# The Next Big One: Detecting Earthquakes and other Rare Events from Community-based Sensors

Matthew Faulkner
Caltech

Andreas Krause
Caltech

Michael Olson
Caltech

Rishi Chandy
Caltech

Jonathan Krause
Caltech

K. Mani Chandy
Caltech

## ABSTRACT

Can one use cell phones for earthquake early warning? Detecting rare, disruptive events using community-held sensors is a promising opportunity, but also presents difficult challenges. Rare events are often difficult or impossible to model and characterize a priori, yet we wish to maximize detection performance. Further, heterogeneous, community-operated sensors may differ widely in quality and communication constraints.

In this paper, we present a principled approach towards detecting rare events that learns sensor-specific decision thresholds online, in a distributed way. It maximizes anomaly detection performance at a fusion center, under constraints on the false alarm rate and number of messages per sensor. We then present an implementation of our approach in the Community Seismic Network (CSN), a community sensing system with the goal of rapidly detecting earthquakes using cell phone accelerometers, consumer USB devices and cloud-computing based sensor fusion. We experimentally evaluate our approach based on a pilot deployment of the CSN system. Our results, including data from shake table experiments, indicate the effectiveness of our approach in distinguishing seismic motion from accelerations due to normal daily manipulation. They also provide evidence of the feasibility of earthquake early warning using a dense network of cell phones.

## Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design; G.3 [**Probability and Statistics**]: Experimental Design; I.2.6 [**AI**]: Learning

## 1. INTRODUCTION

Privately owned commercial devices equipped with sensors are emerging as a powerful resource for sensor networks. Community sensing projects are using smart phones to monitor traffic and detect accidents [13, 17, 15]; monitor and improve population health [5], and map pollution [18, 30]. Detecting rare, disruptive events, such as earthquakes, using community-held sensors is a particularly promising opportunity [4], but also presents difficult challenges. Rare events are often difficult or impossible to model and characterize a priori, yet we wish to maximize detection performance. Further, heterogeneous, community-operated sensors may differ widely in quality and communication constraints, due to variability in hardware and software platforms, as well as differing environmental conditions.

In this paper, we present a principled approach towards detecting rare events from community-based sensors. Due to the unavailability of data characterizing the rare events, our approach is based on anomaly detection; sensors learn models of normal sensor data (e.g., acceleration patterns experienced by smartphones under typical manipulation). Each sensor then independently detects unusual observations (which are considered unlikely with respect to the model), and notifies a fusion center. The fusion center then decides whether a rare event has occurred or not, based on the received messages. Our approach is grounded in the theory of decentralized detection, and we characterize its performance accordingly. In particular, we show how sensors can learn decision rules that allow us to control system-level false positive rates and bound the amount of required communication in a principled manner while simultaneously maximizing the detection performance.

As our second main contribution, we present an implementation of our approach in the *Community Seismic Network (CSN)*. The goal of our community sensing system is to detect seismic motion using accelerometers in smartphones and other consumer devices (Figure 1(c)), and issue real-time early-warning of seismic hazards (see Figure 1(a)). The duration of the warning is the time between a person or device receiving the alert and the onset of significant shaking (see Figure 1(b)); this duration depends on the distance between the location of initial shaking and the location of the receiving device, and on delays within the network and fusion center. Warnings of up to tens of seconds are possible [1], and even warnings of a few seconds help in stopping elevators, slowing trains, and closing gas valves. Since false alarms can have high costs, it is important to accurately control the false positive rate of the system.

Using community-based sensors for earthquake early warning is particularly challenging due to the large variety of sensor types, sensor locations, and ambient noise characteristics. For example, a sensor near a construction site will have different behavior than a sensor in a quiet zone. Moreover, sensor behavior may change over time; for example, construction may start in some places and stop in others. With thousands of sensors, one cannot expect to know the precise characteristics of each sensor at each point in time; these characteristics have to be deduced by algorithms. A system that scales to tens of thousands or millions of sensors must limit the rate of message traffic to the cloud computer to a rate that can be handled efficiently by the fusion center. For example, one million phones would produce approximately 30 Terabytes of accelerometer data each day. Another key challenge is to develop a system infrastructure


*IPSN'11,* conference info


(a) *Earthquake (Chino Hills, Magn. 5.4, July 2008)* (b) *Early warning* (c) *Phidget sensor with housing*

**Figure 1: (a) Seismic waves (P- and S-waves) during an earthquake. (b) Anticipated user interface for early warning using our Google Android application. (c) 16-bit USB MEMS accelerometers with housing that we used in our initial deployment.**

that has low response time even under peak load (messages sent by millions of phones during an earthquake). Moreover, the Internet and computers in a quake zone are likely to fail with the onset of intensive shaking. So, data from sensors must be sent out to a distributed, resilient system that has data centers outside the quake zone. The CSN uses cloud-computing based sensor fusion to cope with these challenges. In this paper, we report our initial experience with the CSN, and experimentally evaluate our detection approach based on data from a pilot deployment. Our results, including data from shaketable experiments that allow us to mechanically play back past earthquakes, indicate the effectiveness of our approach in distinguishing seismic motion from accelerations due to normal daily manipulation. They also provide evidence towards feasibility of earthquake early warning using a dense network of cell phones.

In summary, our main contributions are

- a novel approach for online decentralized anomaly detection,
- a theoretical analysis, characterizing the performance of our detection approach,
- an implementation of our approach in the Community Seismic Network, involving smartphones, USB MEMS accelerometers and cloud-computing based sensor fusion, and
- a detailed empirical evaluation of our approach characterizing the achievable detection performance when using smartphones to detect earthquakes.

## 2. PROBLEM STATEMENT

We consider the problem of decentralized detection of rare events, such as earthquakes, under constraints on the number of messages sent by each sensor. Specifically, a set of $N$ sensors make repeated observations $\mathbf{X}_t = (X_{1,t}, \ldots, X_{N,t})$ from which we would like to detect the occurrence of an event $E_t \in \{0, 1\}$. Hereby, $X_{s,t}$ is the measurement of sensor $s$ at time $t$, and $E_t = 1$ iff there is an event (e.g., an earthquake) at time $t$. $X_{s,t}$ may be a scalar (e.g., acceleration), or a vector of features containing information about Fourier frequencies, moments etc. during a sliding window of data (see Section 4.2 for a discussion of features that we use in our system).

We are interested in the decentralized setting, where each sensor $s$ analyzes its measurements $X_{s,t}$, and sends a message $M_{s,t}$ to the fusion center. Here we will focus on binary messages (i.e., each sensor gets to vote on whether there is an event or not). In this case, $M_{s,t} = 1$ means that sensor $s$ at time $t$ estimates that an event happened; $M_{s,t} = 0$ means that sensor $s$ at time $t$ estimates that no event happened at that time. For large networks, we want to minimize the number of messages sent. Since the events are assumed to be rare, we only need to send messages (that we henceforth call *picks*) for $M_{s,t} = 1$; sending no message implies $M_{s,t} = 0$. Based on the received messages, the fusion center then decides how to respond: It produces an estimate $\widehat{E}_t \in \{0, 1\}$. If $\widehat{E}_t = E_t$, it makes the correct decision (*true positive* if $E_t = 1$ or *true negative* if $E_t = 0$). If $\widehat{E}_t = 0$ when $E_t = 1$, it missed an event and thus produced a *false negative*. Similarly, if $\widehat{E}_t = 1$ when $E_t = 0$, it produced a *false positive*. False positives and false negatives can have very different costs. In our earthquake example, a false positive could lead to incorrect warning messages sent out to the community and consequently lead to inappropriate execution of remedial measures such as slowing down trains. On the other hand, false negatives could lead to missed opportunities for protecting the infrastructure and saving lives. In general, our goal will be to minimize the frequency of false negatives while constraining the (expected) frequency of false positives to a tolerable level (e.g., at most one false alarm per year).

**Classical Decentralized Detection.** How should each sensor, based on its measurements $X_{s,t}$, decide when to *pick* (send $M_{s,t} = 1$)? The traditional approach to decentralized detection assumes that we know how likely particular observations $X_{s,t}$ are, in case of an event occurring or not occurring. Thus, it assumes we have access to the conditional probabilities $\mathbb{P}[X_{s,t} \mid E_t = 0]$ and $\mathbb{P}[X_{s,t} \mid E_t = 1]$. In this case, under the common assumptions that the sensors' measurements are independent conditional on whether there is an event or not, it can be shown that the optimal strategy is to perform *hierarchical hypothesis testing* [28]: we define two thresholds $\tau, \tau'$, and let $M_{s,t} = 1$ iff

$$\frac{\mathbb{P}[X_{s,t} \mid E_t = 1]}{\mathbb{P}[X_{s,t} \mid E_t = 0]} \geq \tau. \tag{1}$$

i.e., if the likelihood ratio exceeds $\tau$. Similarly, the fusion center sets $\widehat{E}_t = 1$ iff

$$\frac{\mathrm{Bin}(S_t; p_1; N)}{\mathrm{Bin}(S_t; p_0; N)} \geq \tau', \tag{2}$$

where $S_t = \sum_s M_{s,t}$ is the the number of picks at time $t$; $p_\ell = \mathbb{P}[M_{s,t} = 1 \mid E_t = \ell]$ is the sensor-level true ($\ell = 1$) and false ($\ell = 0$) positive rate respectively; and $\mathrm{Bin}(\cdot, p, N)$ is the probability mass function of the Binomial distribution. Asymptotically optimal decision performance in either a Bayesian or Neyman-Pearson framework can be obtained by using the decision rules (1) and (2) with proper choice of the thresholds $\tau$ and $\tau'$ [28].

