# Exploiting Spatiotemporal and Device Contexts for Energy-Efficient Mobile Embedded Systems

Brad Donohoo[†], Chris Ohlsen[†], Sudeep Pasricha[†‡], Charles Anderson[‡]

[†] Department of Electrical and Computer Engineering
[‡] Department of Computer Science
Colorado State University, Fort Collins, CO
bdonohoo@rams.colostate.edu, ohlsensc@rams.colostate.edu, sudeep@colostate.edu, anderson@cs.colostate.edu

## ABSTRACT

Within the past decade, mobile computing has morphed into a principal form of human communication, business, and social interaction. Unfortunately, the energy demands of newer ambient intelligence and collaborative technologies on mobile devices have greatly overwhelmed modern energy storage abilities. This paper proposes several novel techniques that exploit spatiotemporal and device context to predict device interface configurations that can optimize energy consumption in mobile embedded systems. These techniques, which include variants of linear discriminant analysis, linear logistic regression, non-linear logistic regression with neural networks, and k-nearest neighbor are explored and compared on synthetic and user traces from real-world usage studies. The experimental results show that up to 90% successful prediction is possible with neural networks and k-nearest neighbor algorithms, improving upon prediction strategies in prior work by approximately 50%. Further, an average improvement of 24% energy savings is achieved compared to state-of-the-art prior work on energy-efficient location-sensing.

**Categories and Subject Descriptors**: H.1.2 [**Models and Principles**]: User/Machine Systems – *human factors*

**General Terms –** Algorithms, Performance, Human Factors

**Keywords:** Smartphone, Energy Optimization, Machine Learning

## 1. INTRODUCTION

Mobile phones and other portable devices (tablets, PDA's, and e-readers) are fundamental everyday tools used in business, communication, and social interactions. As newer technologies (e.g. 4G networking, multicore/GPUs) and applications (e.g. 3D gaming, Apple's FaceTime™) gain popularity, the gap between device usage capabilities and battery lifetime continues to increase, much to the annoyance of users who are now becoming more and more reliant on their mobile devices. The growing disparity between functionality and mobile energy storage has been a strong catalyst in recent years to develop software-centric algorithms and strategies for energy optimization [1]-[17]. These software techniques work in tandem with well-known energy optimizations implemented in hardware including CPU DVFS, power/clock gating, and low power mode configurations for device interfaces and chipsets [18]-[21].

The notion of "smart" mobile devices has recently spawned a number of research efforts on developing "smart" energy optimization strategies. Some of these efforts employ strategies that are *context-aware* including utilization of device, user, spatial, temporal, and application awareness that attempt to dynamically modify or learn optimal device configurations to maximize energy savings with little or negligible impact on user perception and quality of service (QoS) [10][13][17]. This general theme of a *smart* and *context-aware* energy optimization strategy is further explored in this paper, in which a select number of machine learning algorithms are proposed and evaluated for their effectiveness in learning a user's mobile device usage pattern pertaining to spatiotemporal and device contexts, to predict data and location interface configurations. These resulting predictions manage the network interface states allowing for dynamic adaptation to optimal energy configurations when respective interfaces are not required while maintaining an acceptable level of user satisfaction. This idea is further motivated by considering the power distributions of the Google Nexus One smartphone illustrated in Figure 1 [22]. Even when 3G, WiFi, and GPS interfaces are all enabled and idle, they account for more than 25% of total system power dissipation. Furthermore, when only one of the interfaces is active, the other two idle interfaces still consume a non-negligible amount of power. Our work exploits this fact to save energy by dynamically managing data and location interfaces, e.g., turning off unnecessary interfaces at runtime.

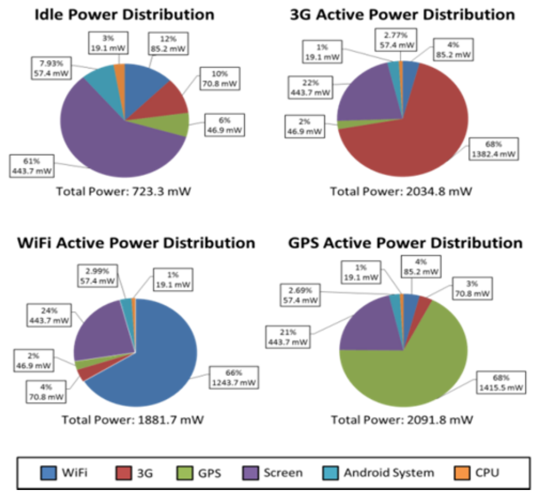

**Figure 1. Google Nexus One smartphone power distributions**

In this paper, we propose and demonstrate the use of four different classes of machine learning algorithms *(i) linear discriminant analysis, (ii) linear logistic regression, (iii) k-nearest neighbor, and (iv) non-linear logistic regression with neural networks*, on predicting user data/location usage requirements using spatiotemporal and device contexts. These strategies are tested on both synthetic and real-world user usage patterns, which demonstrate that high and consistent prediction rates are possible. The proposed techniques are also compared with prior work on device configuration prediction using self-organizing maps [13] and energy-aware location sensing [9], showing an improvement upon these state-of-the-art techniques.

