This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBME.2016.2570210, IEEE Transactions on Biomedical Engineering

Dynamic Computation Offloading for Low Power Wearable Health Monitoring Systems

Haik Kalantarian, Costas Sideris, Bobak Mortazavi, Nabil Alshurafa, Majid Sarrafzadeh

***Abstract*—*Objective:* The objective of our work is to describe and evaluate an algorithm to reduce power usage and increase battery lifetime for wearable health-monitoring devices. *Methods:* We describe a novel dynamic computation offloading scheme for real-time wearable health monitoring devices that adjusts the partitioning of data processing between the wearable device and mobile application as a function of desired classification accuracy. *Results:* By making the correct offloading decision based on current system parameters, we show that we are able to reduce system power by as much as 20%. *Conclusion:* We demonstrate that computation offloading can be applied to real-time monitoring systems, and yields significant power savings. *Significance:* Making correct offloading decisions for health monitoring devices can extend battery life and improve adherence.** [1]

***Index Terms*—power optimization, pervasive computing, signal processing, machine learning**

I. Introduction

A. Introduction to Wearables

As smartphones have entered ubiquity in recent years, various wearable wireless health monitoring gadgets have been proposed. These devices have broad applications ranging from physical activity monitoring [1], to more experimental applications such as diet tracking [3], mental stress detection [6], and rehabilitation [8]. Spurred by a rapidly aging global population and the emergency of lightweight, affordable microelectronics, wearable devices will have increasingly broad applications for consumers and clinicians in the years to come. One of the most significant challenges in medicine is non-adherence, as prior studies have shown that non-compliance to treatments is associated with a host of negative health outcomes [10]. Among other factors, battery life has been reported to be a significant contributor to non-adherence [12], as frequent battery recharging may present a repeated burden to the user. Moreover, the device cannot report user activity when the battery has been depleted, resulting in a loss of data. For these reasons, an increasing amount of attention has been directed towards improving the battery life of wearable health monitoring devices is recent years.

B. Energy Usage

Regardless of function or application, wearables have many fundamental architectural similarities. The primary components typically consist of:

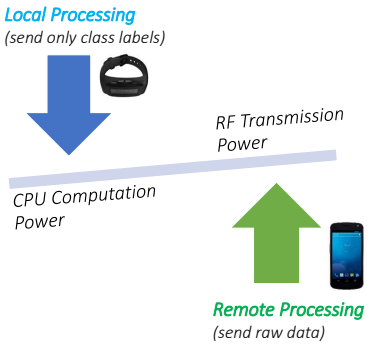

Fig. 1. Wearable devices must evaluate the performance penalty of RF transmission when deciding to perform data computation locally vs. remotely.

1) Sensors: One or more sensors, sampled at a particular frequency depending on the application. These sensors are the foundation through which the device can detect user activity.

2) Microcontroller: A microcontroller that performs the sensor sampling, processes the data, and interfaces with other peripherals.

3) Transceiver: A wireless transceiver which transmits data remotely to a mobile phone or cloud services for analysis, visualization, and feedback.

In wearable devices, it is often the case that a significant amount of system power is expended on wireless transmission and local computation. Wireless transmission overhead refers to the power necessary to transmit data from the microcontroller to a mobile application using a technology such as Bluetooth. Local computation overhead refers to the power requirements of data processing performed on the wearable device's microcontroller. Unfortunately, as Figure 1 illustrates, it is often the case that optimizing the energy of the microcontroller and wireless transceiver are diametrically opposing goals. Assume there is some feature or property *f*, that we are interested in detecting from a continuous signal. There are two possible approaches to detect such an event. The first option is to process the data locally on the wearable device, and only transmit the minimal information: that we have detected *f*. This puts a significant burden on the computation resources of the microcontroller, as processing the sensor data can be expensive and resource intensive. However, in this approach it is only necessary to transmit the final class label to the mobile phone, rather than the raw data, which minimizes total Bluetooth overhead.

