A Learning-based Approach to Confident Event Detection in Heterogeneous Sensor Networks

MATTHEW KEALLY, College of William and Mary
GANG ZHOU, College of William and Mary
GUOLIANG XING, Michigan State University
DAVID T. NGUYEN, College of William and Mary
XIN QI, College of William and Mary

Wireless sensor network applications, such as those for natural disaster warning, vehicular traffic monitoring, and surveillance, have stringent accuracy requirements for detecting or classifying events and demand long system lifetimes. Through quantitative study, we show that existing event detection approaches are challenged to explore the sensing capability of a deployed system and choose the right sensors to meet user-specified accuracy. Event detection systems are also challenged to provide a generic system that efficiently adapts to environmental dynamics and works easily with a range of applications, machine learning approaches, and sensor modalities. Consequently, we propose Watchdog, a modality-agnostic event detection framework that clusters the right sensors to meet user-specified detection accuracy during runtime while significantly reducing energy consumption. Watchdog can use different machine learning techniques to learn the sensing capability of a heterogeneous sensor deployment and meet accuracy requirements. To address environmental dynamics and ensure energy savings, Watchdog wakes up and puts to sleep sensors as needed to meet user-specified accuracy. Through evaluation with real vehicle detection trace data and a building traffic monitoring testbed of IRIS motes, we demonstrate the superior performance of Watchdog over existing solutions in terms of meeting user specified detection accuracy, energy savings, and environmental adaptability.

Categories and Subject Descriptors: C.3 [**Special Purpose and Application-based Systems**]: Real-Time and Embedded Systems

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Wireless sensor networks, sensing, event detection, classification, machine learning

ACM Reference Format:
Keally, M., Zhou, G., Xing, G., Nguyen, D. T., Qi, X. 2012. A Learning-based Approach to Confident Event Detection in Heterogeneous Sensor Networks. ACM Trans. Sensor Netw. 1, 1, Article 1 (January 2012), 28 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

Wireless sensor network deployments have been widely used for event detection in military surveillance [He et al. 2006], environmental and wildlife monitoring [Mo et al. 2009; Dyo et al. 2010], as well as vehicle tracking [Duarte and Hu 2004]. These event

A preliminary version of this work was published in IEEE RTAS 2010.
This work was supported in part by NSF grants ECCS-0901437 and CNS-0916994.
Author's addresses: M. Keally, G. Zhou, D. T. Nguyen, and X. Qi, Computer Science Department, College of William and Mary; G. Xing, Department of Computer Science and Engineering, Michigan State University.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

detection scenarios usually require high accuracy to achieve application goals. For example, urban planners may wish to monitor traffic flow at a troublesome intersection [Duarte and Hu 2004] with less than 5% false positive and false negative vehicle detection rates. A high false positive rate may precipitate a costly and unneeded road expansion. Similarly, a high false negative rate in detection may cause the planners to cancel a proposed road expansion, leading to worsening traffic conditions. Such an event detection application must meet a user's event detection accuracy requirements with a long deployment lifetime. When a framework makes event detection decisions that meet a user's accuracy requirements in terms of desired false positive and false negative rates, we say it is *confident*. Several challenges exist to provide a confident event detection framework:

— How to find the most energy efficient sensor clusters that meet user accuracy requirements? Previous work [He et al. 2006] has shown that clustering sensors can significantly improve detection accuracy over individual sensors. How to explore the detection capability of a specific deployment through clustering sensors and choose the right sensor clusters to save energy and meet the user requirements?
— How to create a generic solution that can work easily and efficiently with a wide range of deployments, sensor modalities, and machine learning methods? Since many real deployments use multiple modalities, using different event detection solutions for each modality can be difficult. How to perform efficient collaboration among heterogeneous sensors that works in a generic context?
— How to efficiently adapt to environmental dynamics? A small energy efficient cluster of sensors may be sufficient to meet user requirements most of the time. Sometimes, however, more detection capability may be required, prompting collaboration between lower power and higher power clusters to meet user requirements and save energy. Clusters and machine learning detection models may also need to be updated during runtime.

Existing approaches for event detection do not provide a holistic solution with respect to addressing these challenges. Some approaches, such as sensing coverage [Yan et al. 2003; Hwang et al. 2007], ignore the sensing capability differences among different sensors and sensor clusters, and hence do not cluster the right sensors to meet user detection requirements. Other approaches do not work in heterogeneous sensor deployments, such as data fusion-based modality-specific sensing models [Chakrabarty et al. 2002; Yuan et al. 2008]. Other sensing models [Rachlin et al. 2005; Wang et al. 2004] ignore different sensing modalities and event detection methods altogether by using abstract fidelity functions. Other approaches, such as those that use machine learning [Benbasat and Paradiso 2007] or aggregation [Deshpande et al. 2004], attempt to capture the sensing capability differences among different sensors but do not provide confidence.

In this paper, we first show that existing approaches have difficulty capturing the sensing capability of different sensors and clusters and demonstrate that the capability of a specific deployment must be fully explored to efficiently meet user requirements. Consequently, we propose Watchdog, an event detection framework which explores the detection capability of a specific deployment and chooses the right sensors to meet accuracy requirements. We investigate several machine learning methods that our generic approach can use for event detection in heterogeneous sensor deployments. To adapt to environmental dynamics, Watchdog uses an energy efficient *sentinel* sensor cluster to make easy event detection decisions. When the sentinel cluster cannot make a confident decision, a more capable *reinforcement* sensor cluster ensures the user requirements are met. When persistent changes in the environment are detected

which significantly impact meeting user accuracy requirements, we update the detection models during runtime and form new clusters. Our main contributions are:

— With trace data from a vehicle detection application, we show the drawbacks of existing solutions and motivate the need for a confident event detection framework.
— We propose Watchdog, a generic event detection framework which clusters the right sensors to enforce user-defined event detection accuracy during runtime.
— Watchdog efficiently adapts to environmental dynamics by requesting only the sensing capability needed to meet user requirements and updating clusters when user requirements can no longer be met.
— We evaluate Watchdog in two scenarios: a vehicle detection application using real trace data and a building traffic monitoring application using IRIS motes. Watchdog can meet user-specified accuracy with reduced energy usage, while in many cases existing solutions cannot.

The rest of this paper is organized as follows: We present related work in Section 2 and motivate our Watchdog design in Section 3. We describe our detailed Watchdog design in Section 4 and present its performance evaluation in Section 5. Finally, we present conclusions in Section 6.

2. RELATED WORK

Many sensing and event detection works ignore both user accuracy requirements as well as the sensing capability differences among different sensors and sensor clusters. In sensing coverage approaches [Xing et al. 2005; Yan et al. 2003; Abrams et al. 2004; Hsin and Liu 2004; Kumar et al. 2005], energy savings is emphasized by ensuring at least one node is awake to cover a detection location, leaving all other nodes asleep. Due to sensing irregularity [Hwang et al. 2007], these coverage approaches may underestimate or overestimate the coverage needed to provide confident event detection. Such coverage approaches may also assume each sensor exhibits a disc-shaped sensing radius [Ermis and Saligrama 2005; Wang et al. 2006; Shrivastava et al. 2006; Chakrabarty et al. 2002; Giusti et al. 2009]. Other works [Dutta et al. 2005; Malinowski et al. 2008] use thresholds per modality to perform detection. Modality in the context of this work represents a reference to sensor types. Finally, the recent work by [Faulkner et al. 2013] proposes a sparse coding based approach to accurately aggregate event detection decisions from sensors in large networks. In contrast, Watchdog addresses the problems of how to select sensors for confident event detection and how to locally aggregate sensor readings to save transmission energy.

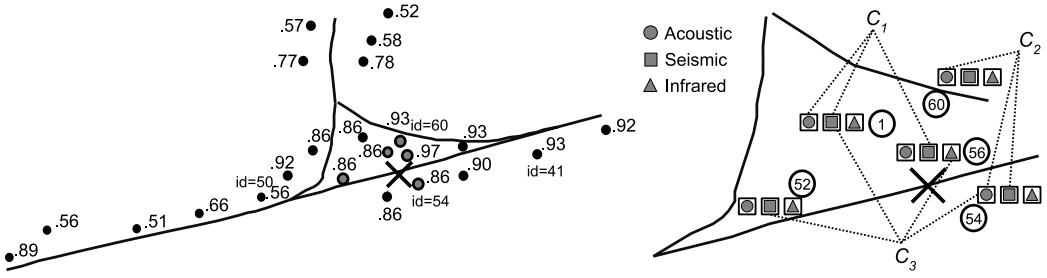
Other works attempt to meet user detection accuracy requirements through modality-specific sensing models and data fusion-based [Varshney 1996] collaboration approaches. However, these models must train each sensor modality individually, making collaboration difficult in heterogeneous deployments. Signal attenuation models are used in [Yuan et al. 2008; Xing et al. 2009; Simon et al. 2004; Volgyesi et al. 2007; Bisnik et al. 2006; Yang et al. 2008], which give a false positive rate and false negative rate for a given modality and set of training data, allowing for data fusion between multiple sensors. Other sensing models are used for cameras [Isler and Bajcsy 2005], accelerometers [Hackmann et al. 2008], as well as magnetic and PIR sensors [Gu et al. 2005]. Other works [Rachlin et al. 2005; Wang et al. 2004; Zhao et al. 2002] provide an abstract sensing fidelity function which leaves out details on the sensing modalities used as well as how sensing and detection is performed.

Some approaches attempt to address sensing capability differences among different sensors but do not provide sensing confidence. Some sensing models do not focus on event detection or classification, instead facilitating sensor data querying [Deshpande et al. 2004] or aggregation [Zhuang et al. 2007; Subramaniam et al. 2006] in a sensor

network. Other works use machine learning, such as feature classification [Eriksson et al. 2008; Kang et al. 2008; Lorincz et al. 2008; Benbasat and Paradiso 2007; Greenstein et al. 2006], Hidden Markov Models [Ganti et al. 2006; Singh et al. 2008], or both [Zappi et al. 2008].

Lastly, in [Keally et al. 2010], we demonstrate that unlike other approaches, we can meet user accuracy requirements by exploring sensing capability differences among different sensors and sensor clusters. The use of both generative and discriminative event detection techniques is demonstrated in [Hill et al. 2007; Hill et al. 2009; Lerner 2002]. However, in this paper, we also explore the effect of different machine learning approaches on meeting user requirements and energy usage, provide further adaptability to environmental dynamics, and present an additional energy saving approach by limiting radio transmissions. Unlike other works which provide high level systems for sensing and data routing [Deshpande et al. 2004; Zhuang et al. 2007; Zhao et al. 2002], we focus exclusively on exploring and capturing sensing capability and clustering the right sensors to meet user accuracy requirements. The recent work by [Krause et al. 2011] studies the problem of optimal sensor placement, and we are aware that it may assist accurate event classification, but our main goal in this paper is to cluster the right sensors to perform confident event detection.

3. MOTIVATION



(a) Individual acoustic sensors, labeled by accuracy. (b) Sensor clusters: with 100% accuracy.

Fig. 1. Event detection performance with vehicle trace data. The target location is marked by the "X" on the road.

In this section, we demonstrate the need for a new approach to confident event detection with reduced energy consumption by showing that performance differences among different sensors and sensor clusters cannot be ignored. Our goal is to provide confident event detection at a critical point, such as monitoring vehiclular traffic flow, detecting soldiers crossing a bridge, or detecting natural disasters, such as an earthquake. As an example, we use the Wisconsin SensIT experiment [Duarte and Hu 2004] to perform vehicle detection at a specific location. The SensIT experiment consists of a 23 node network with acoustic, seismic, and infrared sensors. Vehicles make 20 passes along a road through the network with ground truth provided by a GPS trace. Trace data of raw sensor energy is provided for each sensor at a raw sampling rate of 4960Hz. We provide this unmodified real sensor data and ground truth as input to a Java-based trace-driven wireless sensor network simulation run on a PC. While the sensor data and ground truth is real, we simulate communication for low power mote class devices with 802.15.4 radios, such as the Crossbow IRIS [CROSSBOW]. While we are aware that radio communication can be lossy in wireless sensor networks, in this paper we

focus on sensing accuracy, not communication quality, and assume reliable communication.

