can remain in a low power state under most circumstances, being enabled only when more resource-demanding computation is required. Thus, the system could be modeled as a two-stage classifier: the first stage comprising a high-recall, low-precision classifier and the second stage representing a low-recall, high-precision classifier. The authors present an optimization algorithm in which the optimal number of screening blocks is selected, based on system accuracy requirements.

In [11], Shih et al. propose a similar approach to eliminate irrelevant signals in multi-channel EEG data using a two-stage classifier. In [12], Ghasemzadeh et al. propose a power-aware feature selection scheme for real-time systems in which the cost of each individual feature is factored into the dimensionality reduction algorithm. The key insight of these works is that it is not always necessary to run the classifier at its highest accuracy setting; often, a low-power detection strategy can be used instead, which transitions into a more expensive recognition stage when needed (i.e. criteria are met) [13]–[15].

These works demonstrate an ability to adapt classification algorithms to external conditions to maintain high accuracy while reducing power. However, the primary limitation associated with these techniques is their inability to retrospectively improve the accuracy of prior estimates when hardware resources become available. This is the limitation which we address in our work, with an emphasis on the development of efficient iterative approaches which improve classification accuracy monotonically as a function of feature set size.

III. Wearable System Algorithm Flow

A. *Typical pipeline*

Figure 2 shows an example of a typical processing pipeline for a wearable health monitoring or activity recognition device. First, signals are acquired from the sensor (such as an accelerometer, microphone, or gyroscope) using an on-board microcontroller. After transmission to a mobile aggregator, this data is segmented into discrete windows, each of which are assigned a class label corresponding with the activity associated with this time period. Assigning a class label requires feature extraction, and classification using a pre-trained classifier.

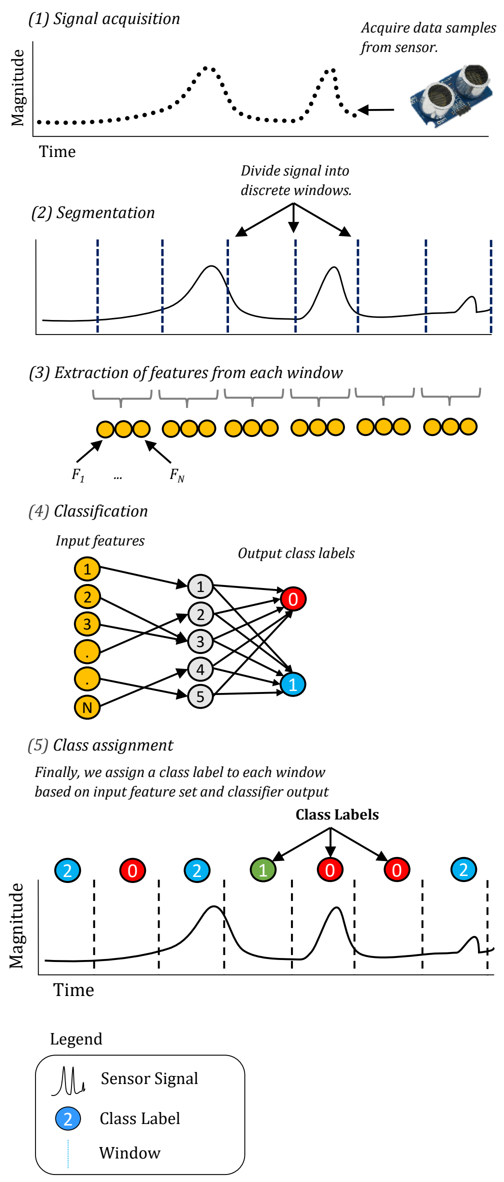

Fig. 2. This figure shows the typical real-time processing pipeline of signals in wearable health monitoring systems. After data is acquired from the sensor, it is segmented into individually processed windows. Statistical features are extracted from each window and input to a pre-trained classifier which assigns labels to each window. Most systems use a constant number of features per window. Those systems which downscale classification parameters when resources are limited do not supplement earlier windows with additional features when resources become available again.