There has also been work in *quickest detection* or *change detection* (*cf.*, [22] for an overview), where the assumption is that there is some time point $t_0$ at which the event occurs; $X_{s,t}$ will be distributed according to $\mathbb{P}[X_{s,t} \mid E_t = 0]$ for all $t < t_0$, and according to $\mathbb{P}[X_{s,t} \mid E_t = 1]$ for all $t \geq t_0$. In change detection,

the system trades off waiting (gathering more data) and improved detection performance. However, in case of rare *transient* events (such as earthquakes) that may occur repeatedly, the distributions $\mathbb{P}[X_{s,t} \mid E_t = 1]$ are expected to change with $t$ for $t \geq t_0$.

**Challenges for the Classical Approach.** Detecting rare events from community-based sensors has three main challenges:

(i) Sensors are highly heterogeneous (i.e., the distributions $\mathbb{P}[X_{s,t} \mid E_t]$ are different for each sensor $s$)
(ii) Since events are rare, we do not have sufficient data to obtain good models for $\mathbb{P}[X_{s,t} \mid E_t = 1]$
(iii) Bandwidth limitations may limit the amount of communication (e.g., number of picks sent).

Challenge (i) alone would not be problematic – classical decentralized detection can be extended to heterogeneous sensors, as long as we know $\mathbb{P}[X_{s,t} \mid E_t]$. For the case where we do not know $\mathbb{P}[X_{s,t} \mid E_t]$, but we have training examples (i.e., large collections of sensor data, annotated by whether an event is present or not), we can use techniques from nonparametric decentralized detection [20]. In the case of rare events, however, we may be able to collect large amounts of data for $\mathbb{P}[X_{s,t} \mid E_t = 0]$ (i.e., characterizing the sensors in the no-event case), while still collecting extremely little (if any) data for estimating $\mathbb{P}[X_{s,t} \mid E_t = 1]$. In our case, we would need to collect data from cell phones experiencing seismic motion of earthquakes ranging in magnitude from four to ten on the Gutenberg-Richter scale, while resting on a table, being carried in a pocket, backpack, etc. Furthermore, even though we can collect much data for $\mathbb{P}[X_{s,t} \mid E_t = 0]$, due to challenge (iii) we may not be able to transmit all the data to the fusion center, but have to estimate this distribution locally, possibly with limited memory. We also want to choose decision rules (e.g., of the form (1)) that minimize the number of messages sent.

# 3. ONLINE DECENTRALIZED ANOMALY DETECTION

We now describe our approach to online, decentralized detection of anomalous events.

**Assumptions.** In the following, we adopt the assumption of classical decentralized detection that sensor observations are conditionally independent given $E_t$, and independent across time (i.e., the distributions $\mathbb{P}[X_{s,t} \mid E_t = 0]$ do not depend on $t$). For earthquake detection this assumption is reasonable (since most of the noise is explained through independent measurement noise, and manipulation through the sensors' owners). While spatial correlation may be present, e.g., due to mass events such as rock concerts, it is expected to be relatively rare. Furthermore, if context about such events is available in advance, it can be taken into account. We defer treatment of spatial correlation to future work. We do *not* assume that the sensors are homogeneous (i.e., $\mathbb{P}[X_{s,t} \mid E_t = 0]$ may depend on $s$). Our approach can be extended in a straightforward manner if the dependence on $t$ is periodic (e.g., the background noise changes based on the time of day, or day within week). We defer details to a long version of this paper.

**Overview.** The key idea behind our approach is that since sensors obtain a massive amount of normal data, they can accurately estimate $\mathbb{P}[X_{s,t} \mid E_t = 0]$ purely based on their local observations. In our earthquake monitoring example, the cell phones can collect data of acceleration experienced under normal operation (lying on a table, being carried in a backpack, etc.). Further, if we have any hope of detecting earthquakes, the signal $X_{s,t}$ has to be sufficiently different from normal data (thus $\mathbb{P}[X_{s,t} \mid E_t = 0]$ must be low when $E_t = 1$). This suggests that each sensor should decide whether to pick or not based on the likelihood $L_0(x) = \mathbb{P}[x \mid E_t = 0]$ only; sensor $s$ will pick ($M_{s,t} = 1$) iff, for its current readings $X_{s,t} = x$ it holds that

$$L_0(x) < \tau_s \tag{3}$$

for some sensor specific threshold $\tau_s$. See Figure 5(c) for an illustration. Note that using this decision rule, for the probability of a pick it holds that $\mathbb{P}[M_{s,t} = 1 \mid E_t = e] = \mathbb{P}[L_0(X_{s,t}) < \tau_s \mid E_t = e] = p_e$. This anomaly detection approach hinges on the following fundamental anti-monotonicity assumption: It assumes that

$$L_0(x) < L_0(x') \Leftrightarrow \frac{\mathbb{P}[x \,|\, E_t{=}1]}{\mathbb{P}[x \,|\, E_t{=}0]} > \frac{\mathbb{P}[x' \,|\, E_t{=}1]}{\mathbb{P}[x' \,|\, E_t{=}0]}, \tag{4}$$

i.e., the less probable $x$ is under normal data, the larger the likelihood ratio gets in favor of the anomaly. The latter is the assumption on which most anomaly detection approaches are implicitly based. Under this natural anti-monotonicity assumption, the decision rules (3) and (1) are equivalent, for an appropriate choice of thresholds.

PROPOSITION 1. *Suppose Condition* (4) *holds. Then for any threshold $\tau$ for rule* (1)*, there exists a threshold $\tau_s$ such that rule* (3) *makes identical decisions.*

Since the sensors do not know the true distribution $\mathbb{P}[X_{s,t} \mid E_t = 0]$, they use an online density estimate $\widehat{\mathbb{P}}[X_{s,t} \mid E_t = 0]$ based on collected data. The fusion center will then perform hypothesis testing based on the received picks $M_{s,t}$. In order for this approach to succeed we have to specify:

(i) How can the sensors estimate the distribution $\widehat{\mathbb{P}}[X_{s,t} \mid E_t = 0]$ in an online manner, while using limited resources (CPU, battery, memory, I/O)?
(ii) How should the sensors choose the thresholds $\tau_s$?
(iii) Which true positive and false positive rates $p_1, p_0$ and threshold $\tau'$, *cf.*, (2), should the fusion center use?

We now discuss how our approach addresses these questions.

**Online Density Estimation.** For each sensor $s$, we have to, over time, estimate the distribution of *normal* observations $\widehat{L}_0(X_{s,t}) = \widehat{\mathbb{P}}[X_{s,t} \mid E_t = 0]$, as well as the activation threshold $\tau_s$. There are various techniques for online density estimation. Parametric approaches assume that the density $\mathbb{P}[X_{s,t} \mid E_t = 0]$ is in some parametric family of distributions:

$$\mathbb{P}[X_{s,t} \mid E_t = 0] = \phi(X_{s,t}, \theta).$$

The goal is then to update the parameters $\theta$ as more and more data is obtained. In particular, mixture distributions, such as mixtures of Gaussians, are a flexible parametric family for density estimation. If access to a batch of training data is available, algorithms such as the Expectation Maximization algorithm can be used to obtain parameters that maximize the likelihood of the data. However, due to memory limitations, it is rarely possible to keep all data in memory; furthermore, model training would grow in complexity as more data is collected. Fortunately, several effective techniques have been proposed for incremental optimization of the parameters, based on Variational Bayesian techniques [25] and particle filtering [8]. Online nonparametric density estimators (whose complexity, such as the number of mixture components, can increase with the amount of observed data) have also been developed [10]. In this paper, we use Gaussian mixture models for density estimation.

**Online Threshold Estimation.** Online density estimators as introduced above allow us to estimate $\widehat{\mathbb{P}}[X_{s,t} \mid E_t = 0]$. The remaining question is how the sensor-specific thresholds $\tau_s$ should be chosen. The key idea is the following. Suppose we would like to control the per-sensor false positive rate $p_0$ (we need to do that in order to perform hypothesis testing in the fusion center). Since the event is assumed to be extremely rare, with very high probability (close to 1) *every* pick $M_{s,t} = 1$ will be a false alarm. Thus, we would like to choose our threshold $\tau_s$ such that, if we obtain a measurement $X_{s,t} = x$ at random, with probability $1 - p_0$, it holds that $\widehat{L}_0(x) \geq \tau_s$.

This insight suggests a natural approach to choosing $\tau_s$: For each training example $x_{s,t}$, we calculate its likelihood $\widehat{L}_0(x_{s,t}) = \widehat{\mathbb{P}}[x_{s,t} \mid E_t = 0]$. We then choose $\tau_s$ to be the $p_0$-th percentile of the data set $\mathfrak{L} = \{\widehat{L}_0(x_{s,1}), \ldots, \widehat{L}_0(x_{s,t})\}$. As we gather an increasing amount of data, as long as we use a consistent density estimator, this procedure will converge to the correct decision rule.

