AMERICAN UNIVERSITY OF BEIRUT

CONTEXT-AWARE DYNAMIC DESIGNS FOR ENERGY EFFICIENT MOBILE SENSING

by
SIRINE HASSAN TALEB

A dissertation
submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
to the Department of Electrical and Computer Engineering
of the Faculty of Engineering and Architecture
at the American University of Beirut

Beirut, Lebanon
April 2017

AMERICAN UNIVERSITY OF BEIRUT

CONTEXT-AWARE DYNAMIC DESIGNS FOR ENERGY EFFICIENT MOBILE SENSING

by

SIRINE HASSAN TALEB

Approved by:

Dr. Ayman Kayssi, Professor	Chair of Committee
Electrical and Computer Engineering	[illegible]
Dr. Zaher Dawy, Professor	Co-Advisor
Electrical and Computer Engineering	[illegible]
Dr. Hazem Hajj, Associate Professor	Co-Advisor
Electrical and Computer Engineering	[illegible]
Dr. Fadi Karameh, Associate Professor	Member of Committee
Electrical and Computer Engineering	[illegible]

Dr. Wassim El-Hajj, Associate Professor	Member of Committee
Computer Science	[illegible]
Dr. Brian Evans, Professor	Member of Committee
The University of Texas at Austin	[illegible] on behalf of Prof. Evans
Dr. Rached Zantout, Associate Professor	Member of Committee
Rafik Hariri University	[illegible]

Date of the dissertation defense: May 10, 2017

AMERICAN UNIVERSITY OF BEIRUT

THESIS, DISSERTATION, PROJECT RELEASE FORM

Student Name: TALEB SIRINE HASSAN

Last	First	Middle
TALEB	SIRINE	HASSAN

○ Master's Thesis ○ Master's Project [✓ Doctoral Dissertation]

☐ I authorize the American University of Beirut to: (a) reproduce hard or electronic copies of my thesis, dissertation, or project; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

☒ I authorize the American University of Beirut, to: (a) reproduce hard or electronic copies of it; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes after: **One X year from the date of submission of my thesis, dissertation or project.**
Two ___ years from the date of submission of my thesis , dissertation or project.
Three ___ years from the date of submission of my thesis , dissertation or project.

[illegible]
Signature

Sept. 12, 2017
Date

This form is signed when submitting the thesis, dissertation, or project to the University Libraries

Acknowledgements

First and foremost, my profound praises and thanks to the Almighty, Allah, for His showers of blessings throughout my research work, and for bestowing upon me the knowledge to reach my goals.

There are no proper words to convey my sincere gratitude and respect for my thesis advisors, Professor Hazem Hajj and Professor Zaher Dawy, for their continuous guidance, patience, advice and encouragement throughout my doctoral study. They have always encouraged me to strive to overcome the challenges I thought unsolvable. Both provided me with exceptional mentorship which has positively influenced my problem-solving and research skills. Their profound knowledge and dedication will remain an inspiration for me throughout my future career. It has been an honor and privilege to work under their guidance.

I would like to acknowledge my thesis committee members, Prof. Ayman Kayssi, Prof. Fadi Karameh, Prof. Wassim El Hajj, Prof. Brian Evans, and Prof. Rached Zantout for generously granting me their time, support, guidance, and constructive comments. I would also like to thank the American University of Beirut for offering me the opportunity to join its accelerated PhD track, the Department of Electrical and Computer Engineering, the AUB University Research Board (URB), King Abdulaziz City for Science and Technology (KACST), and the National Council for Scientific Research - Lebanon (CNRS-L) which granted me the CNRS-L/AUB doctoral scholarship to support my PhD study.

I am extremely grateful to my parents, Hassan and Samia, for their love and prayers. They have taught me, and showed me, the meaning of hard work and persistence. Thank you both for always expressing how proud you are of me. To my mother, thank you for being my first teacher, for providing me with unflinching support. To my father, thank you for being my source of inspiration and strength throughout my life. I am also truly grateful for the love, encouragement, and patience of my husband, Ali. Thank you for joining me in this journey. Without you by my side, it would not have been possible to accomplish this dream. My very special thanks go to my baby, Karim, who was growing inside me while writing this thesis. He was, and still is, the guiding light which made all the difference in my life, and who motivated me to successfully finish this work.

I also express my thanks to my entire extended family for always showing confidence in me, and supporting me to complete this thesis successfully. I owe thanks to a very special person, my Aunt, Samira for her generous care and valuable prayers. I appreciate the love and unconditional support of my in-laws.

Last, but certainly not least, I would like to gratefully acknowledge the support of my friends and colleagues. Special thanks go to my friend and sister, Nadine, for her frequent support and help. To my roommate and sister, Maya, thank you for helping me get through difficult times. I am indebted to my many colleagues who provided a stimulating and inspiring work environment. I am grateful to Rasha, Lama, Gilbert, Ramy, Reem, Carine, and Wissam. Special thanks to my AUB friends: Nour, Al Zahraa, Nadine, Layal, and Raneem. I would also like to thank my best friends since school: Eman, Zainab, Nouhad, Mariam, Lamis, Fatima, and Sara.

Finally, my deep gratitude goes to all the people who have encouraged me to complete this scholarly journey.

An Abstract of the Dissertation of

Sirine Hassan Taleb for Doctor of Philosophy
Major: Electrical and Computer Engineering

Title: Context-Aware Dynamic Designs for Energy Efficient Mobile Sensing

Technological advances in the past decade have driven a significant evolution of various technologies towards Internet of Things (IoT) in domains such as sensing, communications, and computing. In particular, today's smart mobile devices have become equipped with various specialized sensors and can be augmented with external wearable sensors to collect vital data. As a result, new context recognition applications have been developed to understand and analyze user's context such as activity, location or health conditions. Often times, multiple context applications are running simultaneously on smart devices placing strenuous demands on their battery-limited resources. As a result, to support the growing requirement and proliferation of such applications in IoT, there is a need to optimize the usage of the limited computing and sensing resources. To this end, this dissertation proposes a novel context-aware dynamic sensing framework that enhances the trade-off between energy consumption and accuracy in context detection and recognition. The goal is to have decisions customized for each context, and thus develop optimized context-aware designs for three aspects of sensor usage: how many samples to collect every time a sensor is triggered, when to schedule sensors' data collection, and which sensors to use. For the choice of sensors' samples, we propose two sampling mechanisms, where one is based on information theory, and the second is based on recent advances in deep learning. For sensor scheduling, we design an optimized real-time strategy based on the Viterbi algorithm with customized rewards to decide dynamically on when to trigger the sensors for data collection. Finally, for sensor selection, we develop a new mechanism to alleviate the energy limitations by considering synergy in sensor usage in multi-context setting. A new context ontology is built to support the framework selection strategy and the extraction of commonalities and differences between multiple context recognition models along with descriptive specifications for each model. The ontology enables the framework generalization and application to any context already captured. Performance evaluations of the proposed framework components for a wide range of scenarios demonstrate their effectiveness and applicability compared to state-of-the-art techniques from the literature.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Nearly 20 years ago, the founder of ubiquitous computing -Mark Weiser- foresaw that in the 21st century the technology revolution will move into the everyday, the small and the invisible [1, 2]. By invisible, he meant profound technologies that weave themselves into everyday life until they are indistinguishable. In a way, Weiser foresaw an environment with the Internet of Things (IoT) that includes invisible tiny processors which ubiquitously communicate with each other without us noticing. IoT environment will embrace everyday objects, which will become smart and invisibly interconnected. According to Intel's analysts [3], more than 50 billion devices will be connected by 2020. As part of the IoT framework, smart mobile devices have evolved into multi-purpose devices equipped with multiple sensors. Cisco's annual report stated that by 2021, there will be nearly 1.5 mobile devices per capita, and more than 929 million wearable devices will be in use [4]. Today, we live in a ubiquitous computing environment where computing devices are present anywhere and at anytime. In support of this evolution, computing power has also been evolving to having pocket-size mobile devices with powerful processing capabilities at widely accessible prices [5]. These advances have enabled mobile devices to pervasively recognize and analyze the human context such as activity, location or health conditions. This environment has led to several opportunities and challenges which are further discussed in this chapter.

1.1 Motivation

User's context, such as activity, location and emotions, has gained popularity in recent decades with the increase use of pervasive computers and mobile devices in everyday life. Context awareness is a concept which has been firstly introduced in 1994 by Schilit et al. [6] as the ability of computers to sense and react to a user's situation. Therefore, context awareness has been a popular research topic for a number of years now. Furthermore, the use of mobile devices, and particularly smartphones, has been growing exponentially so that they have become the central computing and accompanying device in peoples' lives. Therefore, mobile devices took a prominent place in the progress of context awareness where the concept of mobile context awareness has been introduced and has become a popular research trend in the field of ubiquitous computing.

Recent smartphones have evolved from being mobile devices used only for communications into multi-purpose devices combining a plethora of functionalities. A recent smartphone can be described as a gateway which connects the user with several domains including healthcare, activity detection, and navigation. Today's smartphone hosts a growing set of small size, cheap and powerful embedded sensors such as GPS, accelerometer, light, temperature, a high quality microphone and a camera. These sensors collect physical data about the user and her environment to support personalized analytic services. Therefore, more sensors are being deployed on a large scale and it is predicted that the numbers will continue to grow rapidly [4]. Moreover, wearable external sensors such as ECG and EEG offer sensory data which can be collected by mobile devices through short range wireless communication technologies such as Bluetooth or ZigBee. Recently, users are carrying more mobile devices besides the smartphone such as wearing smartwatches or smartglasses. For example, the number of smartwatch users is expected to reach 28.5 million units sale this year, and vendors expect this number to reach 54 million units by 2021 [7].

As a result of this wide deployment of sensors, large amounts of raw data will be generated and present a wealth of information for extracting context that describes the circumstances surrounding the user such as user's activity, health condition, and location. In other words, context recognition refers to the automated identification of the following aspects related to the user: what is the user doing? Who is the user interacting with? How is the user achieving the task at hand? When is the user in a particular condition? Each context can have several states; for example, user's activity can be running, meeting, or shopping as shown in Figure 1.1. Extracting context allows understanding of the human behavior and facilitates both user-to-machine and machine-to-machine communications. Studies show that people's behaviors and lifestyles are highly dependent on their own context, their surrounding environment, and their interaction with others [8]. Furthermore, people are driven by desire for self reflection and self improvement to understand themselves and self-organize their daily lives [9].

The emergence of wearable devices such as the Apple Watch [10], has facilitated the development of new applications that infer context from smartwatches and external sensors [11, 12]. These advances have provided a fertile ground for researchers to explore efficient solutions for automated recognition of user's context such as activities, social interactions or emotions. There has been a growth of context-aware applications and systems, that go beyond context recognition to make optimized decisions based on the extracted context. Such applications first extract contexts, and then make decisions based on the context to assist users in improving their lifestyles, and providing them with personalized situation-specific

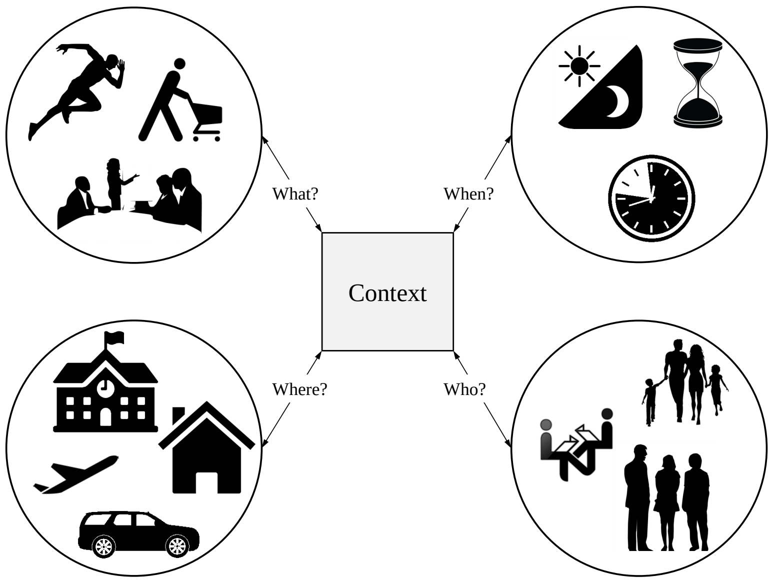


Figure 1.1: An operational overview of context.

services. A multitude of such applications is currently distributed through commercial app stores such as Apple App Store and Google Play.

Research in this field has recently evolved where several systems and frameworks have been proposed by researchers to make use of context at multiple scales. For example, context information plays a key role that drastically enriches fast developing fields such as automated driving [13], drones, mobile health, etc. Of particular interest is the emergence of context-aware applications for the health industry which has been considered as a cornerstone technology. Using pervasive mobile healthcare, the patient can monitor her own health condition and the doctor can check remotely the patients' health and provide immediate urgent care services in case of emergency. Furthermore, healthy users can also exploit the advantages of pervasive mobile healthcare through monitoring their bodies, improving their lifestyles, and early diagnosing any health problem. Smart health applications depend on physiological data collected from sensors which are either wearable or embedded in mobile devices. This data (blood pressure, temperature, heart rate, etc...) is used either for patients' treatment procedures [14, 15] or for chronic disease monitoring such as heart problems [16]. Statistics show that by 2020, IoT healthcare market segment will be worth 117 million dollars [17]. By 2018, 200 million people will use wearable devices that measure heart rates [18].

Another trending research is the use of sensors to improve the general lifestyle of users. For example, wearable physiological sensors are used to test the fitness level of users [19, 20]. In addition, fitness level has been tested against weight loss and dietary programs by tracking the user's activity [21]. In [22], the author investigates the possibility of recognizing bipolarity from behavior changes using smartphone sensors. In addition, the users' psychological life is studied through measuring stress using the smartphone's built-in accelerometer to detect behaviour and correlate with stress levels and emotions [23]. One cannot easily overstate the magnitude of the capabilities provided by recent sensors. Due to the importance and potential of mobile sensing in the activity tracking and emotion recognition fields, we aim to use activity and emotion as the requested contexts in our case study for implementing and evaluating the thesis's proposed sensing designs.

A rapidly growing research is to design context-aware systems which enhance the sensing functionality so that it provides advanced sensing decisions to optimize the usage of the limited mobile computing and sensing resources while still meeting the applications' requirements. Since each context has its own computing and sensing requirements, there is a great opportunity for devising strategies to optimize usage of resources based on context.

1.2 Challenges

Despite the rapid evolution of embedded and external sensors in recognizing context, there are several challenges that face the full deployment of available sensors in context recognition applications. Continuous and extensive sensing to support context recognition places a heavy workload on smartphones and sensing devices with limited battery capacity. Unfortunately, the recent developments in smartphone sensing and computing have not been accompanied by comparable advancements in battery lifetime enhancement.

Furthermore, deploying embedded and wearable sensors for context recognition impose additional heavy workloads on smartphone's limited resources; hence, the battery depletes faster when turning on sensors. For example, a typical GPS consumes 166 mW power which completely drains the smartphone's battery in six hours [24]. In addition, machine learning algorithms used for context detection require sufficient raw data from sensors in order to extract accurate context. Therefore, an efficient sensing system should decide on the amount of raw data samples to extract from each sensor such that accurate classification is obtained.

There have been some approaches to solve the battery drainage problem where early efforts tended to avoid continuous sensing through assigning lower sampling frequencies to sensors which help reduce energy consumption. However, in most cases, using a lower sampling frequency gives a less accurate classification and causes delays in detecting a context state change. Moreover, the response of some critical applications should be almost real-time. For example, fall detection of elderly people requires the fastest possible detection of any change in the body's posture to avoid risks [25]. Such applications rely on continuous recognition of the user's current state and fast detection of any critical context change. In such cases, one can run sensing continuously to ensure best accuracy and lowest delay;

however, that strategy incurs high energy consumption. Therefore, a dynamic sensing schedule is required to decide when to trigger sensors and trade-off energy, accuracy and delay.

While the above challenges exist for every context-aware application, the problems become more significant when multiple applications are running simultaneously. The multiple sources of raw data present the curse of dimensionality challenge [26]. Hence, to alleviate the energy limitation with mobile devices in multi-context setting, a smart sensor selection algorithm is needed to choose the right sensors to achieve synergy across applications.

1.3 Thesis Overview and Proposed Framework

Although sensors' widespread adoption has led to enormous advancements in developing more context applications, they come with challenge of limited resources. Therefore, the thesis of this dissertation is:

> *A novel smart dynamic sensing framework can be designed for the collection of data from external and embedded sensors while trading off resource consumption, application accuracy, and recognition delay.*

Building energy efficient context-aware sensing strategies requires deciding how many samples to collect every time a sensor is triggered, when to schedule sensors' data collection, and which sensors to turn on. Finally, the proposed system will use a case study among multiple context-aware applications to evaluate the feasibility and benefits of the proposed efficient sensing strategies. Therefore, the research questions are:

1. How many samples to collect upon triggering a sensor and for how long should it be kept running to achieve the right balance between energy consumption and application accuracy?

2. When to trigger a sensor for data collection while using energy efficiently and minimizing the delay in detecting a context state change?

3. Which sensors to trigger in a multi-context setting such that the required context is achieved while trading off resource consumption and application accuracy?

4. How to integrate the different proposed context-aware designs so that it provides a more cohesive solution?

To answer the question "How long to collect data?", we intend to study the sensors' properties and the requirements of each state where some states might require less samples to detect context; thus, less energy consumption. To answer

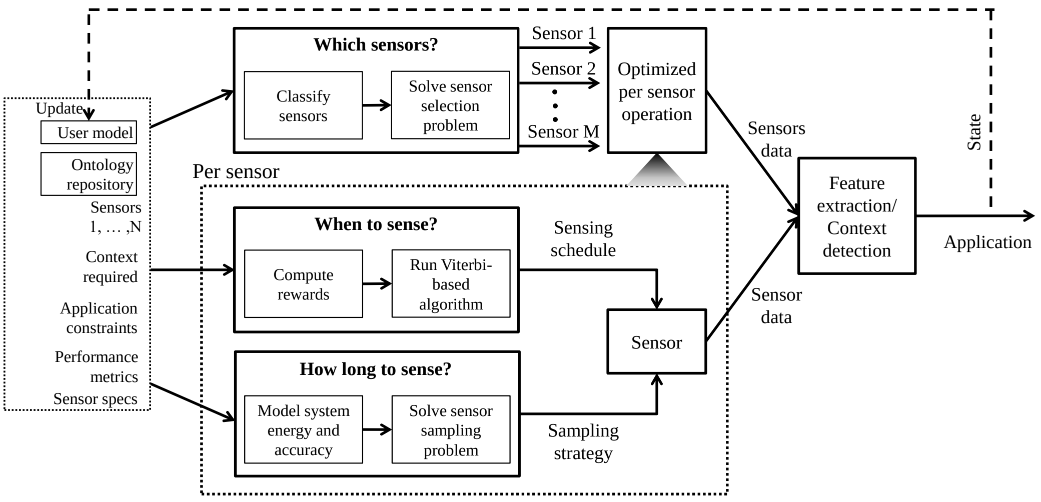


Figure 1.2: Challenges addressed in thesis framework. The boxes in the middle capture the three areas of challenges: how long to sense (Chapter 3), when to sense (Chapter 4), and which sensors (Chapter 5).

the question “When to trigger sensing?”, we intend to explore the idea that non-stop continuous data sensing is not essential for context detection since human context (such as health condition, activity, or location) does not change all the time. Instead, we just need the sensing devices to be ready to collect data when needed. For example, results in [27] showed that people move only 20% of the time during a day; sensing below this frequency will likely result in inaccuracies. Finally, to answer the question “Which sensors to operate?”, we intend to choose sensors which add value to the process of detecting multiple contexts simultaneously. We propose an integration strategy towards a holistic approach that combines sensor scheduling and sensor selection.

The thesis goal is to optimize the trade-off between: the sensing energy, the delay in detecting contextual state change, and the accuracy of extracting the correct contextual state. Figure 1.2 provides the general work flow of the thesis's objectives. The inputs to this system are: 1) the user model which captures user transition model such as the probabilities of transition from one state to another at any time instant, 2) the developed context ontology repository which captures context recognition models along with descriptive specifications for each context recognition model such as required sensors and machine learning parameters, 3) the different available embedded and wearable sensors, 4) the required context for each application, 5) the application constraints where some applications can tolerate errors whereas other applications require the minimum delay possible such as real-time health applications, 6) specifications for each sensor such as its required operating time to obtain a feature and its sleeping time; and 7) the performance metrics: energy, delay and accuracy which are used to define costs and gains. The output of this system is the set of chosen sensors with their corresponding sensing schedules and sampling strategies.

The upper block “Which sensors?” shows how we intend to select sensors. The selection is based on the required context and the application accuracy requirements. Recently, users are carrying more mobile devices besides the smartphone such as wearing smart watches or smart glasses. Hence, multiple alternatives exist for sensing and recognizing the same context. Several contexts can be retrieved via different multi-modal embedded and wearable sensor combinations [28]. For example, a recent study by Miao et al. [29] aimed at monitoring the ECG signal to prevent cardiovascular disease and detect symptomatic signs. They proposed combining ECG with built-in kinematic sensors (accelerometer, gyroscope, and magnetic sensor) so that the system would recognize user's physical activities and identify ECG abnormal patterns. For each selected sensor, we have two additional main objectives: the first “When to sense?” is to derive an optimized sensing strategy that decides when to turn the sensor on. The second objective “How long to sense?” is to derive an efficient sampling strategy providing the number of samples needed and the sufficient duration of sampling to obtain informative features.

The extracted sensor data then goes through the Feature extraction/ Context detection block which detects the contextual state and feeds it to the corresponding application. Each context can have several states. For example, the location of the user can be in her home, her work, or in a given mall. Our system is adaptive in that each time the recognition system detects a state, the user model is updated with the current state as shown by the feedback from the output to update the user model input.

Figure 1.3 provides an example output in relation to a user's timeline. The first timeline of the figure shows the actual user states. The user is in the first contextual state for a specific actual time such as the user walking. Then, the user state switches to the second state and stays for another actual time period such as sitting. The second axis contains the estimated user model which is learned from user behavior based on historical data. This timeline holds the estimated time periods for the current state and the probabilities of switching to another state at any time instant. The remaining timelines show examples of the decision outputs from the proposed framework for the different selected sensors. The system chooses which sensors to be triggered. For each sensor, the system decides when to trigger data collection. These are represented in the figure by the impulse arrows. Additionally, the sensor-specific sampling frequency and the window size decide on the amount of data collected each time the sensor is triggered.

In the remaining of this section, we provide an overview of the dissertation.

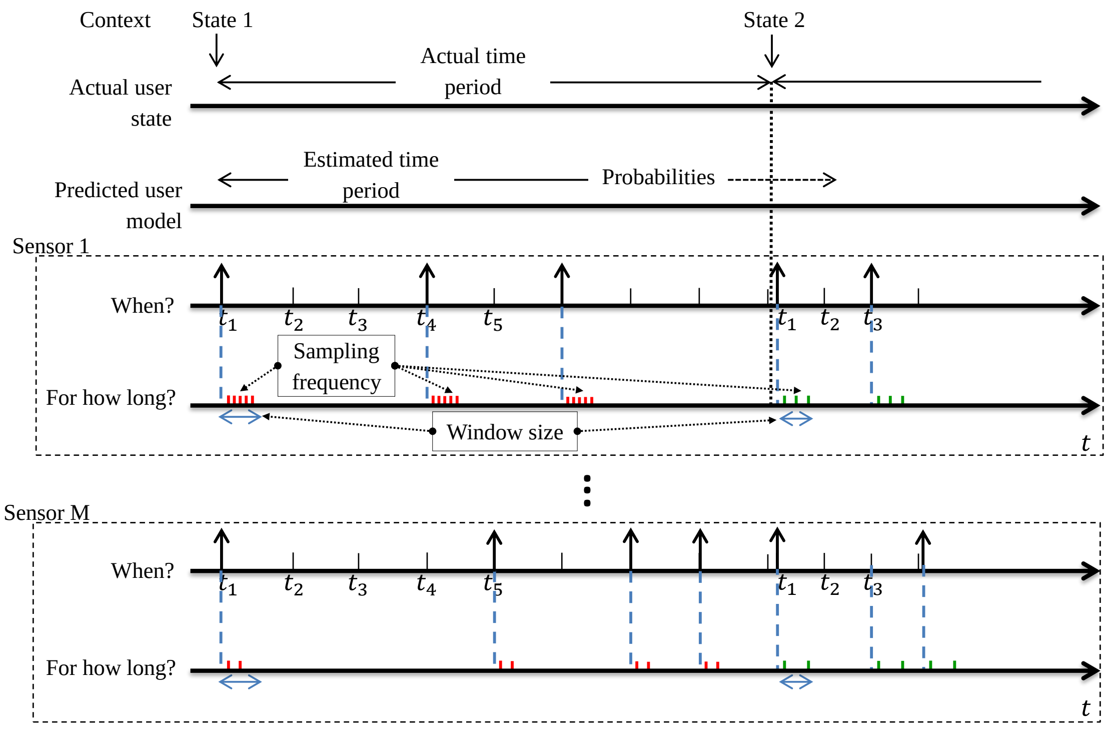


Figure 1.3: A unified model for user context and smartphone sensing: first timeline reflects the real state change, predicted user model defines the derived system parameters needed for the sensing designs, the "for how long?" shows the sensor-specific and state-dependent sampling parameters, the "when?" timeline shows the sensing schedule which presents the instants at which sensing should be triggered for each of the M selected sensors.

1.4 Dissertation Structure

The rest of the dissertation is organized as follows. Chapter 2 includes background information, and introduces some existing context-aware sensing framework designs and recent existing context-aware applications. Chapter 3 includes two proposed sampling mechanisms that determine for how long the data needs to be collected once the sensor is triggered. The first mechanism is based on information theory, and the second is based on recent advances in deep learning. Chapter 4 provides a mechanism for sensor scheduling based on using Viterbi algorithm with customized rewards to decide on when to trigger the sensors for data collection. Chapter 5 describes our sensor selection mechanism which is ontology-based to enable the determination of commonalities and differences between several context recognition models.

1.4.1 Chapter 2: Background and Literature Review

This chapter introduces the main concepts that we use in this dissertation such as defining context and context awareness. In particular, we focus on how context awareness has evolved through years to serve as the core feature of ubiquitous and pervasive computing systems [30]. Many researchers have studied context awareness and proposed several approaches to capture contextual information to be deployed in fields such as assisted living, smart cities, automated driving, etc. Hence, in this chapter, we also analyze some existing context-aware frameworks along with context-aware available applications. Moreover, thorough analyses of existing energy efficient designs are provided in this chapter to compare with our proposed context-aware sensing designs.

1.4.2 Chapter 3: Sensor Sampling

This chapter proposes two methods to decide on how many samples to collect upon triggering a sensor. These methods build upon the fact that continuous sensing mechanisms in sensors cost high energy consumption to support accurate contextual detection. However, lowering sampling frequency may lead to the misclassification of critical contextual events which results in lower accuracy. Hence, there is a trade-off between the classification accuracy and the energy consumption. In the first method, we formulate the energy-accuracy trade-off as an entropy-based optimization problem, in order to propose an efficient algorithm based on user activity and phone sensor parameters. The ultimate goal of this method is to design and evaluate an optimized sampling mechanism that trades-off energy and accuracy based on the current physical activity of the user. Experiments demonstrate the gains of the proposed algorithm with 43% reduction in energy consumption for a case study based on real data traces. In the second method, we exploit the advantages of Deep Neural Network (DNN) with ensemble classification of other complementary machine learning approaches to determine the best sensor sampling frequency for the recognition of a given context. DNN relies on raw data for classification while the other complementary methods (such as Decision Tree and Naïve Bayes) use feature recognition to classify data. Therefore, our approach provides a range of granularity from raw data. We prove the robustness of our approach in experiments which show high accuracy in context recognition. In addition, real experiments demonstrate the energy gains of the proposed algorithm which reach 87% reduction in energy consumption when compared to continuous sensing. Finally, this chapter is summarized by providing a comparison between these two proposed methods. This chapter is based on the following publications: [31] and [32].

1.4.3 Chapter 4: Sensor Scheduling

This chapter proposes a context-aware mechanism to optimize the trade-off between delay in sensing context change and energy consumption via deciding when to trigger the sensors for data collection based on the user's behavior. Monitoring context depends on continuous collection of raw data from sensors which are either embedded in smart mobile devices or worn by the user. However, continuous sensing constitutes a major source of energy consumption; on the other hand, lowering the sensing rate may lead to missing the detection of critical contextual events. Hence, deciding on when to trigger sensors becomes very critical for energy reduction. In this manner, this chapter proposes VCAMS: a Viterbi-based Context Aware Mobile Sensing mechanism that adaptively finds an optimized sensing schedule to decide when to trigger the sensors for data collection while trading off the sensing energy and the delay to detect a state change. Each context can have several states; for example, the location of the user might be at home, at work, or in mall. For a given user in a specific state, the objective of the system is to dynamically provide the time instants at which a sensor needs to be triggered, also called the sensing schedule for the specific user and the particular state. The sensing schedule is adaptive from two aspects: 1) the decision rules are learned from the user's past behavior, and 2) these rules are updated over real time whenever there is a significant change in the user's behavior. VCAMS is validated using multiple experiments, which include evaluation of model success when considering binary and multi-user states. VCAMS is also implemented on an Android-based device to estimate its computational costs under realistic operational conditions. In conclusion, test results show that the proposed strategy provides better trade-off than previous state-of-the-art methods under comparable conditions and provides 78% energy saving when compared to continuous sensing. This chapter is based on the following publication: [33].

1.4.4 Chapter 5: Sensor Selection

This chapter proposes a context-aware sensor selection mechanism that alleviates the energy limitation with mobile devices in multi-context setting. Extensive sensing may cause fast battery drainage for mobile devices. Furthermore, running multiple context recognition simultaneously will cause an even more extensive energy drain. In this sense, this chapter proposes an ontology-based framework for group sensor selection to achieve synergy across applications and thus reduce energy consumption while trading off accuracy and delay in context recognition. A new context recognition ontology is designed to capture context recognition models along with descriptive specifications for each context recognition model such as required sensors and machine learning parameters. Reliability of the ontology data is ensured by selecting data sources that have been tested and validated from existing literature publications. It captures the alternative groups of sensors for each context, the feature set, and the parameters for context recognition models. The framework also provides integration and synchronization with an optimized sensor scheduling mechanism. The proposed framework is implemented on an Android platform that provides accessibility to on-board mobile sensors and external wearable sensors, and the sensor selection strategy is tested in a multi-context setting to simultaneously detect activity and emotion contexts. The results show that our proposed strategy provides better trade-off between energy consumption, number of recognized contexts, accuracy of context recognition and delay in detecting a state change. Furthermore, the method provides 39% energy saving with comparable accuracy when compared to the most accurate group of sensors. This chapter is based on the following publications: [34] and [35].

1.4.5 Chapter 6: Conclusion and Future Work

Finally, in chapter 6, we summarize the main contributions of this dissertation. In addition, we present open research challenges and topics that need future investigation.

Chapter 2

Background and Literature Review

This chapter first provides a background that introduces the concepts of context and context awareness, with a focus on the mobile context awareness concept. The second part of this chapter provides a literature survey of state-of-the-art research in this field. Finally, Section 2.3 presents a summary which highlights the limitations of the studies conducted in the literature.

2.1 Background

The objective of this background section is to give a general overview about some important concepts related to the thesis work. We start by giving an overview about context and context awareness. Then, we explore the required tasks to infer relevant context. Application-specific frameworks have been proposed in various domains including healthcare, location detection, and activity recognition. We present some recent works in this field in Section 2.1.3.

2.1.1 Defining Context and Context Awareness

Figure 2.1 illustrates the concepts described in this section. It provides an overview of mobile context awareness starting from the sensory raw data extracted from mobile devices which undergoes several functionalities to recognize the context, then context is processed to develop meaningful context-aware applications which provide the users with recommendations, alerts or anticipations of future states.

A context is defined by Dey as:

> *Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is*

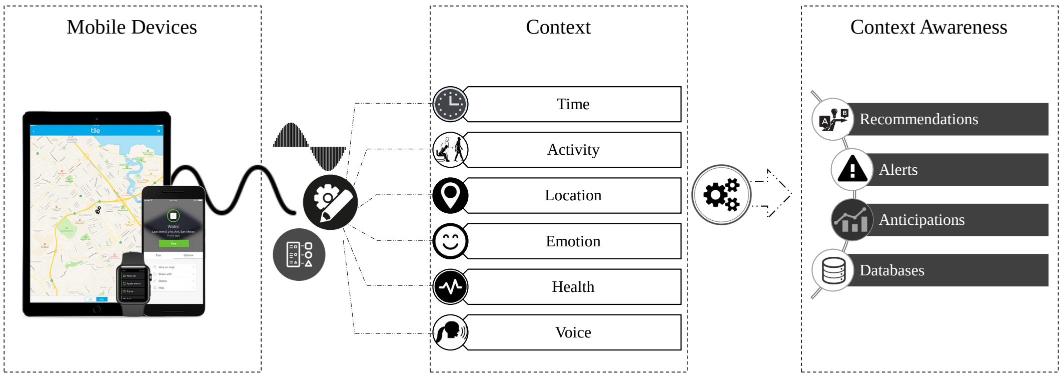


Figure 2.1: An overview of mobile context awareness from low-level raw data generated by sensors in mobile devices to high-level meaningful contextual information.

> *considered relevant to the interaction between a user and an application, including the user and applications themselves.[36]*

This definition of context is regarded as the standard description; however, it can be further particularized depending on the application. Following Dey's definition, the context covers a vast collection of information such as the user's activities, plans, emotions, health conditions, etc. Furthermore, the definition of context also describes the environment surrounding the user such as the location, surrounding people, social interactions, noises around, etc. Therefore, context encompasses two levels: personal and environmental.

In this dissertation, we refer to context as the representation of a user or entity's situation; therefore, we capture both personal and environmental contextual levels. Moreover, each context can be further categorized into states. For example, if the requested context is emotion, the states may be either neutral, happy or angry. The contextual states can sometimes be related to different granularity levels; for instance, the location of the user can be either described by GPS coordinates (latitude and longitude) or by user-defined terminologies and fingerprints such as at home, at work, or in mall.

Once the context is recognized, smart applications often make use of the inferred context to provide beneficial decisions to the mobile user. This interpretation and usage of context is known as context awareness. Context awareness was first introduced by Schilit and Theimer [6] in 1994. Later, Abowd et al. [37] criticized Schilit's definition as being too specific and defined context awareness as:

> *A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task. [37]*

Following Abowd's definition, any system which proactively provides the user with valuable information, enhances the user's life and provides tailor-made services to this user, is a context-aware system. The existence of context-aware systems created an interaction gateway between users and smart devices which is inevitable nowadays and will persist for future technologies.

The usage of mobile devices, mobile phones in particular, is constantly increasing. Today, mobile devices can be found everywhere such as Personal Device Assistants (PDAs), mobile phones, laptops and notebooks. According to Manish [38], mobile devices can be divided into three classes: mobile phones, portable computers and wearable computers. More wearable external devices are being also deployed and used in everyday life such as smartwatches, smartglasses, fitness wearables and clothing sensors. The vast majority of these mobile devices have embedded small-size sensors which are continuously developing to provide pervasive recognition of contextual information. Today, users own multiple mobile devices and move seamlessly between them throughout the day. The advancements in the mobile device technology facilitated the recognition of context and the development of a new class of context-aware applications. Therefore, the concept of context awareness has been extensively used along within mobile sensing domain; thus introducing the concept of mobile context awareness.

2.1.2 Inferring Contextual Information

The general aim of context-aware applications is to evaluate the user's situation and then take the necessary actions and decisions based on the contextual information. However, to obtain the contextual information, raw data collected from sensors undergoes several stages to infer the required context. Figure 2.2 shows the different layers that the sensory readings undergo to reach the application layer. The figure shows the flow for two context recognition techniques: the traditional machine learning approach and the deep learning approach.

For the traditional machine learning flow, the low level sensor layer identifies the sensors needed to provide the required context, then it triggers those sensors for data acquisition and obtains the raw sensory data. Different sensors have different data collection specifications such as the frequency by which data is collected and the sliding window which defines the time length needed for context recognition. In addition, some applications prefer to have an overlap between two subsequent window readings to help detect changes in the state. At the signal level, raw data is preprocessed to filter out signal variability and it is sampled based on the application's requirement.

At the features layer, feature extraction is applied to each time window to exploit hidden patterns inside the raw data. Different characteristics can be obtained from raw data such as the time domain features and frequency domain features. Time domain features use signal processing primitives such as computing the mean, standard deviation, peak and range. On the other hand, frequency domain features require more computational power to calculate characteristics

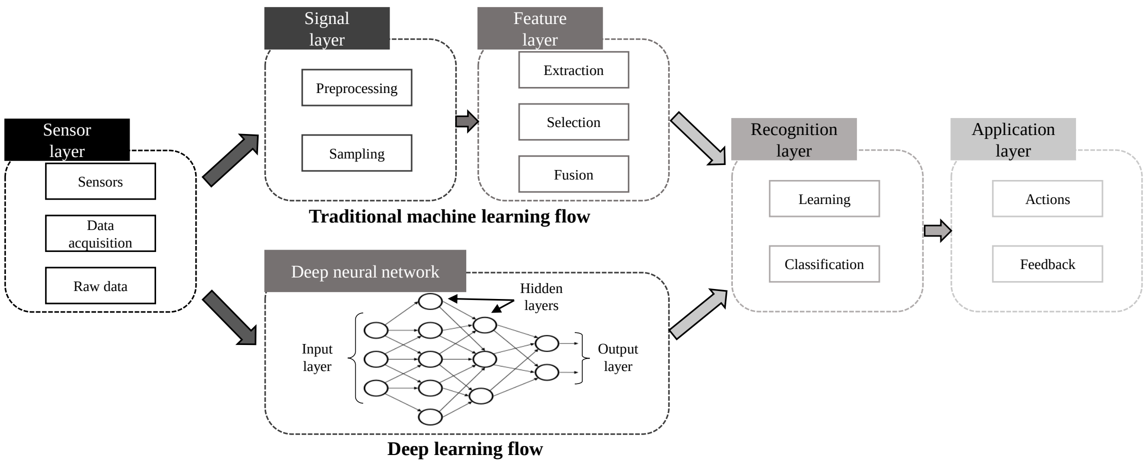


Figure 2.2: Standard context recognition layers and tasks for traditional machine learning algorithms and deep learning algorithms.

such as the Fast Fourier Transform (FFT). Several contexts can be retrieved via different multi-modal embedded and wearable sensor combinations [28]. Therefore, sensor fusion is being investigated by researchers [39]. Sensor fusion is the process of combining data extracted from multiple sensors so that it forms a more accurate and relevant information [38]. This sensor fusion might happen at different layers through the context recognition process such as sensor level, feature level, or decision level [40].

At the recognition layer, the previously collected data and its corresponding annotations are used to train the classifier. In real-time, the derived model is used to recognize the contextual state of the user. In this regard, several classifiers have been used by researchers. Some of the commonly used classification algorithms are: Naïve Bayesian, Support Vector Machines, and Decision Trees.

Extracting features and selecting relevant ones are time consuming and laborious tasks [41]; therefore, deep learning has been introduced as a new breakthrough research field. Deep learning methods eliminate the need for tedious feature engineering as shown in Figure 2.2. It provides an end-to-end computation starting from the sensory raw data all the way to the final recognized context. Deep neural network (DNN) is a deep learning architecture that contains many layers of non-linear hidden units. There are three main steps in deep learning [42]. The first step is to define the number of hidden layers to construct the Deep Neural Network (DNN). Each hidden layer's output serves as the input to the subsequent layer. The second step is the unsupervised pre-training where each layer is trained to learn weights. The third task of deep learning includes supervised fine tuning where the learned weights are fine tuned using backpropagation algorithm.

Table 2.1: Examples of sensor-based commercial applications

Application	**Context**
ElectricSleep [48]	Record sleep cycles
Smart Thermometer [49]	Check temperature near the phone
GPSLogger [50]	Check location updates
MapMyRun [51]	Record daily workout details including location
Muse [52]	Meditation
EpSMART [53]	Monitoring epileptic seizure

2.1.3 Recent Works in Context-Aware Applications

Mobile devices have gone far beyond their traditional voice communication usage. These devices introduced intelligence into their behavior and they are currently known as smart devices. They have introduced context recognition by embedding a large number of sensors that can detect user's context such as activity, emotion and location. Therefore, smart devices can run heterogeneous context-aware applications that are used to assist the users in their lives and provide personalized services. In addition, external wearable sensors have been introduced and they are advancing the level of technology and providing a boom of context. Hence, diverse context-aware applications have emerged to monitor various contexts including healthcare [43], emotion recognition [44], location detection [45], activity recognition [46] and social interactions [47]. The proposed methods in this dissertation tackle applications that request sensor data from any embedded or external sensor. In this section, context-aware applications are introduced for the following contexts: activity, location, emotion and health. We limit this context-aware applications' survey to publications since year 2010 with a focus on the most recent trends in each field. Furthermore, Table 2.1 shows examples of recent commercial context-aware applications each with its requested context.

Activity

Human activity recognition is one of the most important topics studied by an innumerable researchers. Activity recognition aims at recognizing the real-time behavior of the user and the activities that she is performing. It has been used in many aspects such as offering a better lifestyle, monitoring the fitness level or providing an ambient assisted living environment for elderly people [54]. There are studies that recognize the user's activities using solely the smartphone since it accompanies its user almost most of the time. An experiment conducted by Dey et al. [55] shows that smartphones are found in the same room as the user almost 90% of the time.

There are recent research directions in activity recognition. For example, researchers are currently investigating the use of recent machine learning algorithms such as neural networks and deep learning. Extreme Learning Machine (ELM) has been used in [56] to recognize activities. ELM is a recently introduced term in machine learning which is a form of feed-forward neural network. Another example is the use of deep learning and active learning from Hasan et al. [57]. The introduction of such revolutionized machine learning algorithms has been discussed thoroughly in [58]. The authors describe using deep learning in mobile computing as game-changing trend that exploits both the CPU and DSP of a mobile device. Hence, the field of context-aware computing is making the best use of technological advances.

Another recent trend in activity recognition is the shift from application-dependent solutions into more general solutions that can be applied to a diverse range of situations and users. For example, the work presented in [59] aims at collecting an extensive amount of data by proposing a newly developed virtual environment simulation modeling (VESM) that generates datasets suitable for testing. In [60], the authors use embedded smartphone sensors to implement a HAR scheme. They generalize their method by proposing a position-independent scheme where the proposed work deals with different positions. In [61], the authors use accelerometer-embedded mobile phones to monitor one's daily physical activities. They intend to recognize the physical activities where the mobile phone's position and orientation are varying. Using the accelerometer data, Capela et al. build a robust classifier that can be generalized and automated to all cases of elderly people [62]. Furthermore, a recent activity recognition approach is the use of more complex sensors than the accelerometers. There is a huge amount of work for the use of external sensors and devices such as the use of visual features and videos [63, 64].

Location

A variety of applications focus on obtaining the user's location to provide location-based services [65] that extract the user's whereabouts to provide directions, recommendations, and "check in" services. Most of these applications use exclusively the GPS to recognize the user's longitude and latitude which gives the location of the user with location accuracy of around 10 meters [66]. Several researchers focus on GPS traces to extract location and merge it with activities of users [67]. The authors in [76] use the GPS and knowledge about the transportation network (such as bus locations, spatial rail and spatial bus stop information) in order to infer the means of transportation [68]. However, GPS is an energy-hungry sensor which consumes enormous amounts of energy [69]. For instance, the operation time of Nexus 4 declines from 170 hours to 7 hours when using GPS [70]. In addition, GPS is not available in indoor scenarios.

Researchers have recently focused on finding alternatives that are less energy-hungry and can detect location in both indoor and outdoor scenarios. Most researchers are using radio fingerprints which are obtained either from WiFi [71] or from cellular networks [72]. Other studies propose using lower power sensors such as the barometer which is a relatively new sensor now present in devices [73].

Furthermore, some location aware applications use short range communication such as Bluetooth when probing a user's adjacent surrounding [74].

Some researchers proposed selecting new novel features for more accurate recognition of location and transportation mode. The authors in [75] conclude that features such as the heading change rate, the velocity change rate, and the stop rate are more robust and informative than the traditional speed and acceleration features. Another recent study which is based on location is the community sensing such as monitoring traffic on highways to offer the user with real-time traffic conditions and reroute in case of congestion [76]. In addition, researchers are making use of multi-modal sensor availability where recent efforts have been made to incorporate multiple sensors to recognize location [77, 78].

Emotion

Emotion recognition is by itself a recent field introduced into mobile computing area. The developments of wearable and sensing technologies are driving the emotion recognition where its market size is expected to grow from 6.72 billion dollars in 2016 to 36.07 billion by 2021 [79]. Pervasive sensing has facilitated measuring our feelings and monitoring our affective states. Emotions affect our everyday life and our performance since the human body responds to different emotions by several psychological and physiological expressions, biological changes, and mental thoughts. The most deployed mechanism for emotion detection is the facial recognition where a camera captures the person's face to know the current expressed emotion. For example, MoodMeter is an application which uses facial expressions to know the collective emotional conditions by counting smiles on a college campus [80].

Recently, researchers are using biosensors to detect emotions. For instance, some researchers use several biosensors through a biosignal acquisition apparatus that collects multimodal data, such as combining the Electromyography (EMG), Electrocardiography (ECG), Electrodermal Activity (EDA), Blood Volume Pulse (BVP), Peripheral Temperature (SKT), and Respiration (RESP) in [81]. Other studies prefer using cheap and low-power sensors such as the accelerometers to detect the sitting position of the user which reflects the current emotional state [82]. In [83], the authors detect emotion from gestures based on the arm movements using 3D inertial sensors.