Feature extraction can be computationally demanding, regardless of whether it is performed on a wearable device or mobile phone. The extraction of each feature is associated with a cost, C, representing the amount of time necessary for the system to produce the output attribute from the raw data. Thus, the total cost of the feature extraction phase can be approximated as:

$$C_{extract} = W \cdot \sum_{i=1}^{N} f_i \qquad (1)$$

As shown in this equation, the classifier requires N input features from f_1 to f_N. W refers to the number of signal segmented window frames since each must be processed individually, which impacts cost because each window must be processed individually. Subsequently, features are extracted from each window frame, and the feature values must be entered into the model of a pre-trained classifier to classify each window frame. This is also associated with computational overhead, and is typically a function of classifier type as well as the size of the input feature set. Depending on the application, the cost of sampling and analog-digital conversion can also be significant. However, for low sample rates, this is typically negligible compared to other sources of power dissipation. After a trained classification model is designed, the cost of classifying data is a sum of the following costs: acquiring the data ($C_{acquire}$), extracting features from each window frame ($C_{extract}$), applying the model ($C_{classify}$), transmitting data from one device to another ($C_{transmit}$). A typical power dissipation model is summarized in Equation 2 below.

$$C_{total} = C_{acquire} + C_{extract} + C_{classify} + C_{transmit} \qquad (2)$$

In this particular work, we emphasize optimizing the feature extraction phase by adjusting the dimensionality (number of features) based on available system resources.

B. Accuracy improvement

Generally, a classifier continues to improve its accuracy the more relevant features (correlated with the output) it is provided, until a certain point (where an increase in features no longer provides any gains). This observation is used in our proposed system to improve accuracy when resources are available. In this effort, we assume a simplified feature extraction method in which the time required to extract features from a particular window frame of the signal increases linearly with the number of features extracted, although in practice some features will cost more (require greater computation) to extract than others.

It should be noted that a separate classifier must be trained for each set of input features. And because training a unique model for each subset of features is prohibitive in cost, the primary challenge is to develop a series of models, where the features used for one (smaller) model is a subset of the other (larger) model. This prevents redundancy of computing time, allowing for savings in computation and complexity.

IV. Algorithm

A. Feature extraction

While the feature extraction of our proposed scheme is slightly more complex than that of traditional wearable systems, it comes at a benefit of power savings. Figure 3 shows the proposed processing pipeline. The device samples data at a predefined rate, which is generally an interrupt-based, timer-driven process. After the data is acquired, it is segmented into discrete window frames. At this point, the system performs an evaluation of additional system resources, based on an external heuristic that our algorithms leave undefined for generalizability. However, likely choices include timing constraints (such as the amount of time until the next sample must be acquired), or the available power budget. If resources are available, an additional feature is extracted from a previous window. At a particular stopping threshold, the extracted features will be entered into a classifier, based on the size of the feature set, to obtain the predicted class label.

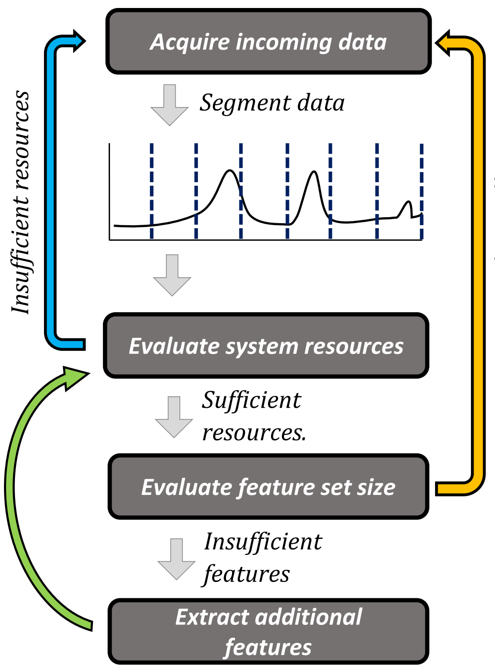

Fig. 3. This figure shows the classification pipeline in our proposed scheme. As before, data is acquired and segmented into discrete windows. However, additional features are incrementally added when system resources are available. The set of features is carefully pre-selected to ensure high classification accuracy can be achieved with various subsets of the original set.