Using the trace data, we define a target location at the "X" along the road in Figure 1. Data is aggregated into time intervals of 100ms length for a total length of 6763 intervals. At each interval, we classify sensor and sensor cluster data into events when the vehicle is present within 2 meters of the target location and non-events when the vehicle is farther away or not present. With this in mind, we determine vehicle detection accuracy for individual sensors and sensor clusters using the method we present in Section 4.3 and we plot the results in Figure 1, highlighting the impact on existing solutions.

In Figure 1 (a), we first observe that sensors with the same distance to the target location may exhibit different detection accuracies. For example, nodes 41 and 50 are both 80m from the target location, but their detection accuracies are different, 93% and 56%, respectively. This is because while accuracy generally decreases with distance from the target location, terrain changes and environmental conditions still produce irregularities in sensor performance, which is consistent with the findings in [Hwang et al. 2007]. This observed sensing irregularity can cause modality-specific sensing models to suffer, such as [Yuan et al. 2008]. For example, a signal attenuation model for acoustic sensors [Yuan et al. 2008] derives the same acoustic signal receiving power for sensors with the same distance to the target location. Therefore, the same detection accuracy is statistically derived for nodes with the same distance (node 41 and 50 in our example). This signal attenuation model cannot articulate the accuracy differences among sensors, such as determining which sensor is 93% accurate and which is 56% accurate in our example. For this reason, the system performance suffers and the required detection accuracy can not always be met, which we further demonstrate in Section 5.3.

In Figure 1 (a), we also observe that not all sensors within the 25m sensing range provide the same detection accuracy. For example, even though both node 60 and 54 are within the 25m sensing range of the target location, they have different detection accuracies, 93% and 86%, respectively. This observed sensing difference can cause sensing coverage-based schemes to suffer. For example, in [He et al. 2006], only one of multiple sensors with the sensing range is enabled at a time to provide sensor coverage (or 1-coverage) for energy savings. For our example, this means that either node 60 or 54 can be turned on to provide such 1-coverage. However, it is clear that using node 60 will provide 7% points better accuracy than with node 54. Unfortunately, sensing coverage schemes have no knowledge of such subtle but important detection accuracy differences, and hence cannot provide confident event detection.

Figure 1 (b) illustrates that different sensor clusters are able to provide the same detection accuracy. For example, clusters C_1, C_2, and C_3 (consisting of different sensor modalities) can all provide 100% detection accuracy even though individual sensors cannot. As shown in [Xing et al. 2009], clustering sensors can produce a synergistic effect, allowing sensors with complimentary detection strengths in different scenarios to collaborate. Exploring the detection capability of a deployment by evaluating the performance of different sensor clusters allows the most energy efficient clusters to be chosen to confidently detect events. However, such exploration is not achieved by existing works and thus user-defined accuracy requirements cannot be met with reduced energy usage.

From the trace data analysis, it is very clear that existing approaches have difficulty meeting user required detection accuracy. This is due to lack of detailed detection accuracy knowledge of individual sensors and sensor clusters. Therefore, it is imperative to design a scheme that can provide confident event detection with user-defined accuracy, address in-situ sensing reality, and reduce energy usage.

4. WATCHDOG DESIGN

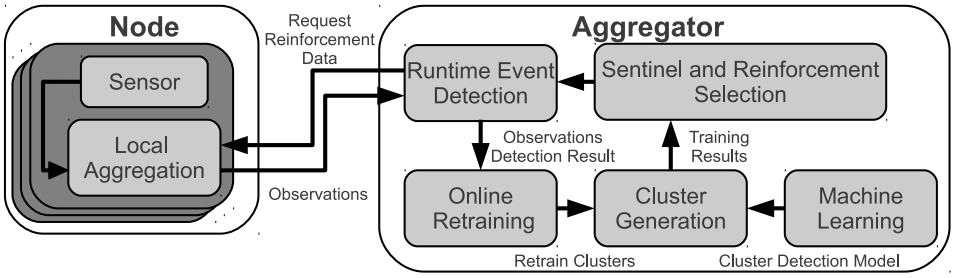


Fig. 2. Watchdog design.

Before we elaborate the details of our Watchdog system, we provide a list of sensor terminology used in the text.

— *Node:* a sensing unit with limited computational capability
— *Aggregator:* a unit with advanced computational capability that provides event detection based on information from the nodes
— *Cluster:* a group of sensor nodes
— *Sentinel Sensors:* a cluster of low-power sensors that is used for simple event detection decisions
— *Reinforcement Sensors:* a cluster of high-power sensors that is used for complex event detection decisions

In our Watchdog architecture, depicted in Figure 2, computationally limited nodes with sensors are connected through a wireless link to a more powerful aggregator, such as a mobile phone. Nodes collect sensor data and return observations to the aggregator which makes event detection decisions. Our architecture is structured to solve the challenges that arise from providing confident event detection through the use of the modules we describe below and elaborate upon in the following subsections.

The Local Aggregation module, located on sensor nodes, is used to provide efficient collaboration between heterogeneous sensors. Sensor data is aggregated such that observations from different sensor modalities can be compared and easily fused at the aggregator to make cluster-level detection decisions.

In Cluster Generation, we explore the detection capability of a deployment by determining the detection capabilities of individual sensors and sensor clusters within the deployment. We use machine learning to perform event detection and determine training accuracy of heterogenous sensor clusters. In Section 4.2, we apply several machine learning techniques to our generic design which we also evaluate in Section 5.

In Sentinel and Reinforcement Selection, clusters are selected that meet user detection requirements and adapt to changes in environmental dynamics. Using the deployment detection capability determined by Cluster Generation, a subset of that capability is selected such that the user requirements can be met. A cluster of low-power *sentinel* sensors is selected to meet the user detection requirements for many runtime observations, when event detection decisions are easy. For more difficult event detection decisions where more detection capability is needed, a cluster of *reinforcement* sensors is selected to ensure the user detection requirements are met.

In Runtime Event Detection, the detection capability is adapted to runtime observations using the clusters selected in Sentinel and Reinforcement Selection. Specifically, a low-power set of sentinel sensors make easy event detection decisions to meet user

accuracy requirements. When the sentinel sensors determine that more detection capability is needed, a second set of reinforcement sensors are used to make a confident detection decision.

With Online Retraining, described in Section 4.7, Watchdog is able to detect significant changes in environmental dynamics which cannot be captured by the existing sentinels and reinforcement clusters. When such changes are detected, new observations are labeled with ground truth and new clusters are generated to ensure user requirements are met. We now describe each design module in detail:

4.1. Local Aggregation

On a sensor node, the Local Aggregation module allows nodes to aggregate data locally at regular intervals, allowing for reduced radio communication and heterogeneous sensor fusion. The module is flexible to allow incorporation of different widely used aggregation algorithms. In our vehicle detection scenario, for each sensor, our aggregation method returns the normalized sample variance of the raw sensor data (sampled at 4960 Hz) every 100ms. In this aggregation algorithm, presented in [Duarte and Hu 2004], each raw sensor sample is represented as a 16-bit integer and divided by 32767 ($2^{16} - 1$) for the normalization. The normalized sample variance is the variance of all normalized raw sensor samples. The normalized sample variance is returned because it amplifies the changes in the raw data samples that occur when a vehicle is nearby. The aggregation interval length is selected such that an event can be captured.

For sensor j, and aggregation interval t, aggregated sensor data is represented as observation $O_{j,t}$. The aggregator fuses observations from each sensor j in a sensor cluster C_i to form an observation $O_{C_i,t}$ for that cluster. The fused observations can then be used by the aggregator to determine sensor cluster accuracy or make runtime detection decisions. We describe how sensor observations are fused together in Section 4.2, as different approaches are used for each detection algorithm.

Transmission Energy Savings. From training observations, a default observation value is determined for each sensor, which is associated with non-events. To save energy, a node only transmits observations when at least one of its sensors makes a non-default observation that is above the configured transmission threshold values. The aggregator is the receiver of non-default observations. At each aggregation interval, if the aggregator does not receive an observation from a sentinel or reinforcement sensor, it assumes the default observation value.

Since our original approach for transmission energy savings in [Keally et al. 2010] requires discrete sensor data, we expand on this to allow for machine learning methods that continuous data as input. We implement a bandwidth and energy saving approach via a transmission threshold defined for each sensor. Using training observations and labeled ground truth for each observation, each sensor computes a non-event and event centroid during initial training and when clusters are updated. The non-event centroid values are transmitted to the aggregator during the cluster generation process.

A system-defined transmission threshold, α, depicted in Figure 3, resides between the non-event and event centroids for each cluster member sensor. If a member sensor reading falls below the threshold, the reading is not transmitted to the aggregator. Therefore, if the aggregator does not receive a data sample from a cluster member sensor, it assumes the non-event centroid reading for that sensor when forming a cluster observation. We show the effectiveness of this threshold in our evaluation in terms of energy consumption and accuracy.

4.2. Machine Learning Exploration

To discriminate events from non-events, Watchdog can make use of many machine learning techniques, however, we focus on Hidden Markov Models, k-means cluster-

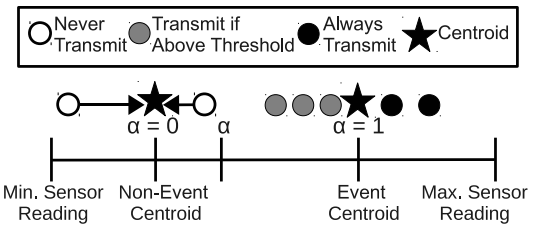


Fig. 3. Transmission threshold $0 \leq \alpha \leq 1$ for a cluster member sensor. Sensor readings below α are never transmitted and assumed by the aggregator to be the non-event centroid, while readings greater than or equal to α are always transmitted.

ing, and Fisher's Linear Discriminant, which we compare in our evaluation. We first explain Hidden Markov Models with uniform discretization. We then illustrate how to use k-means clustering to discretize a vector of real-valued sensor readings as input into a discrete Hidden Markov Model. Next, we explore the use of a continuous Hidden Markov Model. Lastly, we look at Fisher's Linear Discriminant as another approach for event detection and improving detection accuracy to meet user requirements.

Hidden Markov Models with Uniform Discretization. Hidden Markov Models require little initial configuration and are built upon the premise of determining hidden states (events) from a sequence of known observations (sensor readings) [Rabiner 1989]. HMMs assume that events and non-events are correlated with time and make use of transition probabilities to further predict the likelihood of an event at each aggregation interval. HMMs also allow aggregated data from different sensor modalities to be easily fused, providing a generic framework that is adaptable to many application scenarios.

The HMMs we use have 2 hidden states: $E = 0$ for non-events and $E = 1$ for events. The HMM also has m possible observations. To form HMM observations for a sensor cluster C_i, aggregated observations for each sensor in the cluster are independently discretized into one of m evenly sized bins. For a training or runtime aggregation interval t, discretized observations for each sensor are then fused into a single discrete observation $O_{C_i,t}$ which is the average bin value of all observations in the cluster. Using a fused training observation sequence O_{C_i}, a cluster HMM for a cluster C_i is trained using the Baum-Welch algorithm [Rabiner 1989] to form state transition probabilities and observation probabilities. In our vehicle detection scenario, with 1000 training observations (100 sec), the Baum-Welch algorithm would converge within about 20-30 iterations for each cluster (the other HMM techniques we explore also exhibited similar convergence).