Smartphones offer a rich set of user interaction and signals that can be used for emotion recognition. For example, the user's typing characteristics, applications being used, and interactions with others on the phone, are all indicators that can reflect the user's emotional state. MoodScope [84] is a system that studies the communication patterns of the user on the phone and associates it with certain moods. Furthermore, many companies are developing commercial sensors that can be used to monitor the user's emotions and provide meditation sessions such as Affectiva sensors [85], BMS smartband [86], and Muse [87].

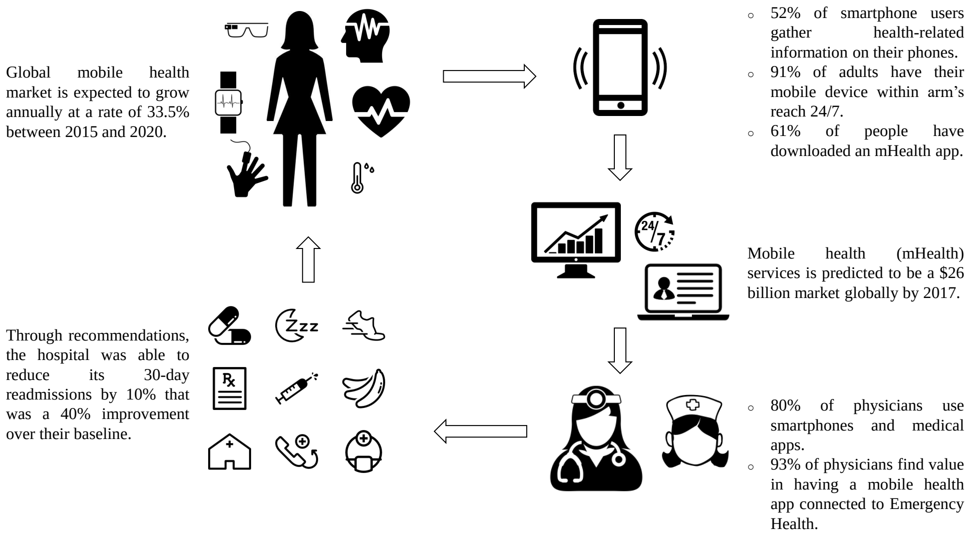

Global mobile health market is expected to grow annually at a rate of 33.5% between 2015 and 2020.
52% of smartphone users gather health-related information on their phones.
91% of adults have their mobile device within arm's reach 24/7.
61% of people have downloaded an mHealth app.
24/7
Mobile health (mHealth) services is predicted to be a $26 billion market globally by 2017.
Through recommendations, the hospital was able to reduce its 30-day readmissions by 10% that was a 40% improvement over their baseline.
80% of physicians use smartphones and medical apps.
93% of physicians find value in having a mobile health app connected to Emergency Health.

Figure 2.3: The flow of data in mHealth domain

Health

The traditional health services and caregivers are usually available within hospitals. Patients along with their families need to contact doctors to know about the patient's health condition. The patient cannot monitor his health context state. However, this is inconvenient since the patient should track her health condition to help improve her overall health and track how her body is affected by new medicine, exercise, and changes in diet. Therefore, pervasive mobile computing gave practical solutions where patients can be monitored remotely in certain health conditions. The integration of context-aware applications in the health domain has led to the appearance of a recent active research field which is mobile health or mHealth [88]. Figure 2.3 shows the flow of data in mHealth. The flow starts by collecting raw data from the user's biosensors which are then processed by health applications and sent to physicians who provide recommendations and actions depending on the user's health situation. The statistics in the figure are taken from referral MD article [89].

The goal of recent pervasive healthcare studies is to sustain a healthy society by limiting stress, mental health conditions, and the spread of contagious diseases [90]. There are several sensors that are used in health care services. Electrocardiogram (ECG) and electroencephalogram (EEG) are the most important wearable sensors which measure the signals generated by the heart and the brain respectively [91]. ECG measures electrical signals produced by the heart during blood pumping and muscle contractions; whereas, EEG detects electrical activity in the brain and provides evidence of how the brain functions. Potential applications of these sensors include diagnosing diseases such as congestive heart failure and managing neurodegenerative conditions such as Parkinson's disease

[92]. Recently, Duc et al. [93] assessed the spine mobility using wearable sensor which measures the kinematics of the head and the thorax kinematics.

2.2 Literature Survey

Recent mobile devices are equipped with high-end processors and several embedded sensors. Therefore, many types of context recognition applications have been developed which recognize several contexts such as activity, location and emotion. Most of these applications require the development of a complete context-aware framework that can abstract the context recognition process. In this section, we present an overview of the proposed context-aware framework designs and applications. In Section 2.2.2, we present some of the approaches that have been considered in the literature to capture contextual information from sensors in an energy-efficient mechanism.

2.2.1 Context-Aware Sensing Framework Design

Over the past recent years, context-aware framework designs have been proposed by researchers and applied in a variety of applications. The growing concept of using context awareness in order to assist users' lives have led to the development of a special context-aware middleware as an abstract layer for context-aware applications. Developing context-aware middlewares and frameworks has been the objective of many researchers in a bewildering set of research areas. There are several research fields that interwind together when designing a context-aware framework such as mobile sensing, human-computer interaction (HCI), machine learning, and context prediction [94]. Most prior studies propose application-specific frameworks and provide partial solutions to the existing trade-offs. In this section, we provide a brief review of the remarkable and latest context-aware framework designs arranged in chronological order:

- Seemon [95, 96] is an early context monitoring framework that uses hierarchical sensor mechanism. The framework defines context to be the combination of location, activity and time. It allows several applications to simultaneously register their queries asking for some context. Seemon also focuses on detecting accurate context by monitoring the feature data without recognizing the exact context; therefore, it avoids collecting continuous raw data. The solution attempts to minimize the number of sensors needed to answer all these queries by maintaining an essential set of sensors. The solution does not provide answers for other types of contexts or what sensors should be used if one particular context is needed all the time.

- In [97], the authors present a sensor management approach for energy efficient mobile sensing system (EEMSS), which is a context-aware monitoring system. It explores hierarchical sensor management by powering only a minimum set of sensors. They aim at reducing energy consumption while maintaining an acceptable accuracy. The authors define context such as motion (such as running and walking), location (such as staying at home or in an office) and background environment (such as sound and quiet). EEMSS heuristically assigns fixed duty cycles to recognize user state. EEMSS defines the probabilities for transitioning from one user state to another; nevertheless, it lacks user profiling features where it cannot dynamically update the user states and respond to variant user behaviors.

- The same authors of EEMSS later proposed an algorithm that complements EEMSS and obtains the optimal sensor sampling policy under the assumption that the user state transition is Markovian [98]. They formulate the problem as a Constrained Markov Decision Process (CMDP) that outputs a sensor sampling policy to minimize user state estimation error while satisfying a given energy consumption budget. Results show that this framework saves energy when compared to uniform sampling; however, user state transitions in real data traces are not strictly Markovian.

- Another context-aware framework to manage sensor operations is Jigsaw which was proposed by Lu et al. [99]. It supports accelerometer, microphone, and GPS sensors. Jigsaw proposes a processing pipeline for each sensor so that it copes with the sensor behavior and the sensing challenges of each. It implements a duty cycling technique to achieve energy efficiency; for example, the accelerometer sampling frequency is decayed when no activity is detected. It builds on activity detection from the accelerometer data to enhance the location tracking pipeline by the GPS. The GPS pipeline formulates the problem as a Markov decision process for duty cycling the GPS based on the activity detected. It uses an optimization formulation to choose the optimal GPS duty cycle for each detected activity assuming that activities with higher speed of motion require more GPS samples. As for the microphone pipeline, it classifies the acoustic sounds as human speech or other noisy sounds. However, Jigsaw uses a Markov Model which does not represent real user traces, and it does not learn from historical user behavior since it only observes a short time window of data.

- A more recent context-aware generic system framework was proposed in [100] focusing on human activity recognition (HAR) as the requested context. The framework consists of a Hidden Markov Model (HMM) that applies duty cycling on accelerometer data to recognize the activity of the user. As for all Markov-based frameworks, this system uses a state transition matrix that saves the transition probabilities between states. The authors focus on the inhomogeneity in the user's behavior; hence, they use time-variant system parameters and update the user profile using the convergence of entropy rate. When an application requests the user's activity context, the proposed framework either recognizes a genuine user state if sensory data is available or estimates the user state using the forward and backward algorithm. Results show a considerable increase in power efficiency for acceptable accuracy range if compared against continuous sampling; however, this framework focuses on Human Activity Recognition (HAR).

- Recently, the authors in [101] propose Orchestrator which is a context-aware framework proposing sensor selection system for multiple context case. It considers sensor-based plans which should be generated by developers. These plans represent the sensing and processing modules to detect a context. In addition, Orchestrator focuses on where to perform feature computations: on the mobile device or offload to the sensor itself. Orchestrator decides which sensor to choose given that the same sensor can be deployed in different devices; for example, their approach selects the best accelerometer out of several available accelerometers on the u-watch or u-wear. However, Orchestrator's cost function to minimize is either related to energy or accuracy; hence, it does not consider the trade-off.

The field of mobile and ubiquitous computing has evolved particularly since the first extensive survey was published in 2000 by Chen and Kotz [102]. The survey discussed context-aware computing in terms of modeling, applications and limitations. Therefore, there are several studies that propose context-aware framework designs, and some are mentioned in Table 2.2. The considered studies are listed in chronological order and compared using several metrics.

Table 2.2: Important proposed framework designs

Reference	Context	Sensors	Classification Algorithms	Energy Efficiency	Implementation	Results
[103]	Activity	Wearable ACCs	HMM followed by NB	Sensor selection	19 nodes placed on the two arms of a tester to recognize 10 activity classes	Extend the network life up to 4 times while reaching 90% correct classification ratio
[95]	Activity, heart rate, stress	Wearable ACC, GPS, BVP, HR and GSR	C4.5 DT	Essential sensor set selection	Eight sensors for 12 hours to recognize binary states for each context	4.2 times greater throughput with 3.6 times less transmission reduction rate
[97]	Activity and background sounds	ACC, GPS, MIC and WiFi	DT	Sensor selection and fixed duty cycles	Participant provided with a Nokia N95 device to recognize different states	A high level of recognition accuracy (> 70%) with 75% energy efficiency
[99]	Activity	ACC, MIC and GPS	J48 DT, GMM, SVM and NB	Duty cycling the GPS using MDP	Apple iPhone and Nokia N95 to recognize activities	Reduces the power usage yet keeps the average error low
[98]	Activity	ACC	DT	Duty cycling	Nokia N95 to classify and detect the transition between Stable and Moving states	20% less state estimation error over periodic sampling
[45]	Location	GPS	Latitude and longitude coordinates	Substitution, suppression, piggybacking and adaptation	G1 Android Developer Phone (ADP1) to recognize the user's location	Reduce the GPS usage by up to 98% and improve battery life by up to 75%.
[104]	Location	MIC	GMM	None	Recorded audio on Nokia N97 containing speech in indoor and outdoor scenarios	The accuracy is low at the beginning of the experiment but starts increasing as the experiment proceeds up to 90%
[105]	Transportation mode, image recognition, sound classification and acceleration classification	GPS, camera, ACC and MIC	HMM, KNN, GMM and DT	Sensor sampling	Evaluate on exiting large and real-world data set provided by [75]	Energy and latency savings of 16% and 76% respectively
[27]	Motion and location	ACC, compass, WiFi and GPS	Decision rules	None	HTC Hero and G1 Android phones to recognize moving and stationary motion states in addition to the location of the user	91% of the points indicate the actual position within the estimated error bound
[106]	Activity	ACC, BT and MIC	GMM	Dynamic adaptation sensor sampling mechanism	Collect data using the EmotionSense [107] running on Nokia 6210 phones	Energy saving 11% with 3% less accuracy
[108]	Activity	ACC	J48 DT, Adaboost, SVM and NB	None	Nokia N95 phone to recognize 7 activities with GPS location obtained from [109]	Average accuracy of 77.14%
[100]	Activity	ACC	HMM	Sensor sampling	Blackberry RIM Storm II 9550 smartphone to recognize activities such as sitting, standing, walking and running	Provides 82% accuracy against 42% power consumption when compared to aggressive sampling
[101]	Activity, light, temperature and humidity	ACC, light, temperature and humidity	C4.5 DT	Sensor selection	Ultra Mobile PC and 6 sensors with dynamic context queries	Achieves better context recognition (95% improvement) and less energy consumption (10.7% reduction) when compared to conventional system

- ACC: Accelerometer; MIC: Microphone; BVP: Blood Volume Pulse; GSR: Galvanic Skin Response; HR: Heart Rate; GPS: Global Positioning System; BT: Bluetooth
- SVM: Support Vector Machine; DT: Decision Tree; GMM: Gaussian Markov Model; NB: Naïve Bayes; HMM: Hidden Markov Model; MDP: Markov Decision Process; KNN: K-Nearest Neighbor

2.2.2 Energy Efficient Mobile Sensing

Smartphone sensors constitute a major source for energy consumption; therefore, several approaches have been considered in the literature to capture contextual information from sensors while minimizing the energy consumed. Moreover, many studies exploit the trade-off between resources and sensing quality which is defined as the accuracy of recognizing the current contextual state of the user or the delay in detecting a state change. There are several surveys that have presented approaches used to solve these challenges from different perspectives ranging from the extracted sensing raw data to the classification algorithms used [110, 111, 112, 113, 114]. The objective of this section is to describe the various methods that researchers have proposed to address the challenges, mainly the trade-off between energy, accuracy and delay. To optimize resource usages, the applied methods in relevant prior works use approaches to select duty cycles which decide when to trigger sensing, adaptive sampling periods and sampling frequencies to decide how long sensing data should be collected, or sensor selection and hierarchical sensing mechanisms.

2.2.3 Sensor Sampling Approaches

In order to fetch data from a sensor, the application's developer needs to specify the sampling frequency. Minimizing the sampling frequency means less data collection; and thus, less sensing and communication energies. However, minimizing the sampling frequency can decrease the classification accuracy. In recent years, several approaches were applied to optimize sampling decisions from sensors embedded in mobile devices and wearable sensors such that the trade-off between energy and accuracy is optimized.

Maurer et al. study the performance of accelerometer data using different sampling frequencies; and they showed that using a sampling frequency above 20 Hz does not provide additional gains [115]. French et al. [116] propose detecting activity using selective sampling strategies which are either based on the distribution of duration times or based on the transition probabilities between activities. They propose the use of Markov model for modeling the activity behavior of a user. In A3R [117], the authors adapt the sampling frequencies according to the current activity of the user to reduce energy consumption. In addition, A3R adapts the classification feature; therefore, it chooses the tuple of sampling frequency and classification feature that provides a trade-off between energy and accuracy. However, A3R determines offline the optimal tuple for each activity; therefore, a new training offline phase would be required when new activities are detected. Furthermore, adapting the sampling frequency has been also proposed in works which detect the location of the user. For example, the proposed approaches by Paek et al. [118] and Lin et al. [119] use lower GPS sampling frequency when the GPS is inaccurate in indoor areas and some urban streets.

AdaSense [46] uses genetic programming to control body sensor sampling frequencies while meeting a user-specified accuracy to trade-off energy and accuracy. They propose the idea that detecting a state change using binary classification requires a lower sampling frequency than classifying the exact state of the user. Therefore, AdaSense uses the lower sampling frequency until it detects a state change where it shifts to the higher sampling frequency. In [120], the authors propose predefined power management policies where each policy has a different sampling rate frequency. The policy is chosen based on the type of application; for example, the sampling rate is lowered when the activity becomes less clinically relevant. Their results show that the sampling frequency has to be kept above 30 Hz to avoid misclassifications. Moreover, there are researchers who propose reducing the sampling frequency for wearable external sensors. In [121], the impact of sampling rate on energy and classification accuracy was studied for the eWatch platform. Li et al. [122] propose decreasing the sampling frequency of pedometer in mobile applications and their results showed that reducing the sensing sampling frequencies can reduce energy consumption by 50% on average. Those approaches trade-off energy and accuracy to obtain accurately the current contextual state, and some can also be used to detect change in state; however, they do not focus on rapidly detecting the critical transition between one state and another with minimal delay.

2.2.4 Sensor Scheduling Approaches

Duty cycling is the process of defining the ON/OFF cycles that trigger the sensor for data collection only when necessary or based on a periodic behavior. There have been attempts in the literature to consider sensor triggering, namely sensor duty cycling or sensor scheduling as we refer to in this thesis.

Markov Decision Process (MDP) frameworks have been used to schedule sensors based on the current state of the user [98, 123]. This statistical approach uses transition probabilities from one state to another to determine the optimal sensor sampling policy assuming a Markovian user state process. In addition, Thiagarajan et al. [124] uses a Hidden Markov Model (HMM) to estimate the user's location. In [125, 100], the authors propose an inhomogeneous framework based on Hidden Markov Model. Their model either recognizes or estimates user states when power optimization is considered. This framework focuses on Human Activity Recognition (HAR). In a very recent study, Yurur et al. [126] proposes a discrete-time inhomogeneous hidden semi-Markov model framework to recognize user activity. In addition, they utilize power-efficient sensor management strategies through changing the duty cycle assuming a Constrained Markov Decision Process (CMDP).

Jigsaw [99] is another similar system which balances the performance needs of an application and resource demands. It uses sensor specific pipelines that have been designed to cope with individual challenges experienced by each sensor.

It uses learning techniques and drives the duty cycle taking into account the activity of the user, energy budget, and duration over which the application needs to operate. This is done by learning an adaptive sampling schedule using a Markov Decision Process. However, Markov models assume predetermined systems which know in advance the true user state which might turn out to be a wrong assumption. In addition, user state transitions in real data traces are not strictly Markovian [98]. In [47, 106, 127], authors use learning techniques to control the sampling periods of the sensors. Their decision whether to sense or not is based on the probability of sensing. The sensing action results in either success if unmissable event or failure if missable event.

Furthermore, in the participatory sensing scenarios where multiple users exist, most researchers focus on giving each user the ability to specify the amount of energy that she wants to dedicate for sensing. For example, the authors in [128] allow the users to define their energy budget which they are willing to spend on sensing. Accordingly, the adaptive scheme defines the corresponding sensing schedule. In addition, there are proposed methods that trigger sensing only when special events happen. For example, the authors in [129] propose an approach that schedules the sensor based on the upcoming location queries from the server. Another approach triggers the sensor when desired events should be captured such as when the user approaches predefined important locations [130].

2.2.5 Sensor Selection Approaches

Sensor group selection is one of the techniques used to minimize the sensing energy consumption through turning off unnecessary sensors while trading off energy with classification accuracy and detection delay. Increasing the number of used sensors induces more ground truth data and thus may lead to higher accuracy; however, it causes extra energy consumption [113].

Zappi et al. [103] consider a gesture recognition application which turns on only necessary sensors depending on their contribution to the recognition accuracy. Hence, it minimizes the number of sensors used, thus minimizing energy. They define a minimum desired accuracy such that the selected sensors should be able to guarantee this accuracy. However, this approach deals only with one application and one sensor; an accelerometer deployed on different positions on the body. Another approach is proposed by Noshadi et al. [131] who rely on the selected set of sensors to predict the unseen data of the sensors which were not selected. They consider 19 sensors placed on the arm of the user. In [132], the authors deal with BAN where several nodes can be deployed on the body of the user. They aim at choosing the best set of nodes such that the system is reconfigured whenever a node drops due to failure. The aforementioned sensor selection approaches assume the existence of numerous similar sensor nodes which is not applicable in our daily lives.

Gordon et al. [133] apply their proposed sensor selection algorithm to activity recognition by considering activity-sensor dependency. They leverage the ability to predict the human behavior; therefore, they identify the most probable next activity. Accordingly, they select the subset of sensors which is necessary to distinguish this future activity at little energy cost. Gao et al. [134] propose multi-sensor fusion framework. They propose a sensor selection module that uses convex optimization to select the active sensor nodes while guaranteeing that their classification accuracy is above a defined threshold. In [45], the authors only considered location and used the alternative between GPS and Network-based triangulation. They build a profile for user's location to alternate between these two choices of sensors. In addition, they use the accelerometer to trigger GPS when user moves from static to movement. They consider multiple applications that request location; thus, they do not not consider multiple contexts. These approaches focus on selecting an energy-efficient sensor set only for one context. They do not tackle the challenging issues that arise when multiple applications simultaneously request different contexts.

In [96, 95], Kang et al. propose SeeMon which aims at minimizing the number of sensors triggered and selecting the Essential Sensor Set (ESS) while answering the context-related queries given by applications. It defines one context as the combination of location, activity, and time states; for example, the query can be (location=library, activity=sleeping and time=evening). SeeMon does not consider different possible sensors for the same context. For example, location is always retrieved by taking the longitude and latitude from the GPS. In [97], the authors propose a framework for an Energy Efficient Mobile Sensing System (EEMSS). It uses a minimum set of sensors for each state. It requires that developers fill in information about sensor management rules for each encountered state. They consider 3 combined sensors; GPS, accelerometer and microphone; to know the state of the user. The authors also propose using duty cycles to achieve energy efficiency. However, they do not propose a solution for all contexts where EEMSS is specific for the states that they define. In [135], Roy et al. select the optimized set of external sensors while minimizing the wireless communication energy overhead. They do not consider state-based decision as they assume that all context states are equally likely. Furthermore, combining more than one sensor is essential in some healthcare applications. For example, the authors in [136, 137] use the accelerometers, gyroscopes, force sensors and pressure sensors to detect fall, tremor and dyskinesia.

In [138], the authors propose a sensor and feature selection problem that chooses features based on the power consumed while processing these features to classify context. They use a graph model to remove irrelevant features and choose the optimal set of features when redundant ones exist based on weights they specify. They formulate the problem using integer linear programming (ILP), and they propose a greedy approximation. In [101], the authors propose Orchestrator which considers sensor-based plans generated by developers. These plans represent the sensing and processing modules to detect a context. It decides which sensor to choose given that the same sensor can be deployed in different devices or locations. Furthermore, their cost function to minimize is either related to energy or accuracy; hence, they do not trade-off factors. In these approaches, sensors are selected in a way to resolve contentions among concurrent applications and maximize sharing whereas the proposed sensor selection approach in this thesis aims at selecting the sensor set which gives the best trade-off between energy, accuracy and delay.

2.3 Summary

A major component in context recognition applications is the ability to recognize context accurately while optimizing the use of constrained resources in mobile devices and sensors. Researchers have seen that optimizing resources is becoming feasible with the recent technological developments; thus, researchers tend to provide limited access to local resources by offloading services to cloud servers whenever possible, selectively cache data to save memory space, and design hardware components taking into account the limited processing units and battery life of mobile devices. An alternative option to minimize the energy consumption is to provide sensor-level designs as presented in Section 2.2.2. However, there is still no generic framework that guides the complete sensing designs.

Table 2.3 illustrates how the proposed smart dynamic sensing framework in this dissertation differs from existing prior works. These framework designs are compared in terms of different metrics: whether they propose sensor selection, sensor sampling and sensor scheduling. In addition, we differentiate these proposed prior works based on whether they consider accuracy and delay when optimizing the trade-off with energy. Furthermore, we highlight whether the framework is adaptive such that it changes based on the user's state and whether it is time-variant and dynamic. Our contribution, is to optimize various sensing design alternatives to trade-off the following metrics: energy consumption, recognition accuracy, and detection delay.

Table 2.3: Comparing the dissertation's proposed framework with other prior framework designs

Reference	**Sensor Selection**	**Sensor Sampling**	**Sensor Scheduling**	**Optimize Accuracy**	**Optimize Delay**	**Adaptive Framework**	**Dynamic Framework**
[97]	X			X			
[95]	X			X			
[98]			X		X		X
[99]			X	X			X
[47]		X	X	X		X	
[100]		X	X	X		X	X
Dissertation	X	X	X	X	X	X	X

Chapter 3

Sensor Sampling: Trade-off Energy and Accuracy for Activity Mobile Sensing

Critical applications rely on accurate and continuous detection of user state; and they provide personalized situation-specific services to the user such as real time traffic monitoring and personal assistance [97, 96, 98]. Such applications require continuous sensing. However, continuous sensing mechanisms constitute a major source for energy consumption and impose extra overhead on mobile devices' and sensors' limited resources; hence, the battery depletes rapidly. One solution is to have only the needed sensors turned on with activity-based duty cycles assigned to avoid continuous energy consuming data extraction. This solution builds upon the idea that non-stop continuous data sensing is not essential for context detection since users tend to perform activities (such as sitting, sleeping, or studying) over long time periods and activity recognition can tolerate few misclassifications. Hence, assigning appropriate activity-based sampling frequencies to sensors can help reduce energy consumption while preserving some minimum accepted accuracy level. We will show that the accuracy of detecting some activities does not decrease even when decreasing the sampling frequency from the maximum to the minimum acceptable value.

As explained in Section 2.2.3, most of the referenced literature assigns fixed sampling frequencies which are not adjustable to different user states [97]. In [97], the authors presented a sensor management system: EEMSS. It explores hierarchical sensor management by powering only a minimum set of sensors and assigning fixed duty cycles to recognize user state. However, active sensors are assigned fixed duty cycles which are not activity dependent; thus, they are not adjustable to different user behaviours. Recently, there has been some research work to minimize energy consumption through optimizing user-activity dependent sampling frequency. In Kobe framework [105], the goal is to optimize energy-latency-accuracy trade-off for classification which is not activity specific. In A3R

[117], the authors use activity adaptive sampling rates to detect context and reduce energy consumption. However, A3R determines offline the optimal tuple for each activity; therefore, a new training offline phase would be required when new activities are detected.

In this chapter, we aim at applying appropriate activity-based sampling frequencies by using entropy as the optimization criterion to quantify activity classification accuracy. In addition, we aim at applying a new emerging concept in deep learning: DNN which outputs the appropriate state-based sampling frequencies and shows better accuracy levels compared to other proposed algorithms.

This chapter is organized as follows. Section 3.1 presents the first entropy-based approach we used with its own experiments and results. The second approach is based on deep learning and is described and evaluated in Section 3.2. Finally, a comparison between the two approaches along with the outcomes of the chapter are summarized in Section 3.3.

3.1 Entropy-based Optimization to Trade-off Energy and Accuracy for Activity Mobile Sensing

This section proposes an entropy-based optimization algorithm to trade-off energy and accuracy with mobile sensing for a particular activity. A key aspect of our work is the design and evaluation of optimized energy-accuracy activity-dependant sensing algorithm for recognizing human physical activity. The models for activity recognition are trained using a publicly available dataset collected from seven subjects performing six everyday activities. To support real-world implementation, energy and accuracy models are evaluated from experiments conducted on real smartphones.

The rest of this section is organized as follows. Section 3.1.1 presents the proposed optimization formulation to derive an energy efficient algorithm. Section 3.1.2 presents the results of the experiments.

3.1.1 Proposed Method

This section presents the proposed method for the entropy-based optimization with trade-off between energy and accuracy for activity recognition using smartphone sensors. First, we present mathematical models for user activity and phone sensing. Then, we present the details of the optimization formulation, and present the steps to solve the optimization formulation.

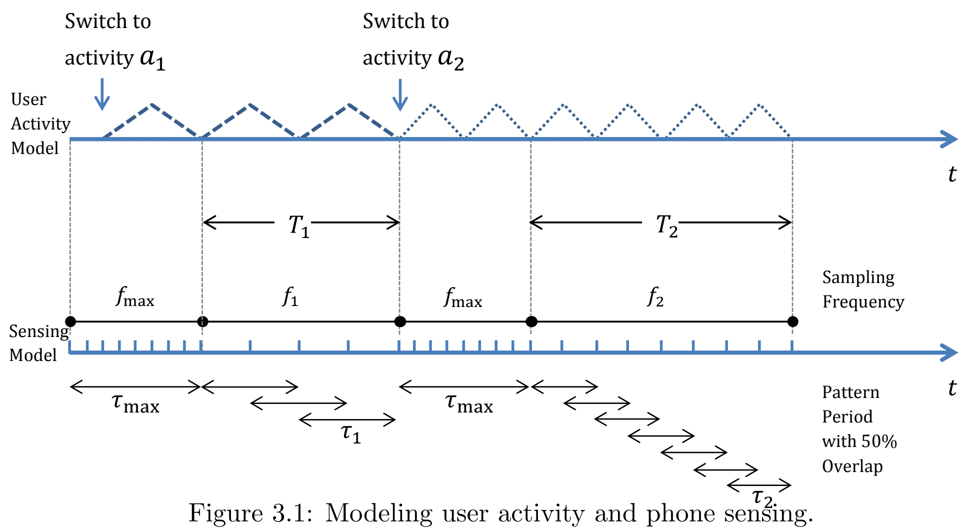


Figure 3.1: Modeling user activity and phone sensing.

Mathematical Models for User Activity and Phone Sensing

Assume we have a phone user who undergoes different activities during a typical day. As an example, as illustrated in Figure 3.1, assume that the person goes for a walk, then transitions into jogging. Let's denote each of the M different types of activities (e.g., walking, jogging, ...) by a_i where $1 \leq i \leq M$; for example, a_1 corresponds to walking state and a_2 to jogging state. Furthermore, let's assume that during walking and jogging, the user goes through some repetitive pattern of activity such as moving the legs. We represent these repetitive motions in the figure by the triangle waveforms. As an example, the waveform can represent the distance between the two feet. As the person walks or jogs, the distances spread, join, then spread again. The period of these repetitive patterns is dependent on the pace of the activity and on the user's motion; e.g., walking takes longer periods to produce steps.

Furthermore, Figure 3.1 shows the phone sensing parameters that are considered for user activity detection.

- f_i represents the minimum sampling frequency for how often the collection of sensor data needs to be initiated for detecting an activity of type a_i.

- τ_i represents the minimum period for how long the sensor data needs to be collected to compute the needed feature for activity a_i. τ_i depends on the type and pace of the activity since it should be directly related to the repetitive patterns found in each activity cycle. During the time window τ_i, the collected raw data is used to generate the feature for activity recognition. For example, the feature can be standard deviation, and a typical strategy is to choose 50% overlapping τ's. Previous work has shown success with 50% overlap in feature extraction [139].

- $f_{\max}$ is the maximum frequency needed to capture any activity. As such, $f_{\max}$ can correspond to the largest f_i, or it can simply be set to the largest sensor frequency available on the phone. For the activity recognition application, we assume that sensor collection is first started at the sampling frequency $f_{\max}$. Once the activity a_i is recognized, the phone sensor is switched to the sampling frequency f_i. At each sample, the application checks for the continuous presence of the activity. When the activity is detected as absent, the application switches to the $f_{\max}$ frequency to make sure it is capable of recognizing any new activity. The process is then repeated, and once the new activity is recognized, the application switches to the proper frequency f_i for the newly detected activity a_i.

- $\tau_{\max}$ denotes the largest period required to detect any repetitive pattern. As such, $\tau_{\max}$ corresponds to the activity with the slowest repetitive pattern.

- T_i is used to represent the time window during which the user is detected practicing activity a_i.

- $N_i = f_i \cdot T_i$ is the expected number of samples if f_i is used during T_i.

- E_{sample} is the energy consumed each time the sensor is triggered for a reading; i.e., it is the energy per one sample.

To illustrate these parameters with walking and jogging activities, $f_{\max}$ would be set to the frequency required for jogging since it needs higher sampling frequency than walking. $\tau_{\max}$ would be equal to the window size for walking activity since walking has a slower repetitive pattern than jogging. In the figure, once the user switches to jogging, and the application detects walking as absent, the system switches to the maximum frequency $f_{\max}$ for a duration of $\tau_{\max}$ to compute a feature and then classify the new activity. In Figure 3.1, $f_2 > f_1$; hence, for equal durations of the two activities, detecting activity of type a_2 leads to more samples and turns ON the sensors more frequently; thus, consuming more energy than that consumed in detecting activity a_1.

For the classification model, let $X_i = \{x_0, x_1\}$ be a Boolean variable representing the presence or absence of the user activity a_i. x_0 indicates absence of the activity and x_1 indicates presence of the activity. Let Y_{f_i} represent the feature that can be collected at frequency f_i for activity a_i. For example if the feature is standard deviation, and τ_i is specified, different sampling frequencies can lead to different feature values. In this proposed approach, we will consider the effect of different sampling frequencies on classification, and we will use standard deviation as the extracted feature; thus, every value in Y_{f_i} represents the standard deviation of data collected during the time window τ_i, i.e., of $\tau_i \cdot f_i$ data tuples.

Table 3.1 presents an example to illustrate the features used in the classification model. In this example, a_1, walking activity, is to be classified and two

Table 3.1: Numerical example for classification model

Y_{50}	Y_{100}	X	Time
0.21	0.18	Walk	τ_1
0.25	0.19	Walk	$2 \cdot \tau_1$
0.19	0.36	Not Walk	$3 \cdot \tau_1$
0.41	0.39	Not Walk	$4 \cdot \tau_1$

feature sets referring to 50 Hz and 100 Hz are under consideration. The table shows X_1 which represents the true class label of the user's activity. Moreover, the table has two columns for the potential sets of features at two different frequencies. τ_1 corresponding to walking activity is used to compute the standard deviations which represent the rows of the features.

Optimization Problem Formulation

Our goal is to achieve activity recognition while achieving highest accuracy and using the least energy from the mobile device. The objective is to derive the best operating conditions for phone sensors and detect the user's activity while trading off the two factors, energy and accuracy. In other words, energy is impacted by the sensor sampling frequency, and accuracy is impacted by both the sensor sampling frequency and the minimum period needed to detect the activity pattern. Energy is reduced with lower usage of sensors, while accuracy is increased when more sensor data is available. As a result energy and accuracy provide conflicting requirements on phone usage. Hence, the objective is to find optimized choices for f_i and τ_i for all M types of activities. Mathematically, the objective function can be formulated as follows:

$$\underset{\substack{f_i,\tau_i \\ i=1,\ldots,M}}{\operatorname{argmin}} \sum_{i=1}^{M} (\lambda_i \cdot E_i + (1-\lambda_i) \cdot P_{e_i}) \tag{3.1}$$

where P_{e_i} represents the expected user state classification error, E_i is the expected energy consumption, and λ_i is a weighting factor where $0 \leq \lambda_i \leq 1$. λ_i is adjusted to achieve the desired energy-accuracy trade-off and guarantee unit compatibility between energy and classification error. Any two consecutive activities are separated by regions of $f_{\max}$ during which the current activity is classified. Thus, the optimization can be applied individually to each activity.

For each activity a_i, the optimization problem becomes:

$$\underset{f_i,\tau_i}{\operatorname{argmin}} \quad \lambda_i \cdot E_i + (1-\lambda_i) \cdot P_{e_i} \tag{3.2}$$

We will show below that (3.14) can be replaced by an alternative entropy-based formulation as follows:

$$\underset{f_i,\tau_i}{\operatorname{argmin}} \quad U(\lambda_i, f_i, \tau_i) = \lambda_i \cdot f_i + (1-\lambda_i) \cdot H\left(X_i | Y_{f_i}\right) \tag{3.3}$$

where U is the utility function to minimize. P_{e_i} has been replaced by the conditional entropy since, as will be shown:

$$P_{e_i} < H\left(X_i|Y_{f_i}\right) \tag{3.4}$$

where X_i represents the user's activity and Y_{f_i} denotes the standard deviation feature collected with a sensor sampling frequency f_i.

We model the energy E_i as

$$E_i = E_{\text{sample}} \cdot N_i \propto f_i \tag{3.5}$$

where E_{sample} is the energy consumed each time the sensor is triggered for a reading and N_i represents the number of samples in a specified time window. E_i can be represented by f_i since E_{sample} is constant for a specific sensor and T_i cannot be known beforehand thus we assume it is constant as well. λ_i can be found using the min-max principle proposed by Gennert and Yuille [140] who suggested that the weights should be selected to maximize the final minimum value of the weighted sum function. The objective function is maximized to balance the trade-off between energy term and classification error term. The min-max principle works as follows: 1) For each value of λ_i, find the values for f_i and τ_i which minimize the objective function $U(\lambda_i, f_i, \tau_i)$ presented in (3.3). 2) Choose the optimal factor λ_i^* which leads to the maximum $U(\lambda_i, f_i^*, \tau_i^*)$.

Accuracy in terms of Entropy and Sampling Choices: The entropy $H(X_i)$ presents the uncertainty in the distribution of the Boolean variable X_i. It is defined as

$$H(X_i) = -\sum_{r=0}^{1} p(x_r)\log_2 p(x_r), \tag{3.6}$$

where $p(x_r)$ denotes the probability that a state in X_i belongs to class x_r. We consider binary classification where the user is either in state a_i or not; hence, the class is either x_1 or x_0 respectively. To reflect the effect of sampling frequency f_i on classification, we generate different sets of feature values for different frequency choices.

Increasing the sampling frequency means collecting more samples and, thus, reducing the possibility of missing relevant samples for feature computation. Hence, each feature corresponds to the conditional entropy $H(X_i|Y_{f_i})$ which depends on sampling frequency f_i. This conditional entropy, $H(X_i|Y_{f_i})$, represents the amount of uncertainty remaining in X_i when knowing the values for feature Y_{f_i}. It is defined as the information needed to classify X_i after using Y_{f_i} with v discretized partitions. The v partitions are found after applying discretization by entropy [141] which discretizes all numeric attributes in the dataset into nominal attributes and, thus, it chooses the best threshold to divide data into different classes. The boundaries of the partitions are chosen as to minimize the entropy.

$$H(X_i|Y_{f_i}) = \sum_{j=1}^{v} \frac{|Y_{f_i} \text{ in partition } j|}{|Y_{f_i}|} \cdot H\left(X_i \text{ in partition } j\right) \tag{3.7}$$

where $|Y_{f_i} \text{ in partition } j|$ represents the count of tuples in Y_{f_i} which belong to partition j and $|Y_{f_i}|$ represents the total count of rows in the feature Y_{f_i}. As an example based on Table 3.1, $H(X_1|Y_{50})$ can be computed as follows. Assume Y_{50} is partitioned into two partitions: below and above 0.26. Then, the first partition (≤ 0.26) has 3 out of 4 samples, with 2 "Walk" and 1 "Not Walk"; whereas, the second partition (> 0.26) has 1 sample with "Not Walk" label. Hence, the entropy in this case is

$$H(X_1|Y_{50}) = \frac{3}{4} \cdot \left(-\frac{2}{3} \log_2 \frac{2}{3} - \frac{1}{3} \log_2 \frac{1}{3} \right) = 0.688 \tag{3.8}$$

The relation between entropy and classification error has been shown by Kovalevskij's upper bound [142] as

$$H(X_i|Y_{f_i}) \geq \log_2 k - k\,(k-1) \left(\log_2 \frac{k-1}{k} \right) \left(P_{e_i} - \frac{k-1}{k} \right), \tag{3.9}$$

where P_{e_i} is the error probability and k is the number of possible outcomes of X_i.

For $k = 2$ which represents the binary classification case, the error probability bound becomes

$$P_{e_i} \leq \frac{H\left(X_i|Y_{f_i}\right)}{2} \tag{3.10}$$

Based on (3.10), classification error is bounded by conditional entropy. Hence, lower entropy would lead to lower error and, thus, higher accuracy. In our problem, we need to choose the feature Y_{f_i} with the minimum classification error; hence, we need to minimize the conditional entropy $H(X_i|Y_{f_i})$.

Energy in terms of Sampling Choices: Assume that E_{sample} represents the amount of energy consumed each time the sensor is triggered for a reading. Assume that the time window when the user is practicing activity a_i is estimated as T_i. Given the sampling frequency f_i, the number of samples during the time window T_i can be estimated by $N_i = f_i \cdot T_i$. E_{sample} is fixed for a specific sensor. T_i is considered to be constant since it cannot be predicted beforehand; thus, the only energy variable in the optimization is f_i, which leads the frequency term in (3.3).

Activity-based Algorithm for Choosing Optimal Sensing Parameters

For each activity a_i, there exist activity-specific parameters f_i and τ_i that minimize the energy while maximizing the classification accuracy. Given the formulation presented in (3.3), Algorithm 3.1 presents the pseudocode to find the optimal sensing parameters. First, the training sets are used to compute the different frequency and entropy components in step (1) of the algorithm. Then, training sets are used to find the optimal weighting factor λ_i using the min-max principle. For

Algorithm 3.1 Choosing the optimal activity-based parameters $< f_i, \tau_i >$

```
Input:
- Training data with activity annotations:
    • $X_i$: user's activity
    • $Y_{f_i}$: standard deviation with frequency $f_i$

Do:
1. Compute $H(X_i|Y_{f_i})$ for all possible sampling frequencies $f_i$ and window sizes $\tau_i$.
2. Find optimal $\lambda_i$ which maximizes $U(\lambda_i, f_i^*, \tau_i^*)$.
3. Find $< f_i, \tau_i >$ from step (2) that correspond to optimal $\lambda_i^*$.
4. When new activity is detected, use $f_{\max}$ for a duration of $\tau_{\max}$ to classify current activity.
5. Repeat step 1.
```

each value of λ_i, find the minimum possible utility function $U_{\min}(\lambda_i)$ using the computations from step (1). Then, choose the optimal factor λ_i which leads to the maximum objective function $\max_{\lambda_i} U_{\min}(\lambda_i)$. In the next step, the optimal sensing parameters f_i and τ_i are found by solving (3.3) for the detected activity. Next, when the binary classification decides that the activity has changed, $f_{\max}$ is used for a duration of $\tau_{\max}$ to classify the current activity. Then, (3.3) is solved again for finding the new activity optimal parameters.

3.1.2 Experiments and Results

Our proposed method was evaluated on a dataset from the Human Activity Sensing Consortium (HASC) [143]. This data was collected from seven subjects holding different accelerometers at 100 Hz sampling frequency. The subjects performed six activities: sit, walk, jog, skip, climb up-stairs, and climb down-stairs. We choose standard deviation to be the feature extracted from the raw accelerometer data with 50% overlapping between consecutive windows consistent with success in related previous work [139]. Several experiments are conducted to validate the proposed model, and evaluate the entropy-based optimizations. This section first gives insight into the energy model, then it provides a validation of the expected relation between entropy and classification error. Finally, we present results of applying the optimization formulation to determine optimal operating conditions for the smartphone.

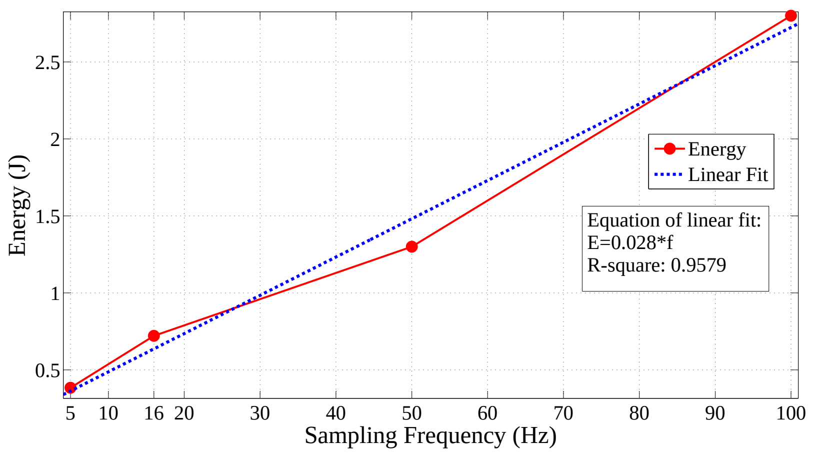


Figure 3.2: Energy consumption versus sampling frequency.

Energy Profile

HASC dataset does not state the amount of energy consumed while collecting data from subjects. Therefore, to capture the effect of sampling frequency on energy, we collected energy measurements for different possible sampling frequencies. Figure 3.2 shows a plot for the energy consumption of the accelerometer turned on for 30 seconds for different sampling frequencies. These measurements were done using Motorola Defy phone which runs Android 2.2 OS and supports sampling frequencies up to 100 Hz. However, this Android API allows four discrete levels of sampling frequencies within the JAVA code which are: Fastest (100Hz), Game (50Hz), UI (16Hz), and Normal (5Hz). To obtain energy values, PowerTutor [144] was used to measure the power consumption by the device [145]. Figure 3.2 shows the resulting measurements, and indicates, as expected from the mathematical models, that sampling frequency f and energy consumption E are directly proportional. Moreover, the figure shows the linear energy model that we assumed. Using the values in the figure, the average E_{sample} is estimated by dividing the energy consumed in 30 seconds as measured by PowerTutor by the number of samples which is $N = 30 \cdot f$. Hence, E_{sample} can be estimated as $\frac{0.028}{30} = 0.93$ mJ.

Classification Error and Entropy Characteristics

The experiments in this section evaluate the expected dependency between classification accuracy and sampling frequency, and window size. In this experiment, we choose standard deviation to be the feature because it has been shown that there is a relationship between the rate of movement in any physical activity and the standard deviation of data in the suitable time window [110]. We considered the different activities available in the HASC dataset. The measurements in this data were originally captured at 100 Hz sampling frequency; thus, to get samples

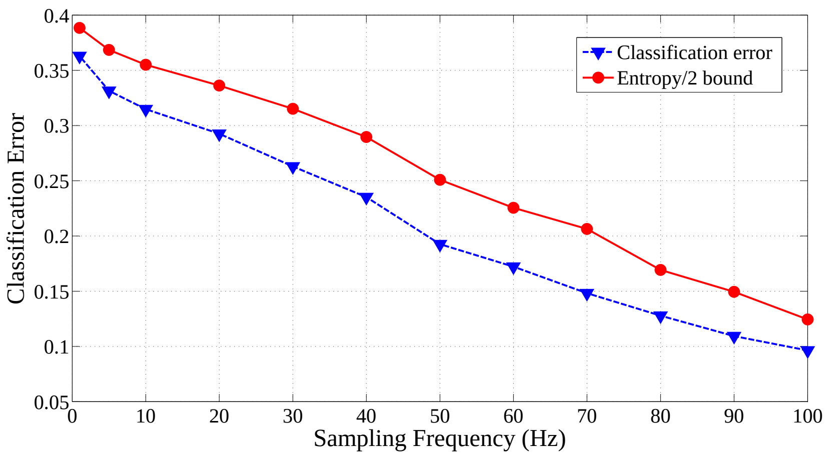


Figure 3.3: Classification error and its entropy upper bound (3.10) for different sampling frequency levels.

at lower frequencies, the data was down-sampled as needed to reflect the desired frequencies. The values obtained from the feature extraction step for different activities are classified using an information-gain based decision tree provided by RapidMiner, and the classification accuracy is estimated by 10-fold validation.