An overview pseudocode of the scheme is shown in Algorithm 1. A list data structure maintains a buffer of the previous α windows, which consist of raw sensor data as well as the features which have been extracted from this data. This data structure is searched when the *OnAvailableSystemResources* function is called. When a window is identified that has less than β features, the next highest priority feature that is missing from the feature set is extracted and added to the data structure. The algorithm for acquiring a new sample of data is shown in Algorithm 2. In this case, a newly acquired window is appended to a list containing the last α windows. When the list is full, the oldest entry is removed and a pre-trained classifier is identified with the same feature set. The window is then classified and the class label is retained locally or transmitted to a mobile phone.

B. *Classifier training*

Based on the feature extraction process shown in Algorithms 1 and 2, it is necessary to train a set of models, K, comprising classifiers, C_l to C_m with dimensionality lower and upper bounds of l and m, respectively. That is, the subscript refers to the dimensionality (feature count) in our notation. The classifiers are subject to several constraints, which we describe in this subsection. It should be noted that this paper refers to the accuracy of a classifier as the percentage of correctly classified instances based on experimental training data. The first constraint is as follows:

$$\forall C \in K, \quad Accuracy(C_i) > Accuracy(C_{i-1}) \tag{3}$$

As shown in Equation 3, with exception to the boundary conditions, each classifier must have demonstrated a total classification accuracy greater than any classifier with a smaller dimensionality on the test set. Thus, classifier accuracy must increase monotonically with dimensionality to ensure that computation is not wasted on feature extraction. This constraint can be enforced by selecting appropriate values of parameters l and m, in addition to hyperparameter optimization and classifier choice.

The second constraint refers to the optimality of the solution. As it is difficult to predict a priori how many features will be extracted before the window must be classified and discarded, it is necessary for every classifier from C_l to C_m to have high classification accuracy. Specifically, our final set of models, K, must have the property such that:

$$\arg\max_{K} \ (\sum_{\forall C \in K} Accuracy(C)) \tag{4}$$

That is, the set K must consist of classifiers whose aggregated classification accuracy is higher than any such set. The third constraint of our set is:

$$(\nexists\ (C_i, C_j) \subseteq K) \mid i = j \tag{5}$$

This condition stipulates that there are no two classifiers in our set with same dimensionality, eliminating wasteful redundancy.

$$(\nexists\ (C_i, C_j) \subseteq K) \mid ((j = i + 1) \wedge i \not\subset j) \tag{6}$$

The last condition, shown in expression 6 states that, given two classifiers C$_i$ and C$_j$ in which j = i + 1, only one feature is present in *j* which is not in *i*. This ensures that selecting the classifier with the next largest dimensionality does not require extracting more than one additional feature.

Algorithm 1: Dimensionality scaling algorithm

```
/* This buffer contains a list of $\alpha$ previous windows that
   have not been assigned a final class label. */
List windows = new List();
/* This function is called by the system whenever there
   are sufficient resources to improve the accuracy of a
   prior estimate. */
function OnAvailableSystemResources()
Begin
    /* This function searches our windows buffer to
       identify the window with the smallest number of
       features. */
    Window current = windowSearch(windows);
    /* The $\beta$ parameter ensures we do not spend system
       resources extracting features that do not further
       improve classification accuracy.*/
    if current.featureSize() $< \beta$ then
        current.addNextFeature(windows);
    return;
End
```

Algorithm 2: Classification condition

```
/* This buffer contains a list of $\alpha$ previous windows that
   have not been assigned a final class label. */
List windows = new List();
/* This function is called when a new window of raw
   sensor data has been acquired. */
function OnNewWindowAcquired(Window NWindow)
Begin
    /* Append the new window to our data structure. */
    windows.append(NWindow);
    if windows.size() $> \alpha$ then
        /* If the buffer is full, we must remove the oldest
           window and classify with the features we have
           had time to extract. */
        Window earliest = windows.Remove(0); /*
          Locate a classifier with the same dimensionality
          as the first window in the buffer. */
        Classifier class =
          searchClassifiers(earliest.featureSize());
        /* Obtain and save the class label corresponding
           with the window. */
        Label label = class.Classify(earliest);
        save(label);
    return;
End
```

V. Experimental methodology

A. *Dataset*

Our experimental methodology was derived from an audio-based nutrition monitoring dataset described in [16]. Data was collected from 20 individuals using a Hyperio Flexible Throat Microphone Headset. The microphone was placed in the lower part of the neck near the collarbone, and connected directly to the mobile phones audio input port using a 3.5mm male audio cable. 16 of the subjects were male, and 4 were female. The ages ranged from 21 to 31 years, with a median age of 22. Commercially available audio-recording technology was used to acquire the audio recordings from the microphone. The primary challenges of automated nutrition monitoring are distinguishing eating features (such as chewing or swallowing) from other ambient noises, and identifying the specific food consumed. The dataset used involved eating the following three foods: nuts, chocolate, and a vegetarian patty. Each food category consisted of sixty quarter-second samples, for a total of 180 samples. Evaluation was conducted using Leave-One-Subject-Out cross validation to reliably assess model generalizability.