In practice, due to memory and computation constraints, we cannot keep the entire data set $\mathfrak{L}$ of likelihoods and reestimate $\tau_s$ at every time step. Unfortunately, percentiles do not have sufficient statistics as the mean and other moments do. Moreover, Munro and Paterson [19] show that computing rank queries exactly requires $\Omega(n)$ space. Fortunately, several space-efficient online $\varepsilon$-approximation algorithms for rank queries have been developed. An algorithm that selects an element of rank $r'$ from $N$ elements for a query rank $r$ is said to be *uniform $\varepsilon$-approximate* if

$$\frac{|r' - r|}{N} \leq \varepsilon$$

One such algorithm which requires logarithmic space is given by [11]. We do not present details here due to space limitations. We summarize our analysis in the following proposition:

PROPOSITION 2. *Suppose that we use a uniformly consistent density estimator (i.e., $\limsup_x \{\widehat{\mathbb{P}}[x \mid E_t = 0] - \mathbb{P}[x \mid E_t = 0]\} \to 0$ a.s.). Further suppose that $\tau_{s,t}$ is an $\varepsilon$-accurate threshold obtained through percentile estimation for $p_0$. Then for any $\varepsilon > 0$, there exists a time $t_0$ such that for all $t \geq t_0$, it holds that the false positive probability $\widehat{p}_0 = \mathbb{P}\left[\widehat{L}_0(x_{s,t}) < \tau_s\right]$ is $|\widehat{p}_0 - p_0| \leq 2\varepsilon$.*

The proof of Proposition 2, which we omit for space limitations, hinges on the fact that if the estimate $\widehat{L}_0(x)$ converges uniformly to $L_0(x)$, the $p_0$-th percentiles (for $0 < p_0 < 1$) converge as well. Furthermore, the use of an $\varepsilon$-approximate percentile can change the false positive rate by at most $\varepsilon$.

Uniform convergence rates for density estimation have been established as well [9], allowing us to quantify the time required until the system operates at $\varepsilon$-accurate false positive rates. Since we assume that communication is expensive, we may impose an upper bound on the expected number of messages sent by each sensor. This can be achieved by imposing an upper bound $\bar{p}$ on $p_0$, again relying on the fact that events are extremely rare. We present more details in the next section.

**Hypothesis Testing for Sensor Fusion.** Above, we discussed how we can obtain local decision rules that allow us to control the sensor-level false positive rate $p_0$ in a principled manner, and in the following we assume that the sensors operate at this false positive rate. However, in order to perform hypothesis testing as in (2), it appears that we also need an estimate of the sensor-level true-positive rate $p_1$.

Suppose that we would like to maximize the detection rate $P_D$ at the fusion center while guaranteeing a false positive rate $P_F$ that is bounded by $\bar{P}$ (e.g., at most one false positive per year). It can

**Data**: Estimated sensor ROC curve, N sensors, communication constraints $\bar{p}$, bound on fusion-level false positives $\bar{P}$
**Result**: sensor operating point $(p_0, p_1)$
**for** *$i^{th}$ operating point $(p_0^i, p_1^i)$ s.t. $p_0^i \leq \bar{p}$* **do**
  *//Do Neyman-Pearson hypothesis testing to evaluate $p_0^i$*
  Compute $N(p_0^i) = \min\{N' : \sum_{S>N'} \text{Bin}(S; p_0^i; N) \leq \bar{P}\}$
  Compute $P_D^i = \sum_{S>N(p_0^i)} \text{Bin}(S; p_1^i; N)$
  Compute $P_F^i = \sum_{S>N(p_0^i)} \text{Bin}(S; p_0^i; N)$
Choose $\ell = \arg\max_i P_D^i$ and set $(p_0, p_1) = (p_0^\ell, p_1^\ell)$

**Figure 2: Threshold Optimization procedure**

be shown that the optimal decision rule (2) is equivalent to setting $\widehat{E}_t = 1$ iff $S_t \geq N(p_0)$, for some number $N(p_0)$ that *only* depends on the total number $N$ of sensors, and the sensor false-positive rate $p_0$. Thus, to control the fusion-level false positive rate $P_F$ we, perhaps surprisingly, *do not need to know* the value for $p_1$, since $P_F$ does not depend on $p_1$:

$$P_F = \sum_{S>N(p_0)} \text{Bin}(S; p_0; N) \text{ and } P_D = \sum_{S>N(p_0)} \text{Bin}(S; p_1; N).$$

Thus, our online anomaly detection approach leads to decision rules that provide guarantees about the fusion-level false positive rate.

Our goal is not just to bound the false positive rate, but also to maximize detection performance. The detection performance $P_D$ above depends on the sensor-level true positive rate $p_1$. If we have an accurate estimate of $p_1$, all sensors are homogeneous and the anti-monotonicity condition (4) holds, the following result, which is a consequence of [28], holds:

THEOREM 3. *Suppose condition* (4) *holds and the sensors are all homogeneous (i.e., $\mathbb{P}[X_{s,t} \mid E_t]$ is independent of $s$). Further suppose that for each sensor-level false-positive rate $p_0$ we know its true-positive rate $p_1$. Then one can choose an operating point $(p_0^*, p_1^*)$ that is asymptotically optimal (as $N \to \infty$).*

Unfortunately, without access to training data for actual events (e.g., sensor recordings during many large earthquakes), we will not be able to obtain an accurate estimate for $p_1$. However, in Section 5, we show how we can obtain an empirical estimate $\widehat{p}_1$ of $p_1$ by performing shaketable experiments. Suppose now that we have an estimate $\widehat{p}_1$ of $p_1$. How does the detection performance degrade with the accuracy of $\widehat{p}_1$? Suppose we have access to an estimate of the sensors' Receiver Operator Characteristic (ROC) curve, i.e., the dependency of the achievable true positive rates $\widehat{p}_1(p_0)$ as a function of the false positive rate (see Figure 7(a) for an example). Now we can view both the estimated rates $\widehat{P}_D \equiv \widehat{P}_D(\widehat{p}_1(p_0)) \equiv \widehat{P}_D(p_0)$ and $\widehat{P}_F = \widehat{P}_F(p_0)$ as functions of the sensor-level false positive rate $p_0$. Based on the argument above, we have that $\widehat{P}_F(p_0) = P_F(p_0)$, i.e., the estimated false positive rate is exact, but in general $\widehat{P}_D(p_0) \neq P_D(p_0)$. Fortunately, it can be shown that if the estimated ROC curve is *conservative* (i.e., $\widehat{p}_1(p_0) \leq p_1(p_0)$ for all rates $p_0$), then it holds that $\widehat{P}_D(p_0) \leq P_D(p_0)$ is an *underestimate* of the true detection probability. Thus, if we are able to obtain a pessimistic estimate of the sensors' ROC curves, we can make guarantees about the performance of the decentralized anomaly detection system. We can now choose the optimal operating point by

$$\max_{p_0 \leq \bar{p}} \widehat{P}_D(p_0) \text{ s.t. } \widehat{P}_F(p_0) \leq \bar{P},$$

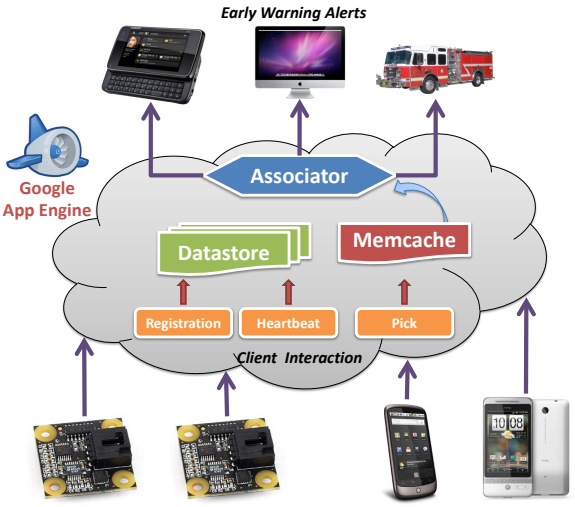



**Figure 3: Overview of the CSN system.**

and are guaranteed that the optimal value of this program is a pessimistic estimate of the true detection performance, while $\widehat{P}_F$ is in fact the exact false alarm rate. Algorithm 2 formalizes this procedure. We summarize our analysis in the following theorem:

THEOREM 4. *If we use decentralized anomaly detection to control the sensor false positive rate $p_0$, and if we use a conservative estimated ROC curve $(p_0, \widehat{p}_1)$, then Algorithm 2 chooses an operating point $p_0$ to maximize a lower bound on the true detection performance.*

# 4. THE COMMUNITY SEISMIC NETWORK

We are building a Community Seismic Network (CSN) to: (a) provide warning about impending shaking from earthquakes, (b) guide first responders to areas with the greatest damage after an earthquake (c) obtain fine-granularity maps of subterranean structures in areas where geological events such as earthquakes or landslides occur, and (d) provide detailed analysis of deformations of internal structures of buildings (that may not be visible to inspectors) after geological events. The CSN is a challenging case study of community sense and response systems because the community has to be involved to obtain the sensor density required to meet these goals, the benefits of early warning of shaking are substantial, and frequent false warnings result in alerts being ignored.