Dependency of Entropy on Sampling Frequency Figure 3.3 shows the average classification error and half the average conditional entropy for all combinations of activities and subjects. We can observe that higher sampling frequency leads to lower classification error and lower conditional entropy. Figure 3.3 indicates, as presented in (3.10), that classification error is upper bounded by conditional entropy.

To further understand which activity requires a higher sampling frequency, we studied the accuracy of classification for each individual activity. Figure 3.4 shows the effect of varying the sampling frequency on the accuracy for each of the activities: sitting, walking, and jogging. It can be observed that classification accuracies of walking and jogging are affected by varying the sampling frequency; i.e., as frequency decreases, classification accuracies sharply decrease. Decreasing frequency for sitting case also decreases classification accuracy but in a lower slope. Hence, more complex activities, such as walking compared to sitting, need more samples to accurately detect the change in activity since more complex activities involve more movement; this demonstrates that classification accuracy is activity-dependent.

Dependency of Entropy on Window Size

The effect of τ can be viewed when comparing the classification accuracies of activities. For example, Table 3.2 shows the average classification accuracies for all subjects for walking and jogging activities at fixed sampling frequency of 50 Hz. For walking activity, the table shows higher classification accuracy for $\tau = 5$ sec than other values of τ; whereas, for jogging, it is more accurate in case of

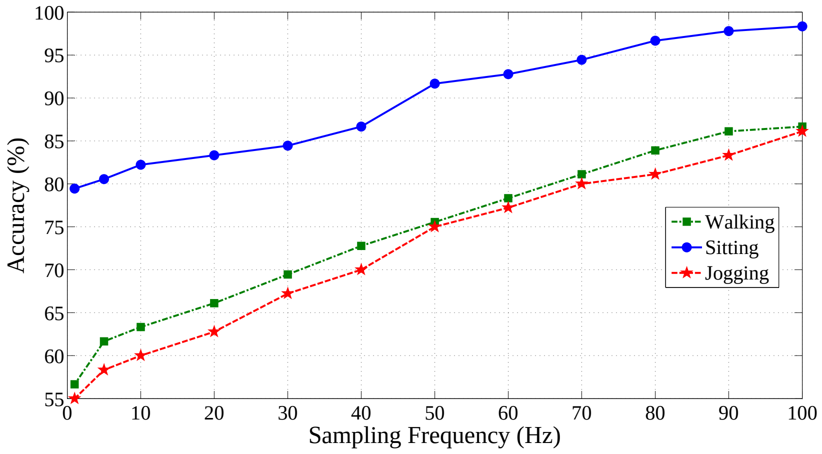


Figure 3.4: Average classification accuracy for different activities versus sampling frequency.

smaller τ (1 or 2 sec). Therefore, jogging needs smaller window size than that of walking because jogging has a faster pace in general.

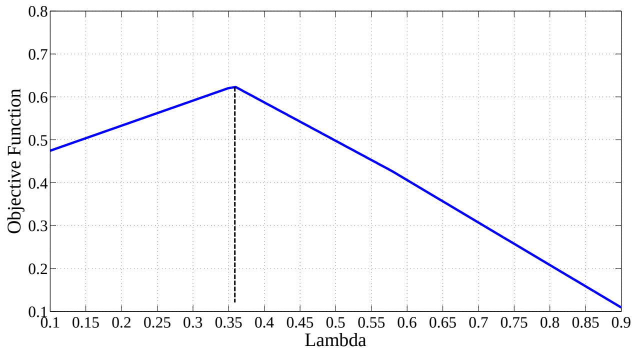
Performance Analysis of Proposed Algorithm

From the optimization formulation in (3.3), and as described in Algorithm 3.1, the first parameter to optimize is the weighting factor λ_i. The experiment considers the three activities: sitting, walking, and jogging. Figure 3.5a shows a sample graph that plots the minimum objective function vs. λ for walking activity. Optimal λ is chosen to be the one that maximizes the objective function. The optimal λ for each activity is shown in Table 3.5b.

Next, to determine the best operating conditions, we consider 11 possible sampling frequencies $f = \{5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$ Hz and 3 possible window sizes $\tau = \{1, 2, 5\}$ seconds. Solving (3.3) for each of the three activities, we obtain the following optimal sensing parameters presented in Table 3.3. The optimal frequency is, as expected, proportional to the complexity of the activity. For example, $f_{sitting}$ is smaller that $f_{walking}$ because sitting does not involve a lot of movement compared to walking. Moreover, the optimal τ is also activity dependant; for example, since jogging has a faster pace than walking, $\tau_{jogging} < \tau_{walking}$. As for sitting, any of the possible τ's could work but we choose the biggest τ to decrease computational energy.

Table 3.2: Classification accuracy for different values of τ

τ	Walk	Jog
1 sec	68.97	67.77
2 sec	70.69	67.77
5 sec	75.33	63.65



(a)

Activity	λ_i
Sitting	0.506
Walking	0.36
Jogging	0.344

(b)

Figure 3.5: Finding optimal weighting factor λ.

Table 3.3: Optimal sensing parameters for each activity

Activity	f_i	τ_i
Sitting	10 Hz	5 sec
Walking	70 Hz	5 sec
Jogging	90 Hz	1 sec

Energy Consumption Gains

To demonstrate the energy savings of our proposed method, we assume that the activity detection application is running for 90 minutes. We also assume uniform distribution of activity, i.e., the user is sitting for 30 minutes, walking for another 30 minutes, and jogging the last 30 minutes. From Figure 3.2 we can estimate the average E_{sample} as 0.93 mJ. Sampling the accelerometer continuously (100 Hz) for 90 minutes = 5400 seconds for the three activities leads to an energy consumption of almost $5400 \text{ sec} \cdot 100 \text{ Hz} \cdot 0.93 \text{ mJ} = 502.2$ Joules. Based on the proposed algorithm, the sampling frequency is chosen to be 10 Hz when sitting is detected, 70 Hz when walking is detected, and 90 Hz when jogging is detected as shown. Running our approach consumes $1800 \text{ sec} \cdot 10 \text{ Hz} \cdot 0.93 \text{ mJ} = 16.74$ Joules to detect sitting, $1800 \text{ sec} \cdot 70 \text{ Hz} \cdot 0.93 \text{ mJ} = 117.18$ Joules to detect walking, and $1800 \text{ sec} \cdot 90 \text{ Hz} \cdot 0.93 \text{ mJ} = 150.66$ Joules to detect jogging. As a result, the proposed algorithm leads to an energy consumption of 284.58 Joules, and an overall 43% reduction in energy consumption.

3.1.3 Summary

In this approach, we have presented an optimization formulation based on entropy to improve the energy efficiency of sensor data collection on smartphones. The method includes a mathematical model for user activity and phone sensor parameters. The algorithm chooses the optimal sensor sampling frequency and the window size for feature derivation. These two parameters are activity-dependent.

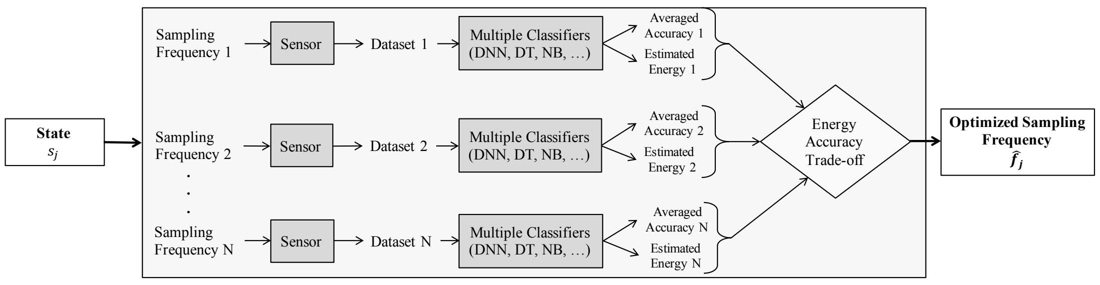


Figure 3.6: Proposed deep learning with ensemble classification for optimized sampling decision.

We experimentally validated the proposed mathematical models, and showed a 43% reduction in energy with the proposed method. Although this work focuses on activity recognition, it can be generalized to any other context detection.

3.2 Deep Learning with Ensemble Classification Method for Sensor Sampling Decisions

In this approach, we exploit the advantages of deep learning. Deep learning has evolved through recent years resulting in high classification accuracy; thus, it can guarantee more efficient sensing from embedded and wearable sensors. Furthermore, deep learning does not need to transform the raw sensor data to intermediate features, and thus eliminates the step of feature extraction required by other conventional machine learning approaches. We use Deep Neural Network (DNN) with ensemble classification of other complementary machine learning approaches (such as Decision Tree (DT) and Naïve Bayes (NB)) to determine the best sensor sampling frequency for context recognition.

A key aspect of our work is the design and evaluation of optimized state-dependant sensing algorithm that trades-off energy and accuracy to recognize user contextual information such as: activity, health condition, and location detection. Moreover, we exploit the advantages of DNN inside an ensemble of classifiers to determine the best state-based sampling frequency. The advantage of this method is that it considers different levels of the data features through DNN. In addition, the effect of this variation is then averaged through the ensemble decision making so that the results are robust to data variation in real applications. To validate our models and algorithm, we used a publicly available dataset collected from seven subjects performing six everyday activities. The rest of this section is organized as follows. Section 3.2.1 presents the proposed architecture and algorithm. Section 3.2.2 presents the experiments and results.

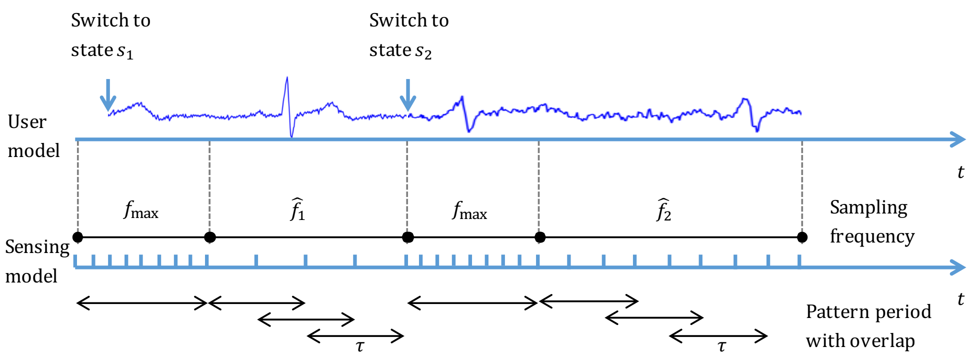


Figure 3.7: Modeling user state and sensing.

3.2.1 Proposed Method

This section presents the proposed method for the energy-accuracy optimization for context detection. The problem can be described as illustrated in Figure 3.6 for each monitored state s_j. The state can be the health condition, the physical activity or the location of the user. In the case of cardiac health monitoring, the electrical activity of ECG wearable sensors are deployed to collect the needed data. We denote each of the J states (e.g., healthy heart, cardiovascular attack, ...) by s_j where $1 \leq j \leq J$; for example, s_1 corresponds to healthy heart and s_2 to risk of heart attack. For each state, different datasets are collected using N sampling frequencies. Each dataset is classified using an ensemble set of classifiers including Deep Neural Network (DNN), Decision Tree (DT), and Naïve Bayesian (NB). The last step of the proposed approach is to input the averaged accuracies from the ensemble classifiers for different sampling frequencies into an optimization algorithm that trades-off energy and accuracy to find the optimized sampling frequency $\hat{f}_j$.

We next present our proposed mathematical model. DNN which is a new machine learning approach has two steps: 1) the unsupervised generative training step which generalizes the model and 2) the discriminate fine-tune step which increases the classification accuracy. We describe those steps briefly before we present the steps to optimize the energy-accuracy trade-off.

Sensing Model

To illustrate the different parameters in our model, consider the example, where we have a phone user with a wearable ECG that monitors her heart activities to detect any critical state. As illustrated in Figure 3.7, assume that the person's first state s_1 is healthy, then it transits into state s_2 which has more fluctuations; thus, it signifies the possibility of having a cardiovascular attack.

Figure 3.7 illustrates the sensing parameters that are considered for user context recognition.

- $\hat{f}_j$ represents the minimum frequency needed for accurate detection of state

s_j. Thus, $\hat{f}_j$ is the output of our proposed method as illustrated in Figure 3.6.

- τ represents the period for how long the sensor data needs to be collected to compute the needed feature for state s_j. During the time window τ, the collected raw data is used to generate one feature.

- $f_{\max}$ corresponds to the largest sensor frequency available. We assume that sensor collection begins with $f_{\max}$. Once the state s_1 is recognized, the sensor is switched to the optimized state-based sampling frequency $\hat{f}_1$. When s_1 is detected as absent, the application switches to the $f_{\max}$ frequency to make sure it is capable of recognizing any new state with high accuracy.

Ensemble Classification with DNN

We propose using an ensemble set of classifiers with averaged accuracy to reduce risk of over-fitting and biasing from a single classifier [146]. Three classifiers are proposed as part of the ensemble decision: DT, NB, and DNN. A decision tree (DT) recursively builds a classification model as a tree structure. The basic idea is to use training data to employ a top-down approach from a root node while partitioning the data into smaller subsets. At each level, the algorithm seeks an attribute that best separates the data at this node until reaching a leaf node. We used the C4.5 DT algorithm which employs the information gain as its criterion [147]. Naïve Bayes (NB) is a probabilistic classifier which is based on Baye's theorem [148]. The algorithm generates a classification model by calculating the posterior probability of each possible class given the attributes while "naïvely" assuming independence between attributes. The outcome is the class with the highest posterior probability.

DNN is a generative-discriminative approach that was proposed in [149]. DNNs can learn high-level invariant features from raw data [150, 151], and this is what makes these networks helpful for context recognition. DNNs are composed of input layer, hidden layers, and output layer. The input layer represents the set of features to be modeled. The hidden layers are trained hierarchically one layer at a time where the output of one layer acts as an input to the next layer. The last step in DNNs is to fine-tune the parameters.

Let x be the input vector which in our case is equal to the sensor feature: ECG features for heart monitoring and accelerometer feature for physical activity detection. A DNN is composed of L layers where the first layer represents the input layer whereas the last layer represents the output layer. These L layers transform the input vector x into a probability distribution y to estimate the output class which is the corresponding activity of the user in our case. Let W be the matrix of weights which controls the relative importance of elements of a layer and the subsequent one. As for the layers in between, they are known as hidden layers. In addition, a bias term exists on each layer except the output layer and

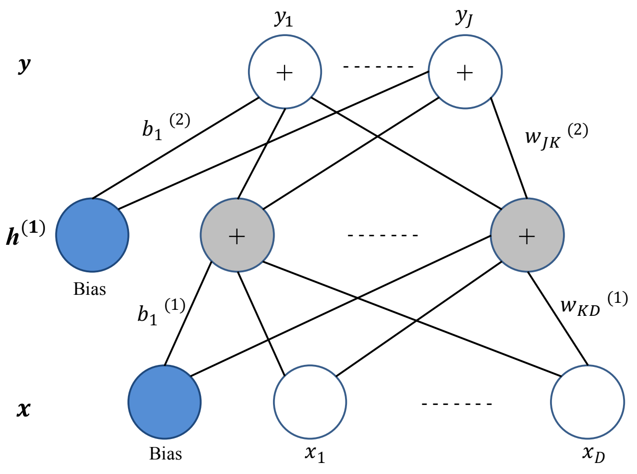


Figure 3.8: Restricted Boltzmann Machine with one hidden layer.

a bias weight vector b exists between this bias term and all nodes in subsequent layer. The DNN is composed of multiple Restricted Boltzmann Machines (RBMs) [152], which represent Hidden Markov chains. Figure 3.8 shows an RBM example with one hidden layer. It has input vector of size D, one hidden layer of size K and an output layer of size J which is the number of possible classes (states). When more than one hidden layer exists, several RBMs are stacked starting with the one whose input is the raw sensor data where the output of the first layer RBM is an input for other subsequent RBMs.

Each layer uses an activation function $g(a)$ that transfers the input data to the output data of that layer. In our DNN, we used hyperbolic tangent (tanh) as an activation function. The output of this function is in the range (-1,1):

$$g(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}} \tag{3.11}$$

For example, we compute the vector $h^{(1)}(x)$ of the first layer as:

$$h^{(1)}(x) = g(W^{(1)T}x + b^{(1)}) \tag{3.12}$$

where $W^{(1)}$ and $b^{(1)}$ are the weight matrix and the bias vector respectively for the first hidden layer. As for subsequent hidden layers, each layer l computes its vector $h^{(l)}(x)$ using previous layer vector $h^{(l-1)}(x)$.

To find the suitable values of W matrices and b vectors, we use training dataset. There are several optimization methods to train data and minimize the classification error. We apply conjugate gradient (CG) method. Although CG is more sophisticated than predominant methodologies (such as stochastic gradient descent (SGD)), it significantly simplifies and speeds up the process of pre-training deep algorithms. We chose CG since it does not use the computationally heavy hessian [153]. The final layer of the DNN uses softmax nonlinearity to output a properly formed probability distribution y over the J possible classes (states). For each state s_j, the probability distribution y_j is defined as:

$$p(s_j|x) = y_j = \frac{exp(W_j^{(L)T} h^{(L-1)} + b_j^{(L)})}{\sum_{k=1}^{J} exp(W_k^{(L)T} h^{(L-1)} + b_k^{(L)})} \tag{3.13}$$

Optimization for Sampling Frequency Selection

Our goal is to detect context with highest level of accuracy while consuming lowest energy from mobile device and wearable sensors and thus extend the usable battery life while minimizing the frequency of charging the devices. The objective is to derive the best sampling frequency for sensors and detect the user's state while trading off the two factors, energy and accuracy. Energy is reduced with lower usage of sensors, while accuracy is increased when more sensor data is available. There are N different sampling frequencies f_n where $n = 1, ..., N$. Hence, for each state s_j, the objective is to find optimized sampling frequency $\hat{f}_j$ out of those N sampling frequencies that minimizes the following optimization problem:

$$\underset{f_n;n=1,...,N}{\text{argmin}} \quad \lambda_j \cdot \frac{E_n(f_n)}{E_{\max}} + (1-\lambda_j) \cdot (1 - \frac{A_n(f_n)}{100}) \tag{3.14}$$

where $A_n(f_n)$ represents the ensemble averaged classification accuracy for each sampling frequency f_n and $E_n(f_n)$ is the sensing energy consumption. We model the energy E_n as f_n since we assume that sensing energy is proportional to sampling frequency, and we assume that computational energy is equal for all choices of sampling frequencies because a feature is computed every τ. We divide $E_n(f_n)$ by $E_{\max}$ to normalize where $E_{\max}$ is the maximum energy consumed when sampling at the highest possible sampling frequency $f_{\max}$ of the sensor. In addition, we divide $A_n(f_n)$ by 100 to normalize such that the energy and accuracy factors have same units. λ_j is a weighting factor that can be tuned from a development data set based on the error tolerance of the application. For example, if the application is critical such as health application, more weight should be given to accuracy thus tuning will result in λ_j having relatively low value.

State-based Algorithm

For each state s_j, there exists state-specific optimized sampling frequency $\hat{f}_j$ that minimizes the energy while maximizing the accuracy. Algorithm 3.2 presents the steps followed for determining the optimized sampling frequency. First, for each state s_j, N sets of data are generated each using different sampling frequency f_n where $n = 1, ..., N$. Then, each dataset is input into the ensemble set of classifiers after which an averaged classification accuracy is computed and consumed energy is estimated. In the next step, the optimized sampling frequency $\hat{f}_j$ is found by solving (3.14) for the detected state. This algorithm is repeated offline for each

Algorithm 3.2 Choosing optimal sampling frequency $\hat{f}_j$ for state s_j

Input:
- Possible sampling frequencies f_n where $n = 1, ..., N$
- λ_j: weighting factor

Output:
The optimized sampling frequency $\hat{f}_j$

```
1: for each $n = 1$ to $N$ do
2:     Generate a set of data collected using sampling frequency $f_n$.
3:     Input this dataset into the classifier ensemble to classify data.
4:     Obtain the classification averaged accuracy $A_n(f_n)$ and the estimated energy $E_n(f_n)$.
5: end for
6: Find $\hat{f}_j$ by solving the optimization formulation (3.14).
```

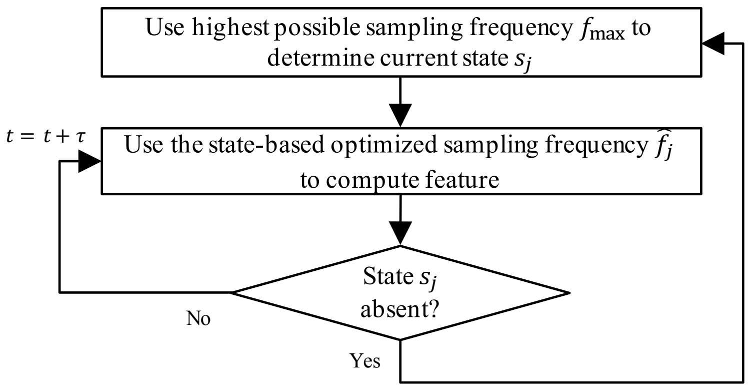

Use highest possible sampling frequency $f_{\max}$ to determine current state s_j
$t = t + \tau$
Use the state-based optimized sampling frequency $\hat{f}_j$ to compute feature
State s_j absent?
No
Yes

Figure 3.9: The sampling frequency resets to $f_{\max}$ with state transitions.

state s_j where $1 \leq j \leq J$. All derived optimized frequencies $\hat{f}_j$s are stored in a look up table that can be used during online processing.

To apply our approach in real time, we use the previously identified sampling frequencies for each state. As soon as the classifier recognizes that the current state is no longer detected, the sampling frequency is switched to the highest possible $f_{\max}$ frequency to determine the new state, and then adjust to a new optimized sampling frequency as shown in Figure 3.9. This optimized sampling frequency $\hat{f}_j$ detects accurately whether the user is still in the same state s_j or has transited into any other state $\bar{s_j}$. Therefore, during transition, the detection is a binary classification.

Computational Complexity

Our algorithm consists of three working phases: 1) training datasets collection, 2) ensemble classifier learning to obtain the averaged accuracies and estimated energies, and 3) choosing the best frequency based on the optimization (3.14). These steps are typically done offline to derive a Look Up Table (LUT) of sensing frequencies for different states. After the offline learning is done, the optimized sampling frequency $\hat{f}_j$ is only fed into the sensor. The device uses this sampling frequency and performs the online activity recognition which is lightweight. The online processing complexity of the recognition algorithms: DNN, DT, and NB are all linear, which provides low overhead.

3.2.2 Experiments and Results

To test our proposed method, we use Human Activity Recognition (HAR) as an application. We used a publicly available dataset Human Activity Sensing Consortium HASC [143] which is collected from seven subjects each holding an accelerometer sampled at 100 Hz which is the maximum possible sampling frequency $f_{\max}$. The subjects performed six activities which include: sit, walk, jog, skip, climb up-stairs, and climb down-stairs. As for the classification feature, we chose the standard deviation computed every 2 sec to calculate the variation in the accelerometer axes data (x, y, and z). This standard deviation with 2 sec sliding window and 50% overlap was proved to be optimal in [31].

To obtain datasets for different sampling frequencies, the data is down-sampled from 100 Hz to obtain 95 sampling frequencies ranging from 5 Hz to 100 Hz; thus, $N = 95$. For Decision Tree and Naïve Bayesian, classification accuracy is estimated using 10-fold cross validation provided by Weka. For Deep Neural Network, MATLAB is used to obtain the classification accuracy. We used a batch size of 2 with 50 epochs where an epoch is the number of rounds carried out over the training dataset. As for the activation functions, we used the hyperbolic tangent function (tanh).

Several experiments are conducted to validate the proposed model and algorithm. First, the experiment gives insight into the importance of Deep Neural Networks compared to other machine learning algorithms. Then, we show the relation between classification accuracy and sampling frequency for each activity. This section also presents the resulting optimized sampling frequencies after applying our proposed approach. Finally, we present an Android-based implementation to show the energy savings in real systems.

Comparison Analysis of Machine Learning Algorithms

We varied the sampling frequency between 5 and 100 Hz for each of the sit, walk and jog activities. Then, each dataset corresponding to a unique combination

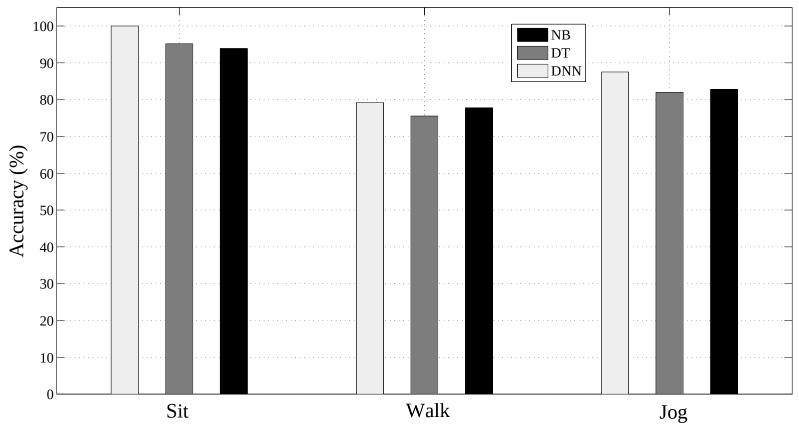


Figure 3.10: Average classification accuracy versus machine learning algorithms for each activity.

of activity and sampling frequency is input into each of the machine learning algorithms (DNN, DT, and NB) one after the other. Thus, for each dataset, we obtain three classification accuracies. To compare the performance of those machine learning algorithms, we compute the average classification accuracy over all sampling frequencies for each activity and each of DNN, DT and NB. Figure 3.10 shows the results illustrated in a bar graph. It can be seen that the overall performance of DNN proves to surpass the performance of DT and NB. This proves the recent claims which state that DNN gives more accuracy due to its pre-training and fine-tuning dual steps. Moreover, the figure shows that DT's performance surpasses NB performance for sit activity, while NB classification accuracy is more for walk and jog cases. Therefore, considering an ensemble of classifiers in our proposed model exploits the advantages of each one of the considered algorithms.

Classification Accuracy versus Sampling Frequency

To validate that sampling frequency depends on the user state, we studied the average accuracy of classification for each activity. In this experiment, we compute the average classification accuracy for each sampling frequency dataset over the three machine learning algorithms as presented in Figure 3.6. Figure 3.11 shows how varying the sampling frequency affects the accuracy of each activity. The figure shows that varying the sampling frequency for sitting activity has little effect on the accuracy since it results in low error even when using low sampling frequencies. Whereas for walking and jogging, the classification accuracy increases as the sampling frequency increases. Therefore, classification accuracy is state-dependant where activities involving more movement (walking or jogging) require more samples compared to steady activities (sitting).

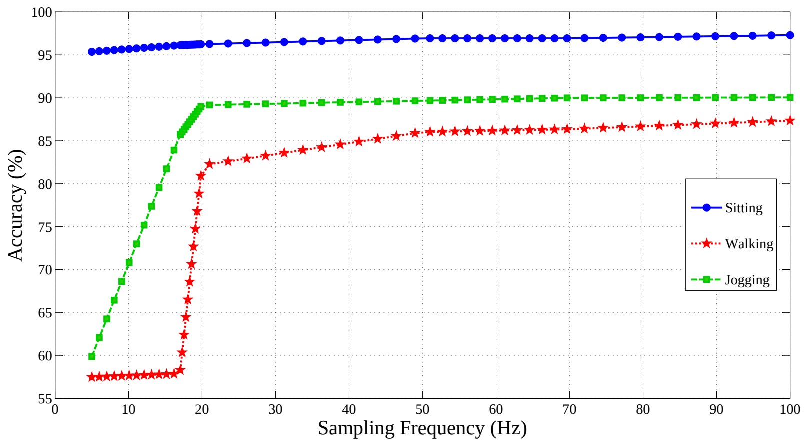


Figure 3.11: Average classification accuracy for different activities versus sampling frequency.

Performance Analysis of Proposed Algorithm

To solve the optimization formulation in (3.14), λ_j depends on the importance of the accuracy with respect to energy consumed. We chose three different λ_j values to show its effect. Solving (3.14) for each λ_j value and each activity, we obtain the optimized sampling frequencies presented in Table 3.4. The first observation is that the optimized sampling frequency for sitting is smaller than other activities since it does not involve complex movements. Another observation is related to the effect of λ_j: for sitting activity it does not vary since the lowest sampling frequency gives the minimum optimization output. As for walking and jogging, decreasing λ_j causes the optimal sampling frequency to increase since it gives more priority to accuracy over energy.

Table 3.4: LUT of optimized sampling frequency for each activity

Activity	$\lambda_j = 0.1$	$\lambda_j = 0.5$	$\lambda_j = 0.9$
Sitting	5 Hz	5 Hz	5 Hz
Walking	50 Hz	20 Hz	17 Hz
Jogging	20 Hz	20 Hz	17 Hz

Energy Gains

Our optimized sensing mechanism is compared with the baseline of continuous sensing which uses 100 Hz all the time for all activities. To analyze the energy consumption of our proposed approach on real systems, we implemented our approach on an Android-based device which is HTC Desire running Android 4.4.2 with a 2100 mAh battery capacity. The conducted experiments involved activity recognition using the embedded accelerometer. We used PowerTutor [144] to measure the energy consumed. We consider the scenario where the user sits for 30 min and walks for 15 min then finally she jogs for another 15 min. Sampling

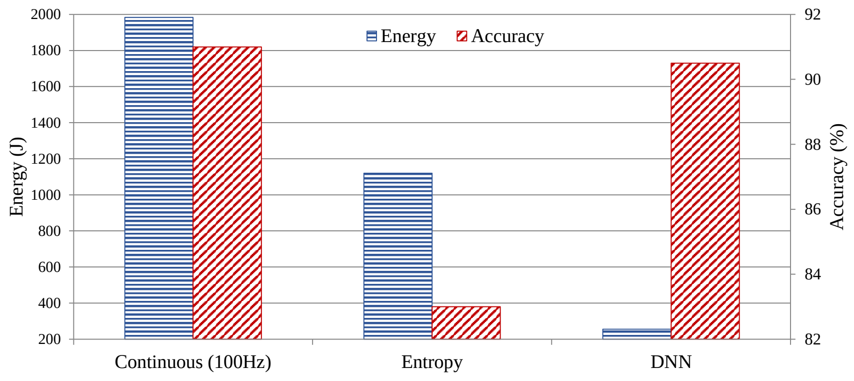


Figure 3.12: Energy and accuracy for our proposed sensor sampling approaches and continuous sensing.

the accelerometer continuously leads to 100 Hz average sampling frequency which consumes 1983.6 J during the one hour duration. Based on our proposed algorithm with λ_j set to 0.5 for all activities, sitting requires 5 Hz while walking and jogging are captured at 20 Hz. Using these optimized sampling frequencies, the application consumed 255.6 J. Therefore, our method saves an overall of 87% of energy consumption.

3.2.3 Summary

In this approach, we exploit the advantages of Deep Neural Network (DNN) with ensemble classification of other complementary machine learning approaches to provide a robust optimized choice of sensor sampling frequency for context recognition. The selection problem is formulated as an optimization problem with trade-off between energy and accuracy. The experiments show that the optimized sensor sampling frequency is state-dependent. Furthermore, the experiments show an 87% energy reduction compared to continuous sensing.

3.3 Comparison and Summary

To compare our two sensor sampling approaches, we implemented both approaches on an Android-based HTC Desire running Android 4.4.2. We used the accelerometer sensor with different sampling frequencies based on the optimized frequencies obtained from the approaches. We used PowerTutor software to obtain an estimation of the consumed energy. As for the scenario, we regenerated the case where the user sits for 30 min then walks for 15 min and finally jogs for 15 min; hence, we recognized the activity of the user for an hour. Figure 3.12 shows the energy and accuracy for our two proposed approaches in addition to continuous sensing which uses 100 Hz as its sampling frequency for all activities. The figure shows that our deep learning approach which uses DNN gives the lowest energy consumption with comparable accuracy level to the continuous sampling. On the other hand, the entropy-based approach saves energy when compared to the continuous case; however, it decreases the accuracy levels from 91% to 83%. Therefore, using an ensemble of DNN with feature-based classifiers allows using low sampling frequencies while providing accurate classification, thus lowering consumed energy demonstrating improved battery life.

Chapter 4

Sensor Scheduling: Viterbi-based Context Aware Mobile Sensing to Trade-off Energy and Delay

A recent smartphone has computer capabilities with a wide range of applications being developed in several domains. On Google Play [154] alone, there are over 1.3 million Android applications, capturing several domains in context-aware computing including healthcare, navigation and personal monitoring. Such proliferation and popularity have given rise to context-aware computing as a branch of mobile computing in which applications detect and exploit contextual information such as locations, health conditions, and activities [102, 155]. Several applications require near real-time response for detecting context changes especially for medical applications. For example, fall detection of elderly people requires the fastest possible detection of any change in the body's posture to avoid risks; hence, it relies on continuous context recognition [156]. However, sensors' continuous usage constitutes a major source for energy consumption that imposes heavy workloads on smartphones. Fortunately, most applications do not need to detect the states of the context continuously, rather they require the detection of critical changes in context. For example, the purpose behind monitoring signals from an electrocardiogram (ECG) sensor of a patient with heart disease is to detect risky heart activity and alert in case of emergency [157]. In such cases, continuous sensing can be substituted by efficient dynamic mobile sensing strategies that allow the smart mobile device to intelligently interact with external sensors and embedded sensors while trading off resources' energy consumption and application delay targets. Therefore, a system is required to select a sensing schedule that optimizes when sensors need to be triggered.

There have been some approaches to solve this problem; however, those approaches focus on energy and accuracy within one state of the context rather than the transitions between states. Recently, stochastic principles have been introduced to assign duty cycles to sensors. Markov Decision Process (MDP) frameworks have been used to schedule sensors based on the current state of the user [98, 123, 125]. However, Markov models assume predetermined systems which know in advance the true user's state which might turn out to be a wrong assumption. In addition, user state transitions in real data traces are not strictly Markovian [98]. These methods focus on one state; on the other hand, our approach aims at detecting the contextual change from one state to another. Those approaches are application-specific and they assume the knowledge of the true context state of the user.

This chapter proposes Viterbi-based Context Aware Mobile Sensing (VCAMS) mechanism to optimize the trade-off between delay in sensing context change and energy consumption via deciding when to trigger the sensors for data collection based on the user's behavior. Typically, Viterbi algorithm [158] is used to find the most probable path of hidden states in Hidden Markov Models used for modeling sequences. In [159], the authors use the Viterbi decoder to extract the sequence of user's locations which are hidden from sequence of observations derived from the signal measurements. In our proposed approach, we define customized reward functions in terms of optimization criteria which depend on the sensing action and the current state. The goal is then to find the optimal sensing schedule that maximizes the cumulative rewards.

VCAMS is context aware system that depends on the situational information about the user and her surrounding. It can be used for applications that anticipate real-time contexts such as location, activity, and health conditions. Each context can have several states; for example, the location of the user might be at home, at work, or in mall. For a given user in a specific state, the objective of the system is to dynamically provide the time instants at which a sensor needs to be triggered, also called the sensing schedule for the specific user and the particular state. The system has two modes: learning mode and execution mode. In the learning mode, the sensing schedules are derived using a Vietrbi algorithm based on historical data of the user model that captures how much time the user spends in particular states. Viterbi is chosen for its low computational complexity. The user model is continuously updated when the system recognizes actual changes of state. The learning mode is executed once to initialize the system, and on occasions while the system runs and after a user's behavior changes significantly from its initial conditions. In the execution mode, the already learned sensing schedules are used to decide on sensing triggers for a particular state.

The main contributions of this chapter include: 1) A formulation for Viterbi implementation with the definition of new customized rewards to learn sensing schedules for real-time decisions on when sensing should be triggered. The energy and delay reward metrics are formulated to best represent the utility associated with each transition between trellis nodes in a Viterbi-based algorithm. The formulation includes a unified model that captures both the user's and the phone's states; and 2) A strategy that triggers learning mode in real-time to update the sensing schedule only when critical changes are captured, thus avoiding unnec-

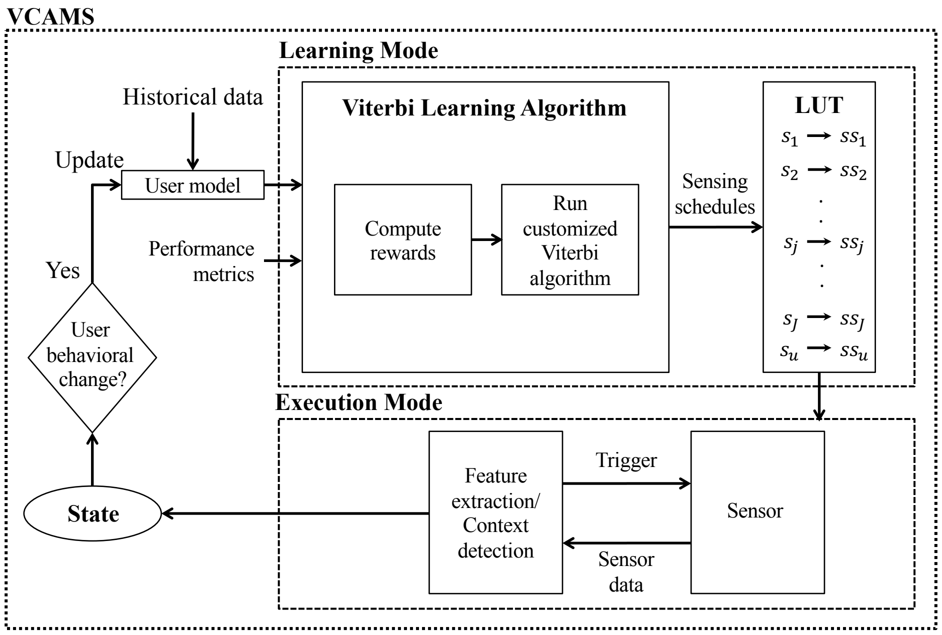


Figure 4.1: The general work flow of the proposed method.

essary computations. To evaluate the performance of VCAMS, we conducted simulation experiments on one state derived using a context simulator. To assess the computational complexity under realistic operational conditions, we implemented VCAMS on an Android-based smartphone. In addition, we investigated a case study using real dataset with multi-state changes to demonstrate the effectiveness of the proposed approach.

This chapter is organized as follows. Section 4.1 describes the overall proposed system model. Section 4.2 presents the details of the proposed Viterbi method and studies its computational complexity. Sections 4.3 and 4.4 present the results obtained by simulations and the case study respectively. Concluding remarks follow in Section 4.5.

4.1 System model and Components

The general work flow of the proposed method is shown in Figure 4.1 for each required context; and the parameters used in VCAMS are listed in Table 4.1. There are two modes in the solution. The learning mode aims at determining the sensing schedule to be used in association with a given user and a given context. This mode is shown in the upper part of Figure 4.1. The inputs to the learning mode of the system are: 1) the user model which captures a transition model from historical behavior of the user such as the probabilities of transition $p_j(t_i)$ from state s_j to another at any time instant t_i, and can also include the statistics of the time duration T_j spent in a specific state s_j and 2) the performance metrics:

Table 4.1: Table of parameters

Groups	**Parameter**	**Description**
State-based parameters	s_j	Particular state out of J total states
	T_j	Time duration spent in s_j
	$\hat{T}_j$	Time limit of s_j
	δ_j	Sampling interval between time instants
	N_j	Number of time instants
	p_j	Probability of state switch
	h_j	Probability's decaying rate
	μ_j	Mean time spent in state s_j
	σ_j	Standard deviation of spent in state s_j
	ss_j	Output sensing schedule for state s_j
Time-based parameters	t_i	Time instant
	a_i	Decision action: 1 for "sense" and 0 for "not sense"
	Δt_i	Time instants accumulated since last sense
Weighting parameters	ω	Pareto weighting factor
	α	Energy-delay weighting factor
	β	Energy-delay weighting factor
Output parameters	D	Output delay
	E	Output energy

energy and delay which are used to define rewards in the customized Viterbi algorithm. The learning part of the system generates a user-specific lookup table (LUT) capturing which sensing schedule should be used for each state. The second part of the solution reflects the execution mode shown in the lower part of Figure 4.1. The execution mode of the system includes integration with a context recognition application. It takes the LUT as input, and chooses the corresponding sensing schedule for the previously detected user's state.

The proposed system starts by developing the initial user model based on the user's historical behavior. The VCAMS learning process requires two steps shown in the "Viterbi Learning Algorithm" box in the figure. The first step is to compute the expected customized rewards to estimate the gain for every combination of state and sensing decision at each time instant. This step is described more in Algorithm 4.1 in Subsection 4.2.3. The second step is to run the customized Viterbi algorithm to maximize those rewards and derive the optimized sensing schedules, and this step is presented in Algorithm 4.2 in Subsection 4.2.3. The learning of the sensing schedule is done offline, and the derived sensing schedules are saved in a LUT where each state s_j has its specific sensing schedule ss_j where $1 \leq j \leq J$. In the execution mode, the system applies these schedules in real-time by inputting the LUT to the online context recognition application which triggers the specific sensor to extract features and recognize the current state of

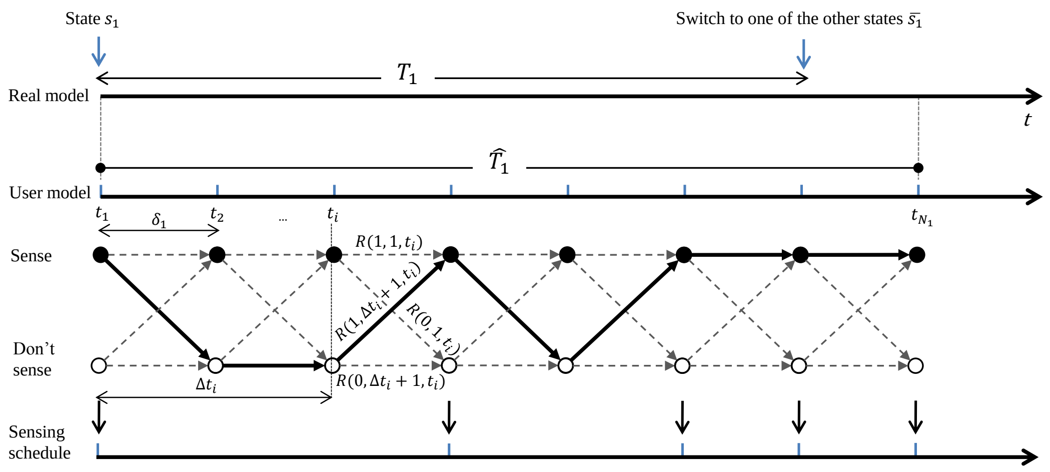


Figure 4.2: A unified model for user context and smartphone sensing: real model reflects the real state change, user model defines the derived system parameters needed for the algorithm, the Viterbi trellis shows the evolution of the algorithm, and the sensing schedule presents the instants at which sensing should be triggered.

the user. The two-step learning mode is executed on occasions when the statistics of the user's behavior change significantly based on our proposed VCAMS trigger strategy as described below in Subsection 4.2.4.

In terms of its flexibility, VCAMS is generic where it can deal with any number of activities. VCAMS can add new states if it is available in the training dataset. Futhermore, VCAMS's framework accounts for an "unknown" state. When an "unknown" state is encountered, the classification algorithm will alert VCAMS that the current state is an unknown state which does not have a specific sensing schedule. The lookup table (LUT) in VCAMS accounts for an unknown state denoted by s_u. It triggers continuous sensing until a change is detected into a state which is recognized by VCAMS. When the system has collected enough training data about the "unknown" state, VCAMS can be updated since it is scalable.

4.1.1 The Sensing Schedule

The problem is to devise an efficient algorithm for an application to detect a critical change in the contextual state of a user. For example, the user can be using an ECG-based health application that needs to discover any sudden change in the user's state from normal heart activity to risky heart beat. As shown in Figure 4.2, at each time instant t_i, we want to decide whether to trigger the sensor (action $a_i = 1$) or not (action $a_i = 0$). The real model axis in Figure 4.2 reflects that the person stays in state s_1 for time duration T_1, then transits to a different state $\bar{s_1}$. Energy is impacted by the number of times sensing mechanism takes place. Energy is reduced with infrequent sensing, and delay is reduced with more frequent sensing. As a result energy and delay provide conflicting requirements on device usage. Hence, the objective is to find optimized choice of when to trigger the sensing mechanism. Triggering the sensor allows detecting whether the user has changed the state. If the state has not changed, then no sensing is needed. As a result, ideally, we do not want to sense until the exact moment of state change. Any early sensing will cause wasted device energy; on the other hand, waiting too long to sense may cause miss and delay in the detection of the state change in the context being monitored. Our goal is to detect this transition as soon as possible while minimizing the amount of consumed energy.

A unified model for user context and phone sensing trigger points is shown in Figure 4.2. The user model is based on the user's historical behavior. It provides the following parameters:

- $\hat{T}_j$ is intended to estimate the time duration the user spends in context s_j. As a result, it is used to represent the time limit of state s_j before the model triggers continuous sensing. $\hat{T}_j$ is chosen based on the historical distribution of the time spent T_j in each state s_j. For example, $\hat{T}_1$ is the time limit for state s_1. If $\hat{T}_1 \leq T_1$, then no delay will be incurred in detecting state change from s_1 since we assume continuous sensing after $\hat{T}_1$; however, if $\hat{T}_1 > T_1$, there might be delay depending on the sensing decision. The choice of $\hat{T}_j$ will be further described in Subsection 4.1.3.