The remainder of this paper is organized as follows. Section 2 reviews several key related works. Section 3 provides a brief overview of the machine learning concepts used in the study. Section 4 discusses the acquisition and creation of real and synthetic user profiles that are used in our analysis studies. Section 5 describes our device power modeling effort. Section 6 presents the results of our experimental studies. Finally, Section 7 presents our conclusions.

## 2. RELATED WORK

A large amount of work has been done in the area of energy optimization for mobile devices in recent years. Much of this work focuses on optimizing energy consumed by the device's wireless interfaces by intelligently selecting the most energy-efficient data interface (e.g. 3G/EDGE, WiFi) [1], [2]. Other work [3]-[8] focuses on energy-efficient location-sensing schemes aiming to reduce high battery drain caused by location interfaces (e.g. WiFi, GPS) by deciding when to enable/disable location interfaces or modify location acquisition frequency. Lee et al. [9] in particular propose a Variable Rate Logging (VRL) mechanism that disables location logging or reduces the GPS logging rate by detecting if the user is standing still or indoors. The authors in [10] propose a context-aware method to determine the minimum set of resources (processors and peripherals) that results in meeting a given level of performance, much like our work. They determine if a user is moving/stationary and indoors/outdoors and control resources using a static lookup table. In contrast, *our work controls resources dynamically by using machine learning algorithms that are trained on real user activity traces*. Zhuang et al. [11] propose an adaptive location-sensing framework that involves substitution, suppression, piggybacking, and adaptation of applications' location-sensing requests to conserve energy. Their work is directed towards LBAs (location-based applications) and only focuses on location interfaces, while *ours is a system-wide optimization strategy that is capable of saving energy regardless of the foreground application type*.

A substantial amount of research has been dedicated to utilizing machine learning algorithms for the purpose of mobile user context determination. Batyuk et al. [13] extend a traditional self-organizing map to provide a means of handling missing values, then use it to predict mobile phone settings such as screen lock pattern and WiFi enable/disable. Other works attempt to predict the location of mobile users using machine learning algorithms. In [14] the authors propose a model that predicts spatial context through supervised learning, and the authors in [15] take advantage of signal strength and signal quality history data and model user locations using an extreme learning machine algorithm. These works are focused on using user context for device self-configuration and location prediction, *whereas our work is focused on using user context for optimizing energy consumed by both data transfer and location interfaces*.

One of the key motivations for applying pattern recognition and classification algorithms to mobile device usage is the observation that user usage patterns are often mutually-independent, in that each user generally has a unique device usage pattern. The use of pattern recognition then allows for energy optimization algorithms to be fine-tuned for each user, achieving energy savings without perturbing user satisfaction levels. This idea is further confirmed in mobile usage studies [16], which additionally focused on smartphone usage pattern analysis and its implications on mobile network management and device power management. Although their work had a slightly different focus than our work, the key relevant take-away is that the authors demonstrated from a two month real smartphone usage study that *all users have unique device usage patterns*. Our previous work [17] also found that usage patterns are unique in the way users interact with different apps on their mobile devices. In Section 4 of this study, the real user usage patterns further confirm this claim – all five users had unique usage patterns in the amount of interaction as well as when and where the interactions most often took place.

## 3. ALGORITHM OVERVIEW

The notion of searching for patterns and regularities in data is the fundamental concept in the field of pattern recognition and data classification. *Machine learning* is often focused on the development and application of computer algorithms in this field [23]. In this work we focus on exploiting learning algorithms to discover opportunities for energy saving in mobile embedded systems through the dynamic adaptation of data transfer and location network interface configurations without any explicit user input. Consequently, we enable energy-performance tradeoffs that are unique to each user. This is accomplished by learning both the spatiotemporal and device contexts of a user through the application of a given algorithm and using these to control device network configuration to save energy. In other words, given a set of input contextual cues, the algorithms will exploit learned user context to dynamically classify the cues into a system state that precisely governs how data and location interfaces are utilized. The goal is to achieve a state classification that saves energy while maintaining user satisfaction. An overview of the basic underlying concepts and application of the machine learning algorithms used in this study is briefly discussed below.