The technical innovations of the CSN include the use of widely heterogeneous sensors managed by a cloud computing platform that also executes data fusion algorithms and sends alerts. Heterogeneous sensors include cell phones, stand-alone sensors that communicate to the cloud computing system in different ways, and accelerometers connected through USB to host computers which are then connected to the cloud through the Internet. Figure 3 presents an overview of the CSN. An advantage of the cloud computing system is that sensors anywhere in the world with Internet access including areas such as India, China, and parts of South America can connect to the system easily merely by specifying a URL.

## 4.1 Community Sensors: Android and USB Accelerometers

The Community Seismic Network currently uses two types of sensors: 16-bit MEMS accelerometers manufactured by Phidgets, Inc., used as USB-accessories to laptops and desktop computers (see Figure 1(c)), as well as accelerometers in Google Android smartphones (see Figure 1(b)) – it will be extended to use other types of phones in the future. Each of the sensors has unique advantages: The USB sensors provide higher fidelity measurements. By firmly affixing them to a non-carpeted floor (preferably) or a wall background noise can be drastically removed. However, their deployment relies on the community purchasing a separate piece of hardware (currently costing roughly USD 150 including custom housing). In contrast, the Google Android platform has a large market share, currently approximately 16.3% (more than the Apple iOS platform, and slightly less than Research in Motion), and is projected to grow further [3]. Android based smartphones typically have 3-axes accelerometers built in, and integration of an Android phone into the CSN only requires download of a free application. On the other hand, the built-in accelerometers are of lower quality (our experiments showed a typical resolution of approximately 13 bits), and phones are naturally exposed to frequent acceleration during normal operation. We have also built early prototypes of stand-alone devices on top of Arduino boards that connect through USB or WiFi to computing systems with access to the cloud.

Are inexpensive accelerometers sensitive enough to detect seismic motion? We performed experiments to assess the fidelity of the Phidgets and a variety of Android phones (the HTC Dream, HTC Hero and Motorola Droid). We placed the sensors on a flat surface and recorded for an hour. We found that when resting, the phones experienced noise with standard deviation $\approx 0.08m/s^2$, while the Phidgets experienced noise with standard deviation of $\approx 0.003m/s^2$. Earthquakes with magnitude 4 on the Gutenberg-Richter scale achieve an acceleration of approximately $.12m/s^2$ close to the epicenter, which can be detected with the Phidgets, but barely exceeds the background noise level of the phones. However, earthquakes of magnitude 5 already achieve acceleration of .5 m/s$^2$, increasing to roughly 1.5 m/s$^2$ for magnitude 6 events. These numbers suggest that cell phone accelerometers should be sensitive enough to be able to detect large earthquakes.

So far, the CSN is a research prototype, and sensors have been deployed and are operated by the members of our research group. We anticipate opening the system to a larger community in the coming weeks. The research reported in this paper presents a feasibility study that we performed in order to justify the deployment of the system. Figure 4(a) presents the locations where messages have been reported from in our network.

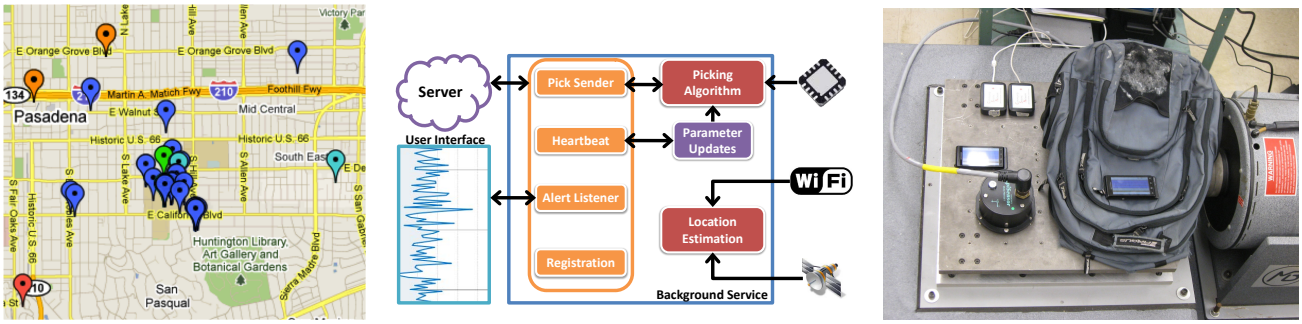
## 4.2 Android Client

Figure 4(b) presents an overview of our Android client application. It consists of several components, which we explain in the following. The client for the Phidget sensors follows a similar implementation, and a detailed discussion is omitted due to space limitations.

A central policy decision of the system was that the only manner in which information is exchanged between a client computer and the cloud computing system is for the client to send a message to the cloud and await a reply: in effect to execute a remote procedure call. All information exchanges are initiated by a client, never by the cloud. This helps ensure that participants in the CSN are only sent information at points of their choosing.

**Registration.** Upon startup for the first time, the application registers with the Cloud Fusion Center (CFC). The CFC responds with a unique identifier for the client, which will be used in all subsequent communications with the CFC.

**Picking Algorithm.** A background process runs continuously on a client computer connected to each sensor and collects data from




(a) *Pick locations* (b) *Android client architecture* (c) *Shaketable*

**Figure 4: (a) Map of locations where measurements have been reported from during our pilot deployment of CSN. (b) Architecture of the phone client software. (c) Experimental setup for playing back historic earthquakes on a shaketable, and testing their effect on the sensors of the CSN system.**

the sensor. The picking algorithm generates "pick" messages by analyzing raw accelerometer data to determine if the data in the recent past is anomalous. The algorithm executes in the background without a user being aware of its existence. It implements the approach discussed in Section 3.

For density estimation, we use a Gaussian mixture model for $\mathbb{P}[X_{s,t} \mid E]$. The most important design choice is the representation $X_{s,t}$ of the sensor data. Our approach is to compute various features from short time windows (similar to phonemes in speak recognition). The idea is that normal acceleration, e.g., due to walking, or manual phone operation, lead to similar signatures of features.

A first challenge is that phones frequently change their orientation. Since accelerometers measure gravity, we first determine (using a decaying average) and subtract out the gravity component from the $[X, Y, Z]$-components of the signal. We then rotate the centered signal such that the estimated gravity component points in the negative Z direction $[0, 0, -1]$. Figures 5(a) and 5(b) illustrate this process. Since we cannot consistently orient the other axes, we use features that are invariant under rotation around the vertical (Z) axis, by replacing the $[X, Y]$ component by its Euclidean norm $||[X, Y]||_2$.

We consider time windows of 2.5 seconds length and, for both the $Z$ and $||[X, Y]||_2$ components calculate 16 Fourier coefficients, the second moment, and the maximum absolute acceleration. This procedure results in a 36-dimensional feature vector. To avoid the curse of dimensionality we perform linear dimensionality reduction by projection on the top 16 components. These principal components can be computed using online algorithms [31]. While PCA captures most variance in the training data (normal acceleration patterns), it is expected that unusual events may carry energy in directions not spanned by the principal components. We therefore add the projection error (amount of variance not captured by the projection) as an additional feature. We arrived at this choice of features, as well as the number $k = 6$ of mixture components through careful cross-validation experiments, using our experimental setup discussed in Section 5.

Our threshold for picking is obtained using online percentile estimation, as detailed in Section 3. In order to bootstrap the deployment of Gaussian mixture models to new phones, our phone client has the capability of updating its statistical model via messages from the CFC. The threshold by which the algorithm on a client computer determines whether an anomaly is present can also be changed by a message from the cloud computer. This allows the CFC to throttle the rate at which a given sensor generates pick messages.

**Pick Reporting.** A background process continually runs, collecting data from the accelerometers onboard the Android phone. Whenever the picking algorithm declares a pick, a message is sent to the CFC, which includes the time, location, and estimated amplitude of the data which triggered the pick. Including the location is important for two reasons. First, for mobile clients, it is more efficient than receiving regular location updates. Second, sending the location is helpful in order to facilitate faster association by avoiding database lookups for every stationary sensor pick. How much battery does the background process draw? In our experiments on the Motorola Droid, the battery life exceeded 25 hours while continuously running the client (but without any further operation of the phone).

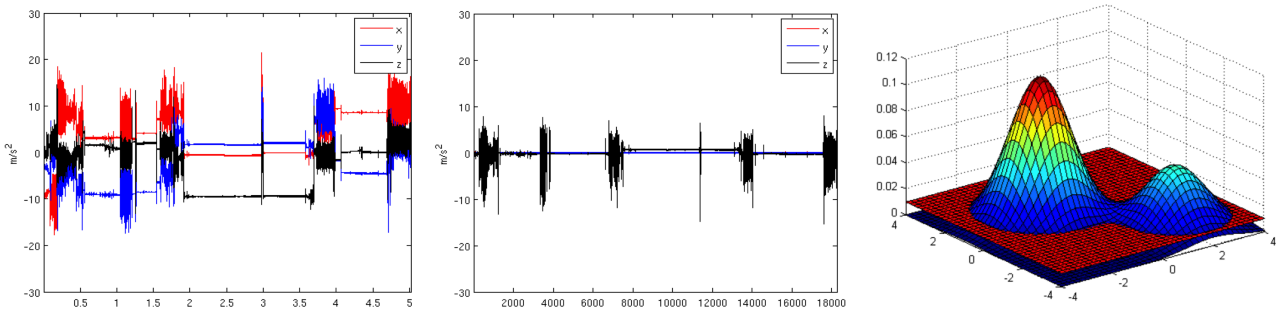
**Heartbeats.** At some prespecified interval, "heartbeat" messages are sent to the CFC, allowing the CFC to keep track of which phones are currently running the program. The heartbeats contain the time, location, waveform logs, and a parameter version number. Using the parameter version number, the CFC can determine whether to send updated parameters to each phone or not. This mechanism allows modifications to the picking algorithm without requiring changes to the underlying client software.