The second option is to perform no data processing on the local device, deferring the processing to the smartphone. This

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBME.2016.2570210, IEEE Transactions on Biomedical Engineering

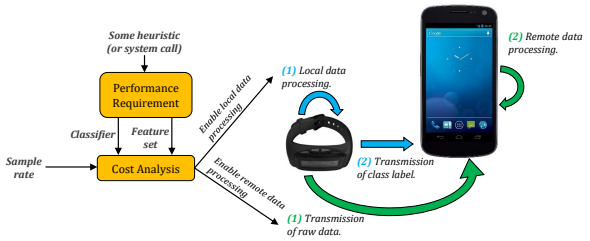

Fig. 2. This figure shows the overall system flow. Each time the required classification accuracy is adjusted, a new decision is made on whether to process the data locally or transmit for remote processing on a smartphone.

approach saves local computation energy on the microcontroller, as it is no longer required to do any expensive processing or classification; the microcontroller can transmit the sensor data as soon as it is acquired. However, this approach may result in higher wireless transmission overhead, which in many cases can be even more than that of local computation. The decision on whether to process data locally or remotely is a function of many parameters, such as Bluetooth connection interval, sample rate, classifier choice, and more.

C. Computation Offloading

Computation offloading is a broad paradigm in which data is outsourced from local computation to a server, cloud, or other form of aggregator. The primary objective of computation offloading is to reduce the energy demands of a small microcontroller with a low battery capacity, or to perform very resource-heavy operations on more powerful hardware for performance reasons. The motivation behind this approach is that mobile phones typically have much larger batteries than small wearable devices, just as servers have more hardware resources than personal computers. In this paper, we propose a novel algorithm for dynamic computation offloading, targeted towards real-time wearable health monitoring applications. While many other works have discussed different strategies for offloading computation [15], [16]. However, our work focuses specifically on modern cutting-edge wearable devices, emphasizing the tradeoffs between local computation and Bluetooth transmission overhead as a function of the required classification accuracy.

II. Classification Flow

In this section, we begin with the preliminaries of a modern real-time health monitoring system. An example architecture for such a system consists of the steps shown in Figure 3.

A. Acquisition

In this stage, the microcontroller on the wearable device acquires data from a sensor and buffers it locally. The choice of sample rate f_s is critical at this stage, and is dependent on the type of sensor used. More data samples may require more Bluetooth transmissions in cases when computation is offloaded. Furthermore, in cases when computation is performed locally, there is still additional overhead associated with processing more data.

TABLE I
Definition of Terms

Term	Description
C_x	Cost (in terms of power) to execute operation x
F	A set of features associated with a window.
f_s	Sample rate: the rate at which data is acquired from the sensor.
M	Function that maps feature set, classifier, and set size to a cost.
K	Variable that represents the classification algorithm used (eg. RandomForest), BayesNet.
n	Number of features extracted.
BW	Bandwidth: the rate at which data is transmitted wirelessly between device and phone.
λ	Connection interval: how often a connection is established between two Bluetooth devices.

B. Segmentation

In the next stage, the signal is divided into shorter windows, each of which are typically processed independently of one another. This is necessary, because it is often impractical to assign a single class label to a very large dataset. Windows that are not associated with any particular activity are known as the *null* class, which can be discarded after processing.

The three primary methods in literature are sliding window approaches, similar to the baseline used in this paper, and recursive techniques that are either partition the entire signal, or merge small denominations of the signal, until a stopping criteria is met. Comprehensive surveys of time-series segmentation are provided by Keogh et al. in [19], and Lovri et al. in [20]. We refer the reader to these works for a more detailed discussion of the topic.

C. Extraction

From each window, a set of representative features are extracted. The number of features that must be extracted at this phase can also have a significant impact on total system power; larger feature sets may provide higher classification accuracy in some cases. The features extracted during this step are pre-selected during the classifier training process using various feature selection and dimension reduction algorithms. However, the feature *extraction* must be performed in real-time.

D. Classification

The features are the inputs to a pre-trained classifier, which outputs a class label that descibes the actions represented by the window. Once again, the classifier may be trained a priori. However, a real-time system would generally require that the classifier be run periodically to provide user feedback.

III. Energy Modeling

In this section, we describe our model for power dissipation in a wearable health-monitoring device. An explanation of terms and symbols is provided in Table I.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBME.2016.2570210, IEEE Transactions on Biomedical Engineering