At each runtime aggregation interval t, a cluster HMM uses the Forward algorithm and a history of fused observations to determine an event probability $\gamma_t \in [0, 1]$. If $\gamma_t \geq 0.5$, the HMM determines that an event has occurred ($E_t = 1$), otherwise the HMM determines there is no event ($E_t = 0$).

Clustering-based Discretization for HMMs. In [Khan et al. 2010], clustering real-valued input data into discrete observations yields significant accuracy improvements over uniform discretization when using discrete Hidden Markov Models. In Watchdog with clustering-based discretization for HMMs, we use k-means clustering [Bishop 2006] to discretize data from a sensor cluster into one of m observations required by the discrete HMM. For each sensor cluster formed during Cluster Generation, the aggregator creates m centroids using k-means clustering of recently collected sensor cluster reading tuples. Unlike the approach used with uniform discretization, each observation input $O_{C_i,t}$ is a tuple of aggregated observations for each sensor in cluster C_i. For each tuple of sensor readings during Cluster Generation or Runtime

Event Detection, the aggregator determines which of the m centroids is closest to the tuple. For a cluster C_i, the centroid index is used as the current observation input to the Baum-Welch algorithm for HMM training or the Forward algorithm for Runtime Event Detection.

Continuous HMMs. In using a continuous Hidden Markov Model, we remove the requirement for discretization of sensor data. A cluster of sensors has a large number of possible reading combinations which must be discretized into a small number of observations. Using continuous Hidden Markov Models [Rabiner 1989] with Gaussian distributions, we can preserve the granularity of the original sensor readings when providing training and runtime observation input. As with the clustering-based HMM, raw sensor readings for each generated cluster are collected by the aggregator and then fed as tuples into the Baum-Welch algorithm for HMM training or the Forward algorithm for detecting events in Runtime Event Detection. Also like the k-means approach, each observation input $O_{C_i,t}$ for a cluster C_i at aggregation interval t is a tuple rather than a single discrete value.

Algorithm 1 Cluster Generation

Input: Set of all sensors in network N, user-defined false positive rate ufp and negative rate ufn, training observations $O = \{O_{C_i,t} | C_i \subset N, 1 \leq t \leq T\}$, ground truth $G = \{G_t | 1 \leq t \leq T\}$, number of clusters for each cluster size M
Output: Set of clusters $C = \{C_i | C_i \subset N\}$
  Randomly generate M clusters for each size $k (1 \leq k \leq |N| - 1)$, add to C
  for all clusters $C_i \in C$ **do**
    Train event detection model for C_i using O_{C_i}
    for Aggregation interval $t (1 \leq t \leq T)$ **do**
      Determine event probability γ_t using C_i and O_{C_i}
      if $\gamma_t \geq .5$ **then** $E_t = 1$ **else** $E_t = 0$
      Compare system event decision E_t with G_t
      Update $fn(C_i)$, $fp(C_i)$, $fn(C_i, \gamma_t)$, $fn(C_i, \gamma_t)$
    end for
  end for

Fisher's Linear Discriminant. While Hidden Markov Models can capture the correlation of events with time as well as the correlation of events with different cluster observations, HMMs cannot fully capture the data dependencies between different sensors in a cluster. To address this concern, we also implement Fisher's Linear Discriminant [Bishop 2006], which attempts to find a projection (i.e. linear combination) of sensor cluster readings that maximizes the separation of event readings from non-event readings. As with clustering-based HMMs and continuous HMMs, the aggregator forms a tuple of aggregated observations from each sensor in a cluster to provide input for training and runtime detection. However, like the uniform discretization HMMs, each individual sensor reading in an observation tuple $O_{C_i,t}$ is uniformly discretized into one of m values. This discretization allows for the construction of discrete observation distributions for each sensor which are then used to partition event and non-event training data. We note that the model-driven scheme assumes Gaussian-distributed sensor readings, which we show in Section 3 and Section 5 does not work well due to the properties of each specific sensor deployment, such as wind, terrain, differences in sensor hardware, sensor placement, and other background noise.

Once a partition is found during training that best separates the event observations from the non-event observations, all training observations are projected onto the line

perpendicular to the partition to determine event and non-event centroids. New observations are then projected onto the same line and are classified by determining the nearest centroid. Like the other HMMs, we can also determine an event probability γ for an observation by comparing the distances between the projected observation and the centroids.

4.3. Cluster Generation

In Cluster Generation, we determine the detection capabilities of individual sensors and different sensor clusters, exploring the detection capability of a specific deployment. To do this, we generate sensor clusters of each possible size, ranging from size 1 for individual sensors to a cluster consisting of all available sensors. Using machine learning, we train a detection model and determine accuracy for each cluster using sensor training data labeled with event ground truth. We explain the cluster generation process in further detail using Algorithm 1.

With N sensors in a network, to completely explore the network detection capability, we ideally would generate all possible clusters from size 1 to $|N|$. However, since computing resources are limited, we compute M random clusters of each possible size from which to choose sentinels and reinforcements. By computing a fixed number of clusters for each size, our exploration approach is comparable in computational efficiency and effectiveness as more advanced feature selection approaches such as simulated annealing [Kirkpatrick et al. 1983] [Cerny 1985].

For each generated cluster C_i in the set C of all generated clusters, we train a machine learning detection model. A model for each cluster is trained using a sequence of training observations O_{C_i} and truth labels for each training observation G. Training observations for each sensor as well as ground truth are collected before runtime or for a short period during a runtime update.

With a trained detection model for each cluster, we can determine a cluster's event decision E_t for each aggregation interval t. E_t is derived from the cluster's event probability γ_t at each training aggregation interval t. As previously explained, the cluster determines an event occurred at interval t ($E_t = 1$) if $\gamma_t \geq .5$ and no event occurred ($E_t = 0$) if $\gamma_t < .5$. We can then use the cluster's event decision sequence $E = \{E_t | 1 \leq t \leq T\}$ to compare with known ground truth $G = \{G_t | 1 \leq t \leq T\}$ at each aggregation interval to determine cluster training accuracy. If, at aggregation interval t, the event detection decision is equal to the ground truth ($E_t = G_t$), then the cluster made a correct decision at t. Otherwise, the decision was a false positive or false negative.

Event Probability Discussion. We can compute the overall accuracy for each cluster C_i by comparing all event detection decisions E_t to ground truth G_t to determine the overall false negative rate $fn(C_i)$ and the overall false positive rate $fp(C_i)$. However, a cluster with an overall low false positive or false negative rate may have all its incorrect decisions result from event probabilities that hover near .5. During runtime detection, it is likely that an event probability near .5 will result in an incorrect decision. Consequently, it is beneficial to differentiate the accuracies between event probabilities. During runtime detection, possible bad decisions made by sentinels due to middle-range event probabilities can be caught and reinforcements can be used to meet the user requirements.

To study the correlation between event probability and detection accuracy, for each cluster C_i, we break down each training event probability γ_t into p ranges of size $1/p$. For each range we compute false positive rates $fp(C_i, \gamma)$ and false negative rates $fn(C_i, \gamma)$. Figure 4 shows an event probability breakdown of a cluster C_i from the Wisconsin vehicle trace data with 97% overall accuracy with $p = 10$ probability ranges. From the figure, it is clear that all negative event decisions have an event probabil-

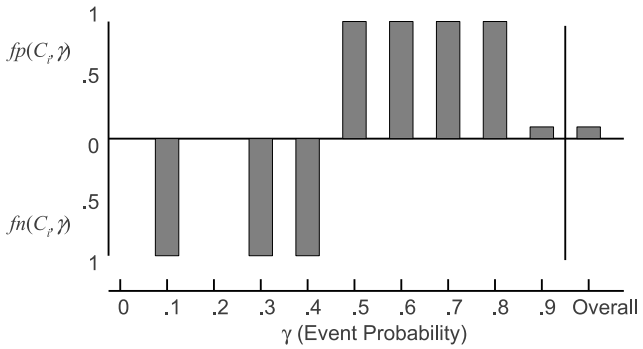


Fig. 4. Event probability breakdown for a cluster C_i with a 6% overall false positive rate and no overall false negative rate. For each .1 event probability range, the associated false positive rate $fp(C_i, \gamma)$ and false negative rate $fn(C_i, \gamma)$ are shown as bars. All ranges that have no observations yield a false positive or false negative rate of 1, since no accuracy can be determined for that range and hence we assume the worst.

ity in the $[0, .1)$ and $[.2, .3)$, ranges, while all event decisions have a probability in the $[.9, 1]$ range. During runtime detection, the event probability breakdown for the sentinel cluster is used to determine if an event probability γ_t does not meet user false positive and false negative requirements and that reinforcement observations should be collected to make a confident decision. Since the scenarios we evaluate only return a finite and discrete number of event prediction probabilities (refer to Section 5.1), using bins does not significantly impact the error analysis.

Algorithm 2 Sentinel and Reinforcement Selection

Input: Set of all sensors in network N, set of trained clusters C, user-defined false positive rate ufp and negative rate ufn
Output: Sentinel sensors s, Reinforcement sensors r

```
/*Sentinel Selection*/
fn(s) = 1; s.numNodes=|N|; s = N;
for all clusters C_i ∈ C do
  /*Meet user FN with least energy*/
  if fn(C_i)≤ufn and C_i.numNodes≤s.numNodes then
    s = C_i
    s.numNodes = C_i.numNodes
  end if
end for
/*Reinforcement Selection*/
fp(r)=1;fn(r)=1; r.numNodes=|N|; r = N;
for all clusters C_i ∈ (C − s) do
  /*Meet user FP and FN with least energy*/
  if (s ∪ C_i).numNodes≤r.numNodes and fp(C_i)≤ufp and fn(C_i)≤ufn then
    r = C_i
    r.numNodes = C_i.numNodes
  end if
end for
```

4.4. Sentinel and Reinforcement Selection

With the deployment detection capability explored by determining accuracy for all generated clusters, we choose a subset of the deployment to remain awake during runtime detection as sentinels and reinforcements to make confident detection decisions. We choose sentinels such that all negative event decisions can be made with confidence: that the user's false negative requirement is met by sentinels. Since communication is the most energy intensive operation in wireless sensor networks [Shnayder et al. 2004], we minimize energy usage by selecting a sentinel cluster with sensors on the fewest number of nodes, for only one radio transmission is needed to report observations from multiple sensors on the same node in one aggregation interval.

Since sentinels are only concerned with determining the lack of an event with confidence, we leave more difficult observations to the more powerful reinforcements when negative event decisions cannot be confidently made by sentinels. Therefore we choose reinforcements so that both the user's false positive and false negative requirements are met. We also ensure that the combined sentinel and reinforcement clusters are located on the fewest number of nodes to save energy. The reinforcement cluster has at least one sensor that is not in the sentinel cluster in order to ensure there is some added benefit from sampling reinforcement data. The sentinel and reinforcement selection algorithm is given in Algorithm 2. We note that false positive and false negative rates are calculated over C_i, since reinforcements are trained as a separate cluster, and their runtime decisions are independent of the sentinels.