### 3.1 Linear Discriminant Analysis

Linear discriminant analysis (LDA) makes use of a Bayesian approach to classification in which parameters are considered as random variables of a prior distribution. This concept is fundamentally different from data-driven linear and non-linear discriminant analyses in which what is learned is a function that maps or separates samples to a class. Bayesian estimation and the application of LDA is also known as *generative modeling*, in that what is learned is a probabilistic model of the samples from each class. By considering parameters as random variables of a prior distribution one can make use of prior known information. For example knowing that a mean $\mu$ is very likely to be between $a$ and $b$, the probability can be determined in such a way that the bulk of the density lies between $a$ and $b$ [24]. Given a prior probability distribution and a class likelihood, Bayes' theorem (equation 1) can be invoked to get an inferred posterior probability to derive a class prediction ($C_k$) for a new observed sample $x_n$ using a *maximum a posteriori* (*MAP*; equation 2) [24]:

$$p(C_k|x_n) = \frac{p(C_k)p(x_n|C_k)}{p(x)} \qquad (1)$$

$$\underset{C}{\operatorname{argmax}}\, p(C_k|x_n) \qquad (2)$$

LDA is applicable to a wide range of classification problems of both univariate or multivariate input spaces and binary- or multi-class classification. A number of statistical distribution functions can be applied, but the most common is the *Gaussian* or *Normal* distribution (which we use in our study) as shown in equation 3 below.

$$\boldsymbol{N}(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\} \qquad (3)$$

There are a number of reasons the Gaussian distribution is a prominent model used in several fields of study, including natural and social sciences, but a simplistic view is that the shape of the Gaussian distribution, the well-known bell shape, is often an acceptable model for complex and partly random phenomena. This notion is brought about by the fact that the shape of the Gaussian is controlled by the mean $\mu$ and variance $\sigma^2$. The location of the mean (or the peak of the curve) gives an expectation of where random variables tend to cluster and the variance (or width of the curve) gives an indication of the variability of the data.

### 3.2 Linear Logistic Regression

Similar to LDA, linear logistic regression (LLR) is a technique used to derive a linear model that directly predicts $p(C_k|x_n)$, however it does this by determining linear boundaries that maximize the likelihood of the data from a set of class samples instead of invoking Bayes' theorem and generating probabilistic models from priori information. LLR expresses $p(C_k|x_n)$ directly by requiring all linear function values to be between 0 and 1 and that they all sum to 1 for any value of $x_n$, as shown in equation 4:

$$p(C_k|x_n) = \frac{f(x_n,\ \beta_k)}{\sum_{m=1}^{K} f(x_n,\ B_m)} \tag{4}$$

With LDA, Bayes' theorem, class priors, and class probability models were used to infer the class posterior probabilities, which were then used to discriminate between the different classes for a given sample $x_n$. In contrast, LLR solves for the linear weight parameters, $\beta_k$, directly using gradients to maximize the data likelihood. This is done by enumerating the likelihood function, $L(\beta)$, using a 1-of-K coding scheme for the target variables, as shown in equations 5, 6 [23], in which every value of $t_{n,k} \in \{0, 1\}$ and each row only contains a single '1'. The class variable transformations are known as indicator variables and are used in the exponents of the likelihood function to select the correct terms for each sample $x_n$.

$$C = \{k_1, k_2, k_3, \dots, k_n\}$$

$$\downarrow$$

$$\begin{pmatrix} t_{1,1} & t_{1,2} & \cdots & t_{1,k} \\ t_{2,1} & t_{2,2} & \cdots & t_{2,k} \\ \vdots & & & \\ t_{n,1} & t_{n,2} & \cdots & t_{n,k} \end{pmatrix} \tag{5}$$

$$L(\beta) = \prod_{n=1}^{N} \prod_{k=1}^{K} p(C_k|x_n)^{t_{n,k}} \tag{6}$$

In order to find the $\beta$ that maximizes the data likelihood, the product of products is transformed to a sum of sums using the natural logarithm to simplify the gradient calculation with respect to $\beta$. Since equation 7 is non-linear, iterative methods like gradient ascent or scaled conjugate gradient [24] can be used to solve for the gradient of the log likelihood, $\nabla_\beta LL(\beta)$, , and obtain the respective $\beta$ weights.

$$LL(\beta) = \sum_{n=1}^{N} \sum_{k=1}^{K} t_{n,k} \log p(C_k|x_n) \tag{7}$$

### 3.3 Non-linear Logistic Regression with Neural Networks

Neural network models, also known as *Artificial Neural Networks*, are inspired by the way the human brain is believed to function. Many of the normal basic everyday information processing requirements handled by the brain, for example sensory processing, cognition, and learning, surpass any capable computing system out there today. Although a human brain is quite different than today's computing hardware, it is believed that the basic concepts still apply in that there is a computational unit, known as a *neuron*, and connections to memory stored in *synapses*. The main difference being that the human brain consists of billions of these simple parallel processing units, neurons, which are interconnected in a massive multi-layered distributive network of synapses and neurons [24].