**User interface.** While the main application runs in the background using Android's multitasking capability, the application provides a user interface to display the recorded waveforms. We are currently collaborating with a USGS led effort in earthquake early warning. Our application will connect to the early warning system and be able to issue warnings about when shaking will occur, as well as the estimated epicenter of the event (see Figure 1(b)). While the application is currently a research prototype and not yet deployed in public use, we anticipate that the capability of real-time early warning may incentivize users to download and install the application.

## 4.3 Cloud Fusion Center

In devising a system to serve as a logically central location for data fusion, we evaluated building our own server network, using virtualized remote servers, having collocated servers, and building our application to work on Google App Engine. App Engine was chosen as platform for the Cloud Fusion Center for several reasons: easy scalability, built in data security, and ease of maintenance.

The App Engine platform is designed from the ground up to be scalable to an arbitrary number of clients. As we expect to grow our sensor network to a very high sensor density, this element of the platform's design is very important. What makes the scalability of the platform easily accessible is the fact that incoming requests are automatically load-balanced between instances that are created and destroyed based on current demand levels. This reduces algorithmic




(a) *Before gravity correction* (b) *After gravity correction* (c) *GMM based picking*

**Figure 5: (a) 5 hours of recording three-axis accelerometer data during normal cell phone use. (b) Data from (a) after removing gravity and appropriate signal rotation. (c) Illustration of the density estimation based picking algorithm. The red plane shows an operating threshold. Acceleration patterns for which the density does not exceed the threshold result in a pick.**

complexity, as the load balancing of the network is handled by the platform rather than by code written for our CSN system.

A second consideration in our selection was data security. With the other solutions we had available to us, if the data we collected was stored on the server network we were using, then, without redundant servers in separate geographies, we risked losing all of our data to the very earthquakes we hoped to record. App Engine solves this problem for us by automatically replicating datastore writes to geographically separate data centers as the writes are processed. This achieves the level of data redundancy and geographical separation we require, without forcing us to update our algorithms. Other network storage solutions would have been possible as well, but having it built into the platform meant that latency for code accessing the storage would be lower.

A final compelling reason to select the App Engine was its ease of maintenance. Rather than spending time building server images and designing them to coordinate with each other, we were able to immediately begin working on the algorithms that were most important to us. Server maintenance, security, and monitoring are all handled by the App Engine team and do not take away from the time of the research team members.

App Engine also includes a number of other benefits we considered. First, it utilizes the same front ends that drive Google's search platform, and, consequently, greatly reduces latency to the platform from any point in the world. Since we plan to expand this project beyond Southern California, this is very useful. Second, the platform supports live updates to running applications. Rather than devising server shutdown, update, and restart mechanisms as is commonly required, we can simply redeploy the application that serves our sensors and all new requests to the CFC will see the new code instead of the old code with no loss of availability.

All of these features do not come without a price, however. We will discuss what we perceive as the two largest drawbacks of the platform: loading requests and design implications.

**Loading Requests.** Because App Engine dynamically scales the number of available instances available to serve a given application as the volume of requests per unit time changes, it creates a phenomenon known as a loading request. This request is named in this manner because it is the first request to a new instance of the application. That is, when App Engine allocates a new instance to serve increasing traffic demands, it sends an initial live request to that instance. In Java, this results in the initialization of the Java Virtual Machine, including loading all of the appropriate libraries.

Over the last three months, we experienced loading requests with a median frequency of 9.52% of all requests. While this means that 90.48% of requests did not experience increased latency as a result of the platform, the remaining requests experienced a median increased processing duration of 5,400 ms. Because of the extreme penalty paid by loading requests, when examining average request duration, their presence dominates the figures. This results in a unique property of App Engine, which is that the system performs much better at higher request loads.

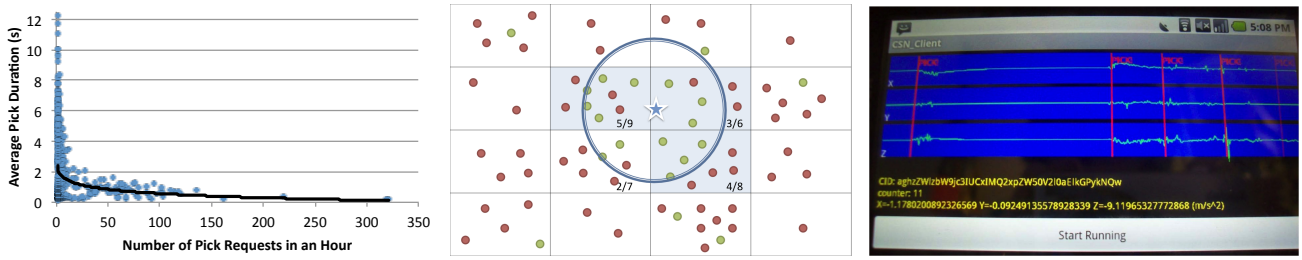
Fig. 6(a), shows that, as the request volume increases, the average duration of each request decreases. This is a result of a reduced impact of loading requests. This data leads to the conclusion that if we avoid potential bottleneck points such as datastore writes, we can expect our performance to stay the same or get better for any increased future load imposed on the system (e.g., as the number of sensors scales up).

**Design Implications.** When designing an algorithm to run on App Engine, the algorithm has to fit inside of the constraints imposed by the architecture. There are a few factors to consider. First, as a result of the automatic scaling done by App Engine, every request to the system exists in isolation. That is, the running requests maintain no shared state, nor do they have any inter-process communication channels. Additionally, since there are no long running background processes, maintaining any form of state generated as a result of successive calls is more difficult. In order to accurately ascertain the number of incoming picks in a unit time over a specified geography, we had to surmount these hurdles.

The only common read/write data sources provided are memcache (a fast key value store) and datastore. The datastore is a persistent object store used for permanent data archiving for future analysis or retrieval. Long term state which changes infrequently, such as the number of active sensors in a given region, is stored and updated in the datastore, but cached in the memcache for quick access. Due to its slower performance, particularly in aggregating writes for immediate retrieval by other processes, it is unsuitable for short term state aggregation.

Short term state, such as the number of picks arriving in an interval of time in a particular region, is stored in memcache. While memcache is not persistent, as objects can be ejected from the cache due to memory constraints, operations that utilize the memcache are much faster. Memcache is ideal for computations that need to occur quickly, and, because memcache allows values to set an expiry time, it is also perfect for data whose usefulness expires after a period of time. That is, after a long enough period of time has passed since a pick arrived, it can no longer be used in detecting an event; therefore, its contributed value to the memcache can be safely expired.

Memcache operates as a key value store, effectively a distributed




(a) *Pick duration* (b) *Associator* (c) *Android Client*

**Figure 6: (a) Average duration of a pick request as a function of system load. (b) Model of dispersed sensors using a hash to a uniform grid to establish proximity. (c) A picture of the CSN android client in debug mode, capturing picks.**

hash table. In order to determine how many sensors sent picks in a given period of time, we devised a system of keys which could be predictably queried to ascertain the number of reporting sensors. We used a geography hashing scheme to ascribe an integer value to every latitude/longitude pair, which generates a uniform grid of cells whose size we can control, with each sensor fitting into one cell in the grid (see Fig. 6(b)). Incoming picks then update the key corresponding to a string representation of the geographical hash and a time bucket derived by rounding the arrival time of the pick to the nearest second.

In this manner, independent processes aggregate their state, and each process runs the hypothesis testing algorithm of Section 3 in the cell whose state it updated to determine the value of $\widehat{E}_t$. If $\widehat{E}_t = 0$, then no action needs to be taken. If $\widehat{E}_t = 1$ a task queue task is launched to initiate the alert process; the task is named using the hash values that generated the alert. Each named task creates a 'tombstone' (a marker in the system) on execution which prevents additional tasks with the same name from being created, so even if successive picks also arrive at the $\widehat{E}_t = 1$ conclusion, we know that only one alert will be sent out for a given set of inputs.

## 5. EXPERIMENTS

Could a network of cheap community sensors detect the next large earthquake? We obtain accurate estimates of the distribution of normal sensor data by collecting records from volunteers' phones and and USB accelerometers. Using an earthquake shaketable and records of ground acceleration gathered by the Southern California Seismic Network (SCSN) during moderately large earthquakes, we obtain estimates of each sensor's ROC curves. These estimates of sensor performance allow us to evaluate the effect of network density and composition on the detection rate. Finally, we apply the learned detection models to data from the 2010 Baja California M7.2 quake.