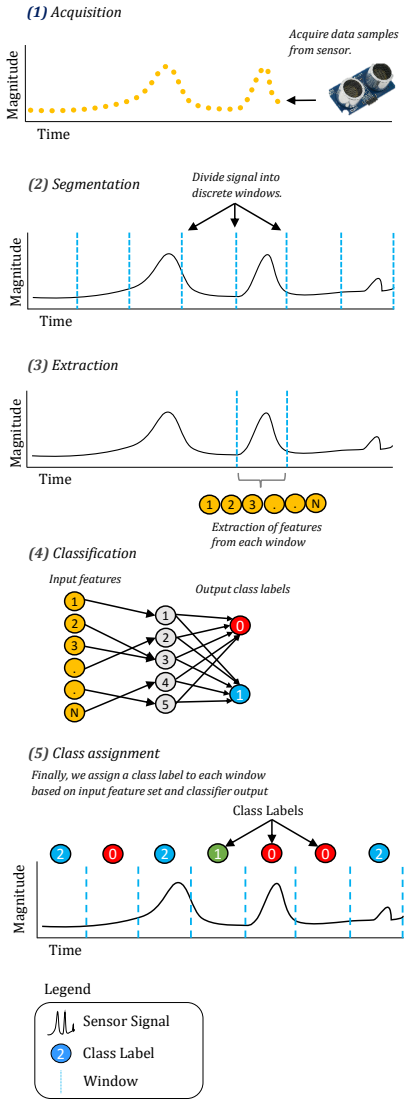

Fig. 3. This figure depicts the overall classification model assumed by this paper. A microcontroller acquires data from a sensor at some sample rate f_s, segments the signal into discrete windows, extracts features from each window, and uses a pre-trained classifier to assign a class label to that segment.

A. A General Model

A simplified model to represent power consumption in a system consisting of a wearable device and a paired smartphone can be represented as the sum of the cost of extracting features, running them through a classifier, and transmitting any necessary information between the phone and device. Though there are other sources of power dissipation such as sampling, segmentation, and leakage, we focus our analysis on the model shown in Equation 1 in this paper.

$$C_{total} = C_{extraction} + C_{classification} + C_{transmission} \quad (1)$$

Assume we have a feature set F, consisting of a total of n features, f_1 through f_n. Similarly, assume each feature f_i has a cost, C_{f_i}. Thus, the cost of the feature extraction phase is:

$$C_{extraction} = \sum_{i=1}^{n} C_{f_i} \quad (2)$$

Modeling the cost of the classification is more challenging. Generally, however, it is a function of the input feature set F, feature size n, and the classifier algorithm used. We define function $\mathbf{M}$ as a function that maps these parameters to the total classifier cost.

$$C_{classification} = \mathbf{M}(F,\ K,\ n) \quad (3)$$

If we perform both the feature extraction and classification locally, the local energy can be modeled as the sum of the feature extraction and classification costs.

$$C_{local} = \mathbf{M}(F,\ K,\ n) + \sum_{i=1}^{n} C_{f_i} \quad (4)$$

We can also model the local cost of transmitting the raw data to the mobile device, and performing both the feature extraction and classification remotely. We assume these features are associated with a window of length L. Thus, the cost of transmitting a window of length L is: $\mathbf{C}_{tx}(L)$. Note that this relationship may not be linear; very high sampling rates may change critical Bluetooth parameters such as the connection and advertisement intervals. Combining these equations gives us the equilibrium point, in which it is roughly equally costly to process the data locally and remotely:

$$\mathbf{M}(F,\ K,\ n) + \sum_{i=1}^{n} C_{f_i} = \mathbf{C_{tx}}(L) \quad (5)$$

An adaptive system could modify several parameters in real-time, to favor either local or remote processing:

1) Algorithm choice: Different classifiers, such as Random-Forest, Support Vector Machines, and k-Nearest Neighbor, have different runtimes. Switching to a lighter classifier can swing the balance in favor of local processing at the cost of classification accuracy.

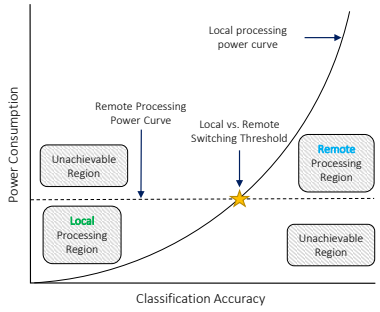

Fig. 4. When the desired classification accuracy increases beyond a certain threshold, it becomes more energy efficient to transmit the data from the wearable to a mobile aggregator for processing.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBME.2016.2570210, IEEE Transactions on Biomedical Engineering

2) Feature count: Choosing a different subset of features may have dramatic performance implications. A higher number of input features may, in some cases, improve classification accuracy. However, the relationship is not linear; very large feature sets may overfit the training data and decrease total classification accuracy.