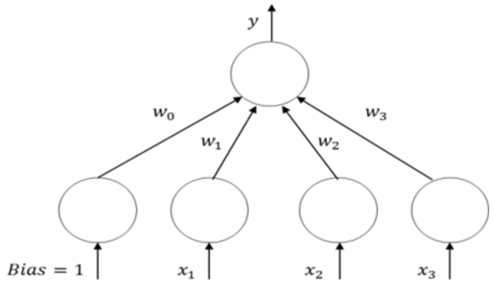

**Figure 2. Neural network perceptron model**

In machine learning, these concepts are modeled as what is called a *perceptron*, the basic processing element, connected by other *perceptrons* through weighted connections, as illustrated in Figure 2. The output of a perceptron is simply a weighted sum of its inputs including a weighted bias, as shown in equation 8.

$$y = \sum_{i=1}^{n} w_i x_i + w_0 \tag{8}$$

To compute the output $y$ given a sample $x_i$, backpropagation using the gradient with respect to the weights is performed using a training dataset to find the weight parameters, $w_i$, that minimize the mean squared error between the neural network outputs, $y_i$, and the target outputs, $t_i$. By default, the neural network consists of a hyperplane (for multiple perceptrons) that can be used as a linear discriminant to linearly separate the classes. To improve prediction accuracy, we make it non-linear, by applying a sigmoidal or hyperbolic tangent to hidden unit layer perceptrons, as denoted in equation 9. This allows for non-linear boundaries with the output of the neural network being linear in the weights, but non-linear in the inputs.

$$y_i' = sigmoid(y_i) = \frac{1}{1 + \exp(\boldsymbol{w^T x})} \tag{9}$$

For classification with a neural network (non-linear logistic regression), the number of parallel output perceptrons is equal to the number of classes. The output from each perceptron, $y_i$, is then sent to post processing as in equation 10 to determine the respective class by taking the maximum of the post-processed outputs:

$$C_i \ if \ y_i = \max_i \frac{\exp(y_i)}{\sum_i \exp(y_i)} \tag{10}$$

One of the biggest criticisms about the use of neural networks is the time required for training. Although this can be a major issue if using a simple gradient descent approach, newer training techniques, such as the scaled conjugate gradient (SCG) [25], can greatly minimize the time required for training. SCG, a method for efficiently training feed-forward neural networks, was used for training the neural networks in this study.

### 3.4 K-Nearest Neighbor

The k-nearest neighbor (KNN) algorithm is a fairly simple non-parametric unsupervised approach for the data classification problem. A key assumption of non-parametric estimation is that similar inputs have similar outputs [23]. In KNN, new samples are classified by assigning them the class that is the most common among the k closest samples in the attribute space. This method requires some form of distance measure for which Euclidean distance is typically used. The Euclidean distance between two points $a$ and $b$, each containing i attributes, is defined in equation 11.

$$d(a,b) = \sqrt{\sum_{i=1}^{n} (b_i - a_i)^2} \tag{11}$$

## 4. USER INTERACTION STUDIES

In order to compare the relative effectiveness of the different algorithms at predicting a user's data/location usage requirements, five real user usage profiles for four different Android smartphones (HTC myTouch 3G, Google Nexus One, Motorola Droid X, HTC G2, and Samsung Intercept) were collected over a one week period with a custom Context Logger application. The application logged user context data on external storage, which was acquired at the end of the one week session and used in our algorithm analysis.

### 4.1 Context Logger

We created a custom *Context Logger* application that ran in the background as an Android service and gathered both spatiotemporal and device usage attributes at a one minute interval. Table 1 lists the attributes recorded by the logger and used for algorithm analysis and indicates whether the attribute was a continuous variable (floating point), discrete variable (integer), or logical variable (true/false). The *GPS Satellites* attribute is used as an indirect correlation to GPS signal strength and *WiFi RSSI* is a measure of WiFi signal strength. In addition to more common device attributes such as *Battery Level* and *CPU Utilization*, we gathered several uncommon OS attributes: *Context Switches*, *Processes Created*, *Processes Running*, and *Processes Blocked*. We hoped to aid prediction by using these as inputs to the machine learning algorithms. The three target variables (*Data Needed*, *Coarse Location Needed*, and *Fine Location Needed*) were obtained by examining the requested Android permissions of all of the device's current running foreground applications and services. The *Device Moving* attribute was determined by using the accelerometer sensor and a metric for movement that is the sum of the unbiased variance of X, Y, and Z acceleration [7], given as:

$$Var(m_1 \dots m_n) = \frac{\sum_{i=1}^{N} m_i^2 - \frac{1}{N}\left(\sum_{i-1}^{N} m_i\right)^2}{N-1} \quad (12)$$

$$Metric = Var(x_1 \dots x_n) + Var(y_1 \dots y_n) + Var(z_1 \dots z_n) \quad (13)$$