- $p_j(t_i)$ is the survival probability at time instant t_i that the context will stay in the same detected state s_j at this time instant. It is presented in more details in Subsection 4.1.2.

- δ_j represents the sampling interval between time instants where decisions are made to sense or not. The choice of δ_j is further described in Subsection 4.1.4.

- N_j is the number of time instants over the duration $\hat{T}_j$ at which decisions will take place.

The optimized sensing schedule is the output of the proposed Viterbi-based algorithm whose trellis is shown in Figure 4.2. In the figure, $R()$ is a probabilistic reward function comprehending the rewards when state changes and the rewards when state does not change. The goal is to find the sensing schedule which consists of an optimized sequence of actions a_i at each time instant t_i by looking forward over the time duration $\hat{T}_j$ and maximizing the cumulative rewards, where the rewards comprehend the trade-off between energy and delay. This problem can be mathematically formulated to maximize the total probabilistic rewards as follows:

$$\arg\max_{a_i;i=1\ldots N_j} \sum_{i=1}^{N_j} R(a_{i+1}, \Delta t_{i+1}, t_i) \tag{4.1}$$

Mathematically, $R(a_{i+1}, \Delta t_{i+1}, t_i)$ represents the probabilistic reward of having action a_{i+1} as the next action at t_{i+1}. As for a_{i+1}, it represents the decision action: 1 for "sense" and 0 for "not sense". Δt_{i+1} is the time instants accumulated since the last sense was triggered till time instant t_{i+1}. Δt_{i+1} plays an important role in determining the incurred delay. Δt_{i+1} depends on the current action at time instant t_i as follows:

$$\Delta t_{i+1} = \begin{cases} 1, & \text{if at } t_i, \text{ action } a_i = 1 \\ (\Delta t_i) + 1, & \text{if at } t_i, \text{ action } a_i = 0 \end{cases} \tag{4.2}$$

Δt_{i+1} resets to 1 when the action at time instant t_i is "sense"; on the other hand, Δt_{i+1} accumulates when no sensing is triggered at time instant t_i.

Each node to node link in the trellis has its own reward $R()$ which is a function of the next action a_{i+1}, the accumulated delay up to t_{i+1}, and the current time instant t_i. In addition, these reward functions need to comprehend energy and delays as will be described in Subsection 4.2.2. The trellis path that maximizes the probabilistic rewards is chosen. An example illustration is shown in bold lines on the trellis. The corresponding sensing schedule in the figure shows the optimized output sensing schedule where each impulsive arrow represents a trigger decision.

4.1.2 Survival Probability $p_j(t_i)$

The survival probability $p_j(t_i)$ represents the probability that the user will survive in state s_j for the current time instant t_i. In other words, it is the probability that the user will not transit to another state at this time instant. This probability is captured in the user model, and it depends on the previous historical behavior of the user and on the time limit that the user spends before transiting and changing current state. Therefore, this probability decreases with time where the highest probability lies at the instant when the state is first recognized and the lowest survival probability of the state s_j lies at the end of $\hat{T}_j$.

We used survival analysis [160, 161] to derive this probability. The duration until a state change happens is modeled as a random variable. The survival probability $p_j(t_i)$ represents the probability that no change has occurred up to the current time instant t_i. The exponential distribution is one of the most used survivor functions and it gives a good fit which matches the context more than any other distribution [160].

$$p_j(t_i) = e^{-h_j \cdot t_i} \tag{4.3}$$

where h_j is the exponential decaying rate that depends on the time limit $\hat{T}_j$ of state s_j and h_j is set such that the survival probability at t_{N_j} is around 0.001 as exponential cannot reach 0.

4.1.3 Choice of Time Limit $\hat{T}_j$

$\hat{T}_j$ represents the time limit at which continuous sensing is triggered. It is based on the estimated time spent by the user in context s_j. From the user's past behavior, the time spent by the user in the ${s_j}^{\text{th}}$ state can be considered as a random variable T_j, with mean μ_j and a standard deviation σ_j. These time properties are learned from previous behaviors, and they are updated dynamically whenever state s_j is encountered. Typically, the choice of $\hat{T}_j$ should not fall below $(\mu_j - 3 \cdot \sigma_j)$ since most of the time, the actual time T_j is above these values. The choice of a low $\hat{T}_j$ will cause high energy consumption. On the other hand, $\hat{T}_j$ should also not exceed $(\mu_j + 3 \cdot \sigma_j)$ because delaying continuous sensing beyond $(\mu_j + 3 \cdot \sigma_j)$ would incur higher delays. Hence, the choice of $\hat{T}_j$ reflects a trade-off. Choosing $\hat{T}_j < \mu_j$ causes more energy due to continuous sensing and does not provide opportunity for the algorithm to make intelligent decisions in the more frequent cases around the mean μ_j. On the other hand, choosing $\hat{T}_j > \mu_j$ minimizes energy but leads to more delay, and also provides the algorithm the opportunity to make efficient decision in the more frequent cases around the mean μ_j. Hence, $\hat{T}_j$ is chosen to be greater than μ_j. This choice allows Viterbi-based method to minimize delay.

4.1.4 Choice of Sampling Interval δ_j

In the user model shown in Figure 4.2, δ_j represents the sampling interval between time instants t_i and t_{i+1} at which our optimized sensing strategy checks whether to trigger sensing or not. When the model decides not to sense, the risk of delay in sensing a contextual state change increases. The time delay D_t is proportional to the sampling interval δ_j and the number of delay intervals D. D is measured as number of instants between the time instant when context change actually happened and the instant when the following sensing decision happened. Hence, the time delay D_t is measured in seconds as follows:

$$D_t = D \cdot \delta_j \tag{4.4}$$

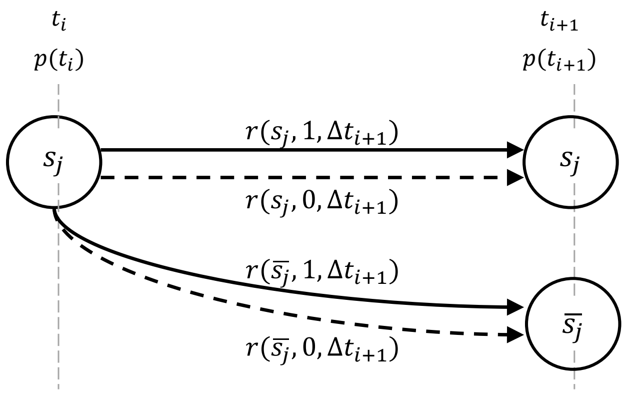
Increasing δ_j increases D_t. However, decreasing δ_j increases the frequency N_j, described in Subsection 4.1.1, at which sensing decisions need to be made; thus, more computational time and energy. As a result, the choice of δ_j provides a trade-off between energy and delay, consistent with the overall objective of the optimization model in (4.1). As a result, we choose to set the value of δ_j to the minimum possible $\delta_{j_{\min}}$ based on the sensor requirements. $\delta_{j_{\min}}$ is composed of the duration required to capture the raw data needed for one feature plus the duration needed to compute this feature; hence, it is sensor-dependent. For example, sensors can be divided into two categories based on their sensing behavior [100]. The first category includes sensors, such as accelerometer and microphone, that require a command from the system to turn on and start collecting samples and another command to turn it off and stop acquiring data. The second category includes sensors, such as GPS and WiFi, which are based on their own protocols to operate. These sensors need a command from the system to turn on; however, they turn into idle mode automatically by themselves after attaining the required data and finishing their tasks. In both cases, there is a minimum sampling interval $\delta_{j_{\min}}$ based on the specifications of each sensor. Furthermore, the acceptable $\delta_{j_{\min}}$ is relative to the time spent in a state; therefore, it is state-dependent. For example, the location of a user driving changes more frequently than the location of a user attending a meeting. In Section 4.3, we will consider the single state of a user during a meeting detected using WiFi hotspot coverage. We choose $\delta_{j_{\min}}$ based on the sensor requirements and state restrictions and delegate the energy-delay trade-off to the proposed Viterbi solution described in Section 4.2.

4.2 The Proposed Viterbi-based Method

This section presents the proposed learning method for the Viterbi-based algorithm with trade-off between energy and delay for context recognition using smart sensors. It describes the Viterbi trellis with its components as was shown in Figure 4.2. The Viterbi algorithm is commonly used to find the most likely sequence of states that maximizes the posteriori probability of a process. In this section, we describe our method when one sensor is considered; however, it can be generalized to multiple sensors. Subsection 4.2.1 gives a background about Viterbi algorithm. Subsection 4.2.2 presents the reward functions defined in terms of energy and delay factors, and Subsection 4.2.3 describes the steps to solve the energy-delay trade-off using Viterbi-based algorithm.

4.2.1 Viterbi Algorithm Background

Viterbi algorithm [158] is applied as a dynamic programming algorithm for an optimization problem that needs to maximize a statistical utility function. For example, it is used as an efficient method of estimating a sequence of hidden states in Hidden Markov Models (HMM) [162, 163]. Viterbi algorithm is also used to find the shortest path through a weighted graph. It is widely used in communication as a decoding algorithm for data encoded with convolutional encoding in digital data transmission [164, 165]. It aims at recognizing data errors caused by communication channels and correcting them. Although Viterbi was initially introduced in 1967 [166], it is still used in various fields as a dynamic programming algorithm which finds the most likely sequence of hidden states. It is recently being used in emerging concepts and domains ranging from communications [167] to target tracking [168], and even biomedical engineering [169, 170] .Viterbi has been shown to be optimal for estimating the state sequence of a finite state process [158]. Hence, Viterbi can be applied to any dynamic problem with



t_i
$p(t_i)$
t_{i+1}
$p(t_{i+1})$
s_j
s_j
$\bar{s_j}$
$r(s_j, 1, \Delta t_{i+1})$
$r(s_j, 0, \Delta t_{i+1})$
$r(\bar{s_j}, 1, \Delta t_{i+1})$
$r(\bar{s_j}, 0, \Delta t_{i+1})$

Figure 4.3: State and reward diagram.

finite states. In our case, the Viterbi states are the sensing decisions which are two: "sense" or "not sense". Our problem deals with real dynamic time limits where some sensing decisions might be taken over long periods of time; hence, Viterbi provides a reduction in computational complexity by using recursion and saving only the most likely path leading to each state [166].

Viterbi algorithm is described using two diagrams. The first diagram is the trellis diagram shown in Figure 4.2 which represents a graph of a finite set of nodes connected using edges that define the possible transitions between nodes at discrete time intervals. Each edge has its probabilistic reward function $R()$. The nodes in the trellis represent the decision where the upper node represents "sensing" decision and the lower node represents "not sensing" decision. The probabilistic state of the user is thus not illustrated in the trellis. Hence, we need the second diagram which is the state diagram shown in Figure 4.3. It defines the instantaneous rewards of the transitions between nodes based on the state of the user. The nodes in the state diagram represent the two states s_j and $\bar{s_j}$. The links between the nodes are the instantaneous rewards $r()$ that depend on whether the state has survived as s_j or transited into $\bar{s_j}$ on time instant t_{i+1}. The Viterbi algorithm uses a set of metrics (costs or rewards) to compare the costs of the various paths through the trellis and then decide the path that maximizes the problem-specific utility function. We define those metrics for our problem in subsection 4.2.2.

The general Viterbi algorithm works as follows. First, the algorithm looks at each node at time t_{i+1}, and for all the transitions from t_i that lead into that node, it chooses the path with the greatest metric, and it discards the other transitions into that node. The algorithm does the same process for all the trellis nodes starting from time t_i by moving forward to time t_{i+1} and repeating the process. When the algorithm reaches the end of the trellis ($t_i = t_{N_j}$), it evaluates the metrics for the different paths that lead to the end and outputs the path with the highest cumulative reward metric $R()$ as presented in (4.1).

Table 4.2: Energy, delay, and recognition components.

	state $= s_j$			state $= \bar{s_j}$		
	energy	delay	recognition	energy	delay	recognition
action = sense	-1	0	0	-1	$-(\alpha \cdot \Delta t_{i+1})$	$(\beta / \Delta t_{i+1})$
action = don't sense	0	0	0	0	$-(\alpha \cdot \Delta t_{i+1})$	0

4.2.2 Rewards

In dynamic programming algorithms, metrics need to be defined to represent the utility associated with each transition between nodes. The metrics used in our proposed Viterbi-based algorithm are defined as $R(a_{i+1}, \Delta t_{i+1}, t_i)$ as shown in the trellis in Figure 4.2. Each edge in the figure holds a defined metric $R()$. The true state of the user is not known unless a sensing mechanism is triggered; therefore, each $R()$ metric is a probabilistic combination of two instantaneous reward functions. At each time instant t_i, there is a probability $p_j(t_i)$ of remaining in state s_j and a probability $1 - p_j(t_i)$ of transitioning into another state $\bar{s_j}$. Therefore, the metric $R()$ is to be computed with probabilities depending on the action considered at each time step. Hence, for each action (sense or don't sense) there are two instantaneous rewards $r()$: the first one $r(\bar{s_j}, a_{i+1}, \Delta t_{i+1})$ representing the reward assuming a state change has happened and the other one $r(s_j, a_{i+1}, \Delta t_{i+1})$ assuming the monitored context is still in the same state s_j. The reward $R()$ for each action a_i becomes:

$$\begin{aligned} R(a_{i+1}, \Delta t_{i+1}, t_i) &= E_s[r(s, a_{i+1}, \Delta t_{i+1})] \\ &= p_j(t_i) \cdot r(s_j, a_{i+1}, \Delta t_{i+1}) + (1 - p_j(t_i)) \cdot r(\bar{s_j}, a_{i+1}, \Delta t_{i+1}) \end{aligned} \tag{4.5}$$

where E_s is the expectation over states s_j and $\bar{s_j}$.

The decision of when to sense depends on the survival probability and the cumulative delay. Intuitively, as the survival probability decreases with time; that is the probability of changing the state of the monitored context increases, we should sense more frequently. In addition, as the risk of accumulated delay increases, sensing mechanism becomes more necessary. Sensing more frequently guarantees lower delays but causes more energy consumption. Thus, energy and delay criteria are represented in rewards $r()$ to define the metrics of each transition in the Viterbi trellis. Table 4.2 shows how we divide the metric into three components: the energy component which represents the cost of sensing when triggered, the delay component which accumulates since the last time instant when sensing was triggered, and the recognition component which defines the reward of recognizing a state transition. Each instantaneous reward $r()$ has an energy, delay, and recognition component.

Energy and delay are measured in terms of time instants. Delay is computed as the number of time instants skipped before detecting a state transition, and energy is computed as the number of time instants at which sensing is triggered. Therefore, energy costs either 0 when no sensing is triggered or -1 when sensing is triggered. Let α and β be energy-delay weighting factors. Δt_{i+1} is used in defining the delay component since the actual delay time is not known as the true state of the user cannot be revealed unless sensing is triggered. α is used in the delay component. It varies the effect of Δt_{i+1} on sensing decision. Increasing α would increase the negative effect of incurred delay; thus, as α increases, sensing is triggered more frequently to avoid excessive delay. On the other hand, β affects the recognition component where VCAMS system gains more reward when β increases; thus, increasing β also leads to more sensing. Therefore, increasing α and β causes more sensing; thus more energy and less delay.

The choice of those weighting factors is application-dependent. For example, if the application is health-related with critical context being monitored, α and β should be chosen such that delay is minimized. Another factor that affects the choice of the weighting factors is the phone status. For example, if the phone battery is low, α and β should be chosen to minimize energy consumption; and thus sense less frequently.

Pareto Optimal (α, β)

Sensing more frequently causes more energy but decreases the delay values; hence, there is a trade-off between energy and delay. Thus, finding the best (α, β) combination is a multi-objective optimization problem which requires an optimal Pareto solution. Pareto solutions find points which are acceptable by both objectives. The optimal (α, β) combination is the one which provides a balance between the incurred delay values and the total consumed sensing energy. Several approximation methods are proposed and used to identify the most desirable Pareto points [171]. The most common approach is an approximation method based on scalarization which combines the multiple objectives into one single-objective function using weighted-sum [172]:

$$\min_{\alpha,\beta} \quad \omega \cdot \frac{E(\alpha,\beta)}{E_{\max}} + (1-\omega) \cdot \frac{D(\alpha,\beta)}{D_{\max}} \tag{4.6}$$

where $0 < \omega < 1$ is the weighting factor, E is the energy function, D is the delay. $E_{\max}$ is the maximum energy and $D_{\max}$ is the maximum delay, and are used to normalize the energy and delay as shown in (4.6). The set of solutions, called non-dominated solutions, which provide a balance between the different objectives, can be generated by varying the weight ω in (4.6) [173]. Since α and β are varied to balance the different objectives, ω can be set arbitrarily. We choose to set ω to 0.5; thus giving equal weight for both energy and delay components in (4.6). It is worth noting that $E_{\max}$ is proportional to the time limit $\hat{T}_j$ at which decisions take place. In addition, the maximum delay $D_{\max}$ that can occur is $\hat{T}_j$ which represents the case when sensing schedule decides to sense only at $\hat{T}_j$. Therefore, if we change the time limit for the same choices of Pareto optimal α and β, we will keep the same equal proportion of energy and delay. By normalizing the energy and delay values, we avoid re-computing the Pareto optimal (α, β) combination again each time a state is encountered in real-time.

Instantaneous Rewards

As shown in Table 4.2, there are four combinations of actions and states; therefore, there are four instantaneous rewards $r()$ as shown in Figure 4.3. The dashed lines in the figure represent the rewards when the action is not to sense. The instantaneous reward $r()$ depends on the current state of the user, the decision taken whether to sense or not at each instantaneous time instant, and the time elapsed since the last sensing mechanism Δt_{i+1}. For each state and action combination, the corresponding reward is obtained by summing up the corresponding energy, delay, and recognition components as follows:

1. When the state is still in the same state s_j and the action is to sense $a_i = 1$ at the corresponding time instant, there is an energy cost.

$$r(s_j, 1, \Delta t_{i+1}) = -1 \tag{4.7}$$

2. When the state is still in the same state s_j and the action is not to sense $a_i = 0$, there are neither energy nor delay rewards.

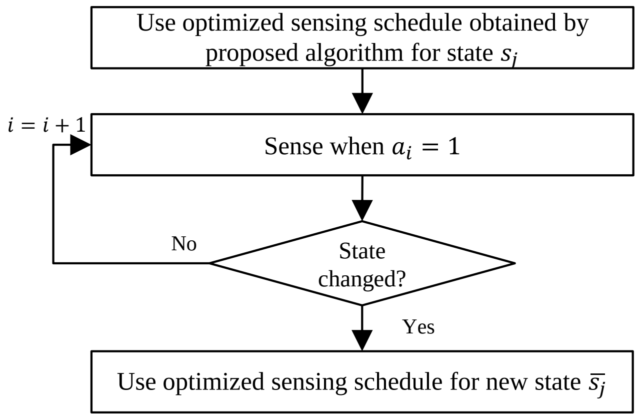
$$r(s_j, 0, \Delta t_{i+1}) = 0 \tag{4.8}$$

3. When the state has changed to $\bar{s_j}$ and the action is to sense $a_i = 1$, there is an energy cost and a negative delay cost which is directly proportional to α and Δt_{i+1}. In addition, there is a positive recognition reward for the correct decision but this reward is inversely proportional to Δt_{i+1}. As β increases, delay is given a higher weight thus emphasizing more sensing.

$$r(\bar{s_j}, 1, \Delta t_{i+1}) = -1 - (\alpha \cdot \Delta t_{i+1}) + \beta / \Delta t_{i+1} \tag{4.9}$$

4. When the state has changed to $\bar{s_j}$ and still the action is not to sense $a_i = 0$, there is negative delay.

$$r(\bar{s_j}, 0, \Delta t_{i+1}) = -(\alpha \cdot \Delta t_{i+1}) \tag{4.10}$$



Use optimized sensing schedule obtained by proposed algorithm for state s_j
$i = i + 1$
Sense when $a_i = 1$
No
State changed?
Yes
Use optimized sensing schedule for new state $\overline{s_j}$

Figure 4.4: Viterbi-based algorithm resets when detecting new state.

4.2.3 Viterbi-based Algorithm for Optimized Sensing Schedule

This subsection presents the Viterbi learning algorithms that output the sensing schedule. Our proposed algorithm is adaptive to the state under consideration, where the optimized sensing schedule depends on the state being monitored. Each state s_j has its own time statistics such as the time limit spent in this state is $\hat{T}_j$. If at any time instant sensing reveals a change in state, a new optimized sensing schedule is started with new state related rewards as shown in Figure 4.4.

Algorithm 4.1 describes how the time statistics for state s_j are computed and shows the steps to compute the rewards $R()$ which are the inputs for the Viterbi-based algorithm described later in Algorithm 4.2. To derive the sensing schedule, Algorithm 4.1 takes as input the whole collected historical data of the user model that captures how much time the user spends in particular contexts. Another input to the algorithm is the energy-delay weighting factors α and β. The first step in Algorithm 4.1 is to compute the mean μ_j and the standard deviation σ_j from the historical data (line 1). Then, the time limit $\hat{T}_j$ is chosen to be greater than the mean μ_j according to Subsection 4.1.3 while δ_j is chosen as the minimum $\delta_{j_{\min}}$ based on Subsection 4.1.4 (lines 2 and 3). The number of time instants N_j at which sensing decision should take place is derived as $N_j = \hat{T}_j \Big/ \delta_j$ (line 4). Survival probabilities are computed based on (4.3), and instantaneous rewards and computed based on Table 4.2 (lines 5 and 6). Finally, the rewards $R()$ are derived using $p_j(t_i)$ and $r()$ as described in (4.5).

Algorithm 4.2 presents the pseudocode to learn the optimized sensing schedule by searching for the best Viterbi path that chooses the best action a_i at each time instant t_i. The general Viterbi algorithm works as follows. At each step $t_i = t_1$ to t_{N_j-1} (line 1), it computes the cost of the different possible paths and keeps the path with the highest reward for each node. There exists a path from each node

Algorithm 4.1 Computing the time statistics and the rewards for state s_j

Input:
- Historical data of statistics in state s_j
- Energy-delay weighting factors α and β

Output:
The rewards $R(a_{i+1}, \Delta t_{i+1}, t_i)$

```
1: Compute the mean $\mu_j$ and the standard deviation $\sigma_j$ for the time
   duration $T_j$ spent in state $s_j$
2: Choose time limit $\hat{T}_j > \mu_j$ based on Subsection 4.1.3
3: Choose sampling interval $\delta_j = \delta_{j_{\min}}$ based on Subsection 4.1.4
4: Derive the number of time instants $N_j = \hat{T}_j \big/ \delta_j$
5: Compute the survival probability $p_j(t_i)$ for $t_i = t_1$ to $t_{N_j}$
6: Compute the instantaneous rewards $r()$ for all time instants based
   on Table 4.2
7: Derive the rewards $R()$ using (4.5)
```

Algorithm 4.2 Choosing the optimized sensing schedule a_i

Input:
- Rewards $R(a_{i+1}, \Delta t_{i+1}, t_i)$ (computed in Algorithm 4.1)
- Number of time instants N_j

Output:
Best action a_i for time instants $t_i = t_1$ to t_{N_j}

```
1: for each $t_i = t_1$ to $t_{N_j - 1}$ do
2:   for each of the $A$ nodes at $t_{i+1}$ representing $a_{i+1} = 0$ to $1$ do
3:     Compare the values $R(a_{i+1}, 1, t_i)$ and $R(a_{i+1}, \Delta t_i + 1, t_i)$ and keep
       the path to $a_{i+1}$ which maximizes the cumulative reward $R()$ up to
       time $t_{i+1}$
4:   end for
5: end for
6: Choose the path with the highest cumulative reward at $t_i = t_{N_j}$ to
   solve (4.1): $\arg\max_{a_i; i=1 \ldots N_j} \sum_{i=1}^{N_j} R(a_{i+1}, \Delta t_{i+1}, t_i)$
```

at t_i to reach each node at t_{i+1}. Let A be the total number of nodes at each time instant which is $A = 2$ in our case. For each node decision at t_{i+1} representing either $a_{i+1} = 0$ or $a_{i+1} = 1$ (line 2) there are two paths: one originates from $a_i = 0$ having $R(a_{i+1}, \Delta t_i + 1, t_i)$ and the other path originates from node $a_i = 1$ having $R(a_{i+1}, 1, t_i)$. For each node decision a_{i+1}, Viterbi algorithm keeps only one path leading to this node which is the path with the highest cumulative metric $R()$ (line 3). This process is repeated for all time instants until reaching t_{N_j} where the algorithm backtracks back to the starting point to find the best path with the highest cumulative metric (line 6).

4.2.4 Triggering of VCAMS Learning Mode

Whenever a new sensing schedule is needed, the system computes the expected rewards using a collected historical dataset for the particular user (Algorithm 4.1) and runs the Viterbi algorithm to maximize those rewards (Algorithm 4.2). However, user's behavior may change over time; therefore, the sensing schedule must adapt to the new data available in real-time by running again the needed learning algorithms and deriving new sensing schedules. To reduce computational overhead, the system should trigger the learning mode for a new sensing schedule only when significant behavioral changes happen. We assume that a behavior has a measurable time length that characterizes it; therefore, a significant behavioral change happens when the time span spent in the state changes significantly from its typical value.

Mathematically, to track the user's behavior and quantitatively explain what significant behavioral change means, the system tracks the statistics associated with state s_j which include the mean value μ_j and the standard deviation σ_j, and constantly updates them as new measurements are captured in real-time. To quantify this relative change from the typical norm, we measure the relative difference between the new time spent and the typical average time spent μ_j. If this normalized difference is more than 1, we assume that there is a significant change in the behavior. Let's denote the new captured time duration spent in s_j as v_j. Behavioral changes are assessed with every new measurement v_j by comparing it to the previously tracked historical average μ_j, and assessing the ration $\kappa = \frac{|\mu_j - v_j|}{\sigma_j}$. When the deviation from the average is less than σ_j, the previously computed sensing schedule for this state is not changed since v_j would still be within the bound of the time limit $\hat{T}_j = \mu_j + \sigma_j$. Behavioral change is flagged as having significantly changed if $\kappa > 1$. When the system detects that a state s_j has been flagged three times, it triggers the learning mode to derive a new sensing schedule ss_j. Therefore, the rules that decide when to trigger the learning mode form our proposed VCAMS adaptive trigger strategy (VT).
The rules for triggering VCAMS's learning mode are as follows:

- $\kappa \leq 1$ (no change in behavior):
    - Update the time statistics.

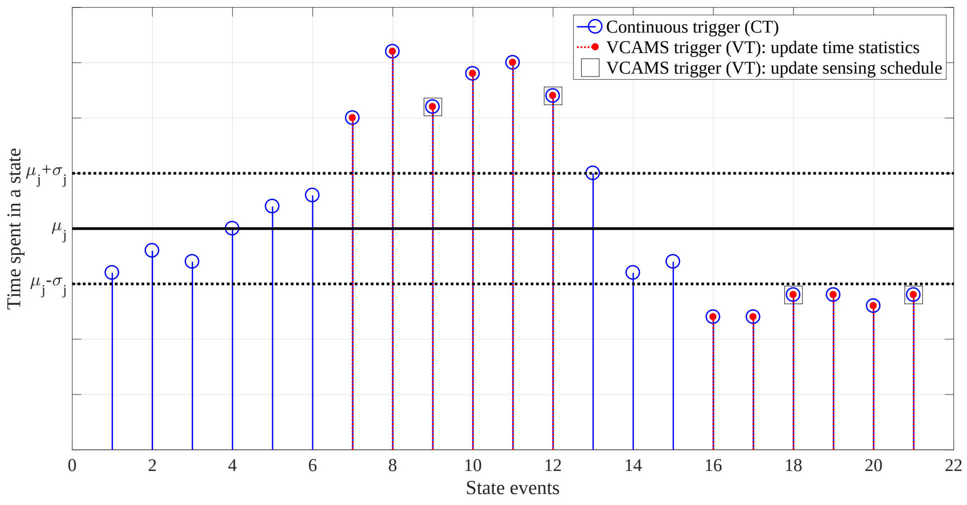


Figure 4.5: Strategy to decide when to trigger VCAMS's learning mode computations.

- $\kappa > 1$ (change in behavior):
  - Update the time statistics.
  - Derive new sensing schedule after the behavior is flagged as having significantly changed for three times.

Figure 4.5 shows a simulation of measurements v_j, and indications of when learning modes are triggered. Without our proposed rules (VT), the default mode is continuous triggering (CT) which triggers learning mode with every new measurement. The x-axis represents the count of state events that increments each time this specific state is encountered. The y-axis represents the time measurement v_j spent by the user in the particular state. The horizontal line in the middle of Figure 4.5 shows the mean value μ_j while the dotted lines illustrate the $\kappa = 1$ threshold of relative differences. The instants at which VT updates the time statistics are illustrated by the red dots whereas the black squares indicate the time instants at which VT triggers VCAMS's learning mode to update the sensing schedule. The blue circles indicate the time instants at which CT triggers the learning mode. The figure illustrates the savings in overhead, where 12 of the 22 events are triggered for updates and only 4 events are triggered for learning.

4.2.5 Computational Complexity Analysis

We now comment on the overall computational complexity of the proposed solution. We distinguish between the learning mode and the execution mode. The learning mode runs offline once to derive Pareto parameters for the multi-objective function and initialize the sensing schedules from historical collected dataset, and it runs as needed with behavioral changes to derive new sensing schedules. In the online execution mode, the system selects based on the current user's state the sensing schedule from the LUT; therefore, there are minimal computations, and thus the complexity is low.

Pareto optimal parameters for the multi-objective function are derived once offline by repeating Viterbi P times where P is the number of possible combinations for the parameters α and β. While α and β are continuous variables between 0 and 1, a finite search grid can be used without loss of accuracy. In our experiments, we varied α and β between 0 to 1 in steps of 0.1 (i.e., 11 values of α combined with 11 values of β, leading to $P = 121$ different combinations). The other parts of the learning mode (Algorithms 4.1 and 4.2) run offline using the derived Pareto points to initialize the system and derive the sensing schedules using a Viterbi algorithm based on historical data of the user model.

The feature extraction and context detection steps shown in the execution mode in Figure 4.1 depend on the classification algorithm which is being used. In general, there is a trade-off between the computational complexity and the accuracy of classification algorithms. To improve the performance, we can use an ensemble of classifiers rather than one classifier; however, ensemble classification comes at the price of more computational energy and complexity. VCAMS is generic and any classification algorithm can be used along with VCAMS depending on the application and the triggered sensor. During execution mode, the system checks continuously whether VCAMS's learning mode should be triggered based on the rules set in VCAMS trigger strategy (VT). When learning mode is triggered online, the time statistics are updated as described in Algorithm 4.1. These time statistics are algebraic aggregation functions; therefore, they can be defined as a scalar function of distributive computations [174, 175]. The computational complexity of calculating each of the mean and the standard deviation is low of order $\mathcal{O}(1)$ with updates.

When VT decides to update the sensing schedule online, Algorithm 4.2 is triggered where the optimized sensing schedule is derived by running Viterbi-based approach. A trellis diagram shown in Figure 4.2 is used to represent the optimal path selection problem. Let us denote "A" as the number of distinct possible actions at each node. There are A paths that reach each node in the trellis. Viterbi algorithm [158, 166] reduces the computational complexity by using the recursion where only the node with the highest cumulative reward survives. The time complexity of the Viterbi algorithm is then $\mathcal{O}(N_j \cdot A^2)$ where N_j is the number of total time instants for state s_j, which has been shown to be smaller when compared to alternative searching algorithms [176, 177].

4.3 Evaluation Using a Context Simulator

Several experiments were conducted with simulation to test the efficiency of our proposed method. The optimized VCAMS sensing schedule was benchmarked against continuous sensing and other state-of-the-art methods to show energy and delay enhancements. We did two sets of experiments. In the first set of experiments, we focused on examining performance with two scenarios with one state change. For one scenario, we used a simulator monitoring one state assessed energy and delay. The second scenario was an implementation on an Android device to reflect real usage and estimate the computational costs. In the second set of experiments covered in Section 4.4, we used real data benchmark with multiple states in real-life situations. Furthermore, we conducted analysis for the different parameters used in the algorithms. Different energy-delay weighting factors (α, β) combinations were studied to investigate their effect on the trade-off between energy consumption and delay. In addition, the effects of the system parameters such as sampling interval δ_j and time limit $\hat{T}_j$ were investigated.

4.3.1 Simulation Experiments' Setup

We used the Siafu platform [178], which is an open-source context simulator developed in Java. We used the office scenario which is modeled based upon the typical behavior of an employee where agents get to work in the morning, work at their desks and attend meetings. To illustrate the method, we chose the state "$s_j =$ AtMeeting" which represents meeting attendance detected using the range of WiFi hotspots. We divided the data into two parts. The first part was used as historical data for initial learning to develop the user model where we obtain the time limit $\hat{T}_j$ for each state/activity s_j and the probability distribution of the time spent in each state and develop the initial sensing schedule for each state. The second part of the data was used for testing in the execution mode to validate the efficiency of the proposed algorithm in detecting state transitions. The time statistics derived from historical data were mean time $\mu_j = 100$ min and standard deviation $\sigma_j = 30$ min. The average time of a single WiFi scan time is 10 sec; however, running a WiFi scan every 10 sec is too restrictive; therefore, we chose to set the value of $\delta_j = 1$ min based on the sensor requirements in addition to state and application restrictions as discussed in subsection 4.1.4. As for the testing data, we ran the simulator to generate 10000 values of the actual time T_j spent in the meeting state. Energy and delay were extracted for each of these 10000 runs, and then the average values of energy and delay for these 10000 runs were computed. For the phone implementation of VCAMS, we selected an Android-based HTC Desire which is equipped with Octa-core 1.7 GHz processor.

4.3.2 Sensing Energy and Detection Delay Evaluation

The objective of VCAMS is to trade-off two factors: sensing energy and delay. Hence, we should evaluate these two criteria.

Sensing Energy in terms of Sensing Schedule

Given the optimized sensing schedule which decides when to trigger sensing, define k to be the number of times the sensor is triggered from the time instant the system recognizes being in state s_j till it detects the transition into state $\bar{s_j}$. Each sensor has its own power characteristics which lead to different energy consumptions per trigger. Hence, rather than specifying the sensor-specific Joules consumed each time the sensor is triggered, we evaluated the general energy E as the number of time instants at which sensing is triggered. E can be used for all sensors. We evaluated E as follows:

$$E = k. \tag{4.11}$$

Delay Intervals in terms of Sensing Schedule

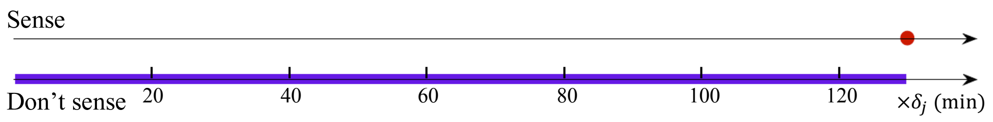
As for the delay, it is the number of time instants between the instant when context change actually happened and the instant when the following sensing decision happened. Assume t_u to be the real transition time instant and t_v be the subsequent time instant when sensing occurred and state transition is detected. Hence, the delay D can be computed as:

$$D = t_v - t_u. \tag{4.12}$$

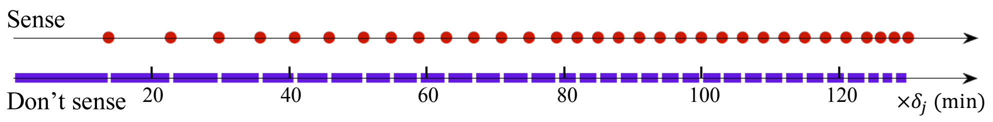
4.3.3 Effect of Energy-Delay Weighting Coefficients α and β

The normalized weighting factors α and β reflect the relative effect of delay with respect to energy in each of the reward functions presented in equations (4.7), (4.8), (4.9), and (4.10). Varying these weighting factors affects the derived sensing schedule. In this experiment, we took one standard deviation above the mean to obtain the time limit $\hat{T_j} = 130$ min before continuous sensing is triggered.

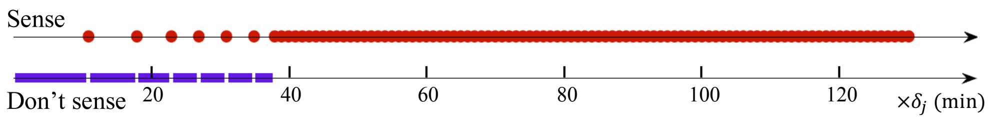
Figure 4.6 demonstrates VCAMS sensing schedule that was obtained for different values of α and β. The figure shows VCAMS decision at each time unit whether to sense or don't sense. For each subfigure, the upper axis represents the time instants at which sensing is triggered, and the lower axis represents the other time instants at which no sensing is triggered. Therefore, the red dots mean "sensing" time instants and the blue boxes illustrate the "no sensing" instants. Figure 4.6 illustrates four specific combinations of (α, β). Subfigures 4.6a, 4.6b, 4.6c, and 4.6d capture (α, β) = (0, 0), (0.7, 0.4), (1, 1), and (2, 2) respectively. We chose those combinations because they are extreme values for the weighting coefficients. It can be seen that as α and β increase, the system approaches continuous sensing. Figure 4.6a shows that when $\alpha = 0$ and $\beta = 0$, VCAMS decision is to sense only once when $t = \hat{T_j}$. Therefore, the lower bound of α and β cannot go below 0. Figure 4.6b presents the Pareto optimal behavior



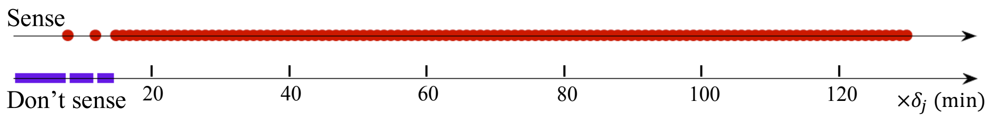
(a) $\alpha = 0$ and $\beta = 0$



(b) $\alpha = 0.7$ and $\beta = 0.4$ (Optimal)



(c) $\alpha = 1$ and $\beta = 1$



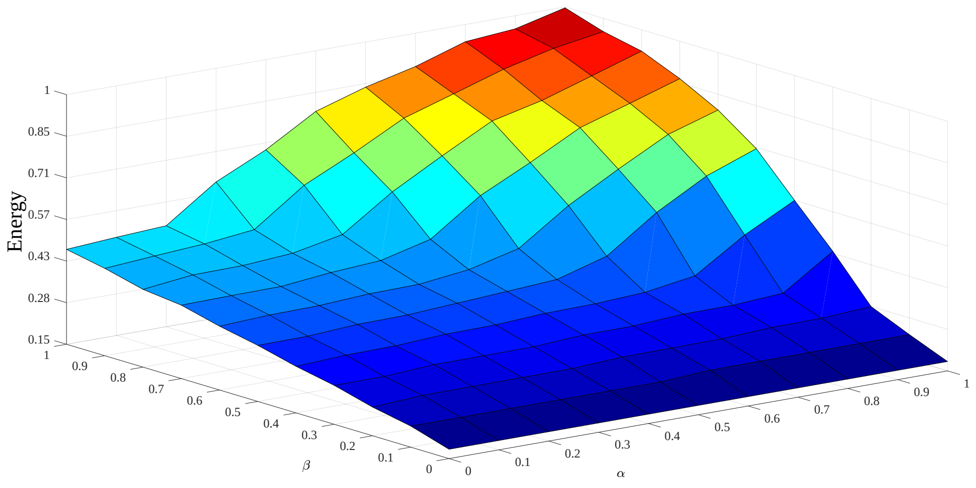
(d) $\alpha = 2$ and $\beta = 2$

Figure 4.6: VCAMS sensing schedule for different (α,β) combinations.

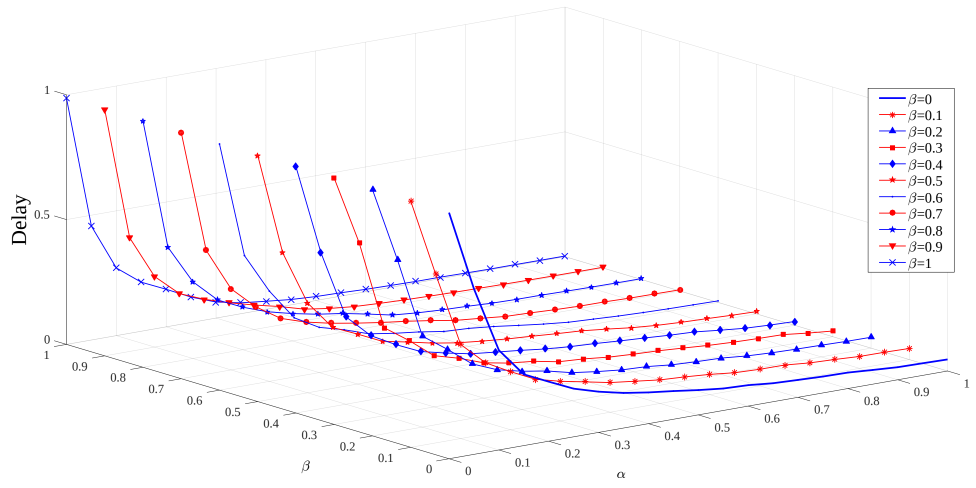
of VCAMS that will be shown in Figure 4.8. The duration of idle times when no sensing is triggered decreases with time as t approaches $\hat{T}_j$. Figure 4.6c shows the behavior of VCAMS when both α and β are unity. The continuous sensing behavior appears in this case. Figure 4.6d shows that increasing α and β beyond 1 will lead to excessive continuous sensing behavior; thus, we restrict α and β to be less than 1 to avoid excessive energy consumption. Therefore, we varied each of α and β between 0 and 1.

Figure 4.7 shows the effect on energy and delay for different values of alpha and beta. We varied each of α and β between 0 and 1 in steps of 0.1 and for each combination, we generated 10000 values of T_j and computed the energy and delay for each. Then, the normalized average energy and delay were computed. Figure 4.7 shows the effect of varying α and β on energy and delay. It can be seen that energy increases with the increase of α and β. Delay decreases with the increase of α, and it slightly decreases with the increase of β. Therefore, increasing α and β causes more sensing; thus more energy and less delay. This result is consistent with the discussion in Subsection 4.2.2.

To solve the trade-off between energy and delay, we searched for the best (α, β) combination which we solved by finding a Pareto optimal point. Pareto point is the one acceptable by both objectives where an improvement in energy requires a degradation in delay and vice-versa. We used the normalized averages of energy and delay by dividing each of the two factors by its corresponding maximum value. A Pareto optimal solution was derived to find the Pareto optimal α and



(a)



(b)

Figure 4.7: Effect of different choices of α and β on energy (a) and delay (b).

β values as shown in Figure 4.8. These values came out to be $\alpha = 0.7$ and $\beta = 0.4$. Therefore, for the remaining experiments, we set α and β to their Pareto optimal values. Figure 4.8 shows the non-dominated points and the Pareto optimal point illustrated as the star point which gives a balance between the total energy consumed and the delay incurred.

4.3.4 Impact of False Positive in Context Classification

When the system triggers sensing, the sensor collects raw data which undergoes processing to recognize the state using a classification algorithm which is characterized by an average classification accuracy. In this section, we study the impact of false positive and false negative events on the system's performance. A

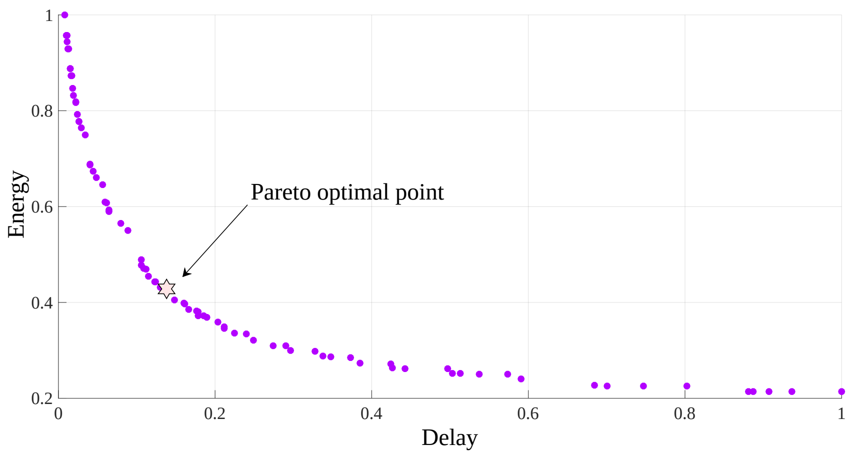


Figure 4.8: Pareto effect.

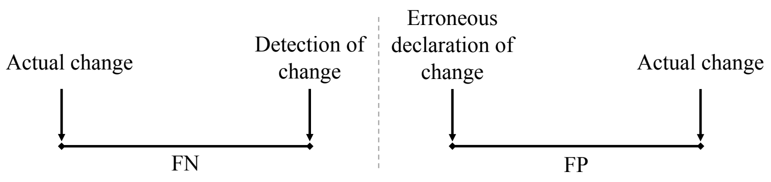


Figure 4.9: False positives (FP) versus false negatives (FN).

false positive occurs when the system erroneously detects a context change, and a false negative occurs when the system erroneously decides that the context has not changed. We will refer to a false negative as FN and the false positive as FP. The impacts of FN and FP are shown in Figure 4.9. If the system makes a FN, extra delay will be accumulated until the system correctly detects that the state has changed. During that period of time, the system will be in an erroneous state period due to the FN. On the other hand, if the system makes a FP for current state, the real state will be lost and the error may accumulate until the system gets back on the correct track. Such a false positive event causes another kind of erroneous state period of time. We use the term "undesired period" to refer to any period caused by a false detection where the system is in an erroneous state.