Algorithm 3 Runtime Event Detection

Input: Sentinels s, reinforcements r, runtime observation for s for the current aggregation interval $O_{s,t}$, may also receive runtime observations for r for the previous and currrent aggregation intervals $O_{r,t-1}$, $O_{r,t}$
Output: Event detection decision for the current aggregation interval E_t and for the previous interval E_{t-1} if E_{t-1}=UNDECIDED

```
if E_{t-1}=UNDECIDED then
    /*Make a confident decision at t − 1 using r*/
    Determine γ_{t-1} using detection model from r and O_{r,t-1}
    if γ_{t-1} ≥ .5 then E_{t-1} = 1 else E_{t-1} = 0
end if
Determine γ_t using detection model from s and O_{s,t}
if γ_t < .5 then
    E_t = 0      /*s confidently determines no event at t*/
else if γ_t ≥ .5 and fp(s, γ) ≤ fp(u) then
    E_t = 1      /*s confidently determines an event at t*/
else if γ_t ≥ .5 and requested O_{r,t} has been received then
    /*Make a confident decision at t using r*/
    Determine γ_t using detection model from r and O_{r,t}
    if γ_t ≥ .5 then E_t = 1 else E_t = 0
else
    /*A confident decision cannot be made at t using s*/
    E_t=UNDECIDED; request O_{r,t} and O_{r,t+1}
end if
```

4.5. Cluster Generation Alternatives

Since randomly generating clusters for sensor selection creates performance concerns, we present two cluster generation and sensor selection alternatives. We first describe

selecting sensors based on distance to the target location and then illustrate selecting sensors based on correlation of event detection decisions. However, in our evaluation in Section 5, we demonstrate that the sparseness of the sensor deployment as well as restricting cluster generation to sensors within 100m of each target location mitigates both the efficiency and accuracy concerns for random cluster generation. Specifically, due to the fusion range constraing and sleeping nodes, the number of available nodes is bounded between 3 and 5 for all locations.

Distance-based Cluster Generation. Sensors are ranked by distance to the target location, with the closest sensors ordered first. Sentinels are selected by adding one sensor at a time in order of distance, until the user's false negative requirement is met. Reinforcements are selected by adding one sensor at a time in order of distance, until both the false positive and false negative requirement are met.

Correlation-based Cluster Generation. A classifier is trained for each individual sensor using training data. Sensors are then ranked by either false negative rate if we are choosing sentinels, or by total accuracy if we are choosing reinforcements. The first sensor chosen, for either the sentinel or reinforcement cluster, is the sensor with the highest rank. Then, we compute the decision correlation between the current cluster and all other remaining individual sensor classifiers as in [Keally et al. 2011]. The decision correlation defines the correlation between sensor and sensor cluster classifier decisions. To compute the decision correlation for a classifier, each correct decision is recorded as one and each incorrect decision is recorded as zero. The sensor with the decision correlation closest to zero is the sensor we add to the cluster. Sensors are added one at a time to the cluster, until the false negative requirement is met if we are choosing sentinels, or until both the false positive and false negative requirements are met if we are choosing reinforcements.

4.6. Runtime Event Detection

In Runtime Event Detection, sentinels and reinforcement sensors sample observations at each aggregation interval while all other nodes are asleep. The aggregator dynamically determines an event detection decision E_t for each interval t using sentinel or reinforcement observations, assuming a default observation value if no transmission is received. The Runtime Event Detection algorithm is described in Algorithm 3.

As shown in the algorithm, for each runtime aggregation interval t, sentinels determine an event probability γ_t using the same method performed in Cluster Generation except runtime observations are used. If $\gamma_t < .5$, the sentinels can confidently determine that no event has taken place ($E_t = 0$) since the sentinels were selected such that the user's false negative requirement is always met. However, if $\gamma_t \geq .5$, we must check if the sentinels meet the user's false positive requirement for the given probability range in which γ_t falls into, $fp(s, \gamma)$. If the user false positive requirement is met, $fp(s, \gamma) \leq ufp$, the sentinels can confidently determine that an event has occurred ($E_t = 1$). Otherwise, when $fp(s, \gamma) > ufp$, then the user false positive requirement is not met, E_t is undecided, and more detection capability is required by requesting reinforcement observations. The aggregator sends a request message to retrieve reinforcement observations for intervals t and $t + 1$ when a confident decision cannot be made by the sentinels. The reinforcement observations for t will be returned at the end of interval $t + 1$. Piggybacking reinforcement observations for interval $t + 1$ along with the observations for t will allow the aggregator to use reinforcement observations to make a decision for $t + 1$ if the sentinels are not confident for $t + 1$. Another reinforcement observation request message for interval $t + 1$ would not be necessary.

When sentinel observations are returned during an interval t for the previous interval $t - 1$, the aggregator can make a confident decision, since the sentinels meet the

user accuracy requirements. γ_t is determined using the reinforcement observations and an event, $E_{t-1} = 1$, is confidently determined if $\gamma_t \geq .5$. Otherwise, $E_{t-1} = 0$.

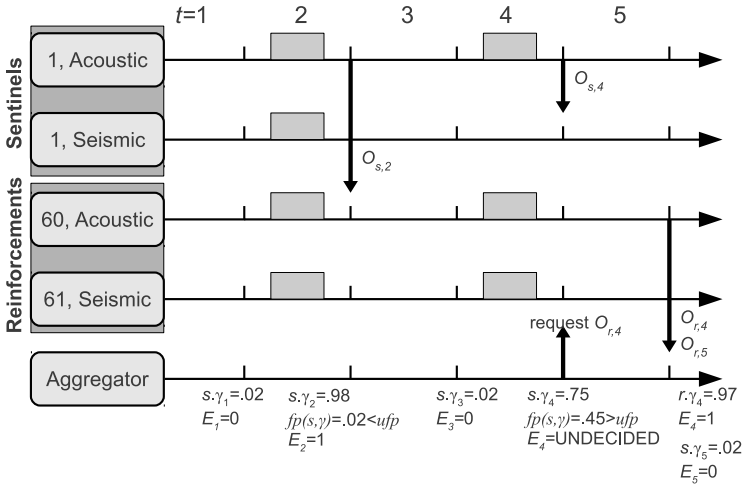


Fig. 5. Runtime detection timeline with sentinel and reinforcement event decisions, where $ufn = ufp = 0.5$. Gray areas indicate sensor readings that trigger non-default observations that are above the configured transmission threshold values. Aggregator-determined event probabilities are indicated by γ_t and event decisions are indicated by E_t. Radio transmissions due to non-default observations are indicated by the arrows.

To illustrate Runtime Event Detection, an example is presented in Figure 5. In the figure, the sensors on node 1 are sentinels while the other two sensors on nodes 60 and 61 are reinforcements. During the first interval $t = 1$, no sensors report non-default observations, so the base station determines an event probability of .02. Since the sentinels have been determined to meet the overall false negative requirement, $fn(s) \leq ufn = .05$, the decision is confident. A similar decision also occurs at $t = 3$. At $t = 2$, the sentinels capture an event and report their observations via radio, yielding an event probability of .98. The false positive rate for sentinels when $\gamma_t = .98$ was determined during training as .02, so this is a confident decision ($.02 \leq ufp = .05$). At $t = 4$ the seismic sentinel sensor does not capture the event, and the sentinel false positive rate for the current observation and event probability was determined from training as .45. Since .45 is greater than $ufp = .05$, the aggregator could not make a confident decision and more detection capability is needed. Therefore, reinforcements are signaled to return their data for $t = 4$ at the end of interval $t = 5$. At $t = 5$, the reinforcement data yields a confident event decision for $t = 4$ since sentinels always meet the user requirements and the sentinel data determines that no event has occurred.

4.7. Online Retraining

During runtime, a sentinel or reinforcement cluster may experience a drop in detection performance, running the risk of not meeting the user detection requirements. Such a performance drop may be due to changes in background noise or to the properties of the event. In these cases, the sentinel and reinforcement clusters are disbanded and new, more accurate, sentinel and reinforcement clusters are regenerated. With Online Retraining, the aggregator receives periodic feedback as to the accuracy of detection decisions. This feedback can be provided in a manner similar to [Keally et al. 2011], where K-L divergence is used to compare training data to current runtime data at each aggregation interval, determining that an update is needed when runtime data is significantly different than training data for each active sensor.

When an update is triggered, the aggregator broadcasts an update message to notify all sensor nodes. Each awake node maintains a cache of the most recent readings for each sensor and upon receipt of an update message, these readings are transmitted back to the aggregator. The aggregator then collects new ground truth for the recent observations and selects new sentinel and reinforcement sensors through the Cluster Generation and Sentinel and Reinforcement Selection processes described in Section 4.3 and Section 4.4, respectively. Candidate sensors are kept awake to be considered as future sentinels or reinforcements. These are chosen randomly after each update using the set of available sensors. We ensure at least two nodes are awake as candidates in the evaluation. Runtime Event Detection then proceeds with the new sentinels and reinforcements as in Section 4.6. Ground truth is obtained via user feedback in a fashion similar to V-SAM [Hwang et al. 2007], where a video camera is utilized. Particularly, a user plays back the most recent activity to see what happened, and records ground truth for retraining.

A new, updated cluster may change with respect to the old cluster in three ways. First, the new cluster may consist of the same exact sensors as the old cluster only with a new detection model at the aggregator. Second, a newly formed cluster may also reside on the same nodes as the old but contain different sensors. Third, a new cluster may also reside on different nodes than the previous cluster. In future work, we will predict how a cluster changes during an update in order to reduce overhead in generating new clusters as well as to ensure energy fairness through load balancing among clusters.

After sentinel and reinforcement sensors are selected during initial training or an update, a subset of nodes is selected as candidate nodes from among all non-sentinel and non-reinforcement sensor nodes. Such candidate nodes remain awake and sample data so that during an update candidates may be selected to become sentinel or reinforcement sensor nodes if the current sentinel or reinforcement nodes cannot meet the user requirements.

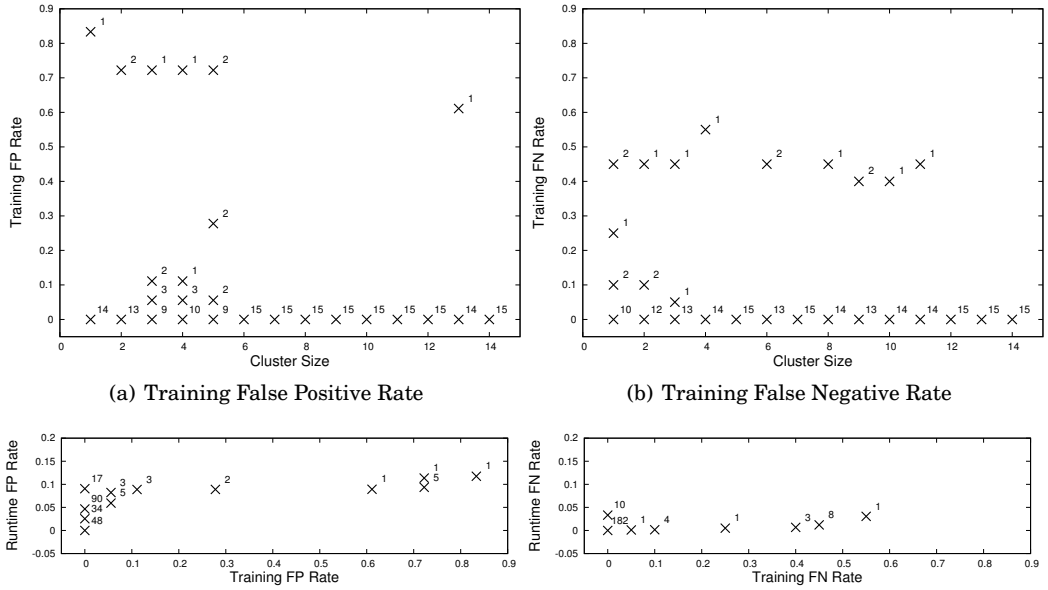
5. EVALUATION

Watchdog is designed as a generic framework, so we evaluate its performance in two different application scenarios using a PC-based Java simulation: vehicle detection using trace data and a building traffic monitoring application using IRIS motes. For the vehicle detection trace, we use the same simulation methodology described in Section 3. We use one pass of the vehicle (70s) as training data and the remaining trace with 10 more passes as runtime data. In the building traffic monitor experiment, we place five IRIS motes with attached MTS310 sensorboards (2-axis accelerometer, 2-axis magnetometer, acoustic, light sensors) [CROSSBOW] on the main entrance door of an academic building to monitor the traffic pattern of when people are most often entering and leaving the building. We define an event and measure the ground truth as the time period during which someone opens the door and walks through (either entering or exiting), with the door automatically closing behind. We obtained ground truth via video recording of the building entrance and sampled data at 20ms intervals using the heterogeneous sensors on the mote sensorboards. We also use a 4s aggregation interval and 2 minutes of data for training. Additionally, 40 observations are used for periodic retraining.