**Table 1. Recorded data attributes**

| Context | Attribute | *Type* |
|---|---|---|
| Temporal | Day of week | *Discrete* |
| | Time of day | *Discrete* |
| Spatial | Latitude | *Continuous* |
| | Longitude | *Continuous* |
| | GPS Satellite Count | *Discrete* |
| | WiFi RSSI | *Discrete* |
| | Number of WiFi APs Available | *Discrete* |
| | 3G Network Signal Strength | *Discrete* |
| | Device Moving | *Logical* |
| | Ambient Light | *Discrete* |
| Device | Call State | *Discrete* |
| | Battery Level | *Discrete* |
| | Battery Status | *Discrete* |
| | CPU Utilization | *Continuous* |
| | Context Switches | *Discrete* |
| | Processes Created | *Discrete* |
| | Processes Running | *Discrete* |
| | Processes Blocked | *Discrete* |
| | Screen On | *Logical* |
| Targets | Data Needed | *Logical* |
| | Coarse Location Needed | *Logical* |
| | Fine Location Needed | *Logical* |

## 4.2 Data Preparation

As mentioned earlier, GPS location coordinates were recorded along with the other attributes at one minute intervals. The Android SDK GPS location data returns the user's longitude and latitude coordinates in decimal degrees as reported by the onboard GPS chipset [26]. Although exact accuracy is dependent on the actual GPS hardware, the returned values were truncated to a given precision and each longitude and latitude coordinate pair was mapped to a unique location identifier. The truncated location resolution generalized the number of unique locations in which a user spends his/her time, given that for example, a user's home may consist of several different samples of different longitude and latitude pairs. In addition, temporal conditions can be applied to further reduce the number of unique locations (e.g. disregard locations where user spent less than x minutes). Figure 3 shows the effect of truncation and application of temporal conditions for a real user and how the primary locations where the user spent most of his/her time are revealed. For our study we used a location precision of 4 decimal places.

The desired data/location interface configurations were partitioned into eight different states based on the desired target variables (*Data Required*, *Coarse Location Required*, and *Fine Location Required*). Table 2 maps the logical values of the three target variables to a state. The states define the device's current required resources. *Efficiently predicting one of these 8 states using temporal, spatial, and device context input variables in Table 1 may ultimately allow opportunistic shutdown of location/wireless radios*. If all interfaces are enabled, they would consume a significant amount of energy in their idle states without this dynamic control.

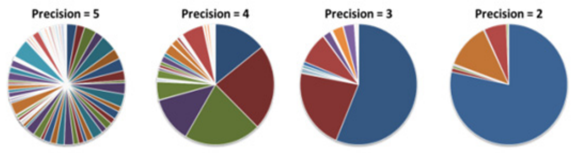

**Figure 3. Unique locations identified for varying GPS precisions**

**Table 2. Interface configuration states**

| State | Data Required | Coarse Location Required | Fine Location Required |
|---|---|---|---|
| 1 | No | No | No |
| 2 | No | No | Yes |
| 3 | No | Yes | No |
| 4 | No | Yes | Yes |
| 5 | Yes | No | No |
| 6 | Yes | No | Yes |
| 7 | Yes | Yes | No |
| 8 | Yes | Yes | Yes |

## 4.3 Real User Profiles

We distributed the Context Logger application to five different mobile device users, and logged their context data over the course of one week. Figure 4 demonstrates the relative state distributions as described in Table 2 for the five different real users. As can be seen, states 2 – 4 are never realized as data was always required when location was required. In addition, the distributions highlight that each user had a considerably different usage pattern. User 1 can be categorized as a minimal user, where the user rarely utilized their phone with only brief periods of interaction, while users 2 and 5 used their phones rather frequently and for longer periods of time. Users 3 and 4 can be categorized as moderate users, primarily utilizing their devices for only certain times of the day.

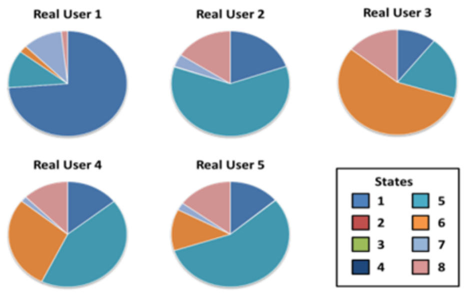

**Figure 4. Real user state distributions**

## 4.4 Synthetic User Profiles

Prior to the algorithm evaluation studies, a subset of synthetic user profiles were created for five different idealized and generalized models of average user usage patterns including the following: *(i) 8 – 5 Business Worker, (ii) College Student, (iii) Social Teenager, (iv) Stay At Home Parent*, and *(v) Busy Traveler*. Both an indoor/outdoor location timeline and an interface state profile were created for each synthetic user. Given the difficulty of generating realistic device system data, such as context switches, CPU utilization, and processes created, only a subset of the attribute space was considered. The remaining attributes were based on both the desired state and/or location. For example, if a user was at an outdoor location, larger GPS satellite values and weak WiFi RSSI values were used as opposed to when the user was indoors.