To analyze the tolerance of our proposed approach to false positives and false negatives of triggered activities, we ran VCAMS for the state "s_j = AtMeeting" and we varied the classification accuracy between 50% and 100%. For each classification accuracy, we ran the simulator to generate 10000 values of the actual time spent in a meeting state. The percentage of undesired periods was extracted for each of these 10000 runs and then the average value was computed. The results illustrated in Figure 4.10 show that running VCAMS using a more accurate classification algorithm leads to lower false positives and accordingly to a lower percent of undesired periods. VCAMS system's performance is impacted linearly with the accuracy of the classifier. From the figure we can see that even when

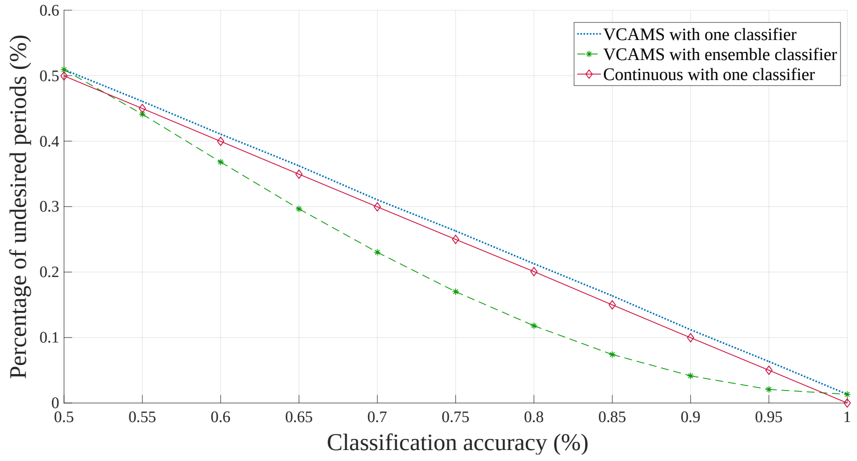


Figure 4.10: Percentage of undesired periods versus the classification accuracy.

classification accuracy is at 100%, there is still a small percentage of undesired periods. This is related to the VCAMS sensing strategy to trade-off energy for delays. Since even if the case of 100% context classifier accuracy, VCAMS may still choose to delay sensing to save energy. To eliminate any delay, VCAMS can switch to continuous sensing, which is shown in the red curve, but such a strategy would have higher energy cost.

To illustrate how classification accuracy can be improved even for low accuracy classifier, we simulated VCAMS with ensemble classification as a means to enhance classification accuracy. Instead of collecting raw sensor data for one classification, we introduced a sequence of three consecutive classifications where the final decision uses a majority rule. When a sensing decision is triggered by VCAMS, the classifier requires three consecutive readings to feed the majority rule. The figure shows that VCAMS with ensemble classifier causes less percentage of undesired periods than that of both VCAMS with one classifier and continuous sensing. It is worth noting that while ensemble classification improves overall performance, it comes at the price of more computational energy and complexity. For the remaining experiments, we set the classification accuracy to a constant value such that we can study the trade-off between energy and delay.

4.3.5 Effect of Sampling Interval δ_j

We described the choice of δ_j in subsection 4.1.4. We chose to set the value of δ_j to the minimum possible $\delta_{j_{\min}}$ based on the sensor requirements and state restrictions. $\delta_{j_{\min}}$ minimizes delay; however, it still consumes more energy than higher values of δ_j and it causes more computational time. In this experiment, we set $\alpha = 0.7$ and $\beta = 0.4$ and we tried four different values for δ_j to show its effect on energy, delay, and computational time. For each value of δ_j, we ran the experiment again for 10000 times to compute the normalized averages of energy,

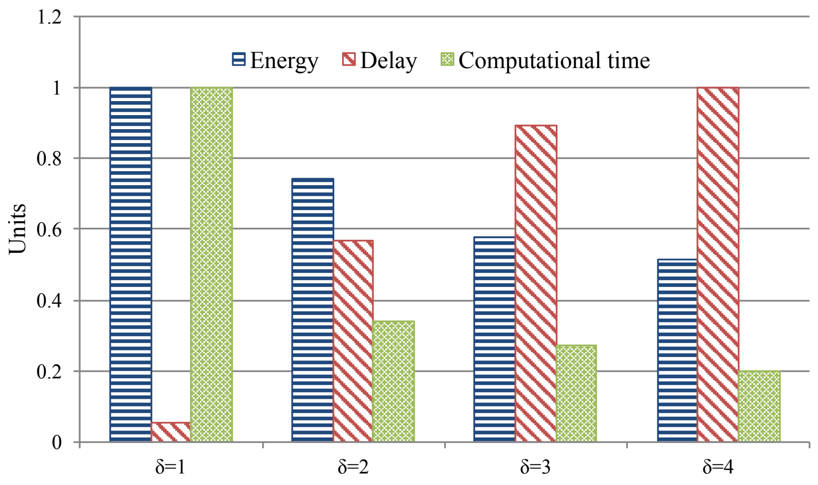


Figure 4.11: Energy, delay, and computational time for different δ_js.

delay, and computational time. Their values were normalized by dividing each of energy, delay and computational time by its corresponding maximum value. Figure 4.11 shows how the choice of δ affects energy, delay, and computational time values. As δ_j increases, energy and computational time decrease while delay increases. In this experiment, we coincidentally noticed also that our choice of $\delta_j = 1$ optimizes the energy-delay trade-off. When δ_j increases from 1 to 4, normalized units of energy decrease from 1 to 0.51 which is a 49% improvement in energy's behavior while normalized units of delay increase from 0.05 to 1 which is a 95% degradation in delay's behavior. On the other hand, normalized units of computational time decrease from 1 to 0.2 which is almost 80% improvement. In our proposed approach, we are trading-off energy and delay; thus, it can be concluded that the proportional delay loss exceeds the proportional energy gain when increasing δ_j. Therefore, taking δ_j to be the minimal is justified.

4.3.6 Effect of Time Limit $\hat{T}_j$

In subsection 4.1.3, we chose $\hat{T}_j$ such that it is greater than the mean value μ_j to avoid excessive early continuous sensing. For experimentation, we choose:

$$\hat{T}_j = \mu_j + \gamma \cdot \sigma_j \tag{4.13}$$

where γ reflects the trade-off in energy versus delay. γ depends on the delay that the application is able to tolerate. In this experiment, the system operated in its optimal condition where $\alpha = 0.7$ and $\beta = 0.4$. δ_j was set to 1 according to the discussion in the previous subsection. To study the effects of varying γ, simulations were conducted for different values of γ. The results are presented in Figure 4.12. Energy and delay values were normalized into units by dividing each by its maximum value to obtain a range of $\{0, 1\}$. Energy was examined for

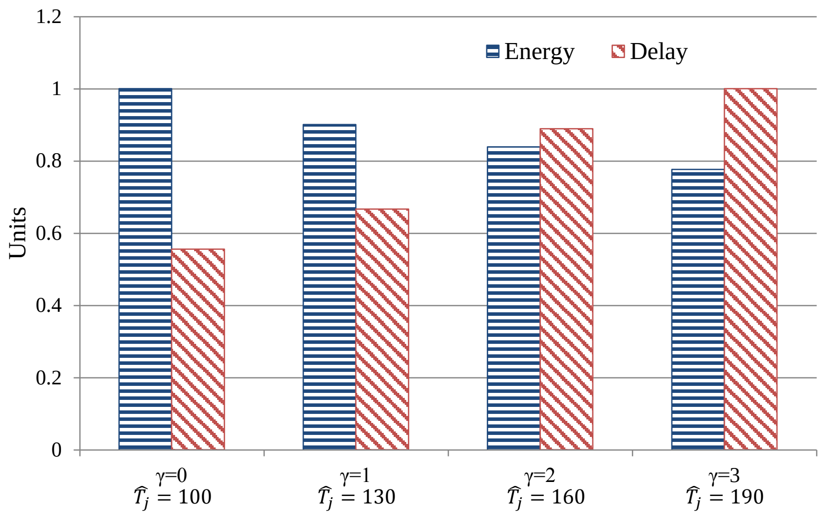


Figure 4.12: Energy and delay for different choices of $\hat{T}_j$s.

all the cases considered. The figure shows that energy decreases as $\hat{T}_j$ increases. Delay in recognizing a state change was also measured. Increasing $\hat{T}_j$ causes more delay since the sensor triggers around the average time are dispersed. Hence, we experimentally choose γ such that the delay is acceptable. When γ increases from 1 to 2, normalized units of energy decrease from 0.9 to 0.82 which is a 9% improvement in energy's behavior while normalized units of delay increase from 0.63 to 0.89 which is a 32% degradation in delay's behavior. Hence, increasing γ beyond 1 leads to delay loss which exceeds the slight energy gains.

4.3.7 Choice of Probability Distribution $p_j(t_i)$ and its Parameters

In subsection 4.1.2, we chose $p_j(t_i)$ to follow an exponential distribution since it gives a good fit that matches the context change. Furthermore, we set the state-based decaying hazard rate h_j shown in (4.3) such that the probability of survival approaches 0 when the time approaches the time limit $\hat{T}_j$. In this section, we will perform supplementary experimentation to show the impact of exponential distribution versus other more general distributions. In this experiment, the system operated in its optimal condition where $\alpha = 0.7$ and $\beta = 0.4$. To study the effect of the different survival probability distributions, we consider the general Weibull distribution [179].

$$p_j(t_i) = e^{(-h_j \cdot t_i)^\xi} \tag{4.14}$$

where ξ is the shape parameter. Weibull distribution reduces to the exponential distribution when $\xi = 1$. We conducted simulations for different values of ξ, and the results are presented in Figure 4.13. We used the normalized averages of energy and delay by dividing each of the two factors by its corresponding

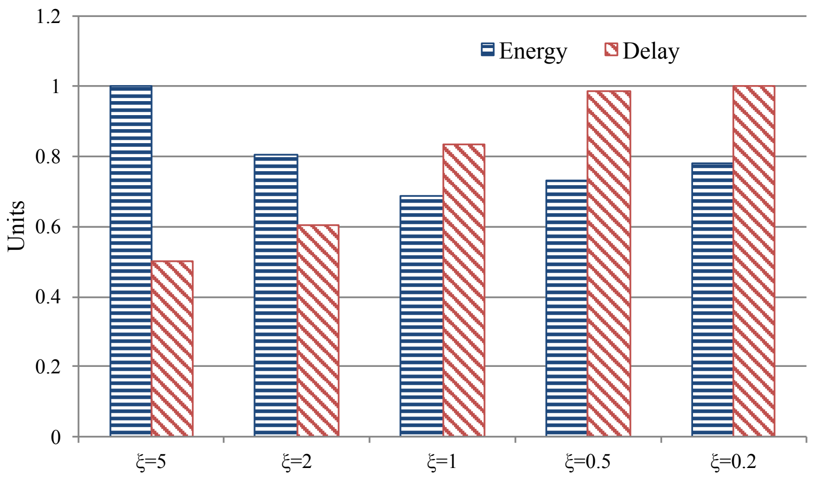


Figure 4.13: Energy and delay for different choices of survival probability distributions.

maximum value. The figure shows that as ξ decreases, energy decreases until it reaches the exponential distribution where $\xi = 1$ after which energy slightly increases. As for delay, it increases as ξ decreases. We chose the exponential distribution since it provides the lowest energy and the trade-off between energy and delay. However, VCAMS is generic and any distribution can be used based on the data available and the context being detected.

We have also investigated the effect of changing the decaying hazard function h_j which is the instantaneous probability that an event will occur at a specific time. In exponential distribution, the decaying rate is constant over time. We consider ${h_j}^0$ to be the decaying rate that we used throughout the simulations. We conducted simulations to show the effect of faster decaying rates. We varied the decaying rate between $\frac{1}{2} \cdot {h_j}^0$ and $2 \cdot {h_j}^0$. The results are illustrated in Figure 4.14. The figure shows that taking the hazard rate as ${h_j}^0$ gives the required balance since we are trading-off energy and delay.

4.3.8 Performance Analysis and Comparison

In this subsection, we first compare our work with existing similar work proposed by other researchers, then we experimentally analyze the computational complexity of our algorithm.

VCAMS versus State-of-the-art Sensing Schedules

There are some existing sensing policies which researchers have proposed [97, 100, 123, 106] whose goal is to detect context in an energy efficient way. These proposed sensing mechanisms aim at trading-off energy and accuracy of state detection. Figure 4.15 illustrates the differences in the kind of problems these different methods try to solve versus our work. While prior state-of-the-art works

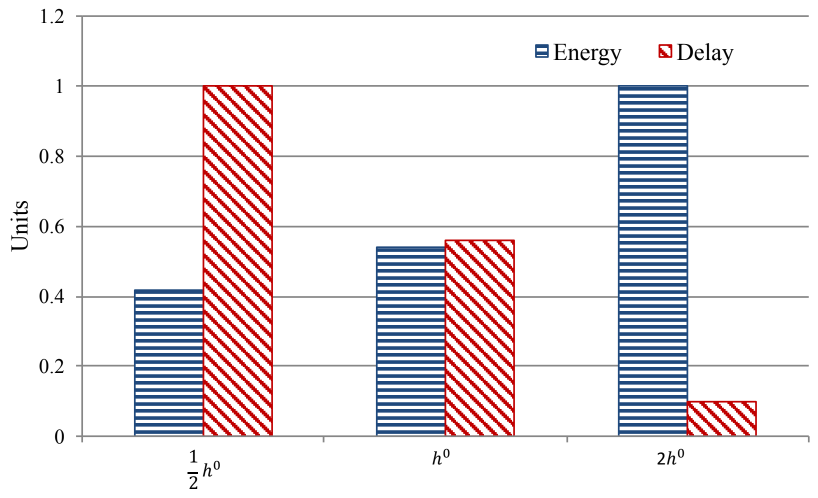
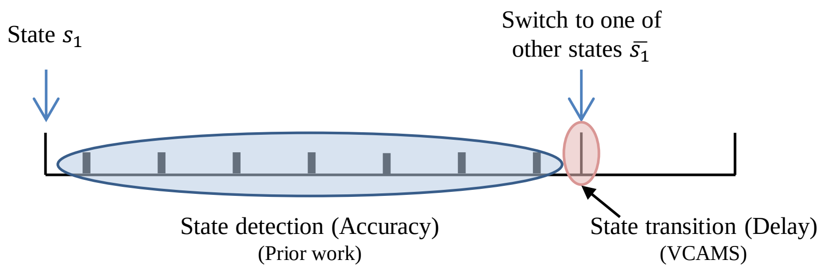


Figure 4.14: Energy and delay for different choices of decaying hazard rate h_j.



State s_1
Switch to one of other states $\bar{s_1}$
State detection (Accuracy)
(Prior work)
State transition (Delay)
(VCAMS)

Figure 4.15: Illustration of VCAMS's aim versus state-of-the-art methods.

focus on optimizing the accuracy of detecting the correct current state, we focus on optimizing the delay of transition detection between states. Previous methods focus on one state and recognize it accurately while minimizing the energy consumed, but can also be used to detect change in state. On the other hand, VCAMS aims at detecting with least delay the contextual change from one state to another while optimizing energy. To compare against these methods, we established continuous sensing as baseline for all. For the other methods, we reproduced their approaches with the aim of detecting state change by setting their parameters as follows:

- Wang et al. propose EEMSS [97] which uses fixed duty cycles; hence, it uses uniform periodic sensing. Their method uniformly skips a number of time units before triggering sensing. EEMSS triggers the accelerometer to sense 6 sec followed by 10 sec of sleep mode; thus, it is sensing 37.5% of the time.

- Lu et al. present Jigsaw [99] which is a continuous sensing engine. The authors use discrete-time Markov Decision Process (MDP) to learn the optimal GPS duty cycle. There are some constants which they did not specify in their work; hence, we assumed the reward adjustment coefficient

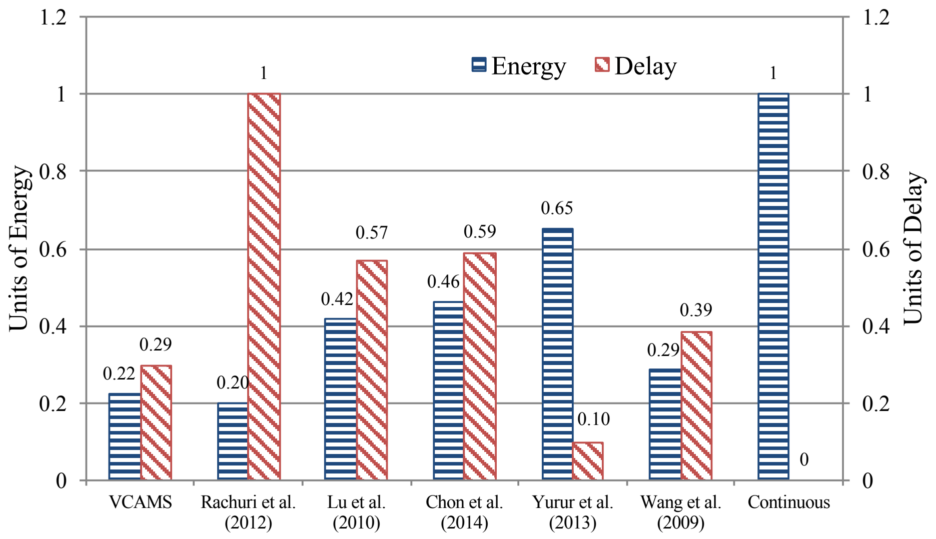


Figure 4.16: Energy and delay for VCAMS and state-of-the-art sensing schedules.

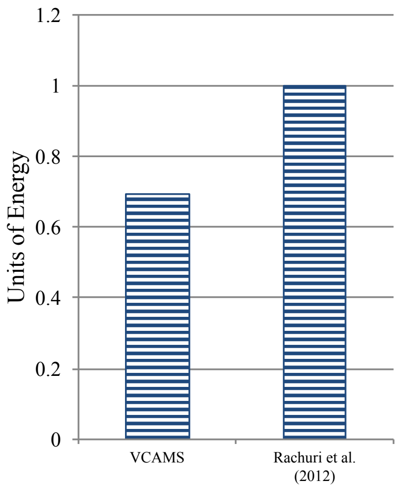
to be 2 and the penalty if the energy budget is depleted before the required duration to be -20 for comparison purposes.

- Chon et al. [123] present SmartDC which is a prediction-based scheme that porposes adaptive duty cycling to provide context while saving energy. The authors use order-1 Markov predictor to predict the time spent in a state; and they formulate the sensing decision problem as a Markov decision process. Through their experiments, they suggest setting the energy budget to 10%; thus, we used the same budget when reproducing their work.

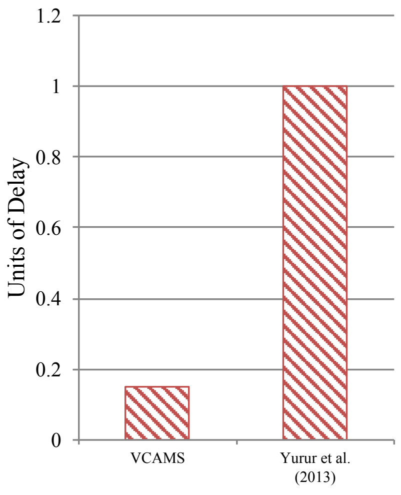
- Yurur et al. [100] propose a sensing mechanism which assigns adaptive duty cycles and sampling frequencies for accelerometer to infer human activity. They propose a novel additive increase additive decrease (AIAD) approach which decreases the duty cycle when the user's state is stable. The authors propose two methods to adjust the duty cycle; however, we compare VCAMS with the first method they propose since it outperforms the second method.

- As for Rachuri et al. [106], their system uses a learning technique to control the sampling rate of the sensors. They classify sensing actions into either success if detecting an unmissable event or failure if it is a missable event. The technique updates the probability of sensing based on the successes and failures. The probability in their work has an α_r as a weighting factor where energy and accuracy decrease as α_r increases. Through their experiments, they specify $\alpha_r = 0.5$; thus, we used the same value when reproducing their work.

We examined the average energy and delay values for each of the proposed



(a)



(b)

Figure 4.17: Comparison between VCAMS and best energy prior work for similar delay (a) and best delay prior work for similar energy (b).

methods. The simulation ran 10000 times for each method to find the average energy and delay, then the metrics were normalized. The largest energy of all methods was used as base for energy normalization, and the largest delay of all methods was used to normalize delays and obtain values between 0 and 1. There were three types of comparisons that were conducted against VCAMS: The baseline case of continuous sensing, previous methods by considering both energy and delay, and best of previous methods when considering energy alone or delay alone. The results are summarized in Figure 4.16. When comparing to continuous sensing, the delay is assumed to be zero for continuous sensing. VCAMS incurs delay, but has significant energy reduction of 78%.

When comparing to previous state-of-the-art methods, the results show that VCAMS gives a balance between energy and delay. It does not provide the lowest energy or the lowest delay of all methods; however, it provides the best trade-off of the two metrics (energy and delay) combined. Although Rachuri et al. (2012) system has energy less than VCAMS, it has the highest delay which is not practical for delay intolerant applications. On the other hand, the work proposed by Yurur et al. (2013) has the lowest delay, but it has higher energy since it applies continuous sensing when the time approaches $\hat{T_j}$. As for Wang et al. (2009), Lu et al. (2010) and Chon et al. (2014), they provide a trade-off between energy and delay; however, they still result in higher energy and delay than VCAMS.

To compare to best previous state-of-the-art methods considering energy or delay alone, VCAMS configuration had to be adjusted to set similar conditions. We used Rachuri et al. as reference for best energy and Yurur et al. as reference for best delay. To compare with prior method with best energy, VCAMS coefficients were set to $\alpha = 0.2$ and $\beta = 0.8$. This (α, β) combination gave similar delay performance of VCAMS compared to [106]; however, the results in Figure 4.17 show that VCAMS outperforms Rachuri's system since it minimizes the amount of energy consumed by 30%. To compare with prior method with best delay, VCAMS coefficients were set to $\alpha = 1$ and $\beta = 1$. This combination of weighting factors gave the maximum energy consumption of VCAMS which is comparable to energy consumption of Yurur's proposed method. The results show that VCAMS provides better delay performance than Yurur's solution as shown in Figure 4.17b. It's energy is comparable to Yurur's proposed method; however, it saves 75% of delay.

In summary, VCAMS outperforms all state-of-the-art methods when considering the combined trade-off of energy and delay. Furthermore, VCAMS can be configured (proper choice of α and β) to achieve lowest possible delay, lowest possible energy, or lowest possible trade-off combination.

Computational Power

To analyze the performance of our proposed approach on real systems, and to also illustrate the computational costs of VCAMS under realistic operational conditions, we implemented VCAMS on an Android-based device which is HTC Desire. The computational power consumed by the smartphone for executing the learning mode of VCAMS and extracting the optimal sensing schedule was measured by PowerTutor [144]. PowerTutor is an Android application that displays the power consumed by major system components: CPU, display, and network interfaces; and it displays the power consumed by each component separately. We ran VCAMS for the state "s_j = AtMeeting". The results summarized in Table 4.3 show that running VCAMS to derive the optimized sensing schedule consumes almost 0.8% of the total CPU. In addition, the table shows that the computational power is almost 5% of the sensing overhead for the cheapest sensor (accelerometer) which is rather negligible, and hence the computational energy can be ignored.

Table 4.3: Computational power

Application	Power (mW)
VCAMS	28
Accelerometer	551
System	3400

4.4 Case Study with Real Activity Context

Several experiments were conducted to test the efficiency of VCAMS and demonstrate its effectiveness in real-life situations beyond simulation. The effects of

Table 4.4: Time properties of activities

Activity	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
Mean (sec) μ_j	33700.8	191.7	403.5	506.3	118.5	6749.1	7134.6	2090.1
Standard deviation (sec) σ_j	2310.6	331.2	267.4	141.0	188.1	5725.8	5360.5	551.4

different system parameters including sampling interval δ_j were investigated. In addition, experiments were conducted to validate the performance of VCAMS and compare it with prior state-of-the-art methods which were described in subsection 4.3.8.

4.4.1 Setup

The real dataset was based on logs of Activities of Daily Living (ADL) [180]. This dataset was collected in smart homes for two users capturing the daily activities performed by each user for 35 days in their own home. The activities accompanied with their labels include sleeping (s_1), toileting (s_2), showering (s_3), having breakfast (s_4), grooming (s_5), spending spare time (s_6), leaving (s_7), and having lunch (s_8). Data was divided into two parts: 1) historical data which was used in the learning mode to obtain the time limit $\hat{T}_j$ of the user in each state/activity s_j and the probability distribution of the time spent in each state and 2) test data which was used in the execution mode to validate the efficiency of our proposed algorithm in detecting state transition. The derived Pareto-optimal values were $\alpha = 0.7$ and $\beta = 0.4$. As expected from Section 4.2.2, they turned out to be the same as the simulation case since if we change the interval for the same choices of α and β, we keep the same relative proportions of energy and delay. Therefore, there is no need to re-compute the Pareto optimal (α, β) combination again each time a state is encountered.

Table 4.4 represents the mean μ_j and the standard deviation σ_j of each state/activity s_j. As expected, for real life activities, the time properties differ between activities. Some last longer than others, for example sleeping's average time was 9 hours; whereas toileting has an average of 3 min. Some have less variations than others, which probably indicates routine scheduled activities versus ad-hoc activities.

4.4.2 Applicability of VCAMS on Real Data Traces

The statistics (μ_j and σ_j) of time spent in each state/activity depended on the behavior of the user in a specific context. To find the time limit $\hat{T}_j$ for each state s_j, we set μ_j and σ_j according to Table 4.4. The time limit $\hat{T}_j$ for each state s_j was set to the mean of the time spent in the activity plus one standard deviation.

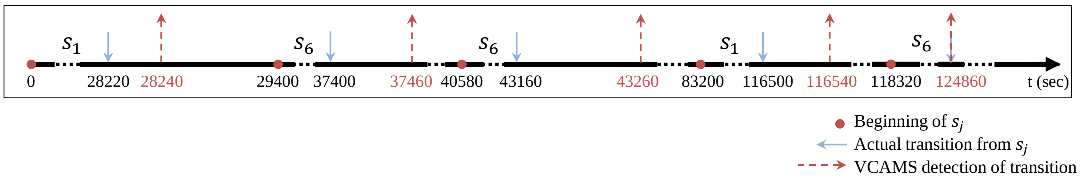


Figure 4.18: Dynamic real-time illustration of applying VCAMS to detecting transitions for s_1 and s_6 activities.

We chose to set the value of sampling interval $\delta_j = 20$ sec so that we use higher granularity than 1 min limited by the sensor requirements and speed of system. However, we also investigated the effect of using higher values of δ_j on dynamic real data to compare with simulation results.

The experiments aimed at determining in real-time the activity of the user, and then detecting changes in activities while optimizing delay and energy consumption. We implemented VCAMS for the 35 days data to monitor the user's activities and test the applicability of our method on dynamic real data. We ran the algorithm on 2.7 GHz Intel Core i5 processor machine. The delays in detecting transitions between states were measured by taking the difference between real transition time and time at which our system detected the transition.

Figure 4.18 shows part of the time line from the results of applying VCAMS's execution mode to two activities: sleeping (s_1) and spending spare time (s_6). We focused on these two activities since they have different behaviors. Sleeping has a very small standard deviation compared to its mean, while spending spare times has a standard deviation which is very high compared to is mean time as shown in Table 4.4. In addition, these two states can be easily illustrated and visualized on a time line. The dot on the time line indicates the beginning of a new activity. The arrow pointing downwards indicates the true transition from s_j and the arrow pointing upwards illustrates the time at which VCAMS detects the transition from s_j. The time difference between the two arrows is the delay in detection which we are trying to minimize while optimizing energy consumed. According to the results, s_1 has delays 20 and 40 sec while s_6 has delays of 60, 100 and 0 sec. However, delay on average will not exceed 1 min for these two states as shown in Figure 4.20. The average delay for sleeping is 15.7 sec which is 0.04% of the mean time of s_1. Whereas, the average delay of s_6 is 47 sec which is 0.7% of the mean time. Both delays are considered acceptable compared to their mean times. s_6 has a higher delay since it has more variability around the mean time as reflected by the standard deviation value. Although some prior methods such as continuous sensing and [100] will cause less delays, they will cause excessive amounts of unnecessary energy.

Figures 4.19, 4.20, and 4.21 demonstrate the energy consumed, the delay incurred and the computational time spent to compute each activity respectively. In addition, the figures show the variation of energy, delay and computational

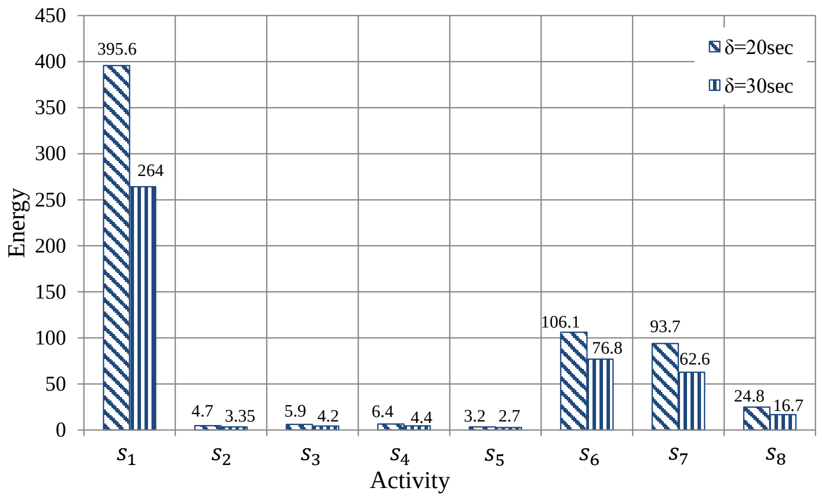


Figure 4.19: Energy for activities considering two different δs.

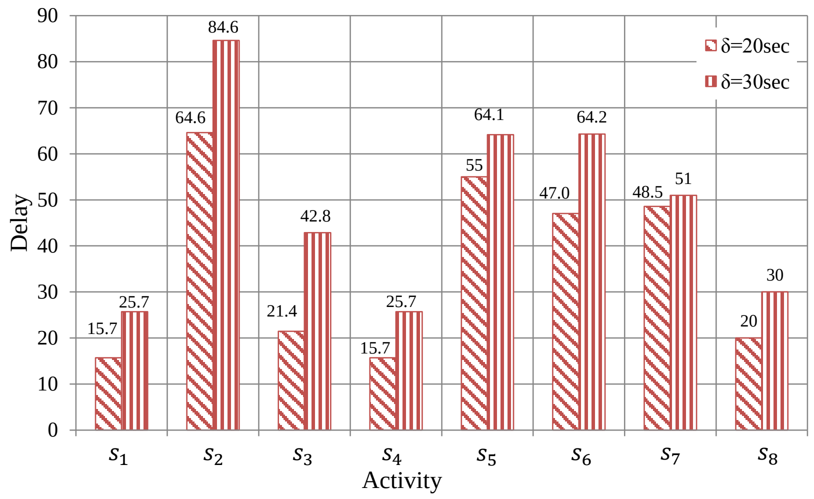


Figure 4.20: Delay for activities considering two different δs.

time versus δ_j. Figure 4.19 shows the number of times the sensor is triggered for the different activities which directly corresponds to the energy consumed since energy is represented in our proposed approach as the number of times sensing is triggered as described in Subsection 4.3.2. Energy increases as the mean of the activity increases. This is due to VCAMS sensing times increasing when the mean time μ_j increases. For instance, considering $\delta_j = 20$ sec, sleeping (s_1) has a mean time of 33700.8 sec and an energy consumption of 395.6 whereas toileting (s_2) has mean time of 191.7 sec and average energy of 4.7. Another observation is the considerable decrease in energy compared to continuous sensing. For example, the energy consumption for state s_1 when VCAMS is used is 395.6; whereas, the energy consumption when using continuous sensing for state s_1 would be 1685. Therefore, VCAMS saves up to 77% energy. Furthermore, increasing δ_j leads to

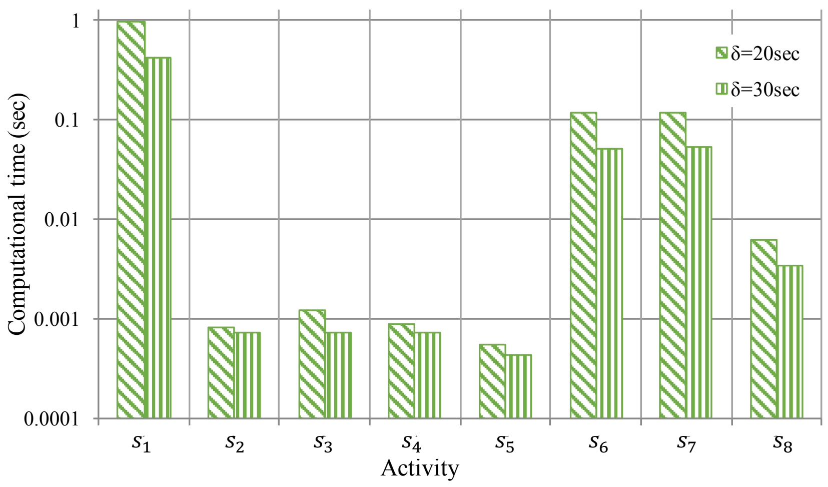


Figure 4.21: Computational time for activities considering two different δs.

less energy consumption in all activities as the sensor in this case has fewer time units at which it is triggered for a reading. This result confirms the results of subsection 4.3.5. Figure 4.20 presents delay versus activity for two values of δ_j. Delay varies between activities, and it reaches maximum of 84.6 sec for activity s_2 which has a standard deviation higher that its mean; hence, data is more dispersed. Delay decreases when minimizing δ_j as the intervals between sensing become smaller, with higher chances of early detection of state changes. Figure 4.21 presents the computational time spent in the learning mode of VCAMS for each activity separately. Computational time varies between activities, where activities with higher time limits require more computational times to obtain the best sensing schedule. In addition, increasing δ_j leads to less computational time since the number of total time instants N_j for state s_j decreases. These results validate Viterbi's complexity analysis presented in Subsection 4.2.5.

4.4.3 Comparison of VCAMS to Other Methods with Real Data Traces

To compare the performance of VCAMS with other existing approaches, we reran each prior state-of-the-art methods presented in subsection 4.3.8. For fairness, we fixed $\delta_j = 20$ sec for all approaches. The overall energy and delay results for activity s_1 are presented in Figure 4.22. Although we focused on one activity due to space limit, other activities have similar analysis when comparing VCAMS to other prior methods. Comparing VCAMS with continuous sensing shows almost 77% decrease in energy with a slight increase in delay from 0 sec to 15.7 sec which is negligible compared to the mean value of sleeping activity. Rachuri's approach results in 16% decrease in energy compared to VCAMS, but it causes 72% increase in delay. As for Wang et al. (2009), Lu et al. (2010) and Chon et

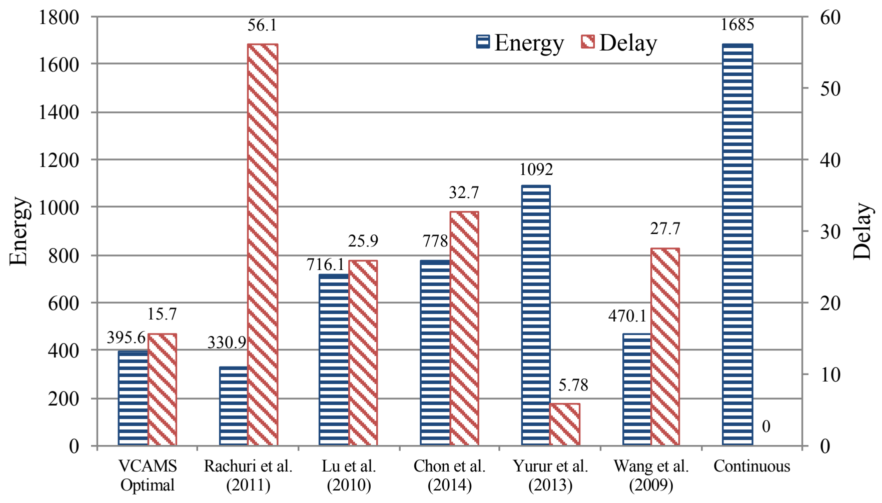


Figure 4.22: Energy and delay for optimal VCAMS and prior state-of-the-art sensing schedules for sleeping activity s_1.

al. (2014) approaches, they result in higher energy and delay than VCAMS. On the other hand, Yurur's approach gains on the delay side; however, it costs 63% more energy. In summary, when applied on real data, the results were consistent with simulations, and VCAMS outperformed other state-of-the-art methods.

4.4.4 Computational Complexity

To show the effectiveness of our proposed VCAMS trigger strategy (VT) described in Subsection 4.2.4, we considered different adaptive strategies and showed that VT is the best in terms of the trade-off between energy, delay and computational time. We considered the following strategies:

- WT "without any trigger": In this case, the learning mode is run offline only once to derive the sensing schedule, and no further adaptation is applied.
- CT "continuous trigger": In this case, the learning mode runs online each time a new state is encountered.
- ST "state-based trigger": For this case, the learning mode runs only during the longest state which can be obtained from analyzing the historical data. The system keeps track of all the behavioral changes encountered in all states but it does not trigger the learning mode except when the longest state is encountered. Since we are monitoring the user's activities during the day, the longest continuous state is sleeping which is s_1 during which most users charge their devices.
- VT "VCAMS proposed selective triggering": This case implements the rules proposed in Subsection 4.2.4 to dynamically decide when to trigger VCAMS's learning mode.

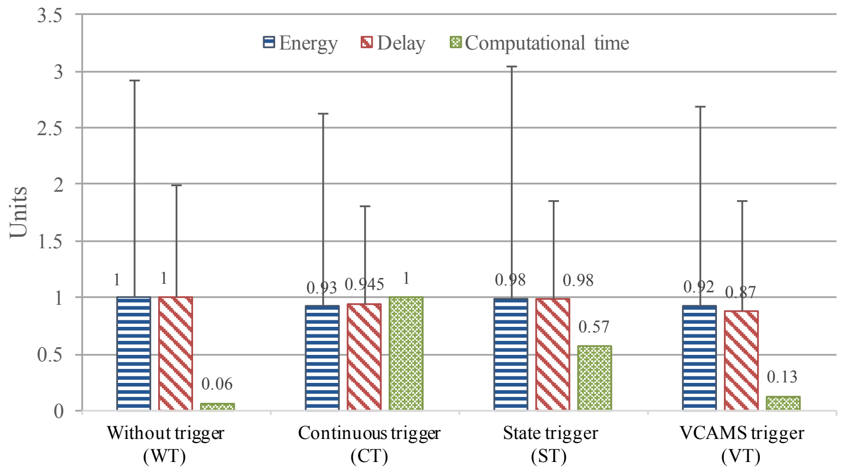


Figure 4.23: Energy, delay and computational time for VCAMS trigger strategy and other strategies.

The results are illustrated in Figure 4.23, which shows the normalized averages for sensing energy, delay, and computational time that were obtained for each of the aforementioned strategies. Our goal is to minimize energy and delay while avoiding excessive computational complexity. When comparing VT with other adaptive strategies, the results show that VT gives the best performance in terms of sensing energy consumed and delay in detecting a state change. The figure shows that VT causes less energy and delay than CT because CT adapts the sensing schedule continuously each time a state is encountered. For example, while CT consumes 0.93 units of energy, VT consumes 0.92 which is slightly less. On the other hand, VT causes 0.87 units of delay while CT causes 0.945; therefore, CT updates the sensing schedule continuously causing sometimes higher time limit $\hat{T}_j$ for state s_j; thus, causing more delay in the upcoming encounters of this state. Regarding the computational time, VT requires slightly more time than the WT strategy and much less time than ST and CT strategies. VT saves 87% from the computational time when compared to CT by avoiding unnecessary computations.

4.5 Summary

In this chapter, we have presented VCAMS: a Viterbi-based Context Aware Mobile Sensing algorithm to trade-off energy consumption and delay when detecting any contextual state change. The method includes a system model for user context and phone sensor parameters. The algorithm chooses an optimized sensor scheduling which maximizes a set of rewards according to Viterbi algorithm. The system has two modes: learning mode and execution mode. In the learning mode, the sensing schedules are initialized offline using a Vietrbi algorithm. The learning mode is also triggered when VCAMS has to update the sensing schedules when significant behavioral changes are observed. In the execution mode, the already learned sensing schedules are used to decide on sensing triggers. We experimentally validated the proposed system models, and showed that our proposed strategy outperforms all state-of-the-art methods when considering the combined trade-off of energy and delay. Results showed that VCAMS saves up to 78% energy when compared to continuous sensing.

Chapter 5

Sensor Selection: Optimized Sensor Selection for Multi-Context Aware Applications with an Ontology for Recognition Models

A growing recent research trend among smartphone applications is to boost the context recognition accuracy by extracting features from different sensors. For instance, in [181], the authors use the accelerometer and a low power ECG to maximize the accuracy of activity recognition. In [29], the authors consider multiple context of activity detection and abnormal ECG pattern; hence, they combine the ECG with accelerometer and gyroscope. Moreover, a plethora of recent research works compare the performance of context recognition when using multiple sensors versus using a single sensor. In [136], the authors compare the performance of having multiple sensors to detect an elderly fall. For example, they compare the performance of microphone and floor sensors and show that the combination gives more accuracy. In [137], the authors show that combining EMG with accelerometer to detect dyskinesia gives better performance than using only accelerometer. Therefore, multi-modal sensor fusion is gaining more attention since it facilitates sensing multiple complementary user properties.

Many phone applications aim at continuously extracting the user's context from sensors' data which places additional burden on energy consumption for devices already constrained on power. Therefore, one key challenge is to minimize the energy consumed by the context-aware application. This problem is expanded when multiple context-aware applications are running concurrently requesting data from multiple sensors to recognize different contexts. Running all these applications independently and without coordination leads to redundant data collection causing unnecessary excess sensing energy. Hence, there is an opportunity for significant energy reduction by having the applications share scarce resources efficiently while still retrieving their requested contexts accurately. Several options can be considered as contexts can be retrieved via different multi-modal embedded and wearable sensor combinations [28]. However, current applications typically utilize modalities that provide highest accuracies, and do not consider alternative sensors with lower energy options. In addition, these applications do not have access to what other applications are concurrently running for potential sharing of data and resources.

To optimize resource usages, the applied methods in relevant prior works use stochastic approaches to select duty cycles which decide when to trigger sensing, adaptive sampling periods and sampling frequencies to decide how long sensing data should be collected, or sensor selection and hierarchical sensing mechanisms. However, most of these proposed approaches target a single context. In this work, we aim at exploring the synergy across simultaneous operation of context applications to minimize the energy consumption by selecting a group of sensors that provides most efficient data and resource sharing among the different applications and without compromising the applications' minimum acceptable accuracies and delays. We introduce EGO – an energy-efficient ontology-based group sensor selection framework that optimally selects a group of sensors exploiting synergy among applications while jointly optimizing the energy-accuracy trade-off of all requesting applications. The contributions of the work include:

- An ontology that captures context recognition models along with descriptive specifications for each model such as required sensors and machine learning parameters. The ontology enables the framework to generalize and be applicable to any context already captured in the ontology. This ontology is used by the framework to examine the different sensor options for each application. It also servers as a rich resource for research in the field and facilitates knowledge sharing for context-aware applications.

- An adaptive sensor selection approach to select the optimal group of sensors that trades-off energy consumption and accuracy of context recognition. The selected group of sensors is adaptively changed based on what contexts are requested by the applications, which sensors are available, what the user state is, and how much energy is available on the smartphone and on external sensors.

- A comprehensive integrated approach that comprehends efficient methods for scheduling sensor data collections, and produces an optimized synchronized sensing schedule which decides when to trigger each of the selected sensors for data collection to trade-off the energy consumption and the delay to detect a context change. The framework uses one of our previously proposed sensor scheduling mechanism VCAMS [33] which is a Viterbi-based Context Aware Mobile Sensing mechanism.

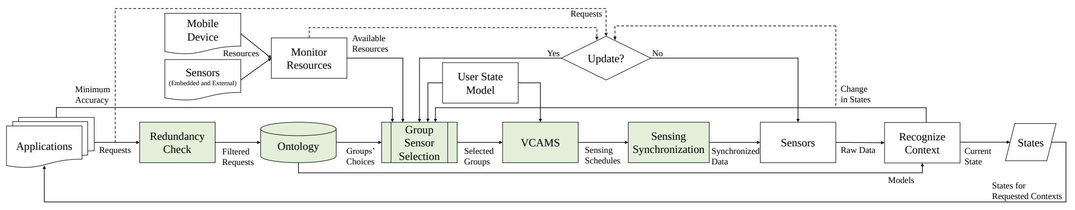


Figure 5.1: The flow diagram of the optimized selection and operation of sensors for multi-context applications. The applications send their context requests upon which sensor choices are generated from the ontology and optimized sensors are selected to recognize the contexts.

- A first of its kind real test-bed implementation that shows simultaneous operation of multiple context-aware applications, and demonstrate the effectiveness of the proposed framework in coordinating efficient operation of the applications. We implement 8 context-aware applications comprehending activity [182, 99, 183, 181] and emotion [82, 184, 185, 44].

This chapter is organized as follows. We present the proposed ontology-based framework in Section 5.1. Sections 5.2 and 5.3 present the results obtained by simulations and real case study respectively. Finally, Section 5.4 provides concluding remarks.

5.1 Proposed Framework

In this section, we provide the details of the proposed framework. The overview of the framework and its flow are described in Section 5.1.1. In Section 5.1.2, the details of the ontology for context recognition models are covered. Section 5.1.3 describes the system model and algorithm for sensor selection. In Section 5.1.4, we describe how sensor selection can be integrated with sensor scheduling.