B. Feature extraction

The feature extraction tool used on the audio dataset was provided by the OpenSMILE framework [17]. This tool is capable of various audio signal processing operations such as applying window functions, FFT, FIR filterbanks, autocorrelation, and cepstrum. In addition to these techniques, OpenSMILE is capable of extracting various speech-related and statistical features. Audio-based features include frame energy, intensity, auditory spectra, zero crossing rate, and voice quality. After data is collected from a each subject, feature selection algorithms are used to identify highly predictive features that accurately predict swallows and bites. The top ten selected features are shown in Table I. These features are produced using an InformationGain attribute evaluation scheme provided by the WEKA data mining software [18]. For a detailed explanation of these features and a more qualitative explanation of what they represent, we refer readers to the OpenSmile documentation [17]; the specific setting used was the emo_large configuration. The WEKA data mining software is used to evaluate the performance of various classifiers [18] based on the input features extracted from the OpenSMILE toolkit from each audio snippet from the nutrition monitoring dataset.

VI. Results and discussion

A. Classifier performance

Figure 4 shows how classification accuracy improved as a function of classifier dimensionality for four different classifiers: RandomForest, SMO (Sequential Minimal Optimization), Logistic Regression, and Naive Bayesian. As shown by this figure, classification accuracy climbs significantly at low feature sizes. The Naive Bayesian classifier failed to improve

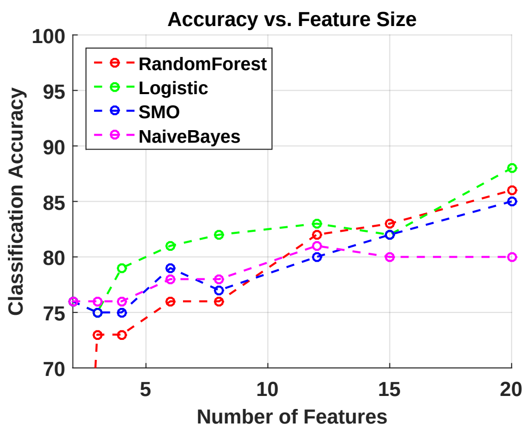

Fig. 4. This figure shows the maximum achievable classification accuracy as a function of classifier type and classifier dimensionality. The maximum attainable accuracy is therefore the maximum value of the best performing classifier at a given feature size.

accuracy after thirteen features, while the other classifiers continued improving their accuracy beyond the 20^{th} feature.

B. Test inputs

We evaluate several test conditions using simulations based on classification results, using function λ(t), which represents the available resources of the system at a particular time t. In our simulation, we fix the amount of resources required to extract one additional feature to be $\lambda(t) = 1$, for a period of 1 second. More generally, we assume that a feature can be extracted between t = a and t = b if the condition shown in Equation 7 is met:

$$\int_{t=a}^{b} \lambda(a) >= 1 \tag{7}$$

If the integral of the λ function exceeds one for a given time interval, we assume multiple features can be extracted in this time interval.

In our simulations, we assume one new window frame per second. In all four of our test conditions, we assume $\lambda(t)$ is a step function with duty cycles of 75%, 50%, 25%, and 10%. In each test input, the total computation power is the same. The "low" state corresponds to 2λ, while the "high" state corresponds to up to 32λ(t) depending on the test condition: lower duty cycles have higher maximum throughput. The test input vectors are shown in Figure 5. As described previously, extra computational resources are used to extract additional features for previous windows, when a sufficient number of features are extracted for the latest window such that additional processing does not significantly improve accuracy. In our experiments, this threshold was set to twelve features.

We compare our scheme to two baselines:

- A fixed scheme that uses the minimum number of features that can be extracted throughout the device lifetime.
- A dynamic scheme in which as many features are extracted in an epoch as possible, but for the most recent window only.