We compare against a sensing coverage-based framework and a modality-specific sensing model using data fusion. The sensing coverage approach, V-SAM [Hwang et al. 2007], is a state of the art scheme which in contrast to conventional coverage approaches, attempts to keep awake sensors that sample similar data. Similarity represents virtual sensing relationship among sensor nodes, and is calculated based on the similarity equation from [Hwang et al. 2007]. We also force k-coverage on V-SAM,

where 1 to 3 nodes are awake to cover an event; only 1 node must detect an event for V-SAM to detect an event. We also compare Watchdog with a classical model-driven event detection solution [Yuan et al. 2008] that uses a modality-specific sensing model. In [Yuan et al. 2008], a signal attenuation model is used to estimate signal energy for targets of different distances with a Gaussian noise distribution model. Given user-defined false positive rate, the model-driven approach can derive an event detection threshold for the average energy readings of all sensors in a cluster.

In Section 5.1, we first demonstrate that Watchdog is able to explore the detection capability of a specific deployment and cluster the right sensors to meet user detection requirements. Next, in Section 5.2, we compare against a sensing coverage-based framework and illustrate that Watchdog achieves a significantly higher performance. In Section 5.3, we compare against a data fusion-based modality-specific sensing model and show that Watchdog can adapt the detection capability to runtime observations and meet user detection requirements while the model-driven approach cannot. We explore the effects of different machine learning approaches in Section 5.4, and investigate in Section 5.5 how Watchdog can create new clusters when the existing sentinel and reinforcement clusters are unable to handle a significant environmental change. Lastly, we demonstrate the benefits of our transmission energy saving approach in Section 5.6 and demonstrate in Section 5.7 that randomly choosing sensors does not impact performance. In the experiments, for Watchdog, we generate $M = 15$ clusters for each possible size. By default, we use HMMs with uniform discretization and Online Retraining disabled. We also set user requirements to 5% for false positives and false negatives.



(a) Training False Positive Rate

(b) Training False Negative Rate

(c) Training vs. Runtime Detection False Positive Rate

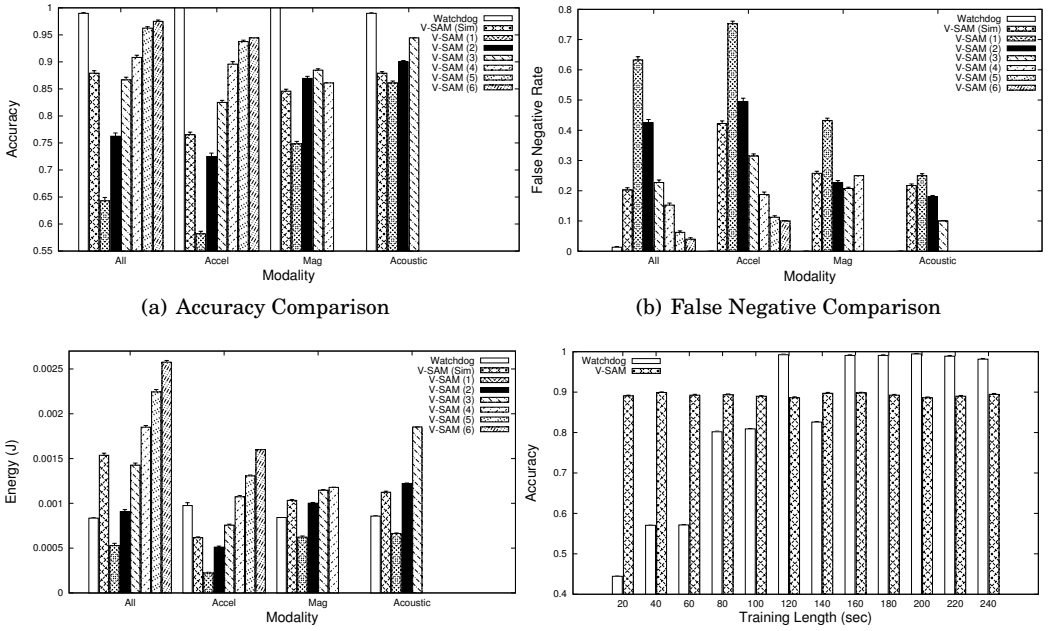
(d) Training vs. Runtime Detection False Negative Rate

Fig. 6. Cluster Training and Runtime Detection. An integer besides the "x" denotes the number of clusters that give the corresponding FP or FN rate.

5.1. Exploring Detection Capability and Meeting Requirements

Using the building traffic monitor trace, we show that by exploring the detection capability of a specific deployment, Watchdog can choose the right sensor clusters to meet user-defined false positive and false negative rates. In Figure 6 (a) (b), we plot the number of clusters for each cluster size that achieve the same training false positive or false negative rate. In Figure 6 (c) (d), we plot cluster training performance compared with runtime performance. In this subsection, we use half of the data (and events) for training and the remaining half for testing.

In Figure 6 (a) (b), there are only a limited and discrete number of false positive and false negative rates that the deployed system can support. To that end, a user can only require a false positive or false negative rate that can be supported by the system. For example, most sensors and sensor clusters have false positive and false negative rates near zero, while only a few experience false positive rates greater than 70% or false negative rates greater than 45%. This set of cluster performances is determined by the sensor hardware and local sensing reality where the system is deployed. Different scenarios may produce different false positive and false negative rates for each cluster.



(a) Accuracy Comparison

(b) False Negative Comparison

(c) Energy Comparison

(d) Training Length Accuracy

Fig. 7. Watchdog and V-SAM comparison for different modalities, levels of V-SAM coverage, and training lengths.

In Figure 6 (a) (b), we also observe that even in a small deployment with "5 IRIS $\times$ 6 sensors each = 30 sensors", there are a large number of sensor clusters available to meet user specified false positive or false negative rate. As shown in Figure 6 (a), there are exactly 3+3+2=8 sensor clusters that demonstrate a 5% false positive rate in the training data and there are 189 sensor clusters in Figure 6 (a) that demonstrate smaller than a 5% false positive rate. So, in total, 8+189=197 different sensor clusters can be chosen to meet the user-specified 5% false positive rate.

In Figure 6 (c) (d), we observe that during runtime detection, Watchdog is able to meet the false positive or false negative rate explored during training. For example, Figure 6 (c) shows that 48 clusters with a training false positive rate of 0% achieve this performance during runtime; Figure 6 (d) shows that 182 clusters with a false negative rate of 0% also demonstrate no false negatives during runtime. In Figure 6 (c) (d), we also observe that clusters with higher training false positive or false negative rates achieve significantly better runtime performance: 6 clusters with a training false positive rate of 72% achieve a runtime false positive rate of 10%, and 13 clusters with a training false negative rate greater than 20% achieve a runtime false negative rate of 5% or less.

To summarize, these data illustrate that Watchdog is able to cluster the right sensors to meet user requirements during runtime. Plus, many clusters of different sizes exist to meet user-required accuracy. This allows for freedom in sentinel and reinforcement selection to adapt the detection capability to environmental dynamics and maximize energy savings.

5.2. Comparison with V-SAM

Using the building traffic monitor trace, we compare Watchdog to a state of the art sensing coverage framework that addresses sensing irregularity, V-SAM [Hwang et al. 2007]. Though V-SAM cannot provide guaranteed accuracy, we set the Watchdog user requirements to the lowest false positive and false negative rates determined from training. Evaluation results are presented in Figure 7 with 95% confidence intervals over 20 runs.

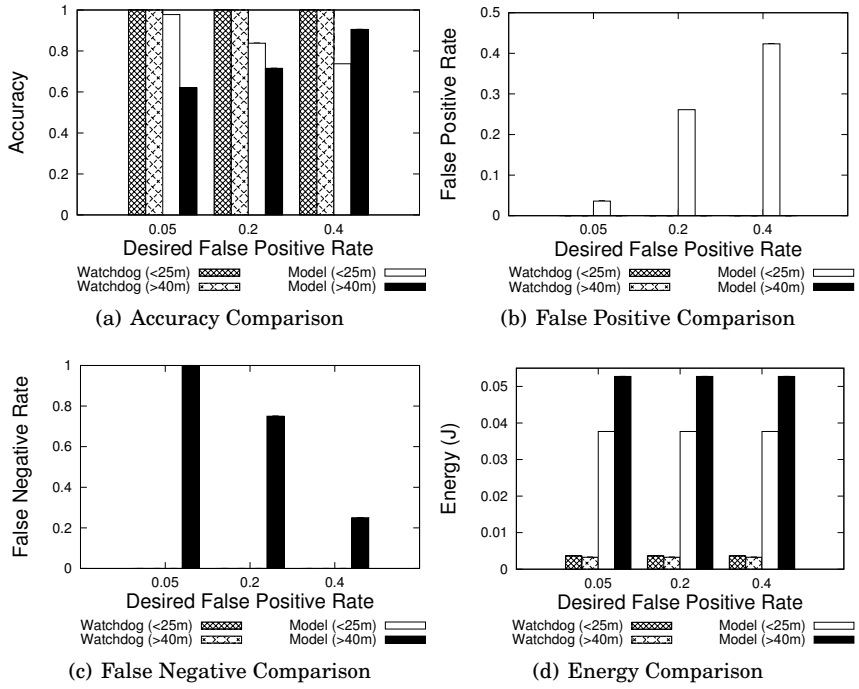
In Figure 7 (a) (b), We observe that Watchdog outperforms V-SAM in every configuration: all modalities, individual modalities, and varying levels of V-SAM coverage. Although using higher k-coverage and similarity-based coverage helps improve V-SAM performance, it is always outperformed by Watchdog, which consistently demonstrates close to 100% detection accuracy in Figure 7 (a) and close to zero false negatives in Figure 7 (b). None of the Watchdog or V-SAM configurations experience statistically noticeable false positives, so false positive rates are not illustrated. Watchdog can consistently outperform V-SAM because Watchdog samples subsets of potential sensor clusters to explore the detection capability of individual sensors and sensor clusters in a deployed system in order to meet user requirements. However, V-SAM has no detailed knowledge of detection accuracy, so the most accurate sensors may be excluded while poor performing sensors may become involved in detection decisions.

In Figure 7 (c), we observe that Watchdog is much more energy efficient than V-SAM. We compute energy to transmit or receive each byte of a 802.15.4 packet on CC2420 radios [ChipconCC2420] using a TDMA-based scheme. We use the default payload of 28 bytes for each payload which is more than sufficient to carry aggregated data for all sensors on the transmitting node. As shown in Figure 7 (c), Watchdog energy consumption is relatively constant for all modalities and for each modality, hovering around 9×10^{-4} J, since typically only 1-2 nodes are used in forming both sentinel and reinforcement clusters. However, V-SAM energy consumption (when achieving good performance) is much more varied: $10 \times 10^{-4} \sim 26 \times 10^{-4}$ J. While Watchdog may use more energy than 1 or 2-coverage V-SAM, Watchdog achieves about 35% points better accuracy compared with those V-SAM configurations. Watchdog is significantly more energy efficient than V-SAM since Watchdog fully explores the detection capability of individual sensors and sensor clusters. Hence, Watchdog can use this knowledge to adapt sensing capability to runtime observations while making confident detection decisions, but V-SAM cannot.