5.1.1 Framework Overview

Figure 5.1 shows the typical task execution for context-aware applications to efficiently select the group of sensors and ultimately recognize the targeted contexts. The applications start by sending their new requested contexts to the *Redundancy Check* component which checks for redundancy with previously submitted requests from other applications. Collecting the same data redundantly can cause extra energy consumption caused by unnecessary sensing and processing. The applications also provide the minimum acceptable classification accuracy. For each of the requested contexts, the *Ontology* (Section 5.1.2) searches its database for the relevant groups of sensors which become alternatives for our sensor selection algorithm.

To monitor the available resources, the *Mobile Device's Resources* block keeps track of the available resources on the mobile device such as remaining energy and available bandwidth. Similarly, the *Sensors' Resources* block monitors the available resources on the external sensors. Moreover, our sensor selection problem is state-based; therefore, the current state of the user is either recognized using context recognition or estimated using a *User State Model* which is based on historical collected data reflecting the behavior of the user.

Given the available resources, the requested accuracy, the groups choices for each context, and the current state of the user, the *Group Sensor Selection* (Section 5.1.3) block selects the best group of sensors to recognize each requested context while minimizing the energy consumption by exploiting the common sensors between the different available groups. The groups selection changes adaptively based on the available sensors, requested contexts, available resources, and current state. Hence, the different dynamic conditions are forwarded periodically to the *Group Sensor Selection* block.

The selected groups of sensors are forwarded to the synchronization step (Section 5.1.4) which integrates with an optimized dynamic sensor scheduling mechanism, such as previously proposed *VCAMS* [33] that adaptively trades off the energy and the delay to detect a state change. The optimized sensing schedules are forwarded into the *Sensing Synchronization* block which synchronizes the data collection from the different triggered sensors resulting in more energy saving. The employed sensors are then triggered based on the synchronized sensing schedules to extract raw data. This data feeds the *Context Recognition* step which extracts the suitable features from each sensor raw data and uses the appropriate classification model to classify the state of each context. The feature extraction method and the classification model are retrieved from the *Ontology Repository.* The recognized states are forwarded back to the requesting applications.

The framework components described in the flow result in the framework architecture shown in Figure 5.2. The framework operates in real time, and is composed of the following key components:

- *Group Sensor Selection* will be described in details in Section 5.1.3.
- *Ontology Repository* will be covered in Section 5.1.2.
- *Sensing Synchronizer* will be discussed in Section 5.1.4.

The other components of the framework serve as sources that provide complementary information to the key framework components. For example, the *System Monitor* component keeps track of the remaining resources on the mobile device and on wearable sensors. It provides this information to the sensor selection block which adapts its selection based on the remaining resources.

User State Model

The user state model is used to estimate the state of the user at times when the actual context recognition is not available based on how frequent the sensor

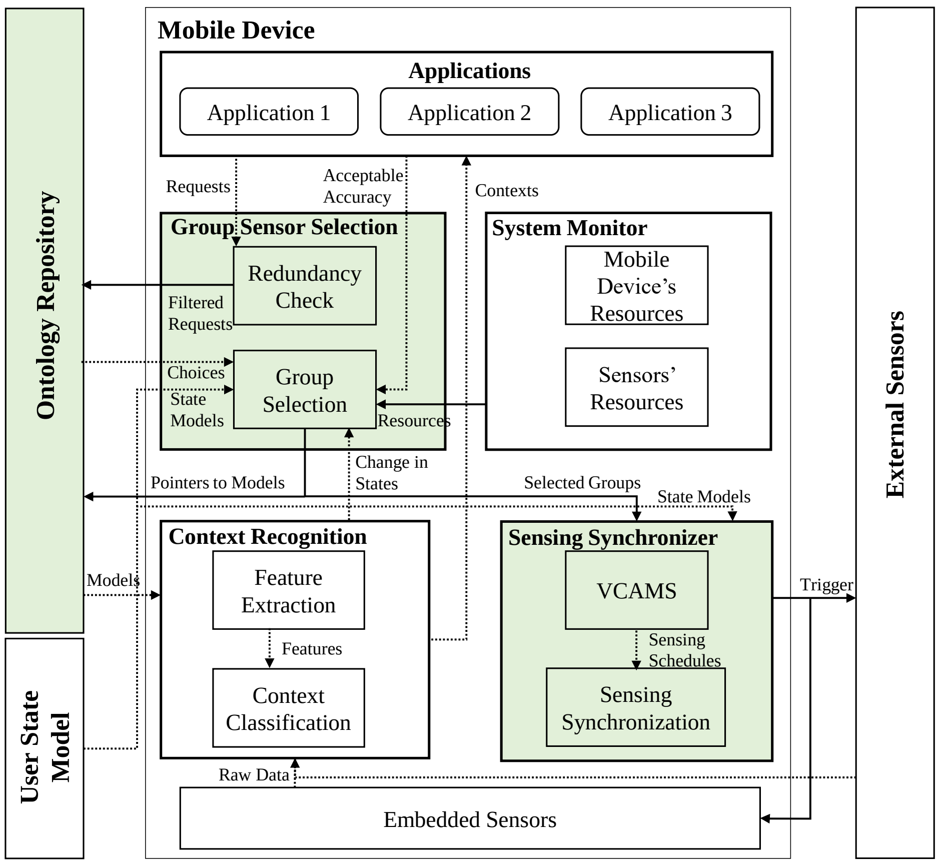


Figure 5.2: The general framework architecture of the proposed method. The colored blocks represent the core components of our framework. The ontology is the knowledge repository used by the framework to examine the different sensor options for each application. The sensor selection block selects the optimal group of sensors that trades-off the energy consumption and the accuracy of context recognition. The synchronization manager outputs an optimized synchronized sensing schedule which decides when to trigger each of the selected sensors for data collection.

is triggering. Therefore, a statistical user state model is built based on prior historical knowledge of human behavior pattern. The user states are contextual inferences that refer to a specific requested context where each context can have several states; for example, the location of the user may be at home, at work, or in mall. Thereby, we propose a statistical user state model similar to what we used in our previous work [33].

The user state model keeps track of the time-variant user contextual behavior to recognize or predict the most likely user current behavior. Furthermore, our proposed user state model tracks the transitions between states and depends on prior knowledge of most recent transition patterns. In general, a user contextual behavior changes in time; therefore, our proposed user state model also predicts when the user may change the current state and the most probable next state to which she will transit. The user state model captures a transition model from historical behavior of the user such as the probabilities of transition from state to another at any time instant, and can also include the statistics of the time duration spent in a specific state. The proposed sensor selection and sensor trigger are modified and updated based on the proposed user state model that takes the state into account, and as a result the system is adaptive to contextual state of the user.

System Adaptability

Our framework dynamically adapts its sensor selection plan. Adaptation as shown in the flow of Figure 5.1 is triggered when:

1. New sensors are available or some sensors have withdrawn due to battery discharge. This depends on what time during the day the user wears a specific sensor.
2. New applications request new context in the background or a previously requested context is no longer needed.
3. User contextual state has changed within one of the monitored concurrent contexts.

5.1.2 Ontology for Context Recognition Models

An ontology is a knowledge repository that uses semantic information to define attributes of a particular field along with relationships between these attributes [186]. There are several advantages for using ontology-based schemes: 1) it forms a gateway between humans' knowledge and computers' representation [187], 2) it facilitates knowledge sharing among researchers [112] and 3) it provides a formal and well defined structure to define entities. Therefore, ontology systematizes information retrieval and paves the way for automating the context recognition process. For example, the authors in [188] represent human behavior ontologies whereas the authors in [189] describe semantic smart homes ontologies. In [190], context modeling is represented as two levels of ontologies, the upper level represents context entities such as person, device or network, while the lower level represents the domain-dependent entities such as home, car or campus. In [191], the authors model human activities based on ontologies using finite state machines. Nevertheless, existing works on ontology-based context-aware systems focus on one context and propose the different relations between attributes that would allow inferring the current context of the user such as the current activity. On the other hand, our ontology captures the mechanisms for context recognition, including the different recognition models for each context, what sensors they use, what data or features they need, and the parameters of the prediction model.

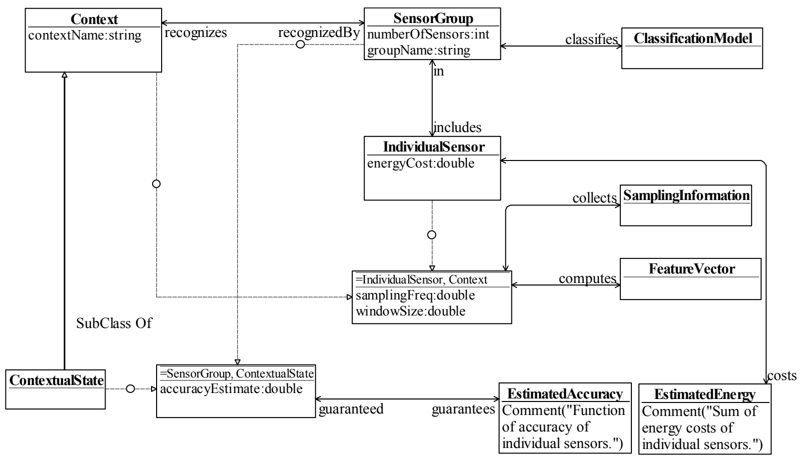


Figure 5.3: The core of the ontology for context recognition models.

Design of the Ontology for Context Recognition

Our ontology uses semantics to represent the attributes relevant to a certain context recognition such as what sensors to operate, what sampling frequencies to use, what features to extract from raw data, and what classification models to consider. To develop the ontology, we have followed the guidelines to build an ontology from Noy and McGuinness, presented in [192]. The main classes and their accompanying properties of the ontology are graphically illustrated in Figure 5.3 by means of Unified Modeling Language (UML) class diagrams [193]. We used OWLGrEd, an online tool to visualize OWL ontologies [194], to draw this UML diagram. The classes of the ontology are *Context, ContextualState, SensorGroup, IndividualSensor, EstimatedAccuracy, EstimatedEnergy, SamplingInformation, FeatureVector* and *ClassificationModel*.

- The class *Context* defines the requested context by the application such as activity, emotion or location. Each context takes contextual state as its value. For example, for emotion context, the state can be either angry, happy or neutral. Therefore, the class *ContextualState* is a subclass of context.

- The class *SensorGroup* defines the groups of sensors that can be used to recognize a certain context. This class is identified by a group name and the corresponding number of sensors.

- The class *IndividualSensor* describes each sensor in terms of energy cost and sampling information.

- The classes *EstimatedAccuracy* and *EstimatedEnergy* define respectively the accuracy and energy for each group of sensors in terms of the individual sensors' accuracy and energy cost.

- The classes *SamplingInformation*, *FeatureVector*, and *ClassificationModel* capture how the data should be sampled from the sensor, what features to extract, and which classification algorithm to use to extract the requested context.

The different instances of these classes are connected via properties that define the relations.

- Each requested context can be *recognizedBy* some sensor groups from which we will select the optimized group of sensors.
- Each group of sensors *includes* a set of individual sensors.
- For each combination of individual sensor and requested context, sampling information is required to *collect* the sensor data and a set of features is *computed* to extract relevant data patterns.
- The *guaranteed* estimated accuracy depends on which group of sensors is triggered and on the current contextual state.

The ontology is queried twice each time the sensors have to be selected. The first query is to get the potential groups of sensors for each context in addition to the estimated energy and classification accuracy for each group. The second query to the ontology takes as input the optimal group of sensors for each requested context selected by the sensor selection algorithm and retrieves from the ontology the feature extraction methods and models for each sensor.

To make sure the data sources for the ontology are reliable, we only allow tested and validated data for recognition models. This is ensured by limiting the source of the ontology to published and validated recognition models, such as [112, 195]. Our designed ontology provides a common data model and an interactive platform for data sharing between researchers in the context awareness field. Table 5.1 shows a sample of the ontology for two contexts: activity and emotion. For each published reference, we capture the deployed sensors, the sampling information for each sensor, the features extracted from raw data, the classification algorithm used, and the contextual states recognized. Therefore, each row of the table represents one entry for the ontology. The columns represent the classes of our context ontology. The full ontology will be made publicly available for sharing and expansion.

5.1.3 Group Sensor Selection

Another important component of our proposed framework is the *Group Sensor Selection* block which selects the optimized group of sensors. The sensor selection mechanism considers the choices provided by the ontology repository to select the appropriate sensors based on the current user state, the available resources, and the requested context recognition accuracies by running applications. The group of sensors is selected to minimize the energy consumption while recognizing the multiple concurrent contexts. This is achieved by exploiting the advantage of having common sensors among different contexts. The system model and parameters for sensor selection are presented in Section 5.1.3. Then, the sensor selection problem in our proposed framework is formulated.

System Model and Parameters

We consider several applications which are requesting different L contexts c_l where $l = 1, 2, ..., L$. For each requested context c_l, consider the ontology section Ω_l which captures the set of all possible sensor groups available to detect context c_l along with their characteristics such as the energy consumed, the accuracy guaranteed, the feature extraction method, and the classification model used. Let K_l be the number of possible sensor groups that detect context c_l. The set of groups for context c_l are denoted by $\mathcal{G}^l_{k_l}$ where $k_l = 1, 2, ..., K_l$. Let M be the total number of available sensors, and let S_m represent a specific sensor where $m = 1, 2, ..., M$. The parameters used in our sensor selection framework are listed in Table 5.2.

Problem Statement: Given multiple potential groups of sensors to recognize each of the requested contexts, the sensor selection problem is to find the group of sensors for each context such that their total energy cost is minimized and their classification model accuracy is above a pre-specified acceptable threshold. The method exploits the advantage of having common sensors among the groups across different contexts.

Given multiple L requested contexts c_l and given the ontology of each context and the corresponding sensor groups, the objective is to identify the best combination of groups that can minimize energy consumption while having constraints on the minimum acceptable accuracy. Let $y^l_{k_l}$ be a decision binary variable which determines whether or not a group of sensors $\mathcal{G}^l_{k_l}$ is chosen to recognize context c_l in the final selected union of groups:

$$y^l_{k_l} = \begin{cases} 1, & \text{if group } \mathcal{G}^l_{k_l} \text{ is chosen} \\ 0, & \text{otherwise.} \end{cases} \tag{5.1}$$

Table 5.1: Sample of the ontology showing samples of selected papers that detect activity and emotion

Context	**Reference**	**Sensors**	**Sampling Information**	**Features**	**Classification Algorithms**	**Recognized States**
Activity	[182]	ACC-E, GPS-E, ACC-W	16Hz with 1 sec interval	Means, variances, correlations, kurtosis, and other statistical measures	MLR	Brushing teeth, hiking, riding bicycle, jogging, standing, strolling, walking downstairs, walking upstairs, and writing on blackboard
	[99]	ACC-E, MIC-E, GPS-E	32Hz sampling rate for the accelerometer; and 8kHz, 16-bit, mono audio for the microphone	Energy, mean, variance and spectral	SVM, NB and GMM	Stationary, walking, cycling, running, and vehicle. With Microphone: brushing teeth, shower, typing, vacuuming, washing hands, crowd noise, and street noise
	[183]	ACC-E, GYR-E, MAG-E	5Hz sampling rate with 3 sec interval	Mean, variance, zero crossing rate, correlation between acceleration and gyroscope, FFT energy and entropy	J48 DT	Slow walking, normal walking, rush walking, running, standing, and sitting
	[181]	ACC-W, ECG-W	50Hz with 5 sec interval for accelerometer; and 200Hz with 20 sec interval for ECG	Mean, median, energy and variance	RVM	Lying, sitting, standing, walking, walking upstairs, walking downstairs and running
Emotion	[82]	ACC-E	Normal sampling rate	The x-value, y-value and z-value of the accelerometer	J48 DT	Stressed, neutral and excited
	[184]	ACC-E, MIC-E, GPS-E	100Hz sensing for the accelerometer which triggers the GPS when necessary	Latitude, longitude, communication frequency, mean, energy, frequency-domain entropy, and correlation	Factor bipartite graph	Very pleasant, pleasant, medium, unpleasant and very unpleasant
	[185]	ACC-W	5Hz followed by a moving average filter	Skewness, kurtosis, standard deviation, correlations between every two axes, Power Spectral Density (PSD), and FFT	DT, SVM, RF, and RT	Neutral, happy, and angry
	[44]	EEG-W, ECG-W	Continuous sensing	KDE and MFCC	MLP	Calm, happy, fear and sad

- ACC: Accelerometer; MIC: Microphone; MAG: Magnetometer; GYR: Gyroscope; ECG: Electrocardiography; EEG: Electroencephalogram
- E: Embedded sensor in smartphone; W: Wearable external sensor
- MLR: Multiclass Logistic Regression; SVM: Support Vector Machine; DT: Decision Tree; GMM: Gaussian Markov Model; NB: Naive Bayes; RVM: Relevance Vector Machine; RF: Random Forest; RT: Random Tree; KDE: Kernel Density Estimation; MFCC: Mel-Frequency Cepstral Coefficients; MLP: Multi-Layer Perceptron

Table 5.2: Table of parameters

Groups	Notation	Description
Sets	L	Set of contexts
	K_l	Set of possible groups for context l
	M	Set of available sensors S_m
	J_l	Set of total states in context c_l
Parameters	c_l	Particular context out of L total requested contexts
	${s_j}^l$	Particular state in context c_l
	$\mathcal{G}^l_{k_l}$	Group of sensors that can detect context c_l
Optimization values	$E^l_{k_l}$	Estimated energy consumed by group $\mathcal{G}^l_{k_l}$
	$A^l_{k_l}$	Accuracy that group $\mathcal{G}^l_{k_l}$ can guarantee for state s^l_j
	E_m	Energy consumption by each sensor S_m
	$A_{m,j}$	Accuracy of sensor S_m in recognizing state s_j
Variables	$y^l_{k_l}$	Binary to determine whether group $\mathcal{G}^l_{k_l}$ is chosen

The energy consumed by a group of sensors $\mathcal{G}^l_{k_l}$ can be computed by summing the individual energies of the sensors that form that group:

$$E^l_{k_l} = \sum_{S_m \in \mathcal{G}^l_{k_l}} E_m \tag{5.2}$$

where $E^l_{k_l}$ is the total consumed energy by group $\mathcal{G}^l_{k_l}$ and E_m is the individual energy of sensor S_m.

Mathematical Formulation

We formulate the problem as a 0-1 integer program where the decision variables are taken to be $y^l_{k_l} = 0$ or 1. The optimized selected group of sensors depends on the state of the user where for each context c_l, there is a set of possible J_l states. For example, for activity context, the state can be walking, sleeping, jogging, etc. Let s^l_j denote the state in context c_l where $j = 1, 2, ..., J_l$. The accuracy of a group $\mathcal{G}^l_{k_l}$ in recognizing a certain state s^l_j is denoted by $A^l_{k_l,j}$. This accuracy is a function of the accuracy of the individual sensors in that group $A_{m,j}$ which represents the contribution of sensor S_m to the accuracy of detecting state s^l_j.

The sensor selection problem can be formulated as follows:
Objective:

$$\min_{\substack{k_l=1,2,\ldots,K_l \\ \forall l \in L}} \sum_{\substack{k_l \in K_l \\ \forall l \in L}} \Big(\sum_{S_m \in (\bigcup_{l\in L} \mathcal{G}^l_{k_l})} E_m \Big) \Big(\prod_{l \in L} y^l_{k_l} \Big) \tag{5.3}$$

Subject to:

$$\sum_{k_l \in K_l} y^l_{k_l} = 1 \qquad \forall l \in L \tag{5.4}$$

$$\sum_{k_l \in K_l} A^l_{k_l,j} y^l_{k_l} \geq A_l \qquad \forall l \in L \tag{5.5}$$

$$E^l_{k_l} \leq E_B \qquad \forall k_l \in K_l, \forall l \in L \tag{5.6}$$

$$y^l_{k_l} \in \{0,1\} \qquad \forall k_l \in K_l, \forall l \in L \tag{5.7}$$

The objective function in (5.3) is to exploit the synergy between the selected groups of sensors by minimizing the total energy cost consumed by the union of sensors in the selected groups. The constraint (5.4) states that exactly one group for each context is to be chosen and the constraint (5.5) guarantees that the minimum acceptable accuracy is attained for each of the L requested contexts by applications. The constraint (5.6) enforces an energy budget E_B on the selected groups and the constraint (5.7) ensures that the variable $y^l_{k_l}$, which decides whether group $\mathcal{G}^l_{k_l}$ is selected, takes binary values.

Group Sensor Selection Proposed Approach

The group sensor selection problem is a variation of the 0-1 knapsack problem [196], which is an NP-hard optimization problem. The optimal solution of 0-1 knapsack problem can be calculated by using brute force algorithm which enumerates all possible combinations and picks the one with the best objective value. The number of possible combinations of groups between the different requested contexts is of order (V), where:

$$V = \prod_{l=1}^{L} K_l \tag{5.8}$$

For each possible combination, the choice of sensors is the union from the groups of sensors to detect the different contexts. There are V possible combinations of sensor groups; therefore, the complexity of brute force algorithm is $O(V)$. To decrease the complexity, we propose an algorithm that filters the possible combinations of sensors groups.

Algorithm 5.1 shows the pseudocode for this proposed solution. The algorithm takes as input the choices of groups of sensors $\mathcal{G}^l_{k_l}$ for each of the requested L contexts. It extracts these choices from the ontology. In addition, the algorithm takes as input the accuracy constraints that define the minimum acceptable accuracy by each application. The basic structure of the algorithm has two main loops. The first loop (line 3) filters the available choices of groups into the set of feasible groups that satisfy the accuracy constraints. In the second loop, the energy consumption of the different combinations of the remaining feasible groups

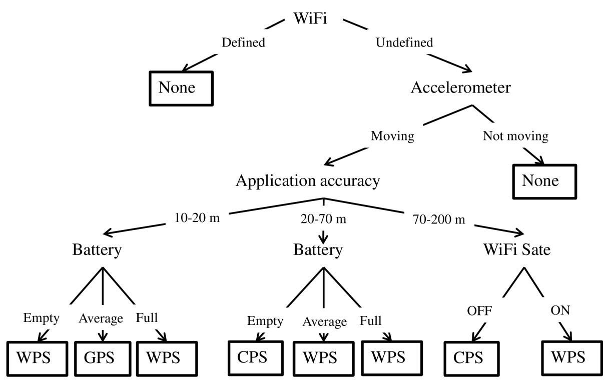


Figure 5.4: Example of single-context setting: location sensor selection.

is computed (line 12) as shown in the utility function (5.3). Those possible combinations are checked whether they consume the minimal energy. Finally, the one combination with the lowest energy consumption is chosen (line 14) such that its energy does not exceed the defined energy budget. Using this algorithm, the complexity decreases to $O(V')$ where $V' \leq V$.

Alternative Sensor Selection in Single Context Setting

For sensor selection in single context case, the method considers the available sensors that can provide the required context; then according to required accuracy and available energy, the algorithm chooses the most effective sensor to use. We develop an energy based algorithm that detects the user context information to achieve a minimum accuracy level. We consider different factors that impact the choice of sensors such as available energy, required accuracy, state of the device (what sensors are already turned on), and state of the user (moving or stationary). A prototype for the algorithm is presented in Algorithm 5.2.

As an example, applying our proposed algorithm on location sensor selection model saves a critical amount of energy. Though GPS provides the highest accuracy between the location sensors available, it is shown through experiments that it is a major source of energy consumption. Figure 5.4 presents the location sensor selection decision tree which was built using decision tree classification technique.

Algorithm 5.1 Group sensor selection algorithm

Input:
- Ω_l: the choices of groups of sensors $\mathcal{G}^l_{k_l}$ for each of the requested L contexts
- A_l: the minimum acceptable accuracy for each context
- E_m: the energy consumption by each sensor S_m
- s^l_j: the current states of the user $\forall l = 1, 2, ..., L$
- E_B: available energy budget

Output:
${\mathcal{G}^l_{k_l}}^*$: Optimized groups of sensors $\forall l = 1, 2, ..., L$

```
1: set Minimum energy $E_\text{min} = \infty$
2: set Feasible set $\Omega'_l = \emptyset$
3: for each requested context $l = 1$ to $L$ do
4:   for each group $k_l = 1$ to $K_l$ do
5:     if the accuracy of group $\mathcal{G}^l_{k_l}$ satisfies the minimum acceptable
       accuracy: $A^l_{k_l,j} \geq A_l$
6:       Add the group $\mathcal{G}^l_{k_l}$ in the set of feasible groups that satisfy the
         accuracy constraint: $\Omega'_l = \Omega'_l \cup \mathcal{G}^l_{k_l}$
7:     end if
8:   end for
9:   The feasible set $\Omega'_l$ for each context has $K'_l$ groups where $K'_l \leq K_l$
10: end for
11: Compute the number of possible combinations of groups as
    $V' = \prod_{l=1}^{L} K'_l$ which represents the union of sensors
12: for each combination $v = 1$ to $V'$ do
13:   Compute the energy consumption $E_v = \sum_{S_m \in (\bigcup_{l \in L} \mathcal{G}^l_{k_l})} E_m$
14:   if $(E_v - E_\text{min}) < 0$ && $E_v - E_\text{B} < 0$
15:     $E_\text{min} = E_v$
16:     for each requested context $l = 1$ to $L$
17:       Optimized group of sensors = ${\mathcal{G}^l_{k_l}}^* \in v$ //
18:     end for
19:   end if
20: end for
```

5.1.4 Integration and Synchronization with Optimized Sensor Scheduler

Once the framework produces the optimized group of sensors, the next step for the framework is to determine the sensing schedule for each sensor. A naive

Algorithm 5.2 Selecting the optimized sensor

```
Input:
- $S_m$ (available sensors on device)
- $A$ (accuracy required by application)
- $c$ (type of context required by application)
Output:
Sensor $S_{\text{opt}}$

1: Find the type of context that can be returned by each sensor. N.B
   some sensors can return more than one type of context.
2: Divide the sensors according to the above classification into K categories.
3: Sort the sensors in each category in descending order of efficiency.
4: Find the category to which c belongs.
5: for all sorted sensors in this category:
   check if $A_m > A$ then choose sensor $S_m$
   else move to the next sensor $S_{m+1}$
```

choice is to use the continuous sensing where a sensor is triggered continuously so that the system does not miss the event of the user changing her state. However, continuous sensing constitutes a major source of energy consumption. Therefore, we propose to use an optimized scheduling approach based on VCAMS (Viterbi-based Context Aware Mobile Sensing) from our previous work [33]. Combining VCAMS with sensor selection provides further energy saving and a trade-off with delay and accuracy.

VCAMS is a context aware system that depends on the situational information about the user and her surroundings. For each requested context, there is a corresponding user state model which describes the states of the user described in Section 5.1.1. For example, if the requested context is emotion, the states may be either neutral, happy or angry. VCAMS dynamically provides the time instants at which a sensor needs to be triggered for the specific user and the particular state. However, VCAMS does not synchronize the sensing triggers among the different selected sensors. The sensing schedules for the different sensors are time synchronized so that the samples from all sensors can be used collectively as features for recognition. Hence, we use the *Sensing Synchronization* block to effectively combine the sensing schedules generated by VCAMS. We propose a hierarchical approach in which the framework first selects the optimized group of sensors and then synchronizes the VCAMS-derived sensing schedule for all sensors.

Let $\hat{T}_j^l$ be the estimated time limit for context c_l; specifically, for state s_j^l,

Algorithm 5.3 Synchronizing the sensing schedules of multiple contexts

Input:
- The sensing schedule ss^l_j generated by VCAMS for each of the requested contexts c_l and particular state s^l_j
- The range threshold th_r to identify the maximum time range between consecutive sensor triggers
- The probability threshold th_p to identify the minimum survival probability beyond which sensing is triggered

Output:
Synchronized sensing schedule

```
1:  set $i = 1$
2:  set $t^m_{\text{last}} = 0$
3:  while running context-aware applications do
4:    for each requesting sensor $S_m$ at $t_i$ do
5:      if $t_i - t^m_{\text{last}} \leq th_r$ and $p_j(t_i) \geq th_p$
6:        Do not trigger sensor $S_m$ at $t_i$
7:      else
8:        Trigger sensor $S_m$ at $t_i$
9:        $t^m_{\text{last}} = t_i$
10:     end if
11:     $i = i + 1$
12:   end for
13: end while
```

reflecting the average time duration which the user spends in contextual state s^l_j. The goal of VCAMS is to find the sensing schedule ss^l_j which consists of an optimized sequence of actions at each time instant by looking forward over the time duration $\hat{T}^l_j$. As a result, $\hat{T}^l_j$ is used to represent the time limit of state s^l_j before the model triggers continuous sensing to avoid delay in detecting a state change which becomes highly probable beyond this time limit. From the user's past behavior, the time spent by the user in the s^l_j state can be considered as a random variable T^l_j, with mean μ^l_j and a standard deviation σ^l_j. These time properties are learned from previous behaviors, and they are updated dynamically whenever state s^l_j is encountered. $\hat{T}^l_j$ is chosen based on the historical distribution of the time spent in each state s^l_j, and it is chosen to be greater than μ^l_j to minimize the sensor triggers, and thus minimize energy. Let $p^l_j(t_i)$ be the survival probability at time instant t_i that the context will stay in the same detected state s^l_j at this time instant.

Algorithm 5.3 shows the pseudocode for synchronizing the sensing schedules for the multiple concurrent contexts. The algorithm is time-variant where different decisions are made at each time instant. The algorithm takes as inputs the sensing schedules generated by VCAMS for the current state of each requested context. In addition, it takes the application-based thresholds upon which it decides whether to sense or not. The first threshold is the range th_r that defines the time instants accumulated since last sense; hence, it plays an important role in determining the incurred delay. th_r is the maximum acceptable time interval between two subsequent sensing triggers from the same sensor; hence, th_r defines the delay interval that the application can tolerate. The second threshold th_p defines the limit of survival probability that the user will stay in the same detected state. This threshold represents the probability beyond which sensing cannot be postponed due to high probability of a state change. For example, when the survival probability is low, it is more likely that the user will change her current state.

For each triggered sensor S_m (line 4), the algorithm checks if the interval between the current time instant t_i and the last time this sensor was triggered t^m_{last} is less than th_r. In addition, check at the same time (line 5) whether the current survival probability of the current state s_j is greater than the probability threshold th_p. If both conditions are satisfied, the sensors are not triggered and the most recent sensor data reading is used (line 6). Otherwise, the sensor is triggered (line 8), and the timing of its latest trigger t^m_{last} is updated (line 9).

5.2 Evaluation of Sensor Selection Using Simulated Dynamic Environment

This section evaluates the performance of EGO sensor selection framework using simulation, and Section 5.3 provides evaluation through a physical testbed implementation. Setion 5.2.1 provides implementation details for our proposed context recognition ontology. We define three conventional sensor selection systems for context sensing to compare their performance with EGO in Section 5.2.2 based on the performance metrics defined in Section 5.2.3. The effect of varying the number of available sensors is illustrated in Section 5.2.4. We compare the performance in terms of context recognition accuracy in Section 5.2.5, and we show that the performance of EGO gives a trade-off between energy, accuracy and the number of recognized contexts in Section 5.2.6. In Section 5.2.7, we evaluate EGO sensor integration with VCAMS sensor schedule to show the extra energy savings.

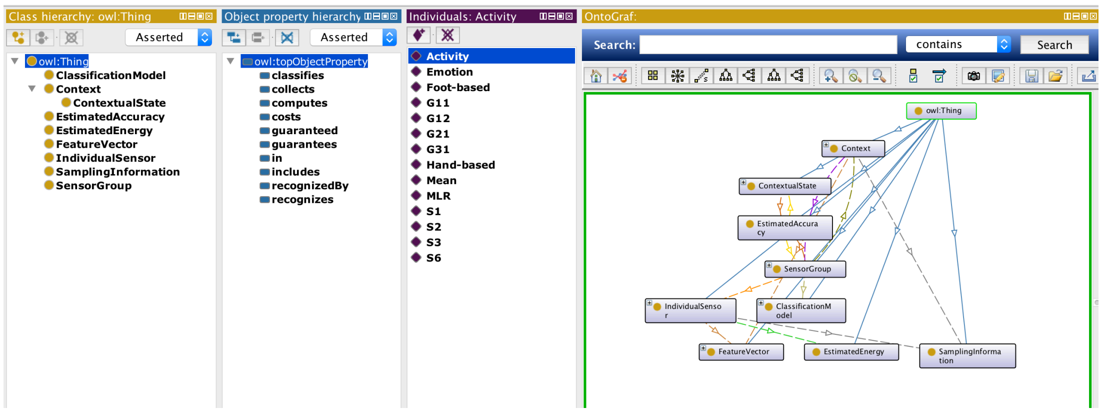


Figure 5.5: Protégé interface showing the ontology's classes, relationships, and a graphical illustration.

5.2.1 Ontology Development

The context-aware ontology was written in Web Ontology Language (OWL) and developed using Protégé [197] which is a graphical tool to develop knowledge-based solutions. Figure 5.5 shows the graphical user interface for Protégé capturing the main classes of the ontology and their accompanying properties. In addition, Protégé provides a high level graphical illustration of our developed ontology.

To retrieve the sensor group, the ontology can be queried with the following using SPARQL syntax:

```
PREFIX : <ContextOntology.owl#>
SELECT DISTINCT ?SensorGroup ?ContextualState ?IndividualSensor ?energyCost ?accuracyEstimate
WHERE { ?SensorGroup :recognizes :Activity .
?SensorGroup :includes ?IndividualSensor .
?ContextualState :recognizedBy ?SensorGroup.
?IndividualSensor :energyCost ?energyCost .
?SensorGroup :accuracyEstimate ?accuracyEstimate .
?ContextualState :accuracyEstimate ?accuracyEstimate .  }
```

Below, we provide a snapshot of the query that retrieves the sensing information from the ontology:

```
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX : <ContextOntology.owl#>
SELECT ?SensorGroup ?IndividualSensor ?samplingFreq ?windowSize ?FeatureVector ?model
WHERE { ?SensorGroup rdf:type :SensorGroup .
?SensorGroup :groupName ?name .
?SensorGroup :includes ?IndividualSensor .
?IndividualSensor :samplingFreq ?samplingFreq .
?IndividualSensor :windowSize ?windowSize .
?IndividualSensor :computes ?FeatureVector .
?SensorGroup :classifies ?model .
FILTER regex(?name, "SelectedGroup") }
```

We selected activity and emotion to be the two contexts simultaneously requested. To get the reliable information and populate the ontology, we selected four literature works for each context, the papers' information is summarized in Table 5.1. The table analyzes published works in detecting activity and emotion contexts by breaking down each study and comparing them in terms of interested contextual states, sensors used, processed features, and applied algorithms. Therefore, we incorporated a group of 8 embedded and external sensors, i.e., the embedded sensors in the smartphone are: 3-axis accelerometer, microphone, GPS, 3-axis gyroscope and 3-axis magnetometer; while the external sensors are: wearable 3-axis accelerometer, wearable EEG headset and wearable ECG. Let M be the total number of available sensors which is $M = 8$ in this example; and S_m be a specific sensor where $m = 1, 2, ..., 8$. Figure 5.6 shows how each of the 8 sensors contribute to the set of groups for contexts c_1 (Activity) and c_2 (Emotion) where each group is denoted by $\mathcal{G}^l_{k_l}$. The colored sensors are the ones which can be exploited for synergy between the two requested contexts.

5.2.2 Comparison Systems for Context Sensing

To show the effectiveness of EGO, we used environments in which the available sensors vary with time and user changes state in real-time to represent typical user behavior. As comparison systems, we considered three standard static sensor selection mechanisms that select pre-defined groups of sensors which does not change dynamically with time and available resources. The first mechanism aims at maximizing the accuracy without any energy considerations or sensor availability check. We denote such a system as Best Accuracy context-aware system (BA). The second mechanism aims at minimizing the energy without considering the achieved accuracy or the sensor availability. Such energy-efficient system selects the group of sensors for each requested context that consumes the least amount of energy. We denote such a system as Best Energy context-aware system (BE). The third mechanism provides a trade-off between energy and accuracy metrics. This energy-accuracy trade-off has been studied extensively in the literature; however, it does not depend on the synergy and the common sensors between multiple concurrent requested contexts. Therefore, such mechanisms select a static group of sensors which is used each time the context is requested without taking into consideration the current user state and other available sensor options. We denote this trade-off context-aware system as (TO).

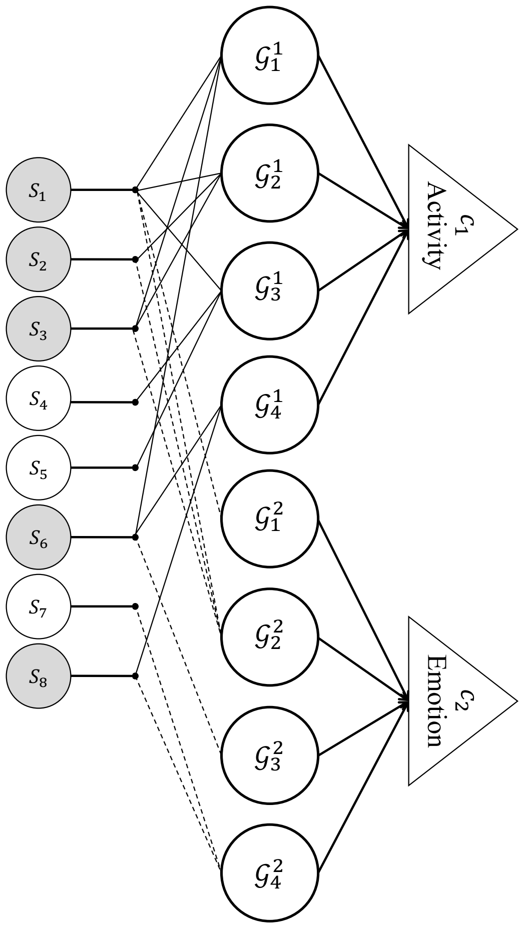


Figure 5.6: The group of sensors for each context c_l.

5.2.3 Performance Metrics

To measure the effectiveness of our proposed system EGO, we used three metrics: fraction of recognized context, energy consumed by triggered sensors and accuracy of recognizing the states in a particular context. In our case, we have two requested contexts: activity and emotion; therefore, the fraction of recognized context is defined as the number of contexts that are recognized by the available sensors. For each possible combination of available sensors, the fraction of recognized context is:

$$\text{Fraction of recognized context} = \frac{N_{\text{rec}}}{N_{\text{req}}} \tag{5.9}$$

where N_{req} is the number of requested concurrent contexts which is $N_{\text{req}} = 2$ in our example (activity and emotion); and N_{rec} is the number of contexts which can be recognized using the available sensors. The total energy consumed is defined as the energy consumed in Joules over the period of time when the application is requesting a particular context and the corresponding group of sensors is turned on and collecting data. It is defined as:

$$\text{Total energy consumed} = \sum_{S_m} E_m \tag{5.10}$$

where E_m is the estimated energy consumption by sensor S_m which belongs to the possible combination of sensors. Each context is recognized using an optimized group of sensors which guarantees a minimum classification accuracy. The average classification accuracy over the different requested contexts is:

$$\text{Average classification accuracy} = \frac{\sum_{l=1}^{N_{\text{req}}} A_{c_l}}{N_{\text{req}}} \tag{5.11}$$

where A_{c_l} is the accuracy of classifying the context c_l using the available sensors.

5.2.4 Performance based on Sensor Availability

Available sensors vary with time based on which sensors the user wears during a particular time of the day. Therefore, we first varied the number of available sensors and evaluated how the fraction of recognized contexts and the energy consumption change with sensor availability. This particular experiment shows how EGO reacts to different sensor availability compared to other static mechanisms.

Let the possible number of available sensors be M' which ranges from 1 to $M = 8$. Let $C_{M'}$ denote the number of possible combinations out of the M' available sensors; hence, $C_{M'} = {}^{M}C_{M'} = \dfrac{M!}{M'!(M-M')!}$. Figure 5.7 shows the average fraction of recognized contexts for each M'. It is defined as:

$$\text{Average fraction of recognized context} = \frac{\sum_{C_{M'}} \frac{N_{\text{rec}}}{N_{\text{req}}}}{C_{M'}} \tag{5.12}$$

As shown in the figure, EGO shows the highest number of recognized contexts. Each of BA, BE and TO selects a static sensor group to recognize a context; on the other hand, EGO checks what sensors are available and chooses the most energy efficient group of sensors while guaranteeing the minimum accepted accuracy. EGO queries the ontology to extract the different possible groups of sensors that can be used to recognize the requested context. When few number of sensors are available, BA, BE and TO do not recognize neither activity nor emotion. As the number of available sensors increases, the fraction of recognized context increases for all four systems BA, BE, TO and EGO. However, EGO guarantees full recognition of requested contexts; i.e. reaches the fraction of 1; earlier than the

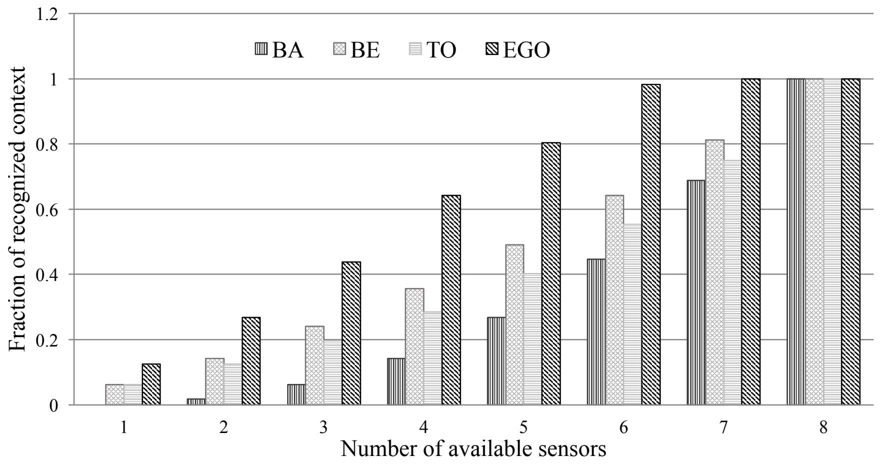


Figure 5.7: The effect of varying the number of available sensors on the fraction of recognized contexts. Our proposed selection mechanism proves to adapt itself to dynamic availability of sensors.

others reflecting its ability to recognize 100% of requested contexts. Therefore, our approach adapts itself to the dynamic availability of sensors.

We also evaluate EGO's energy consumption and compare it with the consumed energy for each of the BA, BE and TO approaches. For this intent, we used the sensor energy consumption values measured in [198] which provides a sensor power model for different sensors. We estimated the average consumed energy for each number of available sensors M', where we varied the sensor composition randomly every 1 minute for 15 minutes; and then we compute the average energy over these possible combinations. Figure 5.8 shows the energy consumed for each possible number of sensors. It is defined as:

$$\text{Average energy consumed} = \frac{\sum_{C_{M'}} \sum_{S_m} E_m}{C_{M'}} \tag{5.13}$$

As shown in the figure, all four considered systems start with similar energy consumption with few sensors. However, the figure shows slightly higher energy consumption by EGO until the number of available sensors hits 6. We previously showed in Section 5.2.4 that for low number of available sensors, EGO recognizes contexts whereas other systems cannot recognize due to lack of required sensors. Hence, although EGO consumes more energy, it guarantees recognizing the requested contexts. In other words, when there are few available sensors, BA, BE and TO approaches decide not to turn on any sensor since its predefined group of sensors is not yet available. Therefore, such approaches consume slightly less energy than EGO; however, at such sensor availability levels, they do not recognize any of the requested contexts. As the number of sensors increases, the BA approach consumes much higher values of energy than EGO. When more sensors

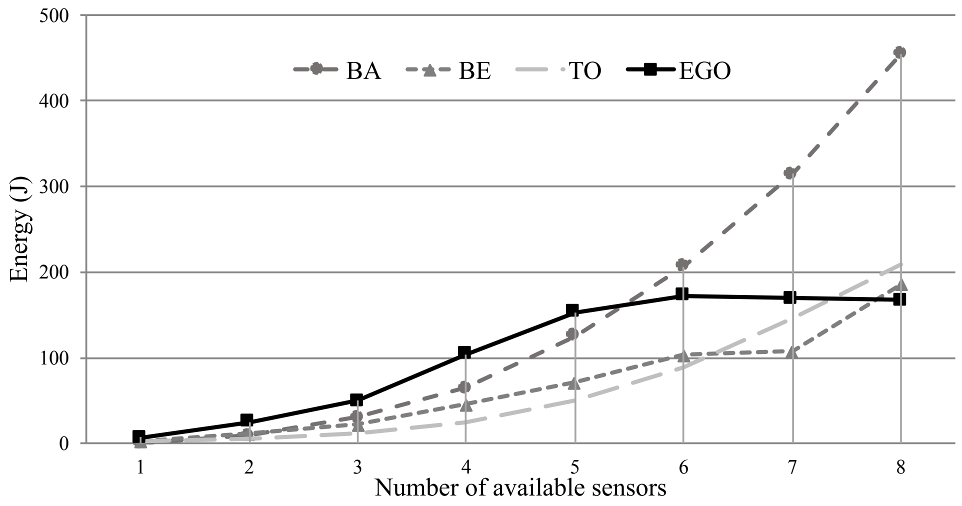


Figure 5.8: Increasing the number of available sensors increases the consumed energy since more contexts are being recognized.

are available, EGO can focus more on energy savings and select sensors which recognize context while saving energy and guaranteeing the minimum accepted accuracy. Hence, EGO shows much less energy consumption than BA and similar performance as BE and TO approaches which consider energy-efficient choices of sensors.