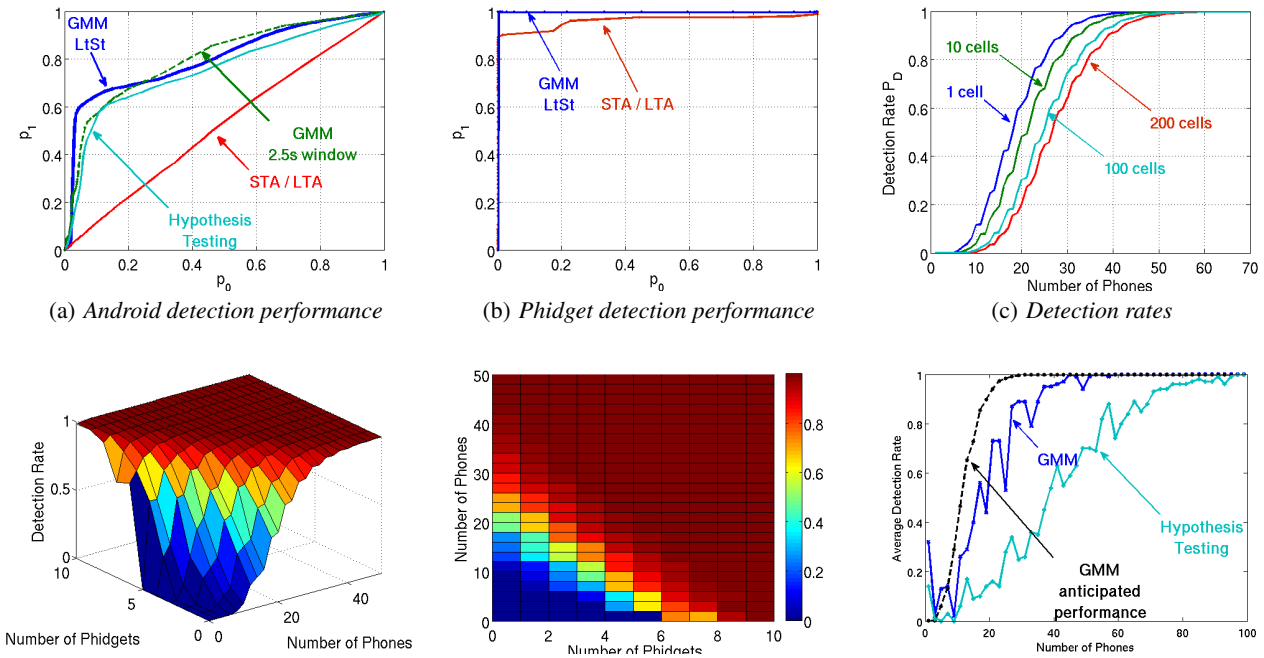
**Data Sets.** While earthquakes are rare, data gathered from community sensors can be plentiful. To characterize "normal" (background) data, seven volunteers from our research group carried Android phones throughout their daily routines to gather over 7GB of phone accelerometer data. Similarly, an initial deployment of 20 USB accelerometers recorded 55GB of acceleration over a period of 4 months. However, due to the infrequent occurrence of large earthquakes, it could require many years of observation to obtain records from several dangerously large events. One approach to overcome this limitation is to simulate sensor observations from existing seismic records, and use these simulated observations for testing. The Southern California Seismic Network, a network of several hundred high-fidelity seismometers, operating since the 1920s, provides a database of such records. We extract a set of 32 records of moderately large (M5-5.5) events from stations at distances of under 40km from the event epicenter. Simulated sensor observations are produced by subsampling these records to 50 samples per second and superimposing them onto segments of Android or Phidget data. As we will see in our shaketable experiments, this method of obtaining simulated sensor data yields a reasonable estimate of detection performance when we reproduce quake records using a shaketable and directly sense the acceleration with both Androids and Phidgets.

**Picking Algorithm Evaluation.** In our first experiment, we evaluate the sensor-level effectiveness of our density-based anomaly detector. We compare four approaches: two baselines and two versions of our algorithm.

1. A hypothesis-testing based approach (as used by classical decentralized detection), which uses a GMM-based density estimate not just for $\mathbb{P}[X_{s,t} \mid E_t = 0]$, but also for $\mathbb{P}[X_{s,t} \mid E_t = 1]$. For training data, we use 80 historic earthquake examples of magnitude 4.5-5, superimposed on the sensor data.
2. A domain specific baseline algorithm, *STA/LTA*, which exploits the fact that the energy in earthquakes is broadband in 0-10Hz. It compares the energy in those frequencies in the last 2.5s to the energy at those frequencies in the previous 5s; a sharp rise in this ratio is interpreted as a quake.
3. A simplified GMM based approach, which uses features from a sliding window of 2.5 s length
4. Our full GMM approach, which combines combines features of the last 2.5s with features from the previous 5s (to better detect the onset of transient events).

Notice that implementing the hypothesis testing baseline in an actual system would require waiting until the sensors experienced such a number of earthquakes, carefully annotating the data, and then training a density estimator. On the other hand, our anomaly detection approach can be used as soon as the sensors have gathered enough data for an estimate of $\mathbb{P}[X_{s,t} \mid E_t = 0]$. We then applied each of these four algorithms to test data based on historic earthquake recordings of magnitude 5-5.5 superimposed on held-out phone / Phidget data (i.e., data that was not used for training). The resulting estimated sensor ROC curves are shown in Fig. 7(a) and Fig. 7(b), respectively.

First note that in general the performance for the Phidgets is much better than for the phones. This is expected, as phones are subject to much more background noise, and the quality of the accelerometers in the Phidgets is better than those in the phones. For example, while the STA/LTA baseline provides good performance for the Phidgets (achieving up to 90% detection performance with minimal false positives), it performs extremely poorly for the phone client (where it barely outperforms random guessing). The other techniques achieve close to 100% true positive rate even for very small false positive rates. For the phone data, both our anomaly detection approaches outperform the hypothesis testing baseline, even though they use less training data (no data about historic earthquakes). In




(a) *Android detection performance* (b) *Phidget detection performance* (c) *Detection rates*

(d) *1 pick per minute, LtSt features* (e) *1 pick per minute, alternate view* (f) *Detection rate on Baja M7.2 (100 iterations averaged)*

**Figure 7: (a,b) Sensor level ROC curves for magnitude 5-5.5 events, for Android (a) and Phidget (b) sensors. (c) Detection rate as a function of the number of sensors in a 20 km $\times$ 20 km cell. We show the achievable performance guaranteeing one false positive per year, while varying the number of cells covered. (d,e) Detection performance for one cell, depending on the number of phones and Phidgets. (f) Actual detection performance for the Baja event. Note that our approach outperforms classical hypothesis testing, and closely matches the predicted performance.**

particular for low false positive rates (less than 5%), the full GMM LtSt model outperforms the simpler model (that only considers 2.5s sliding windows). Overall, we find that both for the phones and the Phidgets, we can achieve detection performance far better than random guessing, even for very small false positive rates, and even for lower magnitude (M5-5.5) events. We expect even better detection performance for stronger events.

**Sensor Fusion.** Based on the estimated sensor-level ROC curves, we can now estimate the fusion-level detection performance. To avoid overestimating the detection performance, we reduce the estimated true positive rates, assuming that a certain fraction of the time (10% in our case) the sensors produce pure random noise. We now need to specify communication constraints $\bar{p}$ on how frequently messages can be sent from the sensors, as well as a bound $\bar{P}$ on the fusion-level false positive rate. We choose $\bar{p}$ to be at most one message per minute, and $\bar{P}$ to be at most one fusion-level false positive per year.

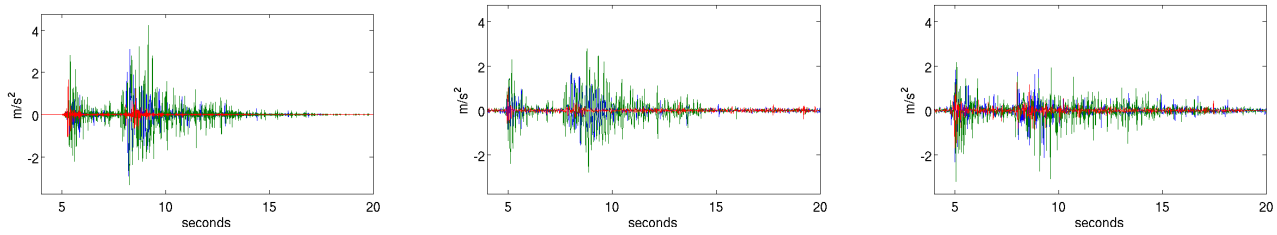
We consider sensors located in a geospatial areas of size 20 km $\times$ 20 km, called *cells*. The choice of this area is such that, due to the speed of seismic waves ($\approx 5 - 10km/s$), most sensors within one cell would likely detect the earthquake when computing features based on a sliding window of length 2.5s. However, in order to achieve larger spatial coverage we will need many spatial cells of 20 km $\times$ 20 km. For example, roughly 200 such cells would be needed to cover the Greater Los Angeles area. Increasing the number of cells additively increases the number of false positives due to the fact that multiple hypotheses (one per cell) are tested simultaneously. Consequently, to maintain our objective of one system-wide false positive per year, we must decrease the rate of annual false positives per cell. The effect on detection rates from this compensation as a function of the total number of cells is shown in Figure 7(c). Notice that even for 200 cells, approximately 60 phones per cell suffice to achieve close to 100% detection performance, as long as they are located close to the epicenter.