## 5. DEVICE POWER MODELING

In order to quantify the energy-effectiveness of using machine learning algorithms to predict energy-optimal device states, power analysis was performed on the real Android based smartphones used, with the goal of creating power models for the data and location interfaces. We use a variant of Android OS 2.3.3, (Gingerbread) and the Android SDK Revision 11 as our baseline OS. We built our power estimation models using real power measurements, by instrumenting the contact between the smartphone and the battery, and measuring current using the Monsoon Solutions power monitor [27]. The monitor connects to a PC running the Monsoon Solutions power tool software that allows real-time current and power measurements over time. We manually enabled the data/location interfaces one by one and gathered power traces for each interface in their active and idle states. The power traces from the Monsoon Power Tool were then used to obtain average power consumption measurements for each interface.

## 6. EXPERIMENTAL RESULTS

### 6.1 Prediction Accuracy Analysis

Recall that the input attributes for the learning algorithms come from the gathered spatiotemporal and device context data (Table 1), and the predicted output is one of the 8 interface configuration states (Table 2). To evaluate the prediction accuracy of the different algorithms, the data for each real user was randomly partitioned into training and test sets using a common 80/20 partitioning scheme. The algorithms were then trained on the training data and evaluated on the test data. This was repeated five times for each implementation and the net prediction accuracy is presented in Figure 5.

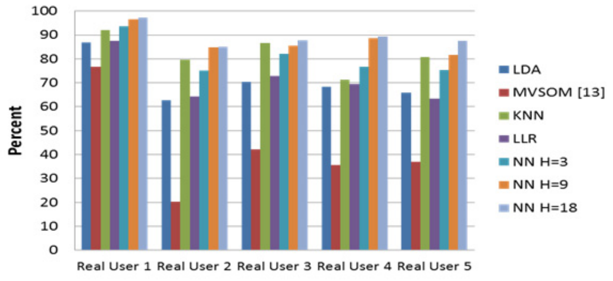

**Figure 5. Real user algorithm prediction accuracy**

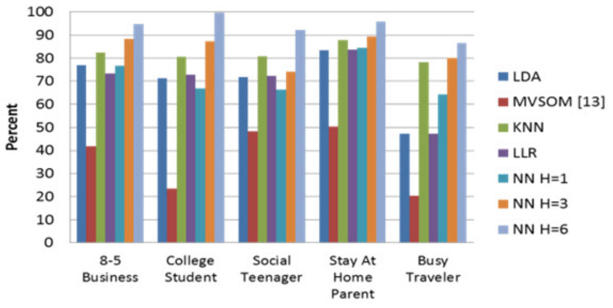

**Figure 6. Synthetic user algorithm prediction accuracy**

Three different neural network (NN) implementations with a varying number of hidden units, equal to the total (H=18), half (H=9), and one-sixth (H=3) the size of the attribute space were evaluated. We compare the prediction accuracy of our algorithms with the configuration prediction strategy presented in [13] (*MVSOM – Missing Values Self-Organizing Map*). As illustrated in Figure 5, the application of neural networks with a number of hidden units of at least half the size of the attribute space resulted in the highest prediction rates. K-nearest neighbor (KNN), linear logistic regression (LLR), and linear discriminant analysis (LDA) also performed fairly well, with prediction accuracies in the range of 60 – 90 %. However these approaches were much more sensitive to the usage pattern. MVSOM performed the worst and had a high degree of variance in both the usage pattern and random training data selection.

The same algorithms were applied to the synthetic user profiles; however, the attribute space was reduced to only *Day*, *Time*, *Location*, *GPS Satellites*, *WiFi RSSI*, *Network Signal Strength*, *Data Needed*, *Coarse Location Needed*, and *Fine Location Needed*. The same strategy for selecting numbers of hidden units for the neural network implementations was applied for the reduced attribute space. Figure 6 illustrates the algorithm prediction rates for the synthetic users showing similar trends as in the real user data.

### 6.2 Energy Savings

It is important to note that despite high prediction accuracy, the amount of potential energy savings is still highly dependent on the user usage pattern and if the algorithms are positively or negatively predicting states where energy can be conserved. Figures 7 and 8 illustrate the energy savings achieved by the individual algorithms when the algorithm's prediction target states are applied to the real and synthetic user profiles, and compare them against the *VRL* technique (*Variable Rate Logging* [9]). Note that as VRL does not predict system state, results for its prediction accuracy are not shown in Figure 5. Although simpler linear models can achieve high energy savings, it is important to note that energy savings themselves are not good discriminants of an algorithm's *goodness* because user satisfaction must also be considered. For example, if a user spends a significant amount of time in the energy consuming state 8 and the algorithms are predicting a less energy-consuming state during these instances, then more energy can be conserved at the cost of user-satisfaction. Directly correlating the prediction accuracy of the algorithms is especially important for highly interactive users such as users 2 and 5, as opposed to minimally interactive users, e.g., user 1.