5.2.5 Context Recognition Accuracy

In this section, we evaluate our proposed approach in terms of classification accuracy. We assumed that all 8 sensors are available during this simulation; therefore, the fraction of recognized contexts is 1 for all approaches: BA, BE, TO and EGO. Throughout this experiment, BA will always choose the same sensors whose overall accuracy is highest, BE will always choose the same sensors whose overall energy is lowest and TO will choose the sensors that trade-off energy and accuracy; therefore, BA, BE and TO's choices of sensors are not based on the current contextual state of the user. On the other hand, EGO will check the current states of the user for each requested context and accordingly select the best group of sensors which saves energy. EGO extracts the recognition models and their parameters from the ontology. To make the simulated environment easier for analysis, we combined the activities of the user into two categories: hand-based activities (such as washing hands or brushing teeth) and foot-based activities (such as walking or climbing stairs). Similarly, we categorized the user's emotions into two states: neutral and expressive (such as happy, fear or angry).

Figure 5.9 shows marginally higher accuracy for BA over EGO for some combinations of states. This is expected since BA cares only about recognizing context

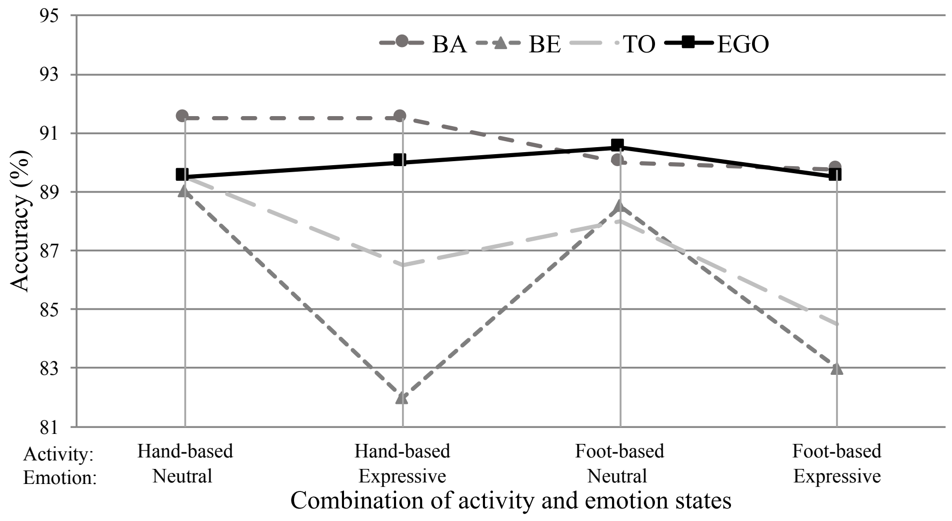


Figure 5.9: The average context recognition accuracy varies depending on the contextual state of the user.

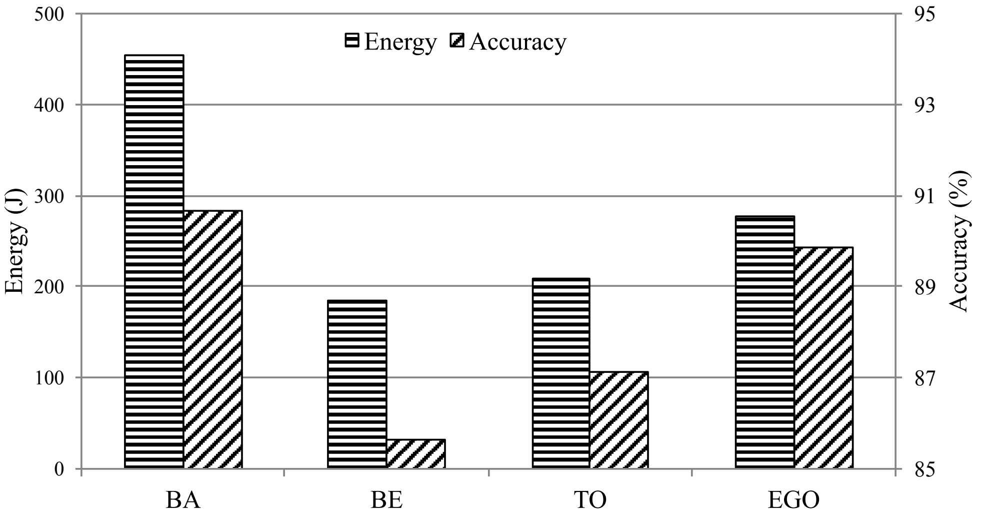


Figure 5.10: The trade-off between the overall average consumed energy and the average context recognition accuracy.

with the highest accuracy; whereas, EGO selects the group of sensors while optimizing energy. On the other hand, TO shows slightly less accuracy that EGO since it trade-off energy and accuracy while EGO guarantees a minimum limit of acceptable accuracy. As for BE, it shows the lowest accuracy levels since it selects its sensors considering only energy without considering accuracy levels.

5.2.6 Energy-Accuracy Trade-off

To show the trade-off between energy and accuracy, we plot the average consumed energy and average classification accuracy for each of the considered context-aware approaches. Figure 5.10 shows the results. The figure shows that both EGO and TO provides a trade-off between energy and accuracy. However, although EGO consumes slightly more energy that TO, it shows higher accuracy value. BE shows the lowest energy consumption but recognizes context with lower accuracy. On the other hand, BA shows the highest accuracy; however, it consumes the highest energy. EGO achieves 39% percent reduced energy consumption compared to BA, while BA has a marginal increase in accuracy which is less than 4%.

5.2.7 Impact of Integration with Optimized Sensing

To evaluate the performance of sensing synchronization after sensor selection, we conducted simulation experiments on two combinations of simultaneous states for activity and emotion contexts. The optimized VCAMS sensing schedules derived for each of activity and emotion contexts were compared with continuous sensing. Furthermore, we conducted analysis for the different parameters used in Algorithm 5.3. Different range threshold (th_r) and probability threshold (th_p) combinations were studied to investigate their effect on the trade-off between energy consumption and delay. The range threshold (thr) represents the maximum number of time intervals between two consecutive sensor triggers. It affects the incurred delay in detecting a state change. The probability threshold (th_p) represents the minimum survival probability beyond which the sensor should be triggered.

Let T_j^l be the actual time duration spent in state s_j^l. From the user's past behavior, T_j^l can be considered as a random variable, with mean μ_j^l and a standard deviation σ_j^l. Therefore, T_j^l follows a time distribution dependent upon the context being monitored and the behavior of the user in such context. For our simulations, we chose for context "c_1 =Activity" the state "s_1^1 = AtMeeting" and for context "c_2 =Emotion" the states "s_1^2 = Neutral" and "s_2^2 = Pleasant". We picked the estimated time for each as follows $\hat{T_1^1}$ = 130 min, $\hat{T_1^2}$ = 60 min and $\hat{T_2^2}$ = 70 min. In the simulation, the user starts the meeting with a neutral emotional state, then after a while she shifts to a pleasant emotional state. As for the testing data, we ran the simulator to generate 10000 values of the actual time spent in each of the states denoted by T_j^l. Figure 5.11 demonstrates VCAMS sensing schedules that was obtained for different contextual states. The first three timelines in the figure show VCAMS sensing decisions at each time unit for each contextual state. Furthermore, the figure shows the synchronized schedule derived from Algorithm 5.3. We compare this synchronized sensing schedule with the schedule which triggers the sensors at all the time instants at which sensing

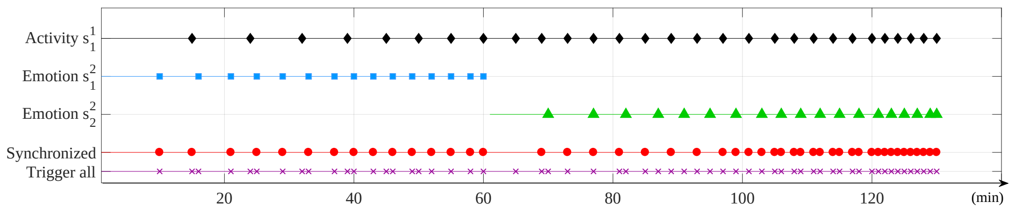
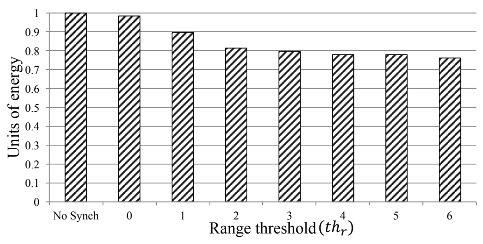
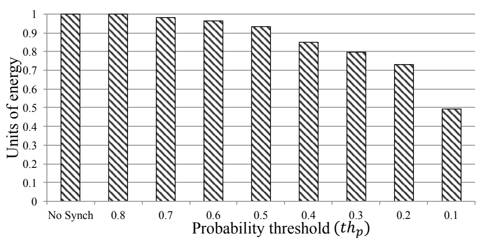


Figure 5.11: The generated sensing schedules after integrating and synchronizing with optimized sensor scheduler. The dots mean the time instants at which sensing is triggered. Our proposed synchronization approach saves sensor triggers.



(a) $th_p = 0.3$



(b) $th_r = 3$

Figure 5.12: The range threshold and the probability threshold reflect the relative effect of synchronizing the sensing schedules generated by the optimized sensing scheduler. These thresholds represent the limits beyond which sensing should be triggered.

is requested by VCAMS denoted in the figure by “Trigger all”. The figure illustrates that less sensing triggers are requested when considering a synchronized schedule compared to the “Trigger all” approach.

Effect of Range and Probability Thresholds th_r and th_p

The range threshold th_r and the probability threshold th_p reflect the relative effect of synchronizing the sensing schedules generated by VCAMS. Varying these thresholds affects the derived synchronized sensing schedule. Figure 5.12 demonstrates how these thresholds affect the generated synchronized sensing schedule; and thus affecting the energy values. The energy metric was normalized where the largest energy was used as base for energy normalization. Figure 5.12a shows the effect of the range threshold th_r for a fixed probability threshold of $th_p = 0.3$ %. The figure shows that as the range threshold increases, less energy is consumed since the algorithm allows older sensing data to be used in the present context recognition. As the range threshold increases, the synchronized schedule accepts a bigger time interval between two subsequent sensor triggers. Figure 5.12b shows the effect of the probability threshold th_p for a fixed range threshold of $th_r = 3$ time intervals. The figure shows that as the probability threshold decreases, less energy is consumed since the sensor is not triggered when the condition

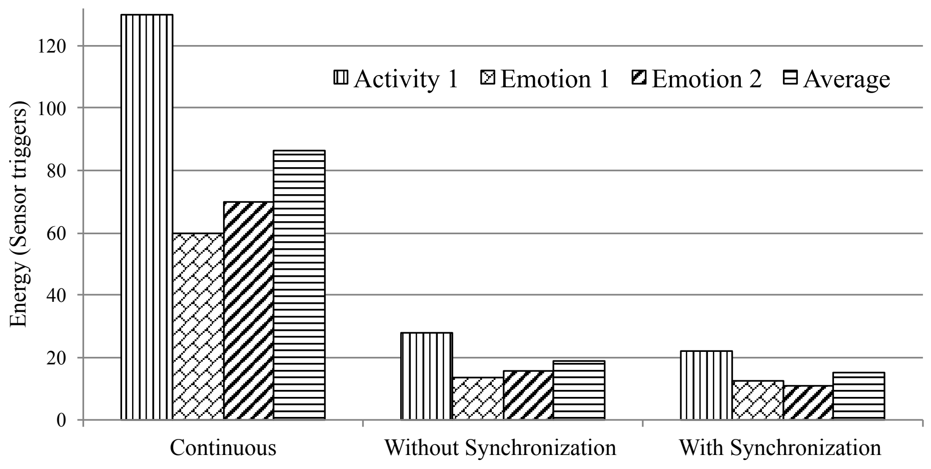


Figure 5.13: The effect of using the optimized sensing scheduler on energy. The figure shows 76% energy gains when compared to continuous sensing.

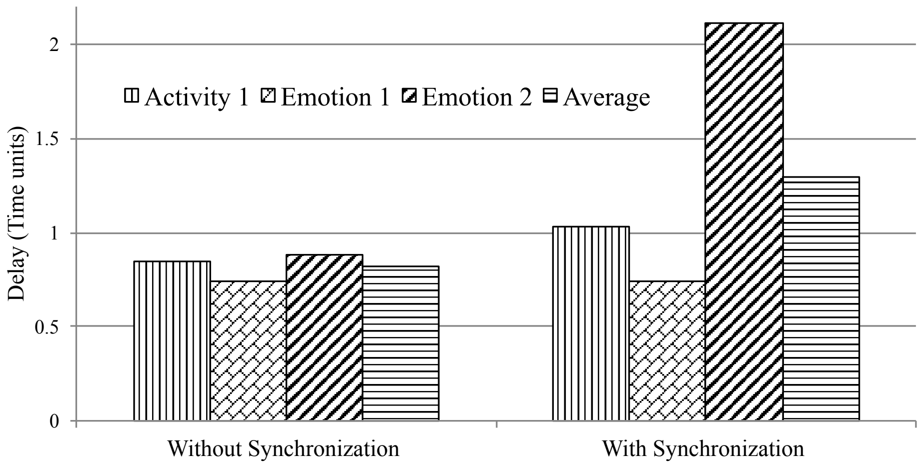


Figure 5.14: The effect of using the optimized sensing scheduler on delay. The figure shows a slight delay increase of 0.5 time unit when synchronizing the sensing schedules of the different sensors.

$p_j(t_i) \geq th_p$ is met.

Performance Analysis of the Integration Strategy

In this section, we compare the performance after synchronization with the performance when only VCAMS is considered without synchronization. In addition, we compare with continuous sensing as baseline for both methods. We examined the average energy and delay values for each of the proposed methods. The simulation ran 10000 times for each method to find the average energy and delay. The choice of the range threshold and probability threshold is application-dependent. For example, if the application is health-related with critical context being monitored, th_r and th_p should be chosen such that delay is minimized. We used $th_r = 3$ and $th_p = 0.3$ as range and probability thresholds respectively which are acceptable by energy and delay objectives and provide a balance between the incurred delay values and the total consumed sensing energy. We measured energy by the number of time instants at which sensing is triggered and delay by the number of time instants skipped before detecting a state transition. Figure 5.13 shows the energy consumed by continuous sensing, VCAMS sensing schedule, and synchronized sensing schedule. We considered each contextual state alone and we computed the average energy consumption. As shown, the consumed energy greatly decreases when using VCAMS without synchronization. In addition, further energy reductions are guaranteed when synchronizing the sensing triggers. Figure 5.14 shows the effect of synchronization on delay. As shown, the average delay for continuous sensing is zero and it slightly increases when using VCAMS by an average of 0.8 min without synchronization; then it reaches its maximum when using synchronization which is slightly larger of average 1.3 min. Therefore, synchronizing the sensing schedules among the different contexts saves almost 22% of sensing triggers; thus, it saves 22% of energy consumption, while increasing the delay by less than 0.5 time unit. The increased delay is 1% of the mean time spent in the state; therefore, it is considered acceptable for the activity recognition application. Therefore, synchronizing the sensing schedules obtained from VCAMS provides a trade-off between energy and delay; however, the proportional energy savings exceed the proportional delay losses.

5.3 Case Study with Real Implementation of Multi-context Operation: Activity and Emotion

To test our sensor selection technique in real-life settings, we have implemented EGO framework on commercially available sensors and mobile devices. We demonstrate the performance of EGO for a human subject using a smartphone and wearable sensors as shown in Figure 5.15. We reproduced the sensor scenarios in Table 5.1 where the possible sensor groups are retrieved from the context-aware ontology. In addition, sensors were used to collect raw data based on the sampling information specified by each of the investigated studies which is retrieved by querying the ontology for the sensing information and recognition models. The subject was instructed to perform 10 activities recorded in reaction to a controlled experiment with 3 emotional movies.

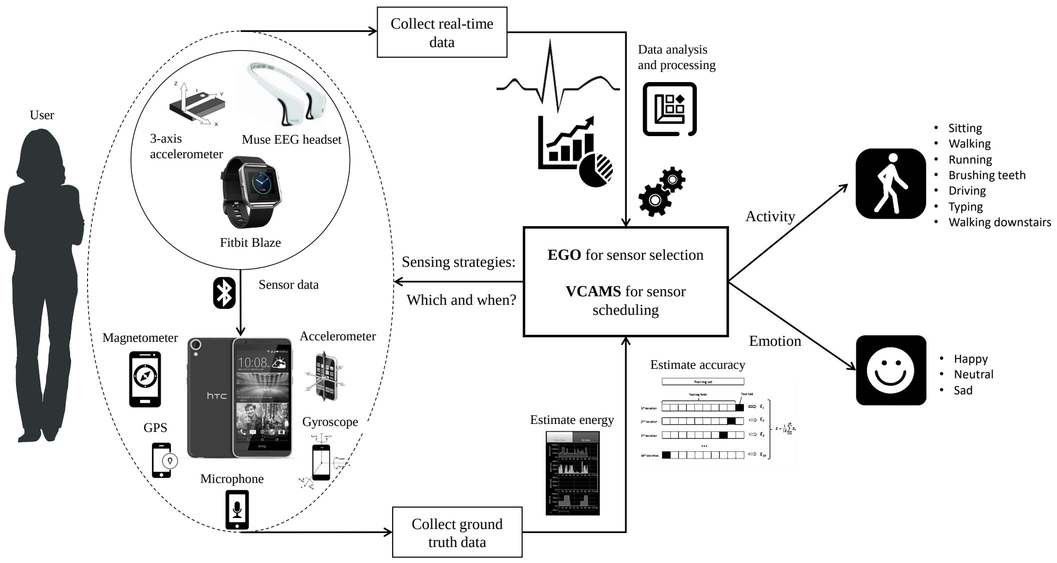


Figure 5.15: The experimental setup of our energy efficient and accurate activity and emotion monitoring system. EGO framework is implemented on commercially available sensors and mobile devices. The subject holds a smartphone and wears different external sensors from which raw data is extracted.

5.3.1 Experimental Setup

The Android device was an HTC Desire 820G device, having an Octa-core 1.7 GHz Cortex-A7 processor and 1 GB RAM. The subject was asked to keep the phone in her pocket. The smartphone has several embedded sensors which collect data. In addition, the subject was asked to wear a BMA050 3-axis accelerometer on the dominant hand, a lightweight 6 channels/electrodes Muse EEG headset on her head [52] and a fitbit blaze to monitor the heart rate using PurePulse [199]. We used Muse Monitor android application [87] to collect the raw data from the EEG headset and Fitbit Data website [200] to capture the raw heart rate from the fitbit. Figure 5.15 illustrates the experimental setup of our case study. The subject was asked to perform different activities, each for 60 seconds. These activities include ‘sitting’, ‘walking’, ‘running’, ‘brushing teeth’, ‘driving’, ‘typing’, and ‘walking downstairs’. Furthermore, for each activity, three emotional states were tested. The emotions were ‘neutral’, ‘happy’, and ‘sad’. The activities and emotions were selected such that they will be inline with the activities that are used to evaluate the different 8 investigated studies. To vary the subject’s emotions, the subject wears a muse headset which is used to meditate her feelings and obtain the neutral emotional state, then the subject was asked to watch a comedian clip to elicit the happy emotional state. Finally, she was asked to watch a sad film clip to elicit her sad emotional state. The readings of all sensors were transmitted to the HTC Desire 820G phone paired with all the wearable sensors worn by the subject. EGO selects the group of sensors based on the state of the user; therefore, we have 21 different activity-emotion state combinations.

For sensor sampling, the sensor readings were captured using the sampling frequency and window size specified by each of the investigated studies in Table 5.1. For feature extraction, we extracted a set of statistical time and frequency features that were proposed by the investigated papers. Finally, the extracted features were partitioned into two sets, we used the first 90% of the data for development and tuning the classification model and the remaining 10% for testing to measure the classification accuracy. We used Statistics and Machine Learning Toolbox in MATLAB, and we performed extensive experiments with different algorithms mentioned in the 'Classification Algorithm' column in Table 5.1, these algorithms are used to derive models for activities and emotions contexts. Finally, these models, derived offline, are communicated into the mobile device to classify the current contextual state in real-time.

5.3.2 Parameter Estimation

In our proposed approach in Section 5.1.3, there are two parameters needed for the optimization problem. The two parameters are the energy consumed by each group of sensors $E^l_{k_l}$ and the state-based classification accuracy by each group of sensors $A^l_{k_l,j}$ which is the accuracy of recognizing state s^l_j using the sensors in sensor group $\mathcal{G}^l_{k_l}$. We used PowerTutor to find the energy consumed by each group of sensors. PowerTutor is an android application which is widely used [144] to estimate the power consumed by major system components: CPU, display, and network interfaces. Therefore, while collecting and processing data from each group of sensors, we ran PowerTutor in the background to collect the energy consumed by the CPU component for this specific group. Table 5.3 shows the energy consumption by each group of sensors which is composed of the energy consumed by embedded sensors, wearable sensors and transmitting the data from wearables to mobile device. This energy was collected over a period of 60 seconds for each group of sensors.

Table 5.3: Energy consumption for sensor groups

Context	**Sensor Group**	**Total Energy (J)**
Activity	$\mathcal{G}_1^1$	5.436
	$\mathcal{G}_2^1$	3.2
	$\mathcal{G}_3^1$	0.856
	$\mathcal{G}_4^1$	16.136
Emotion	$\mathcal{G}_1^2$	1.23
	$\mathcal{G}_2^2$	5.2
	$\mathcal{G}_3^2$	1.614
	$\mathcal{G}_4^2$	28.6

Table 5.4: Classification accuracy of sensor groups in classifying different states in activity context

Sensor Group	**State-based Accuracy**							**Average Accuracy (%)**
	Sitting	Walking	Running	Brushing Teeth	Driving	Typing	Walking Downstairs	
$\mathcal{G}_1^1$	86.7467	87.33	89.1	84.5	87.4	83.623	87.233	86.562
$\mathcal{G}_2^1$	85.59	85.053	86.353	82.2467	87.323	80.203	85.297	84.581
$\mathcal{G}_3^1$	79.09	79.157	82.596	74.38	74.87	73.783	81.956	77.976
$\mathcal{G}_4^1$	84.406	87.36	88.51	85.853	83.506	85.523	86.41	85.939

Table 5.5: Classification accuracy of sensor groups for emotion context

Sensor Group	**State-based Accuracy**			**Average Accuracy (%)**
	Happy	Neutral	Sad	
$\mathcal{G}_1^2$	71.557	72.06	71.526	71.714
$\mathcal{G}_2^2$	62.609	61.727	62.903	62.413
$\mathcal{G}_3^2$	69.967	70.416	69.8	70.061
$\mathcal{G}_4^2$	79.113	78.734	79.08	78.976

To estimate the classification accuracy, we evaluated the accuracy per state for our activity-emotion recognition system. We used 10-fold cross validation which has been widely used in the literature [201, 202]. In 10-fold cross validation technique, we randomly divided our dataset into 10 mutually exclusive subsets. Then, 9 of those subsets are combined to form the training set while the remaining subset is used as the test set. This process is repeated 10 times where each time a different subset is used as a test dataset. Then, the accuracies resulting from all 10 runs were averaged to produce the final classification accuracy. To obtain an accuracy for each state, we considered binary classification where the user is either in state s_j or not; hence, the class is either 1 or 0 respectively. The experimental results are shown in Tables 5.4 and 5.5 for activity and emotion respectively. The tables show that classification accuracy is state-dependent where the same group of sensors return different classification accuracies based on the activity or emotion of the user. The average accuracies over different activities and emotions are proportional to the accuracies obtained by the sensor scenarios listed in Table 5.1; however, they might differ in some cases due to the different experimental setup such as the different brands of sensors used or the list of states considered. These estimated energy and accuracy parameters are input into the ontology so that EGO can select the optimized group of sensors in real-time.

5.3.3 Comparison of EGO to Other Methods with Real Data Traces

To compare the performance of EGO with other existing approaches, we considered again each prior benchmark method presented in Section 5.2.2. The three standard systems use a fixed set of sensors each time the context is requested. In our experiment, the Best Accuracy context-aware system (BA) will choose sensor group $\mathcal{G}_1^1$ to recognize activity and $\mathcal{G}_4^2$ to detect the user's emotion since these sensor groups achieve the highest accuracy as shown in Tables 5.4 and 5.5. Hence, it will trigger sensors S_1, S_3, S_6, S_7 and S_8 corresponding to embedded accelerometer, GPS, wearable accelerometer, EEG and ECG respectively. On the other hand, the Best Energy context-aware system (BE) will choose sensor groups $\mathcal{G}_3^1$ and $\mathcal{G}_1^2$ to recognize activity and emotion contexts respectively. BE chooses these sensor groups since they consume the least amount of energy as shown in Table 5.3. BE will trigger sensors S_1, S_4, and S_5 corresponding to embedded accelerometer, gyroscope and magnetometer. The third approach is Trade-Off context-aware system (TO) which will use sensor groups $\mathcal{G}_2^1$ and $\mathcal{G}_1^2$ such that it provides trade-off between energy and accuracy. For example, sensor group $\mathcal{G}_2^1$ is chosen by TO since it provides an equal trade-off between energy and accuracy when considering activity recognition. Thus, TO will trigger sensors S_1, S_2, and S_3 which are the embedded accelerometer, microphone and GPS respectively.

In this experiment, we asked the user to wear the sensors for 3 hours and

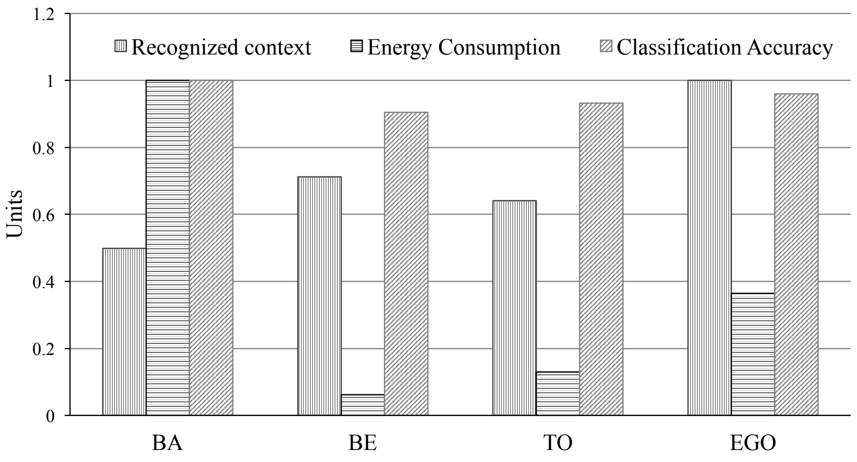


Figure 5.16: Energy, accuracy and number of recognized context for EGO and conventional context-aware sensor selection systems. EGO gives a balance between energy, accuracy and number of recognized contexts.

we randomly changed the number and composition of the available sensors each 15 minutes. Then, we used the three performance metrics: average fraction of recognized context, average energy consumed, and average classification accuracy to evaluate EGO's performance with respect to the other conventional systems. To visualize the gains in terms of the three performance metrics, we normalized these results by dividing each of the average fraction of recognized context, average energy consumed and average classification accuracy by its corresponding maximum value. The results are summarized in Figure 5.16. When comparing to conventional methods, the results show that EGO gives a balance between energy, accuracy and number of recognized contexts. Although BA gives the best accuracy performance, it costs much higher energy consumption and it cannot recognize contexts since it requests the availability of multiple sensors. EGO framework was able to achieve 50% improvement in fraction of recognized contexts and 63% reduction in consumed energy when compared to BA. EGO selects the group of sensors that optimizes energy whereas BA cares only about recognizing context with the highest accuracy and uses a static pre-defined set of sensors each time the context is requested. On the other hand, BE costs the lowest energy consumption; however, it gives lower accuracy value. In addition, BE does not have the ability to recognize the context when few sensors are available. As for TO, it provides a trade-off between energy and accuracy; however, it still does not have flexible choices of sensors to recognize activity and emotion. It always uses the same groups of sensors to recognize context; and it returns void contextual state when any of these sensors is not available. Therefore, there will be times when TO will miss recognizing a context since some of the pre-defined sensors is not available.

5.3.4 Energy Gains of VCAMS with EGO in a Real Application

In this section, we compare the performance of the framework in a real system implementation while measuring energy and delay, and considering three scenarios: 1) EGO with continuous sensing which provides the naive choice of using the continuous sensing where a selected sensor is triggered continuously, 2) EGO with VCAMS but without synchronization which provides the time instants at which a sensor needs to be triggered; however, in this scenario VCAMS does not synchronize the sensing triggers among the different selected sensors, and 3) EGO with synchronization where the sensing schedules generated by VCAMS are combined to produce an optimized synchronized sensing schedule. The user was asked to wear all the sensors for 3 hours during which she was asked to perform a variety of activities. To collect the ground truth, we asked the user to label the activity. We used PowerTutor to estimate the consumed energy and we investigated the delay in detecting a state change from one activity to another.

The overall energy and delay resulting from running each scenario for 1 hour are presented in Figure 5.17. Comparing EGO without any sensing schedule; i.e. continuous; with EGO followed by VCAMS without synchronization shows almost 76% reduction in energy and a slight increase in delay from 0 to 16.2 sec which can be considered negligible compared to the 1 hour that the user spends in an activity. VCAMS gives a balance between energy and delay. As for EGO with synchronization; i.e. VCAMS followed with sensing synchronization, it saves extra energy gains which reaches almost 81% compared to continuous sensing. However, it causes slightly larger delay value of average 21.42 sec. EGO framework with sensing synchronization effectively combines the sensing schedules generated by VCAMS to provide further energy savings. In summary, EGO sensor selection mechanism recognizes the highest number of contexts using available sensors and gives a balance between energy and accuracy values. Furthermore, embedding VCAMS's sensing schedule mechanism with synchronization along the different selected sensors saves extra energy values giving a trade-off between energy and delay in detecting a state change.

5.4 Summary

In this chapter, we have presented EGO: an energy-efficient ontology-based group sensor selection framework that explores the synergy across applications requesting multiple contexts concurrently. EGO reduces energy consumption by considering the different synergy choices and common sensors among the applications. The framework includes a context ontology that captures the knowledge base of context recognition models along with features and machine learning parameters. EGO adaptively selects the groups of sensors that provides a trade-off between

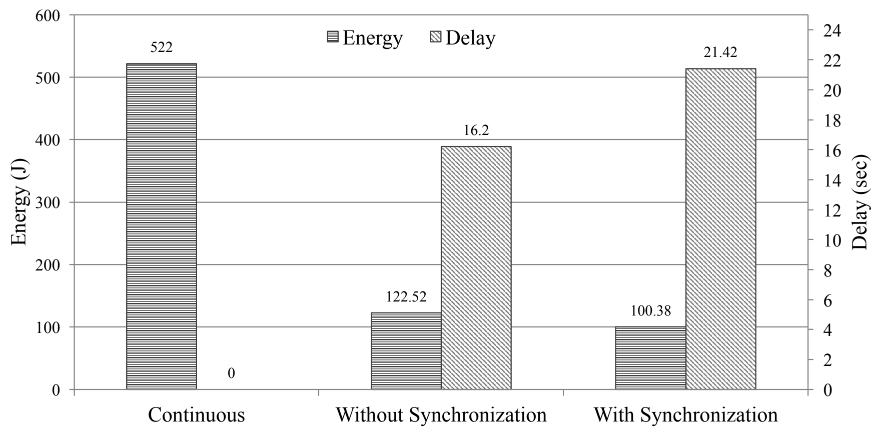


Figure 5.17: The effect of integrating and synchronizing with an optimized sensor scheduler.

the energy consumption, the accuracy of context recognition, and the number of recognized contexts. EGO includes a synchronized sensing schedule that provides a trade-off between the energy consumption and the delay to detect a context change.

We implemented a real test-bed that shows simultaneous operation of multiple context-aware applications, and demonstrates the effectiveness of the proposed framework in coordinating efficient operation of the applications. When compared to conventional sensor selection methods, EGO framework was able to achieve 50% improvement in the fraction of recognized contexts and 63% reduction in energy consumption. EGO provides dynamic sensor selection to recognize requested contexts based on the available sensors while optimizing energy and accuracy; whereas other previous works use static sensor selection. Furthermore, results showed that the integration and synchronization with optimized sensor scheduler saves extra 81% energy when compared to continuous sensing since it provides an optimized synchronized sensing schedule.

Chapter 6

Conclusion and Future Work

This chapter summarizes the main contributions of this PhD dissertation and indicates open research questions in this field which are extensions to the proposed work.

6.1 Thesis Summary and Contributions

As we are shifting into an intelligence technological age [203], we are approaching Mark Weiser's vision of having technologies blending into our everyday life [1]. One of the emerging technologies in this respect is the development of smart mobile and wearable devices which are equipped with a variety of sensors. These devices are nowadays conscious about their users and environments by continuously extracting sensory data and monitoring their context information such as the user's identity, emotion, activity and surrounding environment. Context awareness enables the smart devices to adapt their behavior proactively with the context so that it improves the user's experience and provides user-tailored services. As a result, context-aware applications provide new opportunities for better understanding and enhancing our lives. However, smart mobile devices have limitations and resource constraints where the continuous sensing and context recognition process puts heavy workload on the smart mobile devices and sensors.

In this dissertation, we present a generic context-aware framework composed of sensing designs that enable context recognition. The challenge is to collect relevant sensory data from embedded and external sensors, recognize accurate context, and provide context-aware services with minimal delay, while preserving the scarce resources of mobile devices and sensors. As a result, this thesis has presented novel context-aware dynamic designs and mechanisms that facilitate the process of recognizing context while reducing the amount of energy consumption. We have shown that a smart dynamic sensing framework can improve the process of collecting data from external and embedded sensors while trading off resource consumption, application accuracy, and delay in detecting a state change. In particular, we have presented the following objectives and contributions:

6.1.1 Objective 1: Energy Efficient Sensor Sampling Strategy

In this part of the thesis, we presented two energy efficient sensor sampling strategies that form one component of the proposed smart dynamic sensing framework. In this objective, we prove that the sensing behavior depends on the state of the user. In addition, we explore different techniques to select a sensor sampling mechanism and compare their system performances in terms of energy and accuracy. We have presented the following contributions in this objective:

- Formulating the sensor sampling selection as an optimization problem to trade-off energy and accuracy. The objective is to derive the best operating conditions for phone sensors and detect the user's activity while trading off the two factors, energy and accuracy. In other words, energy is reduced with lower usage of sensors, while accuracy is increased when more sensor data is available. As a result energy and accuracy provide conflicting requirements on phone usage. The objective function is maximized to balance the trade-off between energy term and classification error term.

- Proposing an optimized energy-accuracy activity-dependent sensing algorithm for recognizing human physical activity. We showed the effect of varying the sampling parameters on classification accuracy changes based on the monitored contextual state (activity in our case). Therefore, for each activity, there exist activity-specific sampling parameters (sampling frequency and window size) that minimize the energy while maximizing the classification accuracy.

- Formulating the classification accuracy in terms of entropy and sampling choices. We proved that the classification error is bounded by the conditional entropy. We also demonstrated using experiments on a real dataset that higher sampling frequency leads to lower classification error and lower conditional entropy. Therefore, classification error is upper bounded by conditional entropy.

- Quantifying the impact of sampling parameters (sampling frequency and window size) on the amount of energy consumed while collecting sensory data. We have demonstrated using experiments on a real Android device that the energy consumption depends on the sampling parameters; in particular, the sampling frequency. Consequently, we propose an empirical energy model that estimates the energy consumed by each sensor sampling in terms of the sampling parameters.

- Quantifying the impact of sampling parameters (sampling frequency and window size) on the classification accuracy. We showed through experiments that higher sampling frequency leads to lower classification errors. We used 10-fold validation to estimate the classification accuracy for each sampling frequency. In addition, we demonstrated that the energy consumption depends on the window size which represents the time window during which raw sensory data is collected to generate the feature for context recognition.

- Exploiting the advantages of deep learning to determine the best state-dependent sensor sampling frequency for context recognition. Deep learning has evolved through recent years resulting in high classification accuracy; thus, it can guarantee more efficient sensing from embedded and wearable sensors. We showed using experiments on real dataset that the performance of DNN surpasses the performance of DT and NB.

- Exploiting multiple granularity by ensemble classification for sensor sampling decisions to trade-off energy and accuracy. We used Deep Neural Network (DNN) with ensemble classification of other complementary machine learning approaches (DT and NB). The experiments showed that each considered machine learning approach behaves differently based on the contextual state.

- Designing an optimized state-dependent sensing algorithm that trades-off energy and accuracy to recognize user contextual information such as: activity, health condition, and location detection. The algorithm is based on an optimization formulation that finds the optimized sampling frequency which minimizes energy and maximizes classification accuracy. We demonstrated that classification accuracy is state-dependent where activities involving more movement (walking or jogging) require more samples compared to steady activities (sitting).

6.1.2 Objective 2: Context-Aware Sensor Scheduling to Trade-off Energy and Delay

In this objective, the aim was to decide when to trigger the sensor for data collection so that interesting state changes are captured with minimal energy consumption and lowest delay. This objective builds upon the idea that most applications do not need to detect the states of the context continuously, rather they require the detection of critical changes in context. Hence, we propose an efficient dynamic mobile sensing strategy that allows the smart mobile device to intelligently interact with external sensors and embedded sensors while trading off resources' energy consumption and application delay targets. The contributions in this objective are:

- Formulating the sensor schedule problem using Viterbi implementation to learn sensing schedules for real-time decisions on when sensing should be triggered. The optimal solution determines the time instants at which a sensor should be triggered for data collection. The goal is to find the sensing schedule which consists of an optimized sequence of actions at each time instant by looking forward over the time duration and maximizing the cumulative rewards, where the rewards comprehend the trade-off between energy and delay. We applied Viterbi because it proved to be optimal for estimating the state sequence of a finite state process [158]. Hence, Viterbi can be applied to any dynamic problem with finite states [204]. In our case, the Viterbi states are the sensing actions which are two: "sense" or "no sense". Our problem deals with real dynamic time limits where some sensing decisions might be taken over long periods of time; hence, Viterbi provides a reduction in computational complexity by using recursion and saving only the most likely path leading to each state.

- Customizing new reward functions (Viterbi path metrics). In dynamic programming algorithms, metrics need to be defined to represent the utility associated with each transition between nodes. Therefore, the energy and delay reward metrics are formulated to best represent the utility associated with each transition between trellis nodes in a Viterbi-based algorithm. We defined those customized reward functions in terms of optimization criteria which depend on the sensing action and the current state.

- Proposing a novel user model which captures a transition model from historical behavior of the user such as the probabilities of transition from one state to another at any time instant, and can also include the statistics of the time duration spent in a specific state. Accordingly, the sensing schedule is adaptive from two aspects: 1) the decision rules are learned from the user's past behavior extracted from the proposed user model, and 2) these rules are updated over real time whenever there is a significant change in the user's behavior by updating the user model with new statistics.

- Finding the Pareto optimal parameters in the formulation. There is a trade-off between energy and delay. Thus, finding the best energy-delay weighting factors is a multi-objective optimization problem which requires an optimal Pareto solution. Pareto solutions find points which are acceptable by both objectives. The optimal combination is the one which provides a balance between the incurred delay values and the total consumed sensing energy.

- Proposing a strategy that triggers learning mode in real-time to update the sensing schedule only when critical changes are captured, thus avoiding unnecessary computations. The learning part of the system generates a user-specific lookup table (LUT) capturing which sensing schedule should be used for each state. In addition, when comparing this proposed triggering mode with other adaptive strategies, the results show that our proposed strategy gives the best performance in terms of sensing energy consumed and delay in detecting a state change.

- Showing through experiments using real datasets that our proposed sensor scheduling approach gives the best balance between energy and delay when compared to previous state-of-the-art methods. Furthermore, we also demonstrated that our proposed approach can be configured (by tuning the weighting factors) to achieve lowest possible delay, lowest possible energy, or lowest possible trade-off combination.

6.1.3 Objective 3: Ontology-Based Sensor Selection for Simultaneous Execution of Multiple Context-Aware Applications

The main target of this objective was to alleviate the energy limitation with mobile devices in multi-context setting where multiple context-aware applications are running and requesting multiple contexts. The sensor selection approach builds upon the idea that multiple sensor applications are currently in need to run concurrently so that multiple contexts, such as activity, location or emotion, are simultaneously recognized. Hence, we demonstrated that energy consumed to sense and recognize contexts can be reduced without reducing the recognition accuracy below acceptable values through selecting a group of sensors which exploits the synergy among applications while jointly optimizing the energy-accuracy trade-off of all requesting applications. In particular, we have presented the following contributions in relation to this objective:

- Designing and developing a context ontology that captures context recognition models along with descriptive specifications for each context recognition model such as required sensors and machine learning parameters. This ontology is used by the framework to examine the different sensor options for each application. This ontology can be used by other researchers to facilitate the knowledge sharing and future research in context-aware applications.

- Proposing an adaptive sensor selection approach to select the optimal group of sensors that trades-off the energy consumption and the accuracy of context recognition. The selected group of sensors is adaptively changed based on what contexts are requested by the applications, which sensors are available, how much energy resources are available in the smartphone and external sensors, and what state the user is currently in.

- Implementing a real test-bed implementation first of its kind to show operation of multiple context-aware applications running simultaneously, and demonstrate the effectiveness of the proposed framework in coordinating efficient operation of the applications. We implement 8 context-aware applications comprehending activity and emotion.

6.1.4 Objective 4: An Integration Strategy Towards a Holistic Dynamic Sensing Approach

In this objective, the aim was to propose a cohesive solution that integrates the different objectives together. We propose a hierarchical approach in which sensor selection is followed by selecting the appropriate sensing schedule of each sensor. The main contribution in this objective is the following:

- Proposing a comprehensive approach that uses our previously proposed sensor scheduling mechanism VCAMS proposed in Objective 2 to output an optimized synchronized sensing schedule which decides when to trigger each of the selected sensors for data collection to trade-off the energy consumption and the delay to detect a context change. We then apply synchronization to these selected sensing schedules to save more energy through synchronizing sensor triggers.

6.2 Future Work

In this section, we present future extensions that complement the discussions presented throughout the dissertation. In addition, we highlight some interesting open research problems that need to be addressed in order to improve context recognition and usage so that it can further facilitate users' lives.

6.2.1 Integrating Sensing Strategies: A Holistic Approach for Dynamic Sensing

In the previous objectives, we have proposed a hierarchical approach that integrates the sensor selection and the sensor scheduling mechanisms. We formulate and solve the problem sequentially dealing with the phases of the decisions in a consecutive order with each phase following the other. The first step is in Objective 3 where we should choose what sensors to trigger. For each sensor, we select the time instants at which each sensor should be triggered. Therefore, the variables and the methodologies used in one step depend on the outcomes of the previous step. One advantage of sequential topology when compared to holistic approach is that more specific problems can be tackled in latter steps after finding solutions to prior steps. More specifically, we chose M out of N sensors where

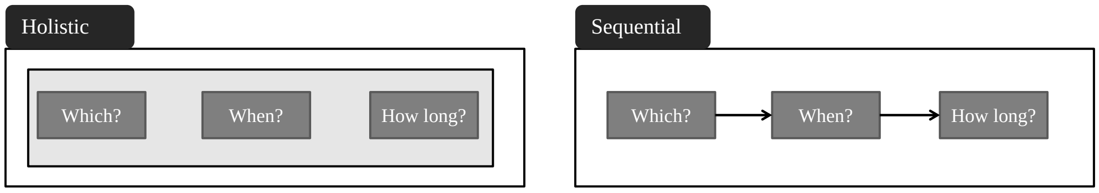


Figure 6.1: Framework approaches for sensing strategies.

each sensor has W_i different number of parameters from Objective 2. Solving Objective 3 has $\sum_{M=1}^{N} \binom{N}{M}$ possible solutions. After finding which M sensors to use, the problem simplifies to finding the sensing parameters W_i for each chosen sensor where $i = 1...M$. Therefore, solving Objective 2 has $\prod_{i=1}^{M} W_i$. Hence, the number of possible solutions:

$$\sum_{M=1}^{N} \binom{N}{M} + \prod_{i=1}^{M} W_i \tag{6.1}$$

One possible extension is to consider another integration approach which takes a holistic look at the problem with real applications where all three decisions need to be made for optimized system operation: Which sensors to operate, when to trigger, and for how long? Based on the required context by the application, there are several topologies for how to integrate the strategies [205]. The integration approaches are illustrated in Figure 6.1, and detailed below:

Holistic Approach

The first approach is to formulate the problem as a holistic optimization problem with all the variables included. In the holistic approach all of the objectives are integrated into one global objective function. Consequently, the novelty in this part consists of developing efficient holistic and simultaneous decisions for sensor selection with corresponding triggering times and sampling options. For example, in some cases, triggering two sensors using lower sampling frequencies will result in less energy consumption than when triggering only one of those sensors using high sampling frequency. The pool of variables is now large, which increases the complexity of the problem. More specifically, to choose M sensors out of N sensors and where each sensor has W_i different number of parameters from Objective 2 and H_i number of parameters from Objective 1, the number of possible solutions becomes:

$$\sum_{M=1}^{N} \left(\binom{N}{M} \prod_{i=1}^{M} (W_i \cdot H_i) \right) \tag{6.2}$$

6.2.2 Fusing Multiple Sensors for Context Recognition

In general, the approaches we proposed in this thesis are applied on individual sensors, except in Objective 3 where we used several sensors to extract context. However, we relied on previous literature works extracted from our ontology to deal with the multiple sensor utilization and fuse sensor data from multiple sources. With the evolution in wearable sensors, data fusion is becoming a necessity where multiple heterogeneous sensor data sources are being deployed. A key challenge is to decide at which level of abstraction should we apply fusion: data-level, feature-level, or decision-level [206]. Therefore, an interesting direction for future work is to investigate how fusion of sensors can be embedded in our proposed framework. In particular, we are interested in proposing a an energy efficient context-aware data fusion mechanism which can be integrated in our proposed framework.

6.2.3 Open Research Directions

Several technical challenges have been detected in the field of mobile sensing and context recognition. There are several research threads that still need further investigation so that pervasive applications can make full use of the available sensors to ubiquitously provide proactive services to the user. We will next highlight two of these interesting research opportunities that complement the context-aware dynamic designs we propose in this dissertation.