Training Length. In Figure 7 (d), we observe that for Watchdog to achieve the aforementioned superior detection accuracy and energy efficiency compared with V-

SAM, only a short training length is needed. As shown in Figure 7 (d), when the training length increases, Watchdog performance improves quickly, surpasses V-SAM performance, and converges to near perfect accuracy after about 2 minutes, which is reasonably short for real applications. Even though V-SAM requires little training, which is invisible in Figure 7 (d), it demonstrates much lower detection accuracy and much higher energy usage than Watchdog. Since the training length is short, the use of periodic retraining can handle environmental changes.

5.3. Comparison with a Modality-Specific Sensing Model



(a) Accuracy Comparison (b) False Positive Comparison

(c) False Negative Comparison (d) Energy Comparison

Fig. 8. Watchdog and modality-specific sensing model comparison with sensors located within 25m of, or more than 40m from, the target location.

In this section, we compare Watchdog with a classical model-driven event detection solution [Yuan et al. 2008] that uses a data fusion-based modality-specific sensing model. For fair comparison, we use the same Wisconsin SensIT experiment trace data [Abrams et al. 2004] used in [Yuan et al. 2008] and make use of acoustic sensors to detect vehicles passing a static target location. Our evaluation is conducted in two scenarios: when the target location is well within the sensing range of all sensors, and when the sensors are located at the fringe of the detection range. In the first scenario, we use 5 acoustic sensors < 25m to the target location; in the second, we use 7 acoustic sensors with distances > 40m from the target location. The results are plotted in Figure 8.

For the $<$25m scenario, we observe from Figure 8 (b) that Watchdog always meets the user false positive requirement while the model-driven scheme cannot. For instance, in Figure 8 (b), the model-driven scheme has a 28% false positive rate when 20% is required, and gives a 42% false positive rate when 40% is required. We also observe

from Figure 8 (a) that Watchdog yields perfect accuracy, while model-driven accuracy drops when the desired false positive rate increases. Watchdog performs better than the model-driven scheme because Watchdog always chooses sentinels and reinforcements that meet user requirements for confident event detection. The model-driven scheme does not exploit such subtle but important information.

For the >40m scenario, we also observe that Watchdog always meets user requirements but the model-driven scheme performs poorly or even fails. For example, when user requires a 5% false positive rate, the model-driven approach experiences very low accuracy, 67% in Figure 8 (a), and a very high false negative rate, 100% in Figure 8 (c). This is because for a low desired false positive rate, the model-driven detection threshold is set too high to detect any events. We also find in Figure 8 (c) that requesting higher false positive rates does not help much. The poor performance of the model-driven scheme and the good performance of Watchdog can be explained with the same reasons attributed to the <25m scenario.

Using the transmission energy model from Section 5.2, in both scenarios, Watchdog is found to consume significantly less energy than the model-driven scheme as shown in Figure 8 (d). This is because the model-driven scheme in [Yuan et al. 2008] has a very simple energy saving scheme: nodes within the 25m "fusion range" are awake and nodes beyond the range all sleep. On the contrary, Watchdog adapts the detection capability to runtime observations through the use of sentinels and reinforcements for more aggressive energy savings. In Figure 8 (d), we also observe that the model-driven scheme consumes more energy in the >40m scenario than the <25m scenario. This is because 7 nodes are used instead of 5.

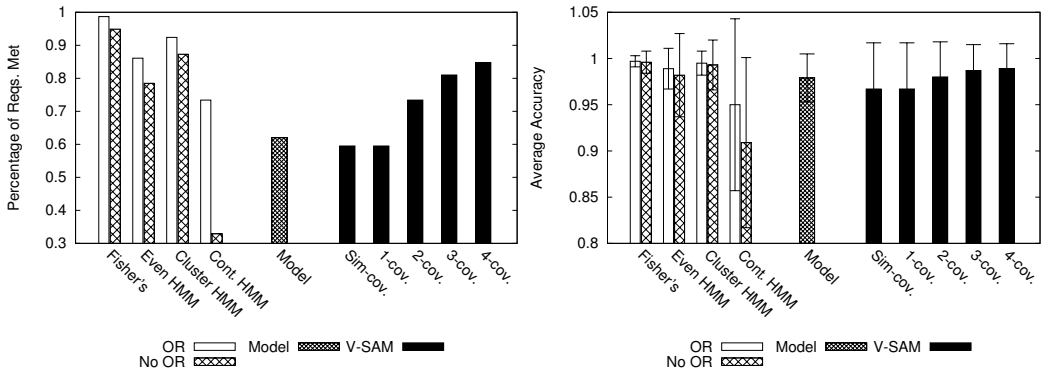
Table I. Adapting detection capability with reinforcements.

Sentinel FP/FN (%)	Reinforc. FP/FN (%)	Reinforc. Requests (%)
9.5/0.0	0.0/0.0	21

Adapting Detection Capability. Using the <25m scenario we illustrate in Table I how Watchdog adapts the detection capability to environmental dynamics. With desired false positive and false negative rates of 0%, a sentinel cluster is selected with a 9.5% false positive rate and 0% false negative rate. A more powerful reinforcement cluster is selected with a 0% false positive and false negative rate. During runtime, 79% observations are comparatively easy and hence confident decisions are entirely made by sentinels. When the sentinels make a decision that does not meet user requirements (for the 21% more difficult observations), reinforcements are used to make a confident decision. The reduction in radio transmissions made by using only the sensors necessary to meet user requirements ensures significant energy savings.

5.4. Machine Learning Comparison

Using the vehicle detection trace, we choose 79 target locations along the road and cluster sensors within a 100m range. Each event location is treated as an independent application instance for Watchdog, and then the results are averaged across the 79 instances. We plot standard deviation error bars for accuracy, false positive rates, and false negative rates. More target locations increases the environmental dynamics and allows us to better investigate the effects of different machine learning algorithms as well as Online Retraining in Section 5.5. Figure 9 a) shows that Watchdog without Online Retraining using Fisher's Linear Discriminant can meet the user requirements for 95% of the target locations, while the modality-specific sensing model and V-SAM meet the user requirements for 63% and 85% of the target locations, respectively. The percentage of target locations meeting user requirements changes significantly between



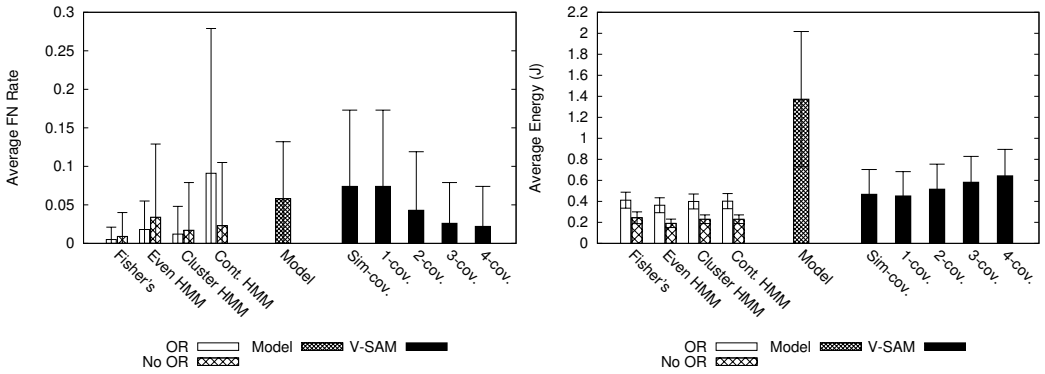
(a) Locations Met (b) Accuracy Comparison

Fig. 9. Multiple location performance with Watchdog, modality-specific sensing model, and V-SAM.

continuous-HMMs with Online Retraining and without Online Retraining due to overfitting. The events change during runtime, so the model must be updated more often since it is overfitting the training data. A similar trend can be seen when comparing average accuracy for all target locations in Figure 9 b). With respect to the different machine learning methods, Watchdog with Fisher's Linear Discriminant exhibits the best performance since it can determine the data dependencies between different sensors in a cluster and use the strengths of each sensor to maximize detection performance. Watchdog with a clustering-based discretization HMM is also able to improve on the default uniform discretization HMM since k-means clustering aids the HMM in separating events from non-events. The continuous HMM meets many fewer target locations compared with other Watchdog implementations since it requires a much longer training dataset to train accurately.

Figure 9 b) shows that a slight accuracy increase allows Watchdog to meet user accuracy requirements for many more locations, making Fisher's Linear Discriminant especially valuable. For example, the continuous HMM without Online Retraining has over 90% average accuracy for all locations, yet only meets user requirements for 34% of these locations. However, with 98% average accuracy, Watchdog with Fisher's Linear Discriminant and no Online Retraining meets user requirements for 95% of locations. Similarly, increasing coverage for V-SAM increases its detection accuracy and the number of locations that meet the user requirements, but even 4-coverage V-SAM meets the user requirements for only 85% of locations. Nearly all of the detection errors for Watchdog, the modality-specific sensing model, and V-SAM are false negatives as shown in Figure 10 a); the average false positive rates for all approaches are less than 1% except the continuous HMM, which has an average false positive rate of 14%.

In Figure 10 b), we show the average energy consumption for each approach across all target locations. We extend energy usage to include not only radio energy but also energy consumed by motes while awake and sensing, with details given in [Shnayder et al. 2004]. Watchdog consistently experiences the lowest energy consumption, ranging from just over 0.2J with the uniform discretization HMM without Online Retraining to 0.4J for the continuous HMM with Online Retraining. Watchdog is able to reduce energy usage compared with other approaches since it chooses only the sensors and nodes needed to meet user requirements. V-SAM has much higher energy consumption, ranging from 0.5J for similarity coverage to 0.6J for 4-coverage. As coverage levels increase, more nodes are awake, increasing V-SAM energy usage. Furthermore,



(a) False Negative Comparison (b) Energy Usage

Fig. 10. False negatives and energy usage comparison for multiple locations.

the standard deviation is higher than Watchdog, for when events are detected, V-SAM awakens all nodes to monitor the event until it is no longer detected. The modality-specific sensing model keeps all nodes awake at all times and has very high energy usage, 1.4J. Its standard deviation is also much higher since some locations have many nodes within the 100m fusion range, while other locations have few nodes.

5.5. Online Retraining

During runtime, Watchdog with Online retraining is able to handle significant environmental changes by retraining sentinel and reinforcement clusters. The 79 different detection locations we choose in the vehicle detection trace provide significant environmental dynamics, as the vehicle path varies widely with each pass. Watchdog configurations with Online Retraining have 2 candidate nodes awake at each time interval and the machine learning approaches that use a transmission threshold have the threshold set to $\alpha = 0.5$. In Figure 9 a), Online Retraining is able to increase the number of locations which meet user requirements for each machine learning method, ranging from 3% points for Fisher's Linear Discriminant to over 30% points for the continuous HMM. Similar increases in accuracy are also observed in Figure 9 b). Figure 10 a) shows that Online Retraining requires slightly more energy due to communication overhead in forming new clusters and awake candidate nodes, but performance is still significantly better than the modality-specific sensing model and V-SAM.

In Figure 11, we show that more powerful reinforcements are requested sparingly, regardless of the machine learning approach or the use of Online Retraining. Using Fisher's Linear Discriminant, some locations request reinforcement data more than 10% of the time. Fisher's Linear Discriminant requests reinforcements slightly more than the other configurations for it can closely capture the data dependencies among sensors in a cluster and better determine when the sentinel cluster cannot make a confident detection decision. This adaptation in detection between clusters of differing sensing capability demonstrates the ability of Watchdog to reduce energy consumption in comparison with other approaches.