TABLE I
Top 10 Selected Features

Rank	Attribute (FFTMag)	Information Gain
1	mfcc sma[5] quartile2	1.292
2	fftMag melspec sma[5] quartile1	1.286
3	mfcc sma[5] amean numeric	1.239
4	fftMag melspec sma[8] quartile1	1.231
5	fftMag melspec sma[7] quartile1	1.21
6	fftMag fband250-650 sma quartile1	1.202
7	fftMag melspec sma[10] quartile1	1.194
8	fftMag fband0-650 sma quartile1	1.18
9	fftMag melspec sma[2] quartile1	1.17
10	fftMag melspec sma[6] quartile1	1.16

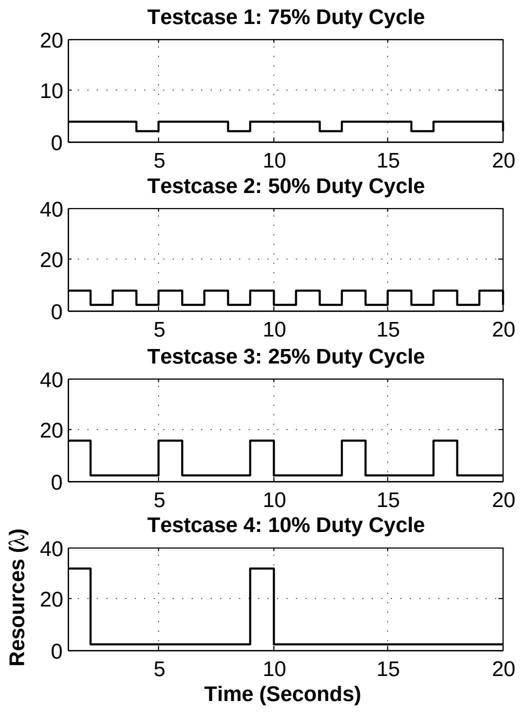

Fig. 5. This figure shows the test inputs used to evaluate our algorithm.

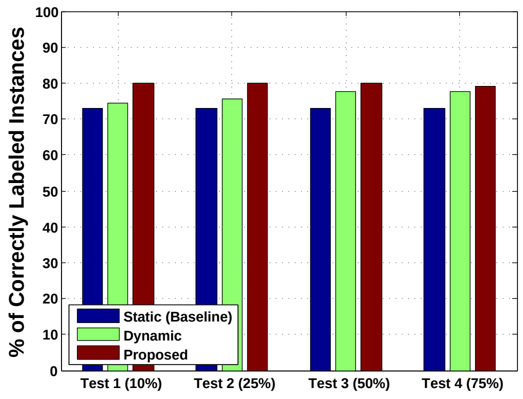

Fig. 6. This figure shows classification results using our proposed scheme, a dynamic scheme which adjusts dimensionality for the current window only, and a baseline static scheme.

Figure 6 shows classification results for all three schemes under four test conditions. Though results were generally high in most cases, the proposed scheme had the highest average classification accuracy under each test condition. The dynamic algorithm outperformed the static baseline by 5%, and the proposed scheme provided an additional 4% improvement (for a total of 9%).

C. Future work

Though preliminary results are promising, there are several limitations in our proposed work. First, future works must implement this system fully on an embedded system and evaluate the feasibility of the algorithm in real-world conditions. Secondly, different datasets must be analyzed, such as those from inertial sensors with much lower sample rates, to evaluate the generality of our proposed solution. Additionally, relaxing some constraints with respect to monotonicity of the function may yield more globally optimal solutions: classifiers with dimensionality at which performance suffers could be skipped, but their features retained for use by the next classifier in the K set at which accuracy improves once again.

VII. Conclusion

In this paper, we propose a novel framework for real-time classification applications in which accuracy of activity recognition is efficiently scaled as a function of available system resources. Future work will demonstrate the applicability of the proposed techniques to other datasets and perform real-world use cases.

Acknowledgment

The authors would like to thank Anne Barrett from Northwestern University for her help in preparing this manuscript and assisting with editing.

References

[1] "Misfit wearables faq," http://www.misfitwearables.com/supportl.

[2] "Jawbone up technical specifications," http://jawbone.com/store/buy/up24.