Mobile Cloud Computing

We are living in the era of Big Data where tremendous amounts of data are being collected. There are 3Vs that define the Big Data: volume, velocity, and variety. In the mobile sensing environment, these 3Vs are clearly obvious. Due to recent developments, there is a “variety” of data sources from multiple embedded and wearable sensors. This data can recognize several context information that describes the user and her environment. In addition, most applications use the recognized context in real-time to take the appropriate context-aware actions. Hence, “velocity” forms a critical component of Big Data. The continuous sensing of multiple sensors leads to a huge “volume” of data that needs to be stored and analyzed. Therefore, mobile cloud computing was recently introduced to combine cloud computing and mobile computing so that cloud resources can be used to store large data [207, 208]. However, it is not an easy task to implement this approach due to several challenges such as privacy, adaptability, mobility, network conditions, and transmission power concerns. Hence, this is still an open research topic especially with the emergence of new fifth generation networks that can offer a higher bandwidth [209].

Privacy and Trust

This aspect in context-aware applications becomes very critical when dealing with urban and participatory sensing where multiple sensors are deployed to share knowledge. For example, traffic monitoring applications rely on real-time context recognized multiple users such as their location and speed. We envision upcoming smart cities, hospitals and schools; however, data privacy constitutes a major challenge for these future advancements. The issue of privacy comes highly into sight when dealing specifically with health applications where information is even more delicate. Some attempts have been done to solve some of the trust challenges that arise when recognizing and sharing context information [210].

Appendix A

Abbreviations

IoT	Internet of Things
DNN	Deep Neural Network
PDA	Personal Device Assistant
FFT	Fast Fourier Transform
HCI	Human-Computer Interaction
EEMSS	Energy Efficient Mobile Sensing System
CMDP	Constrained Markov Decision Process
HAR	Human Activity Recognition
HMM	Hidden Markov Model
ADP	Android Developer Phone
ELM	Extreme Learning Machine
VESM	Virtual Environment Simulation Modeling
EDA	Electrodermal Activity
BVP	Blood Volume Pulse
ECG	Electrocardiogram
EMG	Electromyography
EEG	Electroencephalogram
MDP	Markov Decision Process
GPS	Global Positioning System
GSR	Galvanic Skin Response
HBR	Heart Beat Rate
ESS	Essential Sensor Set
ILP	Integer Linear Programming
HASC	Human Activity Sensing Consortium
DT	Decision Tree
NB	Naïve Bayes
RBM	Restricted Boltzmann Machine
CG	Conjugate Gradient
SGD	Stochastic Gradient Descent
LUT	Look Up Table

VCAMS	Viterbi-based Context Aware Mobile Sensing
VT	VCAMS adaptive trigger strategy
CT	Continuous Triggering
FP	False Positive
FN	False Negative
AIAD	Additive Increase Additive Decrease
ADL	Activities of Daily Living
QoS	Quality of Service
EGO	Energy Efficient Group Sensor Selection
OWL	Web Ontology Language
UML	Unified Modeling Language
PSD	Power Spectral Density
BA	Best Accuracy
BE	Best Energy
TO	Trade-Off

Bibliography

[1] M. Weiser, “The computer for the 21st century,” *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 3, no. 3, pp. 3–11, 1999.

[2] M. Weiser, “Some computer science issues in ubiquitous computing,” *Communications ACM*, vol. 36, no. 7, pp. 75–84, 1993.

[3] Intel, “The smart and connected to the cloud world: 2016 and beyond,” *Intel Newsroom*, 2016.

[4] Cisco, “Cisco visual networking index: Global mobile data traffic forecast update,” *White Paper*, 2017.

[5] R. Schaller, “Moores law: Past, present, and future,” *IEEE Spectrum*, vol. 34, no. 6, pp. 52–59, 1997.

[6] B. N. Schilit and M. M. Theimer, “Disseminating active map information to mobile hosts,” *IEEE Network*, vol. 8, pp. 22–32, Sept 1994.

[7] “US 10 billion-worth of smartwatches to ship in 2017 as traditional watchmakers feel the pressure.” https://www.canalys.com/newsroom/us10-billion-worth-smartwatches-ship-2017-traditional-watchmakers-feel-pressure.

[8] L. Atallah and G.-Z. Yang, “Review: The use of pervasive sensing for behaviour profiling - a survey,” *Pervasive Mobile Computing*, vol. 5, no. 5, pp. 447–464, 2009.

[9] R. Jain and L. Jalali, “Objective self,” *IEEE MultiMedia*, vol. 21, no. 4, pp. 100–110, 2014.

[10] “Apple - Apple Watch.” https://www.apple.com/watch/.

[11] N. Lane, P. Georgiev, C. Mascolo, and Y. Gao, “Zoe: A cloud-less dialog-enabled continuous sensing wearable exploiting heterogeneous computation,” in *13th ACM Conference on Mobile Systems, Applications, and Services (MobiSys ’15)*, pp. 273–286, May 2015.

[12] M. Shoaib, S. Bosch, H. Scholten, P. J. M. Havinga, and O. D. Incel, "Towards detection of bad habits by fusing smartphone and smartwatch sensors," in *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pp. 591–596, March 2015.

[13] E. Ohn-Bar and M. M. Trivedi, "Looking at humans in the age of self-driving and highly automated vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, pp. 90–104, March 2016.

[14] M. Skubic, "A ubiquitous sensing environment to detect functional changes in assisted living apartments: The tiger place experience," *Elsevier Alzheimers and dementia*, vol. 6, no. 4, pp. 1552–5260, 2010.

[15] M. Fahim, I. Fatima, S. Lee, and Y. Lee, "Daily life activity tracking application for smart homes using android smartphone," in *International Conf. on Adv. Commun. Tech.*, pp. 241–245, 2012.

[16] S. Hu, H. Wei, Y. Chen, and T. J., "A real-time cardiac arrhythmia classification system with wearable sensor networks," *Sensors*, vol. 12, no. 9, pp. 12 844–12 869, 2012.

[17] MarketResearch, "Big Data in Internet of Things (IoT): Key Trends, Opportunities and Market Forecasts 2015 2020." http://www.marketresearch.com/Mind-Commerce-Publishing-v3122/Big-Data-Internet-Things-IoT-8926222/, 2015.

[18] Gartner, "Top 10 strategic predictions for 2015 and beyond: Digital business is driving 'big change'," *Gartner Info*, 2014.

[19] A. Rehman, M. Mustafa, I. Israr, and M. Yaqoob, "Survey of wearable sensors with comparative study of noise reduction ecg filters," *International Journal of Computing and Network Technology*, vol. 1, no. 1, pp. 45–66, 2013.

[20] Óscar D. Lara, A. J. Pérez, M. A. Labrador, and J. D. Posada, "Centinela: A human activity recognition system based on acceleration and vital sign data," *Pervasive Mobile Computing*, vol. 8, no. 5, pp. 717–729, 2012.

[21] C.-W. Lin, Y.-T. Yang, J.-S. Wang, and Y.-C. Yang, "A wearable sensor module with a neural-network-based activity classification algorithm for daily energy expenditure estimation," *IEEE Transactions on Information Technology in Biomedicine*, vol. 16, no. 5, pp. 991–998, 2012.

[22] V. Osmani, "Smartphones in mental health: Detecting depressive and manic episodes," *IEEE Pervasive Computing*, vol. 14, no. 3, pp. 10–13, 2015.

[23] E. Ceja, V. Osmani, and O. Mayora, "Automatic stress detection in working environments from smartphones' accelerometer data: A first step," *IEEE Journal of Biomedical and Health Informatics*, 2015.

[24] K. Chen and G. Tan, "Modeling and improving the energy performance of GPS receivers for mobile applications," *CoRR*, vol. abs/1503.02656, 2015.

[25] W. Wibisono, D. Arifin, B. Pratomo, T. Ahmad, and R. Ijtihadie, "Falls detection and notification system using tri-axial accelerometer and gyroscope sensors of a smartphone," in *Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, pp. 382–385, 2013.

[26] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer series in statistics, Springer, 2001.

[27] J. Chon and H. Cha, "Lifemap: A smartphone-based context provider for location-based services," *IEEE Pervasive Computing*, vol. 10, pp. 58–67, 2011.

[28] N. Roy, A. Misra, and D. Cook, "Ambient and smartphone sensor assisted adl recognition in multi-inhabitant smart environments," *Journal of Ambient Intelligence and Humanized Computing*, vol. 7, no. 1, pp. 1–19, 2016.

[29] F. Miao, Y. Cheng, Y. He, Q. He, and Y. Li, "A wearable context-aware ecg monitoring system integrated with built-in kinematic sensors of the smartphone," *Sensors*, vol. 15, no. 5, pp. 11465–11484, 2015.

[30] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey," *IEEE Communications Surveys Tutorials*, vol. 16, pp. 414–454, First 2014.

[31] S. Taleb, H. Hajj, and Z. Dawy, "Entropy-based optimization to trade-off energy and accuracy for activity mobile sensing," in *4th Annual International Conference on Energy Aware Computing Systems and Applications (ICEAC)*, pp. 6–11, 2013.

[32] S. Taleb, A. A. Sallab, H. Hajj, Z. Dawy, R. Khanna, and A. Keshavamurthy, "Deep learning with ensemble classification method for sensor sampling decisions," in *2016 International Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 114–119, Sept 2016.

[33] S. Taleb, H. Hajj, and Z. Dawy, "Vcams: Viterbi-based context aware mobile sensing to trade-off energy and delay," *Submitted to IEEE Transactions on Mobile Computing*, 2016.

[34] S. Taleb, N. Abbas, H. Hajj, and Z. Dawy, "On sensor selection in mobile devices based on energy, application accuracy, and context metrics," in *3rd International Conference on Communications and Information Technology (ICCIT)*, pp. 12–16, 2013.

[35] S. Taleb, H. Hajj, and Z. Dawy, "Ego: An ontology-based framework for optimized selection of sensor groups feeding simultaneous execution of different context-aware applications," *Submitted to IEEE Transactions*, 2017.

[36] A. K. Dey, "Understanding and using context," *Personal and Ubiquitous Computing*, vol. 5, no. 1, pp. 4–7, 2001.

[37] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, *Towards a Better Understanding of Context and Context-Awareness*, pp. 304–307. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999.

[38] M. J. Gajjar, *Mobile Sensors and Context-Aware Computing.* Morgan Kaufmann, 2017.

[39] S. James, "Understanding virtual sensors: From sensor fusion to context-aware applications," *Sensor Platforms*, July 2012.

[40] S. Saeedi, A. Moussa, and N. El-Sheimy, "Context-aware personal navigation using embedded sensor fusion in smartphones," *Sensors*, vol. 14, no. 4, pp. 5742–5767, 2014.

[41] F. J. Ordóñez and D. Roggen, "Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition," *Sensors*, vol. 16, no. 1, p. 115, 2016.

[42] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?," *J. Mach. Learn. Res.*, vol. 11, pp. 625–660, Mar. 2010.

[43] A. Pantelopoulos and N. G. Bourbakis, "A survey on wearable sensor-based systems for health monitoring and prognosis," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 40, pp. 1–12, Jan 2010.

[44] A. AlzeerAlhouseini, I. Al-Shaikhli, A. W. AbdulRahman, and M. Dzulkifli, "Emotion detection using physiological signals eeg and ecg," *International Journal of Advancements in Computing Technology (IJACT)*, vol. 8, pp. 103–112, June 2016.

[45] Z. Zhuang, K.-H. Kim, and J. P. Singh, "Improving energy efficiency of location sensing on smartphones," in *8th International Conference on Mobile Systems, Applications, and Services (MobiSys '10)*, pp. 315–330, 2010.

[46] X. Qi, M. Keally, G. Zhou, Y. Li, and Z. Ren, "Adasense: adapting sampling rates for activity recognition in body sensor networks," in *19th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pp. 163–172, 2013.

[47] K. K. Rachuri, C. Mascolo, M. Musolesi, and P. J. Rentfrow, "Sociablesense: Exploring the trade-offs of adaptive sampling and computation offloading for social sensing," in *17th Annual International Conference on Mobile Computing and Networking (MobiCom '11)*, pp. 73–84, 2011.

[48] "Electricsleep." https://play.google.com/store/apps/details?id=com.androsz.electricsleepbeta.

[49] "Smart thermometer." https://play.google.com/store/apps/details?id=com.colortiger.thermo.

[50] J. Wang, Y. Liu, C. Xu, X. Ma, and J. Lu, "E-greendroid: Effective energy inefficiency analysis for android applications," in *8th Asia-Pacific Symposium on Internetware*, Internetware '16, (New York, NY, USA), pp. 71–80, ACM, 2016.

[51] "Mapmyrun." http://www.mapmyrun.com/.

[52] "Muse: the brain sensing headband." http://www.choosemuse.com. Accessed: 2017-03-20.

[53] S. R. Gouravajhala, D. Wang, L. Khuon, and F. S. Bao, "Epsmart: Epileptic seizure monitoring with alerts in real time: A tablet-based android application for a real-time multi-modal seizure detection system," in *Bioinformatics and Biomedicine Workshops (BIBMW), 2012 IEEE International Conference on*, pp. 959–961, Oct 2012.

[54] G. Singla, D. J. Cook, and M. Schmitter-Edgecombe, "Recognizing independent and joint activities among multiple residents in smart environments," *Journal of Ambient Intelligence and Humanized Computing*, vol. 1, no. 1, pp. 57–63, 2010.

[55] A. K. Dey, K. Wac, D. Ferreira, K. Tassini, J.-H. Hong, and J. Ramos, "Getting closer: An empirical investigation of the proximity of user to their smart phones," in *Proceedings of the 13th International Conference on Ubiquitous Computing*, UbiComp '11, (New York, NY, USA), pp. 163–172, ACM, 2011.

[56] W. Wang, L. Yu, H. Liu, and F. Sun, *Extreme Learning Machine for Linear Dynamical Systems Classification: Application to Human Activity Recognition*, pp. 11–20. Cham: Springer International Publishing, 2015.

[57] M. Hasan and A. K. Roy-Chowdhury, "A continuous learning framework for activity recognition using deep hybrid feature models," *IEEE Transactions on Multimedia*, vol. 17, pp. 1909–1922, Nov 2015.

[58] N. D. Lane and P. Georgiev, "Can deep learning revolutionize mobile sensing?," in *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, HotMobile '15, (New York, NY, USA), pp. 117–122, ACM, 2015.

[59] A. Poshtkar, V. Elangovan, A. Shirkhodaie, A. Chan, and S. Hu, "Physical environment virtualization for human activities recognition," *Proc. SPIE*, vol. 9478, pp. 94780I–94780I–12, 2015.

[60] A. Khan, A. Tufail, A. Khattak, and T. Laine, "Activity recognition on smartphones via sensor-fusion and kda-based svms," *International Journal of Distributed Sensor Networks*, vol. 2014, 2014.

[61] L. Sun, D. Zhang, B. Li, B. Guo, and S. Li, "Activity recognition on an accelerometer embedded mobile phone with varying positions and orientations," in *Proceedings of the 7th International Conference on Ubiquitous Intelligence and Computing*, UIC'10, (Berlin, Heidelberg), pp. 548–562, Springer-Verlag, 2010.

[62] N. A. Capela, E. D. Lemaire, and N. Baddour, "Feature selection for wearable smartphone based human activity recognition with able bodied, elderly, and stroke patients," *PloS one*, vol. 10, no. 4, 2015.

[63] A. Shahroudy, T. T. Ng, Q. Yang, and G. Wang, "Multimodal multipart learning for action recognition in depth videos," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, pp. 2123–2129, Oct 2016.

[64] X. Yang and Y. Tian, "Super normal vector for human activity recognition with depth cameras," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 1028–1039, May 2017.

[65] K. Zickuhr, "Location-based services," *Pew Research*, pp. 679–695, 2013.

[66] M. T. Hallworth and P. P. Marra, "Miniaturized gps tags identify non-breeding territories of a small breeding migratory songbird," *Scientific reports*, vol. 5, p. 11069, 2015.

[67] V. W. Zheng, Y. Zheng, X. Xie, and Q. Yang, “Towards mobile intelligence: Learning from gps history data for collaborative recommendation,” *Artif. Intell.*, vol. 184-185, pp. 17–37, June 2012.

[68] L. Stenneth, O. Wolfson, P. S. Yu, and B. Xu, “Transportation mode detection using mobile phones and gis information,” in *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS ’11, (New York, NY, USA), pp. 54–63, ACM, 2011.

[69] X. Sheng, J. Tang, X. Xiao, and G. Xue, “Leveraging gps-less sensing scheduling for green mobile crowd sensing,” *IEEE Internet of Things Journal*, vol. 1, pp. 328–336, Aug 2014.

[70] T. O. Oshin, S. Poslad, and A. Ma, “A method to evaluate the energy-efficiency of wide-area location determination techniques used by smartphones,” in *Computational Science and Engineering (CSE), 2012 IEEE 15th International Conference on*, pp. 326–333, Dec 2012.

[71] A. Farshad, J. Li, M. K. Marina, and F. J. Garcia, “A microscopic look at wifi fingerprinting for indoor mobile phone localization in diverse environments,” in *International Conference on Indoor Positioning and Indoor Navigation*, pp. 1–10, Oct 2013.

[72] T. Y.-H. Chen, A. Sivaraman, S. Das, L. Ravindranath, and H. Balakrishnan, “Designing a context-sensitive context detection service for mobile devices,” in *DSpace at MIT*, Sep 2015.

[73] K. Sankaran, M. Zhu, X. F. Guo, A. L. Ananda, M. C. Chan, and L.-S. Peh, “Using mobile phone barometer for low-power transportation context detection,” in *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, pp. 191–205, ACM, 2014.

[74] R. Faragher and R. Harle, “Location fingerprinting with bluetooth low energy beacons,” *IEEE journal on Selected Areas in Communications*, vol. 33, no. 11, pp. 2418–2428, 2015.

[75] Y. Zheng, Y. Chen, Q. Li, X. Xie, and W.-Y. Ma, “Understanding transportation modes based on gps data for web applications,” *ACM Trans. Web*, vol. 4, pp. 1:1–1:36, Jan. 2010.

[76] X. J. Ban and M. Gruteser, “Towards fine-grained urban traffic knowledge extraction using mobile sensing,” in *ACM SIGKDD International Workshop on Urban Computing*, UrbComp ’12, (New York, NY, USA), pp. 111–117, ACM, 2012.

[77] T. Alldieck, C. H. Bahnsen, and T. B. Moeslund, “Context-aware fusion of rgb and thermal imagery for traffic monitoring,” *Sensors*, vol. 16, no. 11, p. 1947, 2016.

[78] M.-H. Amri, Y. Becis, D. Aubry, and N. Ramdani, “Indoor human/robot localization using robust multi-modal data fusion,” in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pp. 3456–3463, IEEE, 2015.

[79] MarketsandMarkets, “Emotion detection and recognition market by technology (bio-sensor, nlp, machine learning), software tool (facial expression, voice recognition), service, application area, end user, and region - global forecast to 2021,” *Top Market Reports*, Dec 2016.

[80] J. Hernandez, M. E. Hoque, W. Drevo, and R. W. Picard, “Mood meter: Counting smiles in the wild,” in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp ’12, (New York, NY, USA), pp. 301–310, ACM, 2012.

[81] F. Canento, A. Fred, H. Silva, H. Gamboa, and A. Loureno, “Multimodal biosignal sensor data handling for emotion recognition,” in *2011 IEEE SENSORS Proceedings*, pp. 647–650, Oct 2011.

[82] R. B. Hossain, M. Sadat, and H. Mahmud, “Recognition of human affection in smartphone perspective based on accelerometer and user’s sitting position,” in *2014 17th International Conference on Computer and Information Technology (ICCIT)*, pp. 87–91, Dec 2014.

[83] D. Amelynck, M. Grachten, L. van Noorden, and M. Leman, “Toward e-motion-based music retrieval a study of affective gesture recognition,” *IEEE Transactions on Affective Computing*, vol. 3, pp. 250–259, April 2012.

[84] R. LiKamWa, Y. Liu, N. D. Lane, and L. Zhong, “Moodscope: Building a mood sensor from smartphone usage patterns,” in *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys ’13, (New York, NY, USA), pp. 389–402, ACM, 2013.

[85] “Affectiva’s emotion ai humanizes how people and technology interact.” http://www.affectiva.com/.

[86] “Real-time emotion measurement.” http://bodymonitor.de/.

[87] “Muse Monitor - See what’s really going on inside your head! Real time EEG graphs from your Interaxon Muse headband.” http://www.musemonitor.com/.

[88] R. S. H. Istepanian, E. Jovanov, and Y. T. Zhang, "Guest editorial introduction to the special section on m-health: Beyond seamless mobility and global wireless health-care connectivity," *IEEE Transactions on Information Technology in Biomedicine*, vol. 8, pp. 405–414, Dec 2004.

[89] G. Jonathan, "30 amazing mobile health technology statistics for today's physician," *referralMD*, August 2015.

[90] A. Madan, M. Cebrian, S. Moturu, K. Farrahi, and A. Pentland, "Sensing the health state of a community," *IEEE Pervasive Computing*, vol. 11, no. 4, pp. 36–45, 2012.

[91] "NeuroSky, body and mind." http://neurosky.com/.

[92] P. Bonato, "Wearable sensors and systems," *IEEE Engineering in Medicine and Biology Magazine*, vol. 29, no. 3, pp. 25–36, 2010.

[93] C. Duc, P. Salvia, A. Lubansu, V. Feipel, and K. Aminian, "A wearable inertial system to assess the cervical spine mobility: Comparison with an optoelectronic-based motion capture evaluation," *Medical Engineering and Physics*, vol. 36, no. 1, pp. 49–56, 2014.

[94] V. Pejovic and M. Musolesi, "Anticipatory mobile computing: A survey of the state of the art and research challenges," *ACM Comput. Surv.*, vol. 47, pp. 47:1–47:29, Apr. 2015.

[95] S. Kang, J. Lee, H. Jang, H. Lee, Y. Lee, S. Park, T. Park, and J. Song, "Seemon: scalable and energy-efficient context monitoring framework for sensor-rich mobile environments," in *6th International Conference on Mobile Systems, Applications, and Services (MobiSys '08)*, pp. 267–280, 2008.

[96] S. Kang, J. Lee, H. Jang, Y. Lee, S. Park, and J. Song, "A scalable and energy-efficient context monitoring framework for mobile personal sensor networks," *IEEE Transactions on Mobile Computing*, vol. 9, no. 5, pp. 686–702, 2010.

[97] Y. Wang, J. Lin, M. Annavaram, Q. A. Jacobson, J. Hong, B. Krishnamachari, and N. Sadeh, "A framework of energy efficient mobile sensing for automatic user state recognition," in *7th International Conference on Mobile Systems, Applications, and Services (MobiSys '09)*, pp. 179–192, 2009.

[98] Y. Wang, B. Krishnamachari, Q. Zhao, and M. Annavaram, "Markov-optimal sensing policy for user state estimation in mobile devices," in *9th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN '10)*, pp. 268–278, 2010.

[99] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell, "The jigsaw continuous sensing engine for mobile phone applications," in *8th ACM Conference on Embedded Networked Sensor Systems (SenSys '10)*, pp. 71–84, 2010.

[100] O. Yurur, C. Liu, X. Liu, and W. Moreno, "Adaptive sampling and duty cycling for smartphone accelerometer," in *Mobile Ad-Hoc and Sensor Systems (MASS), 2013 IEEE 10th International Conference on*, pp. 511–518, 2013.

[101] Y. Lee, C. Min, Y. Ju, S. Kang, Y. Rhee, and J. Song, "An active resource orchestration framework for pan-scale, sensor-rich environments," *IEEE Transactions on Mobile Computing*, vol. 13, pp. 596–610, March 2014.

[102] G. Chen and D. Kotz, "A Survey of Context-Aware Mobile Computing Research," Tech. Rep. TR2000-381, Dartmouth College, Computer Science, Hanover, NH, November 2000.

[103] P. Zappi, C. Lombriser, T. Stiefmeier, E. Farella, D. Roggen, L. Benini, and G. Tröster, *Activity Recognition from On-Body Sensors: Accuracy-Power Trade-Off by Dynamic Sensor Selection*, pp. 17–33. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[104] E. Miluzzo, C. T. Cornelius, A. Ramaswamy, T. Choudhury, Z. Liu, and A. T. Campbell, "Darwin phones: The evolution of sensing and inference on mobile phones," in *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, MobiSys '10, (New York, NY, USA), pp. 5–20, ACM, 2010.

[105] D. Chu, N. D. Lane, T. T.-T. Lai, C. Pang, X. Meng, Q. Guo, F. Li, and F. Zhao, "Balancing energy, latency and accuracy for mobile sensor data classification," in *9th ACM Conference on Embedded Networked Sensor Systems (SenSys '11)*, pp. 54–67, 2011.

[106] K. K. Rachuri, C. Mascolo, and M. Musolesi, *Energy-Accuracy Trade-offs of Sensor Sampling in Smart Phone Based Sensing Systems*, pp. 65–76. London: Springer London, 2012.

[107] K. K. Rachuri, M. Musolesi, C. Mascolo, P. J. Rentfrow, C. Longworth, and A. Aucinas, "Emotionsense: A mobile phones based adaptive platform for experimental social psychology research," in *Proceedings of the 12th ACM International Conference on Ubiquitous Computing*, UbiComp '10, (New York, NY, USA), pp. 281–290, ACM, 2010.

[108] Z. Yan, D. Chakraborty, A. Misra, H. Jeung, and K. Aberer, "Sammple: Detecting semantic indoor activities in practical settings using locomotive

signatures," in *16th Annual International Symposium on Wearable Computers (ISWC)*, ISWC '12, (Washington, DC, USA), pp. 37–40, IEEE Computer Society, 2012.

[109] Z. Yan, D. Chakraborty, C. Parent, S. Spaccapietra, and K. Aberer, "Semitri: A framework for semantic annotation of heterogeneous trajectories," in *Proceedings of the 14th International Conference on Extending Database Technology*, EDBT/ICDT '11, (New York, NY, USA), pp. 259–270, ACM, 2011.

[110] N. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. Campbell, "A survey of mobile phone sensing," *IEEE Communications Magazine*, vol. 48, no. 9, pp. 140–150, 2010.

[111] S. A. Hoseini-Tabatabaei, A. Gluhak, and R. Tafazolli, "A survey on smartphone-based systems for opportunistic user context recognition," *ACM Comput. Surv.*, vol. 45, pp. 27:1–27:51, July 2013.

[112] O. Yurur, C. Liu, Z. Sheng, V. Leung, W. Moreno, and K. Leung, "Context-awareness for mobile sensing: A survey and future directions," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 68–93, 2016.

[113] T. Rault, A. Bouabdallah, Y. Challal, and F. Marin, "A survey of energy-efficient context recognition systems using wearable sensors for healthcare applications," *Pervasive and Mobile Computing*, vol. 37, pp. 23–44, 2017.

[114] R. Pérez-Torres, C. Torres-Huitzil, and H. Galeana-Zapién, "Power management techniques in smartphone-based mobility sensing systems: A survey," *Pervasive and Mobile Computing*, 2016.

[115] U. Maurer, A. Smailagic, D. P. Siewiorek, and M. Deisher, "Activity recognition and monitoring using multiple sensors on different body positions," in *Proceedings of the International Workshop on Wearable and Implantable Body Sensor Networks*, BSN '06, (Washington, DC, USA), pp. 113–116, IEEE Computer Society, 2006.

[116] B. French, D. P. Siewiorek, A. Smailagic, and M. Deisher, "Selective sampling strategies to conserve power in context aware devices," in *Proceedings of the 2007 11th IEEE International Symposium on Wearable Computers*, ISWC '07, (Washington, DC, USA), pp. 1–4, IEEE Computer Society, 2007.

[117] Z. Yan, V. Subbaraju, D. Chakraborty, A. Misra, and K. Aberer, "Energy-efficient continuous activity recognition on mobile phones: an activity-adaptive approach," in *16th Annual International Symposium on Wearable Computers (ISWC '12)*, pp. 17–24, 2012.

[118] J. Paek, J. Kim, and R. Govindan, “Energy-efficient rate-adaptive gps-based positioning for smartphones,” in *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, MobiSys ’10, (New York, NY, USA), pp. 299–314, ACM, 2010.

[119] K. Lin, A. Kansal, D. Lymberopoulos, and F. Zhao, “Energy-accuracy trade-off for continuous mobile device location,” in *8th International Conference on Mobile Systems, Applications, and Services*, MobiSys ’10, pp. 285–298, 2010.

[120] F. Casamassima, E. Farella, and L. Benini, “Context aware power management for motion-sensing body area network nodes,” in *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1–6, March 2014.

[121] A. Krause, M. Ihmig, E. Rankin, D. Leong, S. Gupta, D. Siewiorek, A. Smailagic, M. Deisher, and U. Sengupta, “Trading off prediction accuracy and power consumption for context-aware wearable computing,” in *Ninth IEEE International Symposium on Wearable Computers*, pp. 20–26, 2005.

[122] Y. Li, Y. Guo, J. Kong, and X. Chen, “Fixing sensor-related energy bugs through automated sensing policy instrumentation,” in *2015 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 321–326, July 2015.

[123] Y. Chon, E. Talipov, H. Shin, and H. Cha, “Smartdc: Mobility prediction-based adaptive duty cycling for everyday location monitoring,” *IEEE Transactions on Mobile Computing*, vol. 13, pp. 512–525, March 2014.

[124] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson, “Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones,” in *7th ACM Conference on Embedded Networked Sensor Systems (SenSys ’09)*, pp. 85–98, 2009.

[125] O. Yurur, M. Labrador, and W. Moreno, “Adaptive and energy efficient context representation framework in mobile sensing,” *IEEE Transactions on Mobile Computing*, vol. 13, no. 8, pp. 1681–1693, 2014.

[126] O. Yurur, C. Liu, C. Perera, M. Chen, X. Liu, and W. Moreno, “Energy-efficient and context-aware smartphone sensor employment,” *Vehicular Technology, IEEE Transactions on*, vol. 64, no. 9, pp. 4230–4244, 2015.

[127] K. K. Rachuri, C. Mascolo, and M. Musolesi, “Energy-accuracy tradeoffs in querying sensor data for continuous sensing mobile systems,” in *Mobile Context-Awareness Workshop*, 2010.

[128] V. Agarwal, N. Banerjee, D. Chakraborty, and S. Mittal, "Usense – a smartphone middleware for community sensing," in *2013 IEEE 14th International Conference on Mobile Data Management*, vol. 1, pp. 56–65, June 2013.

[129] P. Baier, F. Durr, and K. Rothermel, "Psense: Reducing energy consumption in public sensing systems," in *Proceedings of the 2012 IEEE 26th International Conference on Advanced Information Networking and Applications*, AINA '12, (Washington, DC, USA), pp. 136–143, IEEE Computer Society, 2012.

[130] Y. Man and E. C. H. Ngai, "Energy-efficient automatic location-triggered applications on smartphones," *Comput. Commun.*, vol. 50, pp. 29–40, Sept. 2014.

[131] H. Noshadi, F. Dabiri, S. Meguerdichian, M. Potkonjak, and M. Sarrafzadeh, "Energy optimization in wireless medical systems using physiological behavior," in *Wireless Health 2010*, WH '10, (New York, NY, USA), pp. 128–136, ACM, 2010.

[132] C. Lombriser, R. Marin-Perianu, D. Roggen, P. Havinga, and G. Troster, "Modeling service-oriented context processing in dynamic body area networks," *IEEE Journal on Selected Areas in Communications*, vol. 27, pp. 49–57, January 2009.

[133] D. Gordon, J. Czerny, T. Miyaki, and M. Beigl, "Energy-efficient activity recognition using prediction," in *2012 16th International Symposium on Wearable Computers*, pp. 29–36, June 2012.

[134] L. Gao, A. K. Bourke, and J. Nelson, "Activity recognition using dynamic multiple sensor fusion in body sensor networks," in *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 1077–1080, Aug 2012.

[135] N. Roy, A. Misra, S. K. Das, and C. Julien, "Determining quality- and energy-aware multiple contexts in pervasive computing environments," *IEEE/ACM Transactions on Networking*, vol. 24, pp. 3026–3042, Oct 2016.

[136] S. Chaudhuri, H. Thompson, and G. Demiris, "Fall detection devices and their use with older adults: A systematic review," *J. Geriatric Phys. Therapy*, vol. 37, no. 4, pp. 178–198, 2014.

[137] B. Cole, S. Roy, C. DeLuca, and S. Nawab, "Dynamical learning and tracking of tremor and dyskinesia from wearable sensors," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 22, no. 5, pp. 982–991, 2014.

[138] 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, pp. 800–812, April 2015.

[139] L. Bao and S. S. Intille, “Activity recognition from user-annotated acceleration data,” in *2nd International Conference on Pervasive Computing*, pp. 1–17, April 2004.

[140] M. A. Gennert and A. Yuille, “Determining the optimal weights in multiple objective function optimization,” in *2nd International Conference on Computer Vision*, pp. 87–89, December 1988.

[141] U. M. Fayyad and K. B. Irani, “Multi-interval discretization of continuous-valued attributes for classification learning,” in *13th International Joint Conference on Artificial Intelligence*, pp. 1022–1027, August 1993.

[142] V. A. Kovalevskij, “The problem of character recognition from the point of view of mathematical statistics,” in *Character Readers and Pattern Recognition*, pp. 3–30, 1967.

[143] N. Kawaguchi, N. Ogawa, Y. Iwasaki, K. Kaji, T. Terada, K. Murao, S. Inoue, Y. Kawahara, Y. Sumi, and N. Nishio, “Hasc challenge: gathering large scale human activity corpus for the real-world activity understandings,” in *2nd Augmented Human International Conference (AH ’11)*, pp. 27:1–27:5, March 2011.

[144] “PowerTutor: A Power Monitor for Android-Based Mobile Platforms.” http://powertutor.org/.

[145] L. Zhang, B. Tiwana, R. Dick, Z. Qian, Z. Mao, Z. Wang, and L. Yang, “Accurate online power estimation and automatic battery behavior based power model generation for smartphones,” in *IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pp. 105–114, October 2010.

[146] T. G. Dietterich, “Ensemble methods in machine learning,” in *First International Workshop on Multiple Classifier Systems*, MCS ’00, pp. 1–15, 2000.

[147] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems).* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005.

[148] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics).* Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[149] G. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep neural networks," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[150] D. Yu, M. L. Seltzer, J. Li, J. ting Huang, and F. Seide, "Feature learning in deep neural networks studies on speech recognition tasks," in *ICLR*, 2013.

[151] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[152] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[153] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng, "On optimization methods for deep learning," in *ICML*, pp. 265–272, 2011.

[154] "Google Play Wiki Page." http://en.wikipedia.org/wiki/ GooglePlay/.

[155] J.-y. Hong, E.-h. Suh, and S.-J. Kim, "Context-aware systems," *Expert Syst. Appl.*, vol. 36, pp. 8509–8522, May 2009.

[156] J. K. Lee, S. N. Robinovitch, and E. J. Park, "Inertial sensing-based pre-impact detection of falls involving near-fall scenarios," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 23, pp. 258–266, March 2015.

[157] X. Zhang and Y. Lian, "A 300-mv 220-nw event-driven adc with real-time qrs detection for wearable ecg sensors," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 8, pp. 834–843, December 2014.

[158] G. Forney Jr., "The viterbi algorithm," *Proc. IEEE*, vol. 61, pp. 268–278, March 1973.

[159] N. Viol, J. Link, H. Wirtz, D. Rothe, and K. Wehrle, "Hidden markov model-based 3d path-matching using raytracing-generated wi-fi models," in *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–10, November 2012.

[160] Z. Ma and A. W. Krings, "Survival analysis approach to reliability, survivability and prognostics and health management (phm)," in *IEEE Aerospace Conference*, pp. 1–20, March 2008.

[161] T. Aven and U. Jensen, *Stochastic Models in Reliability.* Applications of mathematics, Springer, 1999.

[162] L. White and H. Vu, “Maximum likelihood sequence estimation for hidden reciprocal processes,” *IEEE Transactions on Automatic Control*, vol. 58, pp. 2670–2674, October 2013.

[163] E. Ramasso and T. Denoeux, “Making use of partial knowledge about hidden states in hmms: An approach based on belief functions,” *IEEE Transactions on Fuzzy Systems*, vol. 22, pp. 395–405, April 2014.

[164] Q. You, Y. Li, Z. Chen, and M. S. Rahman, “A simple near-optimal path selection scheme for multi-hop wireless relay networks based on viterbi algorithm,” *Transactions on Emerging Telecommunications Technologies*, 2014.

[165] M. Eljourmi, H. Elghazi, A. Bennis, and H. Ouahmane, “Performance analysis of channel coding in satellite communication based on vsat network and mc-cdma scheme,” *WSEAS Transaction on Communication*, vol. 12, May 2013.

[166] A. Viterbi, “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm,” *IEEE Transactions on Information Theory*, vol. 13, pp. 260–269, April 1967.

[167] A. Desiraju, M. Torlak, and M. Saquib, “Multiple-attempt decoding of convolutional codes over rayleigh channels,” *IEEE Transactions on Vehicular Technology*, vol. 64, pp. 3426–3439, Aug 2015.

[168] Y. Chen, V. P. Jilkov, and X. R. Li, “Multilane-road target tracking using radar and image sensors,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 51, pp. 65–80, January 2015.

[169] A. Domingues, T. Paiva, and J. M. Sanches, “Hypnogram and sleep parameter computation from activity and cardiovascular data,” *IEEE Transactions on Biomedical Engineering*, vol. 61, pp. 1711–1719, June 2014.

[170] Z. Wang, M. Guo, and C. Zhao, “Badminton stroke recognition based on body sensor networks,” *IEEE Transactions on Human-Machine Systems*, vol. 46, pp. 769–775, Oct 2016.

[171] S. Ruzika and M. Wiecek, “Approximation methods in multiobjective programming,” *Journal of Optimization Theory and Applications*, vol. 126, no. 3, pp. 473–501, 2005.

[172] M. Ehrgott, “A discussion of scalarization techniques for multiple objective integer programming,” *Annals of Operations Research*, vol. 147, no. 1, pp. 343–360, 2006.

[173] S. Gass and T. Saaty, "The computational algorithm for the parametric objective function," *Naval Research Logistics Quarterly*, vol. 2, pp. 39–45, 1955.

[174] A. Vaisman and E. Zimányi, *Data Warehouse Systems: Design and Implementation*, ch. Data Warehouse Concepts, pp. 53–87. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014.

[175] F. Akdag and C. Eick, "An optimized interestingness hotspot discovery framework for large gridded spatio-temporal datasets," in *2015 IEEE International Conference on Big Data*, pp. 2010–2019, Oct 2015.

[176] S. Chatterjee and S. Russell, "A temporally abstracted viterbi algorithm," *UAI*, pp. 96–104, 2011.

[177] J. K. Wolf, "Efficient maximum likelihood decoding of linear block codes using a trellis," *IEEE Transactions in Information Theory*, vol. 24, no. 1, pp. 76–80, 1978.

[178] M. Martin and P. Nurmi, "A generic large scale simulator for ubiquitous computing," in *3rd Annual International Conference on Mobile and Ubiquitous Systems*, pp. 1–3, July 2006.

[179] A. Eyal, L. Rokach, M. Kalech, O. Amir, R. Chougule, R. Vaidyanathan, and K. Pattada, "Survival analysis of automobile components using mutually exclusive forests," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 44, pp. 246–253, Feb 2014.

[180] F. Ordonez, P. de Toledo, and A. Sanchis, "Activity recognition using hybrid generative/discriminative models on home environments using binary sensors," *Sensors*, vol. 13, pp. 5460–5477, 2013.

[181] R. Jia and B. Liu, "Human daily activity recognition by fusing accelerometer and multi-lead ecg data," in *2013 IEEE International Conference on Signal Processing, Communication and Computing (ICSPCC 2013)*, pp. 1–4, Aug 2013.

[182] D. Riboni and C. Bettini, "Cosar: Hybrid reasoning for context-aware activity recognition," *Personal Ubiquitous Comput.*, vol. 15, pp. 271–289, Mar. 2011.

[183] H. Martín, A. M. Bernardos, J. Iglesias, and J. R. Casar, "Activity logging using lightweight classification techniques in mobile devices," *Personal Ubiquitous Comput.*, vol. 17, pp. 675–695, Apr. 2013.

[184] Y. Ma, B. Xu, Y. Bai, G. Sun, and R. Zhu, "Daily mood assessment based on mobile phone sensing," in *2012 Ninth International Conference on Wearable and Implantable Body Sensor Networks*, pp. 142–147, May 2012.

[185] Z. Zhang, Y. Song, L. Cui, X. Liu, and T. Zhu, "Emotion recognition based on customized smart bracelet with built-in accelerometer," in *PeerJ*, 2016.

[186] M. Cai, W. Y. Zhang, and K. Zhang, "Manuhub: A semantic web system for ontology-based service management in distributed manufacturing environments," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 41, pp. 574–582, May 2011.

[187] L. Razmerita, "An ontology-based framework for modeling user behavior; a case study in knowledge management," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 41, pp. 772–783, July 2011.

[188] N. D. Rodríguez, M. P. Cuéllar, J. Lilius, and M. D. Calvo-Flores, "A survey on ontologies for human behavior recognition," *ACM Comput. Surv.*, vol. 46, pp. 43:1–43:33, Mar. 2014.

[189] L. Chen, C. Nugent, M. Mulvenna, D. Finlay, and X. Hong, *Semantic Smart Homes: Towards Knowledge Rich Assisted Living Environments*, pp. 279–296. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[190] D. Ejigu, M. Scuturici, and L. Brunie, "An ontology-based approach to context modeling and reasoning in pervasive computing," in *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops '07. Fifth Annual IEEE International Conference on*, pp. 14–19, March 2007.

[191] U. Akdemir, P. Turaga, and R. Chellappa, "An ontology based approach for activity recognition from video," in *16th ACM International Conference on Multimedia*, MM '08, (New York, NY, USA), pp. 709–712, ACM, 2008.

[192] N. F. Noy and D. L. McGuinness, "Ontology development 101: A guide to creating your first ontology," tech. rep., Stanford, CA, 2001.

[193] L. Balzer, M. Do, and D. Maseluk, "Comparison and evaluation of ontology visualizations," *05 Fakultt Informatik, Elektrotechnik und Informationstechnik*, 2015.

[194] J. Bārzdiņš, G. Bārzdiņš, K. Čerāns, R. Liepiņš, and A. Sproģis, "Owlgred: a uml style graphical notation and editor for owl 2," in *7th International Workshop OWL: Experience and Directions (OWLED-2010)*, Citeseer, 2010.

[195] S. Ilarri, R. Hermoso, R. Trillo-Lado, and M. d. C. Rodríguez-Hernández, “A review of the role of sensors in mobile context-aware recommendation systems,” *Int. J. Distrib. Sen. Netw.*, vol. 2015, pp. 226:226–226:226, Jan. 2016.

[196] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations.* New York, NY, USA: John Wiley & Sons, Inc., 1990.

[197] “Protégé: A free, open-source ontology editor and framework for building intelligent systems.” http://protege.stanford.edu.

[198] S. Tarkoma, M. Siekkinen, E. Lagerspetz, and X. Yu, *Smartphone Energy Consumption: Modeling and Optimization.* New York, NY, USA: Cambridge University Press, 2014.

[199] “Fitbit blaze.” http://www.fitbit.com. Accessed: 2017-03-20.

[200] “View/Download your personal Fitbit data!.” https://www.squashleagues.org/Fitbit/FitbitDataDownload.

[201] R. Fallahzadeh, Y. Ma, and H. Ghasemzadeh, “Context-aware system design for remote health monitoring: An application to continuous edema assessment,” *IEEE Transactions on Mobile Computing*, vol. PP, no. 99, pp. 1–1, 2016.

[202] S. Koldijk, M. A. Neerincx, and W. Kraaij, “Detecting work stress in offices by combining unobtrusive sensors,” *IEEE Transactions on Affective Computing*, vol. PP, no. 99, pp. 1–1, 2016.

[203] F. Piccialli and A. Chianese, “The internet of things supporting context-aware computing: A cultural heritage case study,” *Mobile Networks and Applications*, vol. 22, no. 2, pp. 332–343, 2017.

[204] D. Lee and K. Roy, “Viterbi-based efficient test data compression,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, pp. 610–619, April 2012.

[205] J. C. Greene and V. J. Caracelli, “Defining and describing the paradigm issue in mixed-method evaluation,” *New Directions for Evaluation*, pp. 5–17, 1997.

[206] R. Gravina, P. Alinia, H. Ghasemzadeh, and G. Fortino, “Multi-sensor fusion in body sensor networks: State-of-the-art and research challenges,” *Information Fusion*, vol. 35, pp. 68–80, 2017.

[207] C. Zhu, V. C. M. Leung, L. T. Yang, and L. Shu, "Collaborative location-based sleep scheduling for wireless sensor networks integratedwith mobile cloud computing," *IEEE Transactions on Computers*, vol. 64, pp. 1844–1856, July 2015.

[208] H. Lin, J. Hu, Y. Tian, L. Yang, and L. Xu, "Toward better data veracity in mobile cloud computing: A context-aware and incentive-based reputation mechanism," *Information Sciences*, vol. 387, pp. 238–253, 2017.

[209] M. Agiwal, A. Roy, and N. Saxena, "Next generation 5g wireless networks: A comprehensive survey," *IEEE Communications Surveys Tutorials*, vol. 18, pp. 1617–1655, thirdquarter 2016.

[210] A. Martinez-Balleste, P. A. Perez-martinez, and A. Solanas, "The pursuit of citizens' privacy: a privacy-aware smart city is possible," *IEEE Communications Magazine*, vol. 51, pp. 136–141, June 2013.