Using Online Retraining, Figure 12 a) illustrates the total number of updates for each target location and each Watchdog machine learning configuration. The figure shows that a small number of updates allows Watchdog to maintain its high detection accuracy and meet the user requirements. Most target locations require no updates with all but the continuous Hidden Markov Model, for which most locations require 2, 3, or 4 updates. For Fisher's Linear Discriminant and the discrete HMMs, the ini-

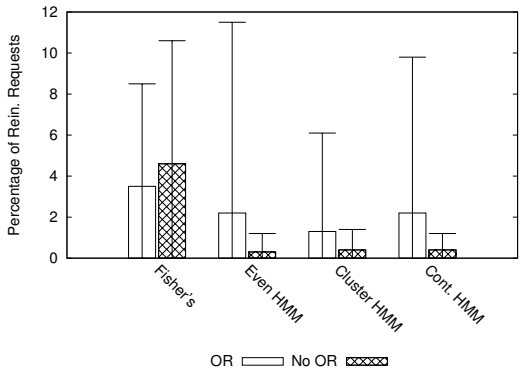
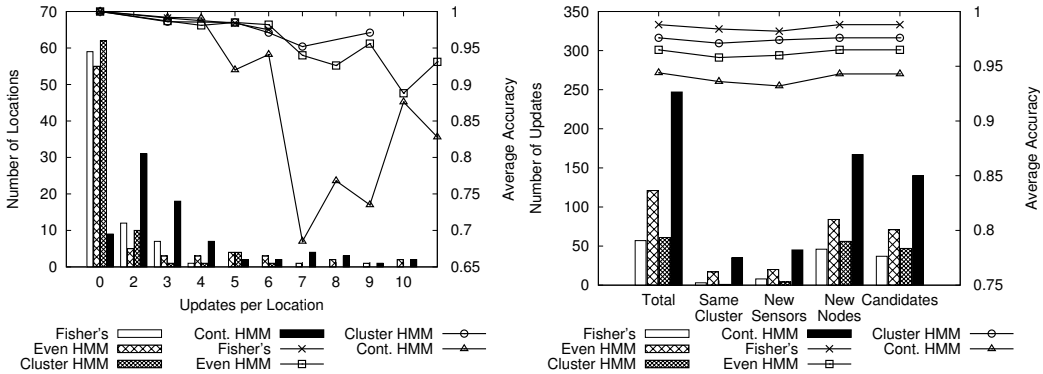


Fig. 11. Reinforcement Requests.



(a) Number of Updates Per Location

(b) Update Types

Fig. 12. Online Retraining.

tial training is good enough to meet the user requirements for most locations, however since the continuous HMM does not receive enough training data, it requires more updates. Figure 12 a) also depicts average accuracy for target locations with each number of updates. As the number of updates per location increases, average accuracy generally decreases, reaching as low as 70% for the continuous HMM. Although locations with the largest amount of environmental change experience lower accuracy and more updates, without such updates accuracy would be even worse for these locations, as illustrated in Figure 9 a).

Figure 12 b) shows how sentinel and reinforcement clusters are reformed during an update. In the figure, the Watchdog configurations with the fewest total updates also experience the highest average accuracy among locations that have updates. Fisher's Linear Discriminant has 51 updates in total and experiences an average accuracy of 98% for locations with updates while the continuous HMM has nearly 250 updates with an average accuracy of 94%. With Fisher's Linear Discriminant, accuracy for most locations is high enough to meet the user requirements and few, if any, updates are needed. Also from the figure, over half of all updates for each configuration require formation of clusters with new nodes that were not members of the previous clusters. Environmental changes cause the current nodes and sensors to fail to meet the user

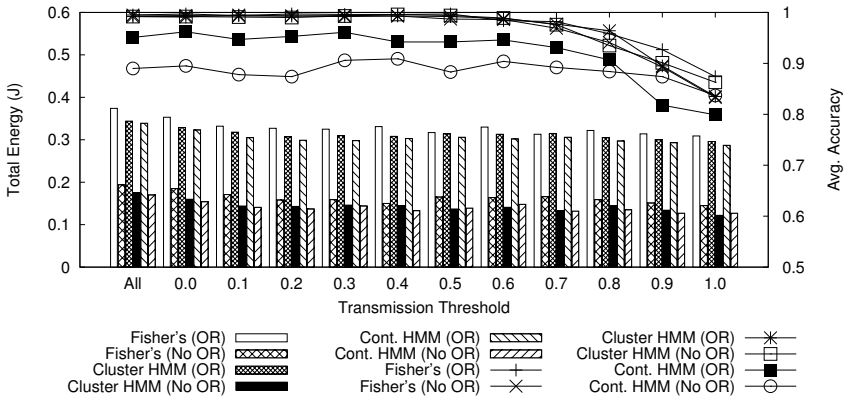


Fig. 13. Total energy and accuracy for different transmission threshold values

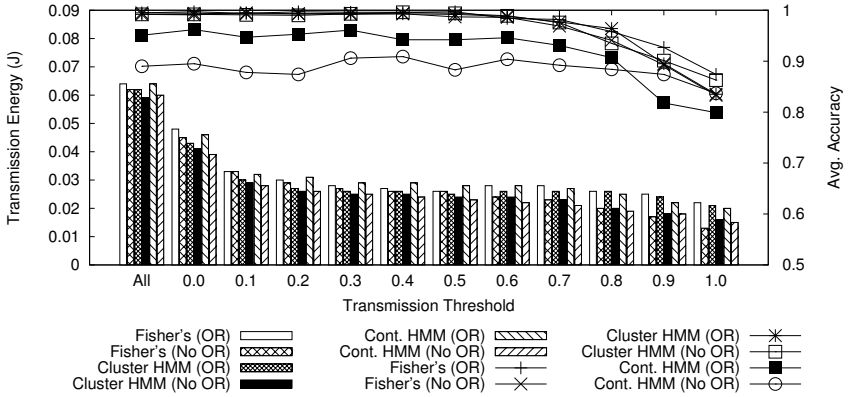


Fig. 14. Transmission energy and accuracy for different transmission threshold values

requirements, which prompts the creation of new clusters with new nodes that can address these changes. Thus, candidate nodes are also heavily used during updates.

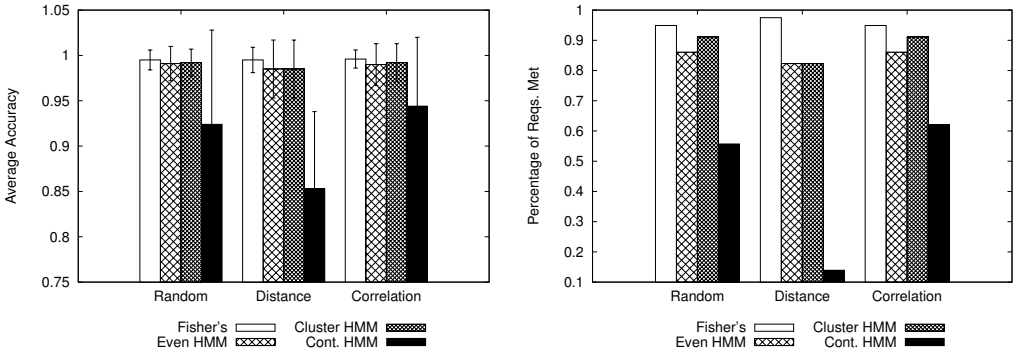
5.6. Transmission Energy Savings

We now justify the choice of the transmission threshold $\alpha = 0.5$ in terms of a tradeoff between accuracy and energy usage for the machine learning approaches that use continuous valued input. We also disable the transmission threshold and show the impact of accuracy and energy when all readings are transmitted. Using the vehicle detection trace data, Figure 13 shows that total energy is affected little by increasing α and transmitting fewer readings. Accuracy remains relatively stable as α increases until $\alpha > 0.5$, when accuracy drops by as much as 12% points for Fisher's Linear Discriminant and the continuous Hidden Markov Model. Figure 14 also shows that for configurations with Online Retraining, candidate nodes consume a significant amount of the total energy budget, for two additional nodes are awake during Runtime Event Detection.

While the transmission threshold may not have a significant impact on total energy, Figure 14 shows that the transmission threshold noticeably affects transmission energy. When all sensor readings are transmitted, all Watchdog configurations consume about 0.065J of transmission energy with while the highest transmission threshold

setting all configurations consume about 0.2J. Energy decreases noticeably for only small and large α, indicating that most sensor readings are closest to the event and non-event centroids and that most detection decisions can be easily made by the sentinel cluster, as is illustrated in Figure 12 a).

5.7. Cluster Generation Alternatives Performance



(a) Impact on accuracy

(b) Impact on meeting user requirements

Fig. 15. Cluster Generation Alternatives Performance.

To evaluate cluster generation algorithms, we compare the random cluster selection with the aforementioned alternative approaches (distance-based and correlation-based). The results show that the alternative schemes have negligible impact on accuracy and the percentage of user requirements met.). In these experiments, online retraining is enabled. Figure 15 a) shows that the alternative methods have little impact on accuracy, and Figure 15 b) shows that the alternative methods also have little impact on meeting user requirements. Therefore, we conclude that choosing sensors and generating clusters randomly has little impact on overall performance.

6. CONCLUSION

Existing works do not provide a holistic solution with respect to clustering the right sensors for confident event detection, heterogeneous deployments, and adaptation to environmental dynamics. Consequently, we present Watchdog, a generic event detection framework which can function in a wide array of applications and deployments. Unlike existing approaches, Watchdog can obtain the detection capability of a specific deployment and use this knowledge to cluster the right sensors to perform confident event detection. With a short training length, Watchdog chooses sentinel and reinforcement sensors which adapt the detection capability to confidently detect events while saving energy. We propose several different machine learning techniques with which Watchdog can use to perform confident event detection and allow these methods to adapt to environmental changes over time. Our evaluation demonstrates that Watchdog largely exceeds the detection accuracy of existing approaches with reduced energy consumption. Our evaluation also demonstrates that Watchdog always meets user detection requirements when in many cases existing approaches cannot.

REFERENCES

ABRAMS, Z., GOEL, A., AND PLOTKIN, S. 2004. Set K-Cover Algorithms for Energy Efficient Monitoring in Wireless Sensor Networks. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN '04)*. ACM/IEEE, 424–432.

BENBASAT, A. AND PARADISO, J. 2007. A Framework for the Automated Generation of Power-Efficient Classifiers for Embedded Sensor Nodes. In *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems (SenSys '07)*. ACM, 219–232.

BISHOP, C. 2006. *Pattern Recognition and Machine Learning*. Springer.

BISNIK, N., ABOUZEID, A., AND ISLER, V. 2006. Stochastic Event Capture Using Mobile Sensors Subject to a Quality Metric. In *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking (MobiCom '06)*. ACM, 98–109.

CERNY, V. 1985. Thermodynamical Approach to the Traveling Salesman Problem. *Springer Journal of Optimization Theory and Applicaitons 45*, 41–51.

CHAKRABARTY, K., IYENGAR, S., QI, H., AND CHO, E. 2002. Grid Coverage for Surveillance and Target Location in Distributed Sensor Networks. *IEEE Transactions on Computers 51*, 1448–1453.

ChipconCC2420. CC2420 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver. http://focus.ti.com/lit/ds/symlink/cc2420.pdf.

CROSSBOW. XBOW Mote Specifications. http://www.xbow.com.

DESHPANDE, A., CARLOS, C., MADDEN, S., HELLERSTEIN, J., AND HONG, W. 2004. Model-driven Data Acquisition in Sensor Networks. In *Proceedings of the 30th International Conference on Very Large Databases (VLDB '04)*. VLDB Endowment, 588–599.

DUARTE, M. AND HU, Y. 2004. Vehicle Classification in Distributed Sensor Networks. *Journal of Parallel and Distributed Computing 64*, 826–838.

DUTTA, P., GRIMMER, M., ARORA, A., BIBYK, S., AND CULLER, D. 2005. Design of a Wireless Sensor Network Platform for Detecting Rare, Random, and Ephemeral Events. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN '05)*. ACM/IEEE, 497–502.