**Sensor Type Tradeoffs.** A natural question is what is the tradeoff between the different sensor types? Figures 7(d) and 7(e) shows the estimated detection performance as a function of the number of Phidgets and number of phones in the area, when constrained to one false alarm per year. Our results indicate that approximately 50 phones or 10 Phidgets should be enough to detect a magnitude 5 and above event with close to 100% success.

The results in Figures 7(d) and 7(e) also allow us to estimate how we could ensure sufficient detection performance if a given area contains only a limited number of active phone clients. For example, if only 25 phones are active in a cell, we could manually deploy 5 additional Phidgets to boost the detection performance from close to 70% to almost 100%.

Notice that all these results assume that the sensors are located close to the epicenter (as they assume the sensors experience maximum acceleration), and are thus to be taken with some care. Covering an area such as Greater Los Angeles likely requires tens of thousands of sensors.

**Shaketable Validation.** Our previous experiments have used synthetically produced data (recorded seismic events superimposed on phone recordings) to simulate how different detection algorithms may respond to a moderately large earthquake. Is such a simulation-based approach valid? Would these sensors actually detect an earthquake from their own recordings? To answer these questions, we




(a) *Episensor Recording* (b) *Android on Table* (c) *Android in Backpack*

**Figure 8: Shake table comparison of 24-bit EpiSensor, Android, and Android in a backpack. Notice that the phone recordings closely match those of the affixed high-fidelity EpiSensor.**

take recordings of three large historical earthquakes, and play them back on a *shaketable* (see Figure 4(c) for an illustration).

First, we test the ability of Android phones to accurately capture seismic events, relative to one of the sensors used in the SCSN. We reproduce records of three large M6-8 earthquakes on the shaketable, and record the motion using one Android placed on the table, and another in a backpack on the table. Ground truth acceleration is provided by a 24-bit EpiSensor accelerometer mounted to the table. A sample record from each sensor is shown in Figure 8. Apart from their lower resolution, the phones are not affixed to the table as is the EpiSensor but rather free to slide, and the backpack introduces an unpredictable source of error. Despite these significant challenges, after properly resampling both signals and aligning them temporally, we obtain an average correlation coefficient of 0.745, with a standard deviation of 0.0168. This result suggests that the phones reproduce the waveforms rather faithfully.

A more important question than the faithful reproduction of the waveform is whether the sensors are able to detect an earthquake played back on the shaketable. To assess this, we use the model trained on background noise data, as described above. We further use percentile estimation to choose the operating point which we experimentally determined to lead to high system-level detection performance above. All six of the recordings (three from the phone on the table and three from the phone in the backpack) were successfully detected.

**The Previous Big One.** To perform an end-to-end test of the entire system, we performed an experiment with the goal to find out whether our CSN would have been able to detect the last big event. A recent major earthquake in Southern California occurred on April 4, 2010. This M7.2 quake in Baja, California was recorded by SCSN, although the nearest station was more than 60km from the event epicenter. Using 8 recordings of this event, at distances of 63km to 162km, we produce simulated Android data and evaluate how many phones would have been needed to detect this event. Specifically, we constrain the system as before to one false alarm per year, and one message per minute in order to determine detection thresholds, sensor operating points and sensor thresholds for both the GMM anomaly and hypothesis testing detector, for each deployment size. We then simulate observations for each sensor in a deployment ranging from 1 sensor to 100 sensors. The models and thresholds are then applied to these observations to produce picks; the fusion center hypothesis test is then performed and the decision is made whether an event has occurred or not. The average detection rates for each deployment size (averaged over 100 iterations, using different Android data to simulate each observation) are shown in Figure 7(f) along with the estimated detection rates for the GMM-based anomaly detection. The latter estimate is based on the ROC that we estimated using a different collection of seismic events, as explained in our Picking Algorithm Evaluation section. Notice that the actual detection performance matches well the predicted detection performance. As baseline, we compare against the hypothesis testing based baseline (trained on 80 smaller-magnitude earthquakes). Anomaly detection significantly outperforms hypothesis testing, and suggests that a deployment of 60 phones in a cell 60 km of the epicenter would have been quite likely to detect the Baja, California M7.2 event.

# 6. RELATED WORK

In the following, we review prior work that is related to various aspects of this paper and that has not been discussed yet.

**Distributed and Decentralized Detection.** There has been a large amount of work in decentralized detection. The classical hierarchical hypothesis testing approach has been analyzed by Tsitsiklis [28]. Chamberland et al. [2] study classical hierarchical hypothesis testing under bandwidth constraints. Their goal is to minimizes the probability of error, under constraint on total network bandwidth (similar to our constraint $\bar{p}$ on the number of messages sent per sensor. Both these approaches require models for $\mathbb{P}[X_{s,t} \mid E_t = 1]$, which are not available in our case. Wittenburg et al. [32] study distributed event detection in WSN. In contrast to the work above, their approach is distributed rather than decentralized: nearby nodes collaborate by exchanging feature vectors with neighbors before making decision. Their approach requires a training phase, providing examples of events that should be detected. Martinic et al. [16] also study distributed detection on multi-hop networks. Nodes are clustered into cells, and the observations within a cell are compared against a user-supplied "event signature" (a general query on the cell's values) at the cell's leader node (cluster head). The communication requirements of the last two approaches are difficult to meet in community sensing applications, since sensors may not be able to communicate with their neighbors due to privacy and security restrictions. Both approaches require prior models (training data providing examples of events that should be detected, or appropriately formed queries) that may not be available in the seismic monitoring domain.

**Anomaly Detection.** There has also been significant amount of prior work on anomaly detection. Yamanishi et al. [33] develop the SmartSifter approach that uses Gaussian or kernel mixture models to efficiently learn anomaly detection models in an online manner. While results apply only in the centralized setting, they support that our approach of using GMMs for anomaly detection could be extended to learn, for each phone, a GMM that adapts to non-stationary sources of data. Davy et al. [7] develop an online approach for anomaly detection using online Support Vector machines. One of their experiments is to detect anomalies in accelerometer recordings of industrial equipment. They use produce frequency-based (spectrogram) features, similar to the features we use. However, their approach assumes the centralized setting.

Subramaniam et al. [26] develop an approach for online outlier detection in hierarchical sensor network topologies. Sensors learn models of their observations in an online way using kernel density estimators, and these models are folded together up the hierarchy to characterize the distribution of all sensors in the network. Rajasegarar et al. [24, 23] study distributed anomaly detection using one-class SVMs in wireless sensor networks. They assume a tree topology. Each sensor clusters its (recent) data, and reports the cluster descriptions to its parent. Clusters are merged, and propagated towards the root. The root then decides if the aggregate clusters are anomalous. Both approaches above are not suitable for the community sensing communication model, where each sensor has to make independent decisions. Zhang et al. [34, 35] introduce intrusion detection and demonstrate online SVMs to detect anomalies in process system calls. Onat et al. [21] develop a system for detecting anomalies based on sliding window statistics in mobile ad hoc networks (MANETs). However, their approach requires for nodes to share observations with their neighbors.

**Seismic Networks.** Perhaps the most closely related system is the QuakeCatcher network [4]. While QuakeCatcher shares the use of cheap MEMS accelerometers in USB devices and laptops, our system is different in its use of algorithms designed to execute efficiently on cloud computing systems and statistical algorithms for detecting rare events, particularly with heterogeneous sensors including mobile phones (which create far more complex statistical challenges). Kapoor et al. [14] use cell phones to detect earthquakes; however, their approach is based on analyzing the increase in call volume after or during an event, and thus cannot be used for earthquake early warning. Another related effort is the NetQuakes project [29], which deploys expensive stand-alone seismographs with the help of community participation. Our CSN Phidget sensors achieve different tradeoffs between cost and accuracy. Several Earthquake Early Warning (EEW) systems have been developed to process data from existing sparse networks of high-fidelity seismic sensors (such as the SCSN). The Virtual Seismologist [6] applies a Bayesian approach to EEW by using prior information and seismic models to estimate the magnitude and location of an earthquake as sources of information arrive. ElarmS [1] collects measurements of the frequency content of the initial P-wave measurements from sensors closest to the epicenter, and applies an attenuation function to estimate ground acceleration at further locations. We view our approach of community seismic networking as fully complementary to these efforts by providing a higher density of sensors and greater chance of measurements near to the epicenter. Our experiments provide encouraging results on the performance improvements that can be obtained by adding community sensors to an existing deployment of sparse but high quality sensors.

**Community and Participatory Sensing.** Community sensing has been used effectively in a variety of problem domains. Several researchers [13, 27, 17, 12, 15] have used mobile phones to monitor traffic and road conditions. Community sensors offer great potential in environmental monitoring [18, 30] by obtaining up-to-date measurements of the conditions participants are exposed to. Mobile phones and body sensors are used to encourage physical activity by categorizing body motion and comparing activities to exercise goals [5]. Like our CSN, these applications stand to benefit from the high density of existing community sensors, but are fundamentally different in their aim of monitoring phenomena rather than detecting infrequent events.