[3] W. Jia, H.-C. Chen, Y. Yue, Z. Li, J. Fernstrom, Y. Bai, C. Li, and M. Sun, "Accuracy of food portion size estimation from digital pictures acquired by a chest-worn camera," *Public Health Nutrition*, vol. FirstView, pp. 1–11, 12 2013.

[4] H. Kalantarian, N. Alshurafa, and M. Sarrafzadeh, "A wearable nutrition monitoring system," in *2014 11th International Conference on Wearable and Implantable Body Sensor Networks*, June 2014, pp. 75–80.

[5] N. Alshurafa, H. Kalantarian, M. Pourhomayoun, J. J. Liu, S. Sarin, B. Shahbazi, and M. Sarrafzadeh, "Recognition of nutrition intake using time-frequency decomposition in a wearable necklace using a piezoelectric sensor," *IEEE Sensors Journal*, vol. 15, no. 7, pp. 3909–3916, July 2015.

[6] F.-T. Sun, C. Kuo, H.-T. Cheng, S. Buthpitiya, P. Collins, and M. Griss, "Activity-aware mental stress detection using physiological sensors," in *Mobile computing, applications, and services*. Springer, 2012, pp. 211–230.

[7] A. Parate, M.-C. Chiu, C. Chadowitz, D. Ganesan, and E. Kalogerakis, "Risq: Recognizing smoking gestures with inertial sensors on a wristband," in *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*. ACM, 2014, pp. 149–161.

[8] S. Bobek, M. layski, and G. Nalepa, "Capturing dynamics of mobile context-aware systems with rules and statistical analysis of historical data," in *Artificial Intelligence and Soft Computing*, ser. Lecture Notes in Computer Science, L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. A. Zadeh, and J. M. Zurada, Eds. Springer International Publishing, 2015, vol. 9120, pp. 578–590. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-19369-451

[9] A. Y. Benbasat and J. A. Paradiso, "A framework for the automated generation of power-efficient classifiers for embedded sensor nodes," in *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems*, ser. SenSys '07. New York, NY, USA: ACM, 2007, pp. 219–232. [Online]. Available: http://doi.acm.org/10.1145/1322263.1322285

[10] H. Ghasemzadeh and R. Jafari, "Ultra low-power signal processing in wearable monitoring systems: A tiered screening architecture with optimal bit resolution," *ACM Trans. Embed. Comput. Syst.*, vol. 13, no. 1, pp. 9:1–9:23, Sep. 2013. [Online]. Available: http://doi.acm.org/10.1145/2501626.2501636

[11] E. Shih and J. Guttag, "Reducing energy consumption of multi-channel mobile medical monitoring algorithms," in *Proceedings of the 2Nd International Workshop on Systems and Networking Support for Health Care and Assisted Living Environments*, ser. HealthNet '08. New York, NY, USA: ACM, 2008, pp. 15:1–15:7. [Online]. Available: http://doi.acm.org/10.1145/1515747.1515767

[12] H. Ghasemzadeh, N. Amini, R. Saeedi, and M. Sarrafzadeh, "Power-aware computing in wearable sensor networks: An optimal feature selection," *IEEE Transactions on Mobile Computing*, vol. 14, no. 4, pp. 800–812, April 2015.

[13] S. Huang, "Adaptive sampling with mobile sensor networks," Ph.D. dissertation, Michigan Technological University, 2012.

[14] A. Jain and E. Y. Chang, "Adaptive sampling for sensor networks," in *Proceeedings of the 1st international workshop on Data management for sensor networks: in conjunction with VLDB 2004*. ACM, 2004, pp. 10–16.

[15] A. D. Marbini and L. E. Sacks, "Adaptive sampling mechanisms in sensor networks," in *London Communications Symposium*, 2003.

[16] H. Kalantarian and M. Sarrafzadeh, "Audio-based detection and evaluation of eating behavior using the smartwatch platform," *Elsevier Computers in Biology and Medicine*, 2015.

[17] F. Eyben, M. Wöllmer, and B. Schuller, "Opensmile: The munich versatile and fast open-source audio feature extractor," in *Proceedings of the International Conference on Multimedia*, ser. MM '10. New York, NY, USA: ACM, 2010, pp. 1459–1462. [Online]. Available: http://doi.acm.org/10.1145/1873951.1874246

[18] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.