DYO, V., ELLWOOD, S., MACDONALD, D., MARKHAM, A., MASCOLO, C., PASZTOR, B., SCELLATO, S., TRIGONI, N., WOHLERS, R., AND YOUSEF, K. 2010. Evolution and Sustainability of a Wildlife Monitoring Sensor Network. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems (SenSys '10)*. ACM, 127–140.

ERIKSSON, J., GIROD, L., HULL, B., NEWTON, R., MADDEN, S., AND BALAKRISHNAN, H. 2008. The Pothole Patrol: Using a Mobile Sensor Network for Road Surface Monitoring. In *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services (MobiSys '08)*. ACM, 29–39.

ERMIS, E. B. AND SALIGRAMA, V. 2005. Adaptive Statistical Sampling Methods for Decentralized Estimation and Detection of Localized Phenomena. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN '05)*. ACM/IEEE, 143–150.

FAULKNER, M., LIU, A. H., AND KRAUSE, A. 2013. A Fresh Perspective: Learning to Sparsify for Detection in Massive Noisy Sensor Networks. In *Proceedings of the 12th International Conference on Information Processing in Sensor Networks (IPSN '13)*. ACM/IEEE, 7–18.

GANTI, R. K., JAYACHANDRAN, P., ABDELZAHER, T., AND STANKOVIC, J. 2006. SATIRE: A Software Architecture for Smart AtTIRE. In *Proceedings of the 4th International Conference on Mobile Systems, Applications, and Services (MobiSys '06)*. ACM, 110–123.

GIUSTI, A., MURPHY, A., PALOPOLI, L., AND PASSERONE, R. 2009. Solving the Wake-up Scattering Problem Optimally. In *Proceedings of the 6th European Conference on Wireless Sensor Networks (EWSN '09)*. Springer-Verlag, 166–182.

GREENSTEIN, B., MAR, C., PESTEREV, A., FARSCHI, S., KOHLER, E., JUDY, J., AND ESTRIN, D. 2006. Capturing High-Frequency Phenomena Using a Bandwidth-Limited Sensor Network. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys '06)*. ACM, 279–292.

GU, L., JIA, D., VICAIRE, P., YAN, T., LUO, L., TIRUMALA, A., CAO, Q., HE, T., STANKOVIC, J., ABDELZAHER, T., AND KROGH, B. 2005. Lightweight Detection and Classification for Wireless Sensor Networks in Realistic Environments. In *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems (SenSys '05)*. ACM, 205–217.

HACKMANN, G., SUN, F., CASTANEDA, N., LU, C., AND DYKE, S. 2008. A Holistic Approach to Decentralized Structural Damage Localization Using Wireless Sensor Networks. In *Proceedings of the 2008 Real-Time Systems Symposium (RTSS '08)*. IEEE, 35–46.

HE, T., KRISHNAMURTHY, S., LUO, L., YAN, T., STOLERU, R., ZHOU, G., CAO, Q., VICAIRE, P., STANKOVIC, J., ABDELZAHER, T., HUI, J., AND KROGH, B. 2006. VigilNet: An Integrated Sensor Network System for Energy-Efficient Surveillance. *ACM Transactions on Sensor Networks 2*, 1–38.

HILL, D. J., MINSKER, B. S., AND AMIR, E. 2007. Real-time bayesian anomaly detection for environmental sensor data. In *Proceedings of the 32nd conference of IAHR*. International Association of Hydraulic Engineering and Research, IAHR, Venice, Italy.

HILL, D. J., MINSKER, B. S., AND AMIR, E. 2009. Real-time Bayesian anomaly detection in streaming environmental data. *Water Resources Research 45,* W00D28, 1–16.

HSIN, C. AND LIU, M. 2004. Network Coverage using Low Duty-Cycled Sensors; Random and Coordinated Sleep Algorithms. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN '04)*. ACM/IEEE, 433–442.

HWANG, J., HE, T., AND KIM, Y. 2007. Exploring In-Situ Sensing Irregularity in Wireless Sensor Networks. In *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems (SenSys '07)*. ACM, 289–303.

ISLER, V. AND BAJCSY, R. 2005. The Sensor Selection Problem for Bounded Uncertainty Sensing Models. In *Proceedings of the 4th International Symposium on Sensor Networks (IPSN '05)*. ACM/IEEE, 151–158.

KANG, S., LEE, J., JANG, H., LEE, H., LEE, Y., PARK, S., PARK, T., AND SONG, J. 2008. SeeMon: Scalable and Energy-efficient Context Monitoring Framework for Sensor-right Mobile Environments. In *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services (MobiSys '08)*. ACM, 267–280.

KEALLY, M., ZHOU, G., AND XING, G. 2010. Watchdog: Confident Event Detection in Heterogeneous Sensor Networks. In *Proceedings of the 16th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS '10)*. IEEE, 279–288.

KEALLY, M., ZHOU, G., XING, G., WU, J., AND PYLES, A. 2011. PBN: Towards Practical Activity Recognition Using Smartphone-Based Body Sensor Networks. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems (SenSys '11)*. ACM, 246–259.

KHAN, M., LE, H., LEMAY, M., MOINZADEH, P., WANG, L., YANG, Y., NOH, D., ABDELZAHER, T., GUNTER, C., HAN, J., AND JIN., X. 2010. Diagnostic Powertracing for Sensor Node Failure Analysis. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN '10)*. ACM/IEEE, 117–128.

KIRKPATRICK, S., GELATT, C., AND VECCHI, M. 1983. Optimization by Simulated Annealing. *Science 220*, 671–680.

KRAUSE, A., GUESTRIN, C., GUPTA, A., AND KLEINBERG, J. 2011. Robust sensor placements at informative and communication-efficient locations. *ACM Transactions on Sensor Networks 7,* 4, 31:1–31:33.

KUMAR, S., LAI, T., AND ARORA, A. 2005. Barrier Coverage With Wireless Sensors. In *Proceedings of the 11th Annual International Conference on Mobile Computing and Networking (MobiCom '05)*. ACM, 284–298.

LERNER, U. 2002. Hybrid bayesian networks for reasoning about complex systems. Ph.D. thesis, Stanford University, Palo Alto, California.

LORINCZ, K., CHEN, B., WATERMAN, J., WERNER-ALLEN, G., AND WELSH, M. 2008. Resource Aware Programming in the Pixie OS. In *Proceedings of the 6th ACM Conference on Embedded Networked Sensor Systems (SenSys '08)*. ACM, 211–224.

MALINOWSKI, M., MOSKWA, M., FELDMEIERA, M., LAIBOWITZ, M., AND PARADISO, J. 2008. CargoNet: A Low-Cost MicroPower Sensor Node Exploiting Quasi-Passive Wakeup for Adaptive Asynchronous Monitoring of Exceptional Events. In *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems (SenSys '08)*. ACM, 145–159.

MO, L., HE, Y., LIU, Y., ZHAO, J., TANG, S., LI, X., AND DAI, G. 2009. Canopy Closure Estimates with GreenOrbs: Sustainable Sensing in the Forest. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys '09)*. ACM, 99–112.

RABINER, L. 1989. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In *Proceedings of the IEEE*. IEEE, 257–286.

RACHLIN, Y., NEGI, R., AND KHOSLA, P. 2005. Sensing Capacity for Discrete Sensor Network Applications. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN '05)*. IEEE/ACM, 126–132.

SHNAYDER, V., HEMPSTEAD, M., CHEN, B., ALLEN, G. W., AND WELSH, M. 2004. Simulating the Power Consumption of Large-Scale Sensor Network Applications. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04)*. ACM, 188–200.

SHRIVASTAVA, N., MUDUMBAI, R., MADHOW, U., AND SURI, S. 2006. Target Tracking with Binary Proximity Sensors: Fundamental Limits, Minimal Descriptions, and Algorithms. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys '06)*. ACM, 251–264.

SIMON, G., MAROTI, M., LEDECZI, A., BALOGH, G., KUSY, B., NADAS, A., PAP, G., SALLAI, J., AND FRAMPTON, K. 2004. Sensor Network-Based Countersniper System. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04)*. ACM, 1–12.

SINGH, A., RAMAKRISHNAN, C., RAMAKRISHNAN, I., AND WARREN, D. 2008. A Methodology for In-Network Evaluation of Integrated Logical-Statistical Models. In *Proceedings of the 6th ACM Conference on Embedded Networked Sensor Systems (SenSys '08)*. ACM, 211–224.

SUBRAMANIAM, S., KALOGERAKI, V., AND PALPANAS, T. 2006. Distributed Real-Time Detection and Tracking of Homogeneous Regions in Sensor Networks. In *Proceedings of the 27th International Real-Time Systems Symposium (RTSS '06)*. IEEE, 401–411.

VARSHNEY, P. 1996. *Distributed Detection and Data Fusion*. Springer.

VOLGYESI, P., BALOGH, G., NADAS, A., NASH, C., AND LEDECZI, A. 2007. Shooter Localization and Weapon Classification with Soldier-Wearable Networked Sensors. In *Proceedings of the 5th International Conference on Mobile Systems, Applications, and Services (MobiSys '07)*. ACM, 113–126.

WANG, H., YAO, K., POTTIE, G., AND ESTRIN, D. 2004. Entropy-based Sensor Selection Heuristic for Target Localization. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN '04)*. ACM/IEEE, 36–45.

WANG, W., SRINIVASAN, V., WANG, B., AND CHUA, K. 2006. Coverage for Target Localization in Wireless Sensor Networks. In *Proceedings of the 5th International Conference on Information Processing in Sensor Networks (IPSN '06)*. ACM/IEEE, 118–125.

XING, G., TAN, R., LIU, B., WANG, J., JIA, X., AND WEI, C. 2009. Data Fusion Improves the Coverage of Wireless Sensor Networks. In *Proceedings of the 15th Annual Conference on Mobile Computing and Networking (MobiCom '09)*. ACM, 157–168.

XING, G., WANG, X., ZHANG, Y., LU, C., PLESS, R., AND GILL, C. 2005. Integrated Coverage and Connectivity Configuration for Energy Conservation in Sensor Networks. *ACM Transactions on Sensor Networks 1*, 36–72.

YAN, T., HE, T., AND STANKOVIC, J. 2003. Differentiated Surveillance for Sensor Networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys '03)*. ACM, 51–62.

YANG, G., SHUKLA, V., AND QIAO, D. 2008. A Novel On-Demand Framework for Collaborative Object Detection in Sensor Networks. In *The 27th Conference on Computer Communications (INFOCOM '08)*. IEEE, 2020–2028.

YUAN, Z., TAN, R., XING, G., LU, C., CHEN, Y., AND WANG, J. 2008. Fast Sensor Placement Algorithms for Fusion-based Target Detection. In *Proceedings of the 2008 Real-Time Systems Symposilum (RTSS '08)*. IEEE, 103–112.

ZAPPI, P., LOMBRISER, C., STEIFMEIER, T., FARELLA, E., ROGGEN, D., BENINI, L., AND TROSTER, G. 2008. Activity Recognition from On-Body Sensors: Accuracy-Power Trade-Off by Dynamic Sensor Selection. In *Proceedings of the 5th European Conference on Wireless Sensor Networks (EWSN '08)*. 17–33.

ZHAO, F., SHIN, J., AND REICH, J. 2002. Information-Driven Dynamic Sensor Collaboration for Tracking Applications. *IEEE Signal Processing 19*, 61–72.

ZHUANG, Y., CHEN, L., WANG, X., AND LIAN, J. 2007. A Weighted Moving Average-Based Approach for Cleaning Sensor Data. In *Proceedings of the 27th International Conference on Distributed Computing Systems (ICDCS '07)*. IEEE, 38–.