Each classifier **M** can be represented by its classifier choice and feature count, and is associated with a particular classification accuracy and power budget. The subsequent challenge is to identify the circumstances in which it is appropriate to vary the desired classification accuracy to save power. However, the specific scenarios are out of scope for this work; we refer the readers to [25], [26], [27] for a discussion of these issues.

B. Connection Interval

The Bluetooth 4.0 LE standard, used in many wearable devices, allows peripherals to suggest a connection interval which specifies how often the device sends data. The Bluetooth standard supports connection intervals ranging from 7.5 ms to four seconds; higher connection intervals reduce system bandwidth, but significantly reduce power consumption as well. Typically, during each connection interval, several Bluetooth packets can be transmitted. However, the Nordic nRF8001 used in our evaluation supports transmission of just one packet [28] per connection event. The payload of each packet is 20 Bytes. According to [28], the total Bluetooth connection bandwidth can be represented as a function of connection interval γ, as shown below in Equation 6:

$$BW_{BT} = \frac{20\ bytes}{packet} \cdot \frac{1\ packet}{conn.\ event} \cdot \frac{1\ conn.\ event}{\gamma\ seconds} \quad (6)$$

Generally, we would select a connection interval to satisfy a particular bandwidth requirement. Simplifying Equation 6 gives us:

$$\gamma = \frac{20\ bytes}{Bandwidth} \quad (7)$$

Expectedly, higher sample rates (f_s) are typically more expensive to process and transmit. In accelerometer-based activity monitoring applications, a sample rate of between 25 Hz and 100 Hz is typical, with some studies claiming that 45 Hz is optimal [29]. Remote computation schemes will generally require that all sampled data is transmitted to the mobile phone for processing. If we assume each data sample is a 32-bit integer (four bytes), we can model the required connection interval as a function of sample rate below:

$$\gamma = \frac{20\ bytes}{4 \cdot f_s} \quad (8)$$

By comparison, local processing schemes can have almost negligible wireless transmission overhead, as it is only necessary to transmit a class label once an event is detected.

IV. Algorithm

A. Classifier Accuracy Adjustment

Many prior works have scaled down accuracy to save power, for various applications including classification and health monitoring. For example, Benbasat et al. propose a power efficient sensor system in [25], in which a wearable gate monitor is optimized by adjusting classification accuracy. In [26], Ghasemzadeh et al. propose a two-tiered classification scheme in which preliminary classification is achieved using lightweight, low-power techniques. A similar two-stage scheme was proposed by Shih et al. in [27] in which a power-efficient screening stage precedes a more computationally expensive analysis stage. The key insight 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, which transitions into a more expensive recognition stage when various criteria are met [31], [32], [33].

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 when a subject begins to move has been shown to maintain high classification accuracy, while reducing power when more simple motions are performed [34]. Though these prior works are able to successfully reduce power consumption, they generally do not evaluate the tradeoffs between wireless transmission and local computation upon changes in classification accuracy.

B. Dynamic Offloading

The proposed computation offloading scheme is as follows. First, the system pre-trains *n* classifiers, $\mathbf{M_1}...\mathbf{M_n}$ Each classifier has the objective of maximizing total classification accuracy for a given power budget, and may have a different number of input features. Based on the desired sample rate, we can predict the benefits of local and remote processing using Equation 5 and select one of the two schemes. When the user program specifies a need to improve the classification accuracy based on some external inputs, we iterate through the *n* possible classifiers and select classifier $\mathbf{M_i}$ that which minimizes power consumption based on the required accuracy threshold. Once this classifier is selected, its predicted cost is computed using Equation 5, and a decision is made with respect to local and remote processing overhead. This procedure is shown in Algorithm 1.

Specifically, once we have selected a potential classifier and feature set, we can compare the power consumption of local and remote data processing. This process continues until the next classification accuracy adjustment. Figure 4 shows the proposed scheme, which depicts both local processing regions, based on the desired classification accuracy. The point denoted by the star represents the condition shown in Equation 5; equal local and remote processing costs. The cost to process

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBME.2016.2570210, IEEE Transactions on Biomedical Engineering

Algorithm 1: Computation Offloading

```
function OnChangeAccuracyRequirement(int pct)
Begin
    /* We select the optimal classifier which minimizes
    power but meets the performance requirement. */
    Classifier K = getClassifier (pct);
    /*Estimate power from classifier, feature count. */
    Power LocalPower = EstimatePower(K.Classifier,
    K.FeatureCount);
    Power RemotePower = EstimateRFPower(L, λ);
    if LocalPower > RemotePower then
        /* Determine that local processing is optimal. */
        RunLocally();
    else
        OffloadComputation();
End
```

data locally is modeled as a function of two parameters: α and β. Parameter α represents the cost to extract a single feature, while β represents the cost to run a classifier on the local application. We assume the classification cost on the mobile device is negligable as the battery life is an order of magnitude larger, and the focus of our work is primarily in the optimization of the wearable device.