# 7. CONCLUSIONS

We studied the problem of detecting rare, disruptive events using community-held sensors. Our approach learns local statistical models characterizing normal data (e.g., acceleration due to normal manipulation of a cellphone), in an online manner. Using local online percentile estimation, it can choose operating points that guarantee bounds on the sensor-level false positive frequency, as well as the number of messages sent per sensor. We then showed how a conservative estimate of the sensors' ROC curves can be used to make detection decisions at a fusion center which guarantee bounds on the false positive rates for the entire system, as well as maximize a lower bound on the detection performance. The pessimistic predicted true positive rates allow us to assess whether a given density of sensors is sufficient for the intended detection task. This online decentralized anomaly detection approach allows us to cope with the fundamental challenge that rare events are very difficult or impossible to model and characterize a priori. It also allows the use of heterogeneous, community-operated sensors that may differ widely in quality and communication constraints.

We then presented an implementation of our approach in the Community Seismic Network (CSN), a novel community sensing project with the goal of rapidly detecting earthquakes using cell phone accelerometers and consumer USB devices. We presented empirical evidence suggesting how cloud-computing is an appropriate platform for real-time detection of rare, disruptive events, as it naturally copes with peaked load, and is designed for redundancy and replication. We furthermore experimentally assessed the sensitivity of our sensors, estimating and evaluating ROC curves using experiments involving data obtained through the playback of historical earthquakes on shaketables. These assessments provide evidence of the likely detection performance of the CSN as a function of the sensor density. For example, we found that approximately 100 Android clients, or 20 Phidgets per 20 km $\times$ 20 km area may be sufficient to achieve close to 100% detection probability for events of magnitude 5 and above, while bounding the false positve rate by 1 per year. While these results are very promising, they have to be taken with some care. In particular, the results are based on the assumption that the phones are located very close to the epicenter of the quake (so they experience maximum acceleration). To enable coverage of the Greater Los Angeles area, this would require a uniformly high density of sensors (tens of thousands of sensors) across the entire domain. We will defer the detailed study of spatial effects, and numbers of sensors needed to achieve spatial coverage, to future work.

We believe that our results provide an important step towards harnessing community-held sensors to provide strong benefits for our society.

# 8. REFERENCES


[1] R. Allen, H. Brown, M. Hellweg, O. Khainovski, P. Lombard, and D. Neuhauser. Real-time earthquake detection and hazard assessment by ElarmS across California. *Geophysical Research Letters*, 36(5), 2009.

[2] J. Chamberland and V. Veeravalli. Decentralized detection in sensor networks. *Signal Processing, IEEE Transactions on*, 51(2):407–416, 2003.

[3] cnet news. Android market share to surge over next four years. http://news.cnet.com/8301-1035_3-20015799-94.html, September 2010.

[4] E. Cochran, J. Lawrence, C. Christensen, and R. Jakka. The Quake-Catcher Network: Citizen Science Expanding Seismic Horizons. *Seismological Research Letters*, 80(1):26, 2009.

[5] S. Consolvo, D. McDonald, T. Toscos, M. Chen, J. Froehlich,

B. Harrison, P. Klasnja, A. LaMarca, L. LeGrand, R. Libby, et al. Activity sensing in the wild: a field trial of ubifit garden. In *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1797–1806. ACM, 2008.

[6] G. Cua, M. Fischer, T. Heaton, and S. Wiemer. Real-time performance of the Virtual Seismologist earthquake early warning algorithm in southern California. *Seismological Research Letters*, 80(5):740, 2009.

[7] M. Davy, F. Desobry, A. Gretton, and C. Doncarli. An online support vector machine for abnormal events detection. *Signal processing*, 86(8):2009–2025, 2006.

[8] P. Fearnhead. Particle filters for mixture models with an unknown number of components. *Statistics and Computing*, 14(1):11–21, 2004.

[9] E. Giné-Masdéu and A. Guillou. Rates of strong uniform consistency for multivariate kernel density estimators. *Ann Inst. H. Poincaré*, 38:907–921, 2002.

[10] R. Gomes, M. Welling, and P. Perona. Incremental learning of nonparametric Bayesian mixture models. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[11] M. Greenwald and S. Khanna. Space-efficient online computation of quantile summaries. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, pages 58–66. ACM, 2001.

[12] R. Herring, A. Hofleitner, S. Amin, T. Nasr, A. Khalek, P. Abbeel, and A. Bayen. Using mobile phones to forecast arterial traffic through statistical learning. *Submitted to Transportation Research Board*, 2009.

[13] B. Hoh, M. Gruteser, R. Herring, J. Ban, D. Work, J. Herrera, A. Bayen, M. Annavaram, and Q. Jacobson. Virtual trip lines for distributed privacy-preserving traffic monitoring. In *Proc. of the International conference on Mobile systems, applications, and services*, pages 17–20. Citeseer, 2008.

[14] A. Kapoor, N. Eagle, and E. Horvitz. People, Quakes, and Communications: Inferences from Call Dynamics about a Seismic Event and its Influences on a Population. In *Proceedings of AAAI Symposium on Artificial Intelligence for Development*, 2010.

[15] A. Krause, E. Horvitz, A. Kansal, and F. Zhao. Toward community sensing. In *Proceedings of the 7th international conference on Information processing in sensor networks*, pages 481–492. IEEE Computer Society, 2008.

[16] F. Martincic and L. Schwiebert. Distributed event detection in sensor networks. In *Systems and Networks Communications, 2006. ICSNC'06. International Conference on*, page 43. IEEE, 2006.

[17] P. Mohan, V. Padmanabhan, and R. Ramjee. Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 323–336. ACM, 2008.

[18] M. Mun, S. Reddy, K. Shilton, N. Yau, J. Burke, D. Estrin, M. Hansen, E. Howard, R. West, and P. Boda. Peir, the personal environmental impact report, as a platform for participatory sensing systems research. In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, pages 55–68. ACM, 2009.

[19] J. I. Munro and M. S. Paterson. Selection and sorting with limited storage. *Theoretical Computer Science*, 12(3):315–323, 1980.

[20] X. Nguyen, M. Wainwright, and M. Jordan. Nonparametric decentralized detection using kernel methods. *Signal Processing, IEEE Transactions on*, 53(11):4053–4066, 2005.

[21] I. Onat and A. Miri. An intrusion detection system for wireless sensor networks. In *Wireless And Mobile Computing, Networking And Communications, 2005.(WiMob'2005), IEEE International Conference on*, volume 3, pages 253–259. IEEE, 2005.

[22] H. V. Poor and O. Hadjiliadis. *Quickest Detection.* Cambridge University Press., 2009.

[23] S. Rajasegarar, C. Leckie, J. Bezdek, and M. Palaniswami. Centered hyperspherical and hyperellipsoidal one-class support vector machines for anomaly detection in sensor networks. *Information Forensics and Security, IEEE Transactions on*, 5(3):518–533, 2010.

[24] S. Rajasegarar, C. Leckie, M. Palaniswami, and J. Bezdek. Distributed anomaly detection in wireless sensor networks. In *Communication systems, 2006. ICCS 2006. 10th IEEE Singapore International Conference on*, pages 1–5. IEEE, 2007.

[25] M. Sato. Online model selection based on the variational Bayes. *Neural Computation*, 13(7):1649–1681, 2001.

[26] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos. Online outlier detection in sensor data using non-parametric models. In *Proceedings of the 32nd international conference on Very large data bases*, pages 187–198. VLDB Endowment, 2006.

[27] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson. VTrack: accurate, energy-aware road traffic delay estimation using mobile phones. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 85–98. ACM, 2009.

[28] J. Tsitsiklis. Decentralized detection by a large number of sensors. *Mathematics of Control, Signals, and Systems (MCSS)*, 1(2):167–182, 1988.

[29] USGS. Netquakes. http://earthquake.usgs.gov/earthquakes/waveforms/netq/, 2010.

[30] P. Völgyesi, A. Nádas, X. Koutsoukos, and Á. Lédeczi. Air quality monitoring with sensormap. In *Proceedings of the 7th international conference on Information processing in sensor networks*, pages 529–530. IEEE Computer Society, 2008.

[31] M. K. Warmuth and D. Kuzmin. Randomized pca algorithms with regret bounds that are logarithmic in the dimension. *Journal of Machine Learning Research*, 9:2287–2320, 2008.

[32] G. Wittenburg, N. Dziengel, C. Wartenburger, and J. Schiller. A system for distributed event detection in wireless sensor networks. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 94–104. ACM, 2010.

[33] K. Yamanishi, J. Takeuchi, G. Williams, and P. Milne. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. *Data Mining and Knowledge Discovery*, 8(3):275–300, 2004.

[34] Y. Zhang and W. Lee. Intrusion detection in wireless ad-hoc networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 275–283. ACM, 2000.

[35] Y. Zhang, W. Lee, and Y. Huang. Intrusion detection techniques for mobile wireless networks. *Wireless Networks*, 9(5):545–556, 2003.