CrowdQuake: A Networked System of Low-Cost Sensors for Earthquake Detection via Deep Learning

Xin Huang[1]*, Jangsoo Lee[2]*, Young-Woo Kwon[2], Chul-Ho Lee[1]
[1]Florida Institute of Technology, [2]Kyungpook National University
xhuang2016@my.fit.edu,dellhart@knu.ac.kr,ywkwon@knu.ac.kr,clee@fit.edu

ABSTRACT

Recently, low-cost acceleration sensors have been widely used to detect earthquakes due to the significant development of MEMS technologies. It, however, still requires a high-density network to fully harness the low-cost sensors, especially for real-time earthquake detection. The design of a high-performance and scalable networked system thus becomes essential to be able to process a large amount of sensor data from hundreds to thousands of the sensors. An efficient and accurate earthquake-detection algorithm is also necessary to distinguish earthquake waveforms from various kinds of non-earthquake ones within the huge data in real time. In this paper, we present CrowdQuake, a networked system based on low-cost acceleration sensors, which monitors ground motions and detects earthquakes, by developing a convolutional-recurrent neural network model. This model ensures high detection performance while maintaining false alarms at a negligible level. We also provide detailed case studies on two of a few small earthquakes that have been detected by CrowdQuake during its last one-year operation.

CCS CONCEPTS

• **Computer systems organization** → **Sensor networks**; • **Computing methodologies** → **Machine learning**;

ACM Reference Format:
Xin Huang[1]*, Jangsoo Lee[2]*, Young-Woo Kwon[2], Chul-Ho Lee[1]. 2020. CrowdQuake: A Networked System of Low-Cost Sensors for Earthquake Detection via Deep Learning. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3394486.3403378

1 INTRODUCTION

Low-cost micro-electro-mechanical systems (MEMS) acceleration sensors have been extensively used over the last few years for monitoring and detecting earthquakes, since they have great potential to complement or possibly substitute traditional expensive seismic networks whose coverage can hardly be dense due to high installation and operational costs. The examples include Quake-Catcher Network (QCN) [7], Community Seismic Network (CSN) [6, 10, 11], Home Seismometer [14], Monitoring of Earthquakes through MEMS Sensors (MEMS) [8] and Palert [24], each of which has employed the low-cost MEMS acceleration sensors to detect earthquakes.

MyShake [16, 18, 19] is the most recent effort in developing a MEMS-sensor-based earthquake detection system, utilizing participatory smartphones as seismic sensors. Unlike the above prior works, it leverages a machine learning algorithm – an artificial neural network (ANN) model, as its core detection algorithm. While each phone records acceleration data with its built-in accelerometer, the ANN model running on the phone determines whether any movement captured based on the acceleration data is caused by an earthquake. A server-side operation then follows up to confirm the earthquake. Despite its great success, it still has limitations. The simple ANN model can hardly be an ideal choice from a perspective of detection accuracy, although it can be a good compromise for the tradeoff between battery and performance. The heterogeneity across different participatory phones can also increase the chance of false alarms.

On the other hand, the recent advances in deep learning have been extensively explored and exploited in seismology for a wide range of applications, which include searching for repeating seismic signals, identifying previously unidentified earthquakes and phase picking, yet from the plethora of *archived* continuous seismic records in databases. For example, Perol et al. [21] proposed a convolutional neural network (CNN) to identify and locate earthquakes, including missing ones, from the existing earthquake waveforms recorded by two local stations in Oklahoma. Zhu and Beroza [25] applied a deep CNN to pick up the arrival time of archived seismic waves. See [17] and references therein for more details. While the deep learning techniques have been powerful tools for the applications, they have not been explored for "real-time" earthquake detection with low-cost MEMS acceleration sensors, which is one of our focuses in this work.

In this paper, we present CrowdQuake – a networked system of hundreds to thousands of low-cost acceleration sensors, empowered by deep learning, for real-time earthquake detection. We envision to use thousands of low-cost sensors, which are firmly installed and regularly power supplied, to continuously stream their acceleration records to a backend system for earthquake detection. We first design the system of CrowdQuake to be capable of handling streaming time-series data from such a large number of sensors and to enable running a deep learning algorithm with the large volume of data. The system is also designed to autonomously identify abnormal sensors that can degrade the detection performance. We propose a convolutional-recurrent neural network (CRNN) model as a core detection algorithm of CrowdQuake, and demonstrate

*Both authors contributed equally to this work.

KDD '20, August 23–27, 2020, Virtual Event, CA, USA
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403378

that it exhibits 99% detection performance with a false alarm rate less than 1% while being robust to the choice of hyperparameters. This CRNN model effectively eliminates the potential drawbacks of the state-of-the-art ANN model, which can lead to excessive false alarm rates for earthquake detection.

As its first phase, CrowdQuake has been deployed with three-hundred acceleration sensors over the southern part of South Korea for about a year. While CrowdQuake is currently being expanded to its second phase with thousands of sensors, it has shown its feasibility of using low-cost sensors to detect small magnitude earthquakes over the last one-year operation, including the ones that would have been considered too small to be detected by other existing systems like MyShake. We expect that CrowdQuake can be complementary and integrative to any existing seismic networks for better coverage and detection performance, and it can also become an effective system model for the areas where a network of seismic stations is not available.

2 CROWDQUAKE

In this section, we present CrowdQuake by providing a system overview and explaining its main components in detail.

2.1 System Overview

CrowdQuake is designed with the following requirements:

- to detect earthquakes in real time, based on acceleration records measured by hundreds to thousands of low-cost MEMS acceleration sensors that are dispersed over a large area.
- to store the past acceleration data in databases and allow post-hoc data analysis to identify the earthquakes that could have been missed by the real-time detection.
- to identify abnormal sensors periodically that can negatively affect the capability of earthquake detection.

We provide a system architecture of CrowdQuake in Figure 1, which consists of low-cost sensors, gateways, a data server that embodies databases and postprocessing engines, and a monitoring server. The