V. Experimental Methodology

A. Dataset

Our experimental methodology was derived from an audio-based nutrition monitoring dataset described in [35]. 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 subects were male, and 4 were female. The ages ranged from 21 to 31 years old, 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 nutrition monitoring is the identification of eating (such as chewing or swallowing) from other ambient noises, and identifying the specific food using various heuristics. The dataset used corresponded to eating of 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 avoid biasing the dataset.

B. Feature Extraction

The feature extraction tool used on the audio dataset was provided by the OpenSMILE framework [36]. This tool is capable of various audio signal processing operations such as applying window functions, FFT, FIR filterbanks, autocorrelation, and cepstrum. After data is collected from a variety of subjects eating several foods, feature selection tools can be used to identify strong features that are accurate predictors of swallows and bites for various foods. The top selected features is shown in Table II. These features are produced using an InformationGain attribute evaluation scheme provided by the WEKA data mining software [37]. For a detailed explanation of these features and a more qualitive explanation of what they represent, we refer the reader to the openSmile documentation [36]; the specific setting used was the emo_large configuration

TABLE II
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

C. Bluetooth Modeling

Power simulations were conducted in the nRFgo Studio software, based on the nRF8002 integrated circuit by Nordic Semiconductor. The simulations assumed no advertisement period, and a default connection interval of 1000 ms. We initially simulated two use cases: one in which 20 bytes (payload) of data are transmitted at a rate of 100 Hz, and another in which data is transmitted at a rate of 1 Hz. These two conditions correspond with remote processing (sending all the data) and local processing (sending class labels) respectively. Subsequently, we experimented with various sensor sample rates and adjusted the connection interval accordingly based on the increase in bandwidth.

D. Computation Modeling

The WEKA data mining software is used to evaluate the performance of various classifiers based on the input features extracted from the openSMILE toolkit from each audio snippet from the nutrition monitoring dataset. The performance of all classifiers was normalized based on our prior results on the MSP430 platform in [39], in which we measured the performance of various algorithms in real-time on a Texas Instruments development board. Similarly, the cost of extracting a particular feature is derived based on our prior work in

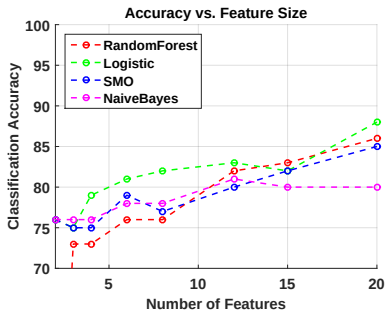

Fig. 5. This figure shows variations in accuracy as a function of classifier, and feature set size.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBME.2016.2570210, IEEE Transactions on Biomedical Engineering

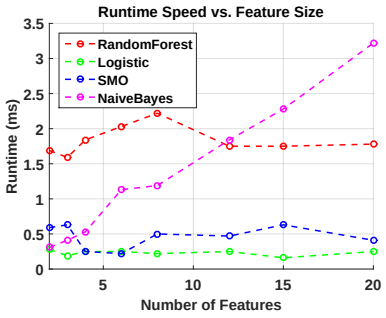

Fig. 6. This figure shows variations in runtime speed as a function of classifier, and feature set size.

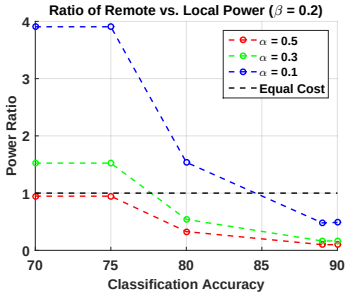

Fig. 7. This figure depicts the ratio of power in the remote and local processing schemes at various values of α, the cost per feature extraction, and β of 0.2 (relative classifier execution cost). A power ratio of 1 represents the equilibrium case.

[39], with the simplifying assumption that each feature has the same cost.