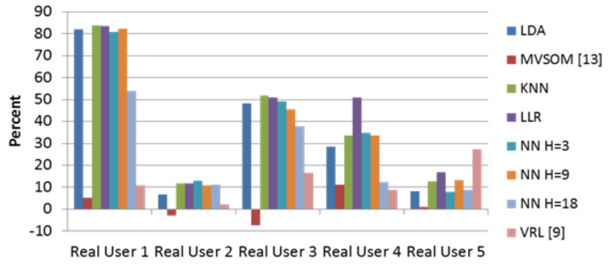

**Figure 7. Percent energy saved for real users**

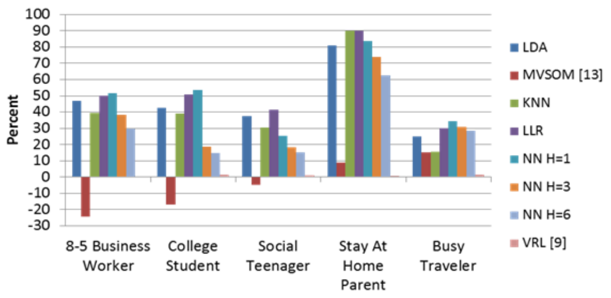

**Figure 8. Percent energy saved for synthetic users**

All energy savings are relative to the baseline case for Android systems without proactive multi-network interface management. Lower prediction and more generalized models result in the highest energy savings as in the case of LDA and LLR. However, again, these higher savings come at the cost of degraded user satisfaction. *KNN overall performs fairly well in terms of both prediction accuracy and energy savings potential, as does the nonlinear logistic regression with NN approach*. With the latter, an important point to note is that prediction accuracy is proportional to the complexity of the neural network and indirectly proportional to the net energy savings. This outcome is expected as less complex neural networks will result in more generalized models relaxing the constraint for inaccurate predictions that result in higher energy savings. MVSOM, with its low prediction rates, also led to instances of negative energy savings, as it often predicted higher energy states when the true target state was one of less energy consumption. Thus we believe that the MVSOM approach is not very viable for use in mobile embedded systems.

VRL's energy saving capability is constrained because it does not disable device interfaces (only deactivates location logging or reduces logging rate), ignoring idle energy consumption. *Overall, compared to VRL, the average energy savings of our KNN algorithm is 25.6%, and that of the NN approaches is 11.7% (H=18), 24.1% (H=9), and 24% (H=3) for real user patterns.*

### 6.3 Implementation Overhead

The prediction and energy savings results presented in the previous sections were obtained using a Python implementation of the algorithms on a 2.6 GHz Intel® Core i5™ processor. When considering real-world implementation, it is important to consider the implementation overhead of the individual algorithms. Current hardware in mobile devices on the market today is quickly catching up to the abilities of modern stationary workstations (e.g. Google's Galaxy Nexus – 1.2 GHz duo-core processor). We determined the maximum implementation overhead for the Google Nexus One 1 GHz Qualcomm QSD 8250 Snapdragon ARM processor [22], as shown in Table 3 below. The values shown in the table are average prediction times for each algorithm at runtime, and do not include initial training time. KNN's run time is several orders of magnitude larger than any of the other algorithms, because all computations are deferred until classification. Therefore, although KNN is as good as or better than the neural network (NN) based approach in terms of energy savings and prediction*, the non-linear logistic regression with NN approach is preferable because of its fast execution time*.

**Table 3. Average algorithm run times**

| Algorithm | Average Run time (seconds) |
|---|---|
| LDA | 0.00361 |
| MVSOM [13] | 2.15023 |
| KNN | 254.131 |
| LLR | 0.00307 |
| NN (all 3 variants) | 0.02501 |
| VRL [9] | 0.05140 |

Learning algorithm training is also an important consideration. There are several approaches to when and how the algorithms can be trained quickly and in an energy-efficient manner without affecting user QoS on a real mobile platform. One possibility is to train only while the mobile device is plugged in and charging. Our plans for future work include implementation of the abovementioned energy-saving training techniques on a real mobile device.

In summary, our LDA and LLR approaches have the lowest implementation overhead and can result in high energy savings, but often at the cost of user satisfaction. Although our KNN approach is very effective in terms of prediction accuracy and energy savings, its unreasonable implementation overhead renders it unacceptable for real-world applications. The prior work with MVSOM [13] provides low energy savings as a result of its poor prediction accuracy, and takes a long time to run; whereas VRL [9] has low run time but also very low energy savings. Our *non-linear logistic regression with neural network approach that uses the fast scaled conjugate gradient training method and with the number of hidden unites equal to half the attribute space* provides good accuracy, good energy savings, and demonstrates the best adaptation to various unique user usage patterns, while maintaining a low implementation overhead.

## 7. CONCLUSIONS

In this work we demonstrated the effectiveness of using various machine learning algorithms on user spatiotemporal and device contexts in order to dynamically predict energy-efficient device interface configurations. We demonstrated up to a 90% successful prediction using neural networks and k-nearest neighbor algorithms, showing improvements over the self-organizing map prediction approach proposed in [13] by approximately 50%. In addition, approximately 85% energy savings was achieved for minimally active users with an average improvement of 24% energy savings compared the variable rate logging algorithm proposed in [9] for our best approach involving non-linear logistic regression with neural networks that also has high prediction accuracy and low overhead.

## 8. REFERENCES

[1] H. Petander, "Energy-aware network selection using traffic estimation," in *MICNET '09*, pp. 55-60, Sept. 2009.

[2] M. Ra, et al., "Energy-delay tradeoffs in smartphone applications," In *MOBISYS '10*, pp. 255-270, Jun. 2010.

[3] I. Constandache, et al., "EnLoc: energy-efficient localization for mobile phones," in *INFOCOM '09*, pp. 19-25, Jun. 2009.

[4] K. Lin, et al., "Energy-accuracy trade-off for continuous mobile device Location," in *MOBISYS*, pp. 285-298. Jun. 2010.

[5] F. B. Abdesslem, et al., "Less is more: energy-efficient mobile sensing with SenseLess," in *MOBIHELD '09*, pp. 61-62, Aug. 2009.

[6] J. Paek, et al., "Energy-efficient rate-adaptive GPS-based positioning for smartphones," in *MOBISYS '10*, pp. 299-314, Jun. 2010.

[7] I. Shafer, M. L. Chang, "Movement detection for power-efficient smartphone WLAN localization," in *MSWIM '10*, pp. 81-90, Oct. 2010.

[8] M. Youssef, et al., "GAC: energy-efficient hybrid GPS-accelerometer-compass GSM localization," in *GLOBECOM '10*, pp. 1-5, Dec. 2010.

[9] C. Lee, M. Lee, D. Han, "Energy efficient location logging for mobile device," in *SAINT '11*, pp. 84, Oct. 2010.

[10] K. Nishihara, K. Ishizaka, J. Sakai, "Power saving in mobile devices using context-aware resource control," in *ICNC '10*, pp. 220-226, 2010.

[11] Z. Zhuang, et al., "Improving energy efficiency of location sensing on smartphones," in *MOBISYS '10*, pp. 315-330, Jun. 2010.

[12] Y. Wang, et al., "A framework of energy efficient mobile sensing for automatic user state recognition," in *MOBISYS '09*, pp. 179-192, 2009.

[13] L. Batyuk, et al., "Context-aware device self-configuration using self-organizing maps" in *OC '11*, pp. 13-22, June 2011.

[14] T. Anagnostopoulos, C. Anagnostopoulos, S. Hadjiefthymiades, M. Kyriakakos, A. Kalousis, "Predicting the location of mobile users: a machine learning approach," in *ICPS '09*, pp. 65-72, July 2009.

[15] T. Mantoro, et al., "Mobile user location determination using extreme learning machine," in *ICT4M*, pp. D25-D30, 2011.

[16] J. Kang, S. Seo, J. W. Hong, "Usage pattern analysis of smartphones," in *APNOMS '11*, pp. 1-8, Nov. 2011

[17] B. K. Donohoo, C. Ohlsen, S. Pasricha, "AURA: An Application and User Interaction Aware Middleware Framework for Energy Optimization in Mobile Devices," in *ICCD '11*, pp. 168-174, Oct. 2011.

[18] S. Choi, et al., "A selective DVS technique based on battery residual microprocessors and microsystems," Elsevier Sc., 30(1):33–42, 2006.

[19] F. Qian, et al., "TOP: Tail Optimization Protocol for Cellular Radio Resource Allocation," In ICNP, 2010.

[20] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani. "Energy Consumption in Mobile Phones: a Measurement Study and Implications for Network Applications," In IMC, 2009.

[21] S. Swanson. M.B. Taylor. "Greendroid: Exploring the next evolution of smartphone application processors," Communications Magazine, IEEE. Vol 49. Issue 4. April 2011.

[22] HTC, "Google Nexus One Tech Specs," http://www.htc.com/us/support/nexus-one-google/tech-specs

[23] C. M. Bishop, "Pattern Recognition and Machine Learning," 1st ed. New York: Springer Science+Business Media, 2006

[24] E. Alpaydin, "Introduction to Machine Learning," 2nd ed. Massachusetts: The MIT Press, 2010.

[25] M. Moller, "Efficient Training of Feed-Forward Neural Networks," Ph.D. dissertation, CS Dept., Aarhus Univ., Arhus, Denmark, 1997.

[26] Android Developers, official website, http://developer.android.com/index.html.

[27] Monsoon Solutions Inc., official website, http://www.msoon.com/ LabEquipment/PowerMonitor, 2008.