The four classifiers used in our evaluation are: Logistic Regression, Sequential Minimal Optimization, Naive Bayesian, and Random Forest. These classifiers were selected on the basis of their high accuracy in our prior audio-based nutrition monitoring study in [35]. Sequential Minimal Optimization is an efficient implementation of the quadratic programming problem in the training of support vector machines that is commonly used in various classification applications [40].

VI. Results and Discussion

A. Classifier Performance

Figure 5 shows the classification accuracy among four classifiers, as a function of feature set size. Note that the classification accuracy does not linearly increase with the number of features extracted; in some cases, more features are detrimental to performance due to overfitting. Moreover, some classifiers such the Naive Bayesian classifier, perform well with small feature sets but fail to improve significantly with greater numbers of attributes. Generally, this figure shows that approximately five features are sufficient for classification accuracy of over 75% for most classifiers, approaching 83% in the case of Logistic Regression. The upper bound for feature count, beyond which there appears to be no significant improvement, appears to be approximately twenty features with 87-88% accuracy.

Figure 6 shows classifier runtime as a function of feature size and classifier choice. Classifier runtime on an embedded microcontroller can be an approximate heuristic for power consumption, as higher algorithm runtimes increase the percentage of time the device is in active mode, rather than one of the low-power modes in which output current is close to in the microamperes range [39]. Simulations show that the Logistic Regression classifier was associated with the lowest runtime, across the entire range of features. Moreover, the runtime of the logistic regression classifier did not noticably increase as a function of number of input features. By contrast, the Naive Bayesian classifier was quick with a small number of features, but did not scale well with larger feature sets, but runtime increased linearly with the feature count. The RandomForest classifier had a somewhat high runtime, especially with small input feature sets. However, the runtime did not increase when sweeping the feature size between 2 and 20.

The implications of these results are shown in Table III: for each desired accuracy level, we can select an optimal combination of classifier and classification feature size. In this case, Logistic Regression was the optimal choice for almost every scenario except accuracy of 85%, in which Sequential Minimal Optimization (SMO) was preferred by a small margin. Though Logistic Regression has a lower *runtime* cost, the *extraction* cost becomes much more significant for larger feature sets.

B. Comparison of Local and Remote Processing

Assuming a connection interval of 10 ms and a payload size of 20 bytes, simulator results show average power as 964 μw. Figures 7 and 8 show the power overhead of both techniques as a function of various values of α and β. More specifically, these graphs show how much power would be spent in a remote processing scheme compared to a local processing approach, at various desired accuracy levels. A power ratio of 1, shown in black, represents the equilibrium case in which local and remote processing powers are equal.

More specifically, we can observe that lower feature extraction costs, as well as lower desired classification accuracies, are dramatically cheaper to execute locally rather than remotely. However, as the required classification accuracy increases, it is often necessary to use more expensive classifiers and larger feature sets. Thus, these schemes favor remote processing, particularly in systems with more expensive features.

TABLE III
Best Classifier and Feature Set Combinations For a Given Accuracy

Accuracy (%)	Lowest-Cost Classifier	Feature Set Size
70%	Logistic	2
75%	Logistic	2
80%	Logistic	6
85%	SMO	20
90%	Logistic	20

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBME.2016.2570210, IEEE Transactions on Biomedical Engineering

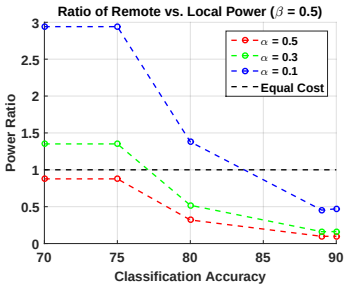

Fig. 8. This figure depicts the ratio of power in the remote and local processing schemes at various values of α, the cost per feature extraction, and β of 0.5 (relative classifier execution cost).

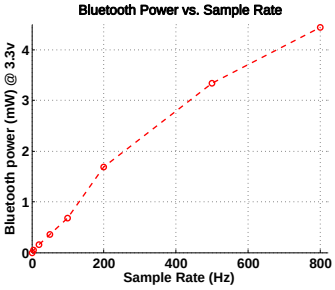

Fig. 9. This figure shows the nonlinearity between sample rate and Bluetooth transmission power. This suggests that Bluetooth offloading may be more practical at higher sample rates, as local computation does not scale as favorably.

C. Variations in Sample Rate

Our previous experiment assumed a connection interval of 10 ms. This connection interval represents one Bluetooth LE connection per second, at which time only one 20-Byte payload can be transmitted based on the limitations of the nRF8001 integrated circuit. Using Equation 8, we know that this connection interval corresponds with a maximum bandwidth of 500 samples per second on our evaluation platform. Though there are applications for which a high sampling rate is sufficient, smaller sample rates can suffice for other uses cases such as environment monitors, dust particle detection, ambient light detection, or smart-home applications. In this section, we analyze the effects of sample rate on transmission power.

Figure 9 shows the Bluetooth power at a 3.3 V supply voltage as a function of sample rate. Interestly, this figure shows that as bandwidth needs increase, wireless power dissipation is less than linear. By comparison, Figure 10 shows the local power dissipation as a function of sample rate and accuracy thresholds. Our model shows a more linear relationship between power and sample rate, given parameters $\alpha = 0.1$ and $\beta = 0.5$. This may suggest high sample rates favor offloading, rather than local computation.

D. Variations in Sample Rate

For a baseline comparison, we consider a scenario in which the system attempts to detect a short-duration event from a large signal in a scenario with the baseline configuration. The

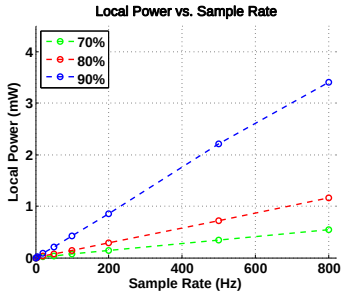

Fig. 10. This figure shows the relationship between local computation power and sample rate for three different accuracies. Higher sample rates reduce the amount of time the device spends in low-power mode.

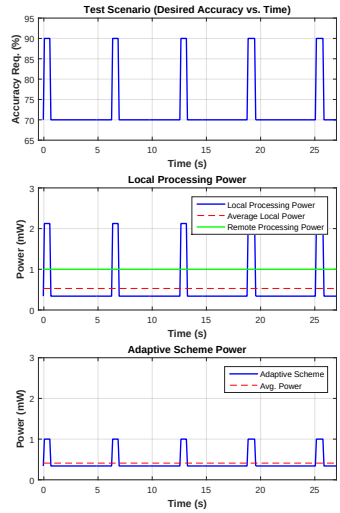

Fig. 11. This figure shows a comparison between local, remote, and adaptive schemes in a system which alternates between low and high accuracy requirements.

system specifications are to operate at 90% accuracy for 10% of the time, and enter low-power mode (70% accuracy) for the remainder of the time. For this experiment, we assume parameters α of 0.1 and β of 0.5. This scenario is shown at the top of Figure 11. As this figure shows, the local processing scheme is optimal at lower accuracies, while remote offloading is favored at high accuracies. By adapting computation as a function of classification accuracy, we are able to reduce power from an average of 0.52 mW to 0.41 mW; a decrease of 21.1% compared to the local processing scheme.

VII. Conclusion

In this paper, we have demonstrated a novel scheme for selective computation offloading based on user-defined accuracy constraints. Our simulations show that with a baseline classifier execution cost of 0.2 mW and feature extraction cost of 0.1 mW, making a correct offloading decision can reduce power by over 20% in certain scenarios. Future work will evaluate these algorithms in a system feature an implementation of real-time classification accuracy adjustment to benchmark our proposed scheme.

REFERENCES

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

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

[3] H. Kalantarian et al., "Monitoring eating habits using a piezoelectric sensor-based necklace," *Elsevier Computers in Biology and Medicine*, vol. 58, no. C, pp. 46–55, 2015.

[4] W. Jia et al., "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.

[5] E. Sazonov et al., "Toward objective monitoring of ingestive behavior in free-living population," *Obesity*, vol. 17, no. 10, pp. 1971–1975, 2009.

[6] F.-T. Sun et al., "Activity-aware mental stress detection using physiological sensors," in *Mobile computing, applications, and services*. Springer, 2012, pp. 211–230.

[7] A. Parate et al., "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. Patel et al., "A review of wearable sensors and systems with application in rehabilitation," *Journal of NeuroEngineering and Rehabilitation*, vol. 9, no. 1, p. 21, 2012.

[9] W. Lutz, W. Sanderson, and S. Scherbov, "The coming acceleration of global population ageing," *Nature*, vol. 451, no. 7179, pp. 716–719, 2008.

[10] J. Dunbar-Jacob and M. Mortimer-Stephens, "Treatment adherence in chronic disease," *Journal of clinical epidemiology*, vol. 54, no. 12, pp. S57–S60, 2001.

[11] B. Blackwell, "Treatment adherence." *The British Journal of Psychiatry*, 1976.

[12] E. L. Murnane, D. Huffaker, and G. Kossinets, "Mobile health apps: adoption, adherence, and abandonment," in *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers*. ACM, 2015, pp. 261–264.

[13] N. Alshurafa et al., "Battery optimization in smartphones for remote health monitoring systems to enhance user adherence," in *Proceedings of the 7th international conference on PErvasive Technologies Related to Assistive Environments*. ACM, 2014, p. 8.

[14] K. Kumar et al., "A survey of computation offloading for mobile systems," *Mobile Networks and Applications*, vol. 18, no. 1, pp. 129–140, 2013.

[15] U. Kremer, J. Hicks, and J. Rehg, "A compilation framework for power and energy management on mobile computers," in *Languages and Compilers for Parallel Computing*, ser. Lecture Notes in Computer Science, H. Dietz, Ed. Springer Berlin Heidelberg, 2003, vol. 2624, pp. 115–131. [Online]. Available: http://dx.doi.org/10.1007/3-540-35767-X8

[16] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, April 2010.

[17] G. Xiaohui et al., "Adaptive offloading inference for delivering applications in pervasive computing environments," in *Pervasive Computing and Communications, 2003. (PerCom 2003). Proceedings of the First IEEE International Conference on*, March 2003, pp. 107–114.

[18] J. Liu, K. Kumar, and Y.-H. Lu, "Tradeoff between energy savings and privacy protection in computation offloading," in *Proceedings of the 16th ACM/IEEE International Symposium on Low Power Electronics and Design*, ser. ISLPED '10. New York, NY, USA: ACM, 2010, pp. 213–218. [Online]. Available: http://doi.acm.org/10.1145/1840845.1840887

[19] E. Keogh et al., "Segmenting time series: A survey and novel approach," *Data mining in time series databases*, vol. 57, pp. 1–22, 2004.

[20] M. Lovrić, M. Milanović, and M. Stamenković, "Algorithmic methods for segmentation of time series: An overview."

[21] Z. Xu et al., "An adaptive algorithm for online time series segmentation with error bound guarantee," in *Proceedings of the 15th International Conference on Extending Database Technology*, ser. EDBT '12. New York, NY, USA: ACM, 2012, pp. 192–203. [Online]. Available: http://doi.acm.org/10.1145/2247596.2247620

[22] M. Okutomi and T. Kanade, "A locally adaptive window for signal matching," *International Journal of Computer Vision*, vol. 7, no. 2, pp. 143–162, 1992. [Online]. Available: http://dx.doi.org/10.1007/BF00128133

[23] V. Katkovnik, K. Egiazarian, and J. Astola, "Adaptive window size image de-noising based on intersection of confidence intervals (ici) rule," *Journal of Mathematical Imaging and Vision*, vol. 16, no. 3, pp. 223–235, 2002. [Online]. Available: http://dx.doi.org/10.1023/A3A1020329726980

[24] R. Klinkenberg and T. Joachims, "Detecting concept drift with support vector machines," in *Proceedings of the Seventeenth International Conference on Machine Learning*, ser. ICML '00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 487–494. [Online]. Available: http://dl.acm.org/citation.cfm?id=645529.657791

[25] 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

[26] 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

[27] 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

[28] S. S. Ole Morten. (2014, jul) How do i calculate throughput for a ble link? [Online]. Available: https://devzone.nordicsemi.com/question/60/what-is-connection-parameters/

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

[30] A. Khan et al., "Optimising sampling rates for accelerometer-based human activity recognition," *Pattern Recognition Letters*, pp. –, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167865516000040

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

[32] 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.

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

[34] 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. Rutkowsk et al.i, Ed. Springer International Publishing, 2015, vol. 9120, pp. 578–590. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-19369-451

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

[36] 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

[37] M. Hall et al., "The weka data mining software: An update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, Nov. 2009.

[38] ——, "The weka data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[39] H. Kalantarian et al., "Power optimization for wearable devices," in *IEEE International Conference on Pervasive Computing and Communication Workshops*, 2015.

[40] J. Platt *et al.*, "Sequential minimal optimization: A fast algorithm for training support vector machines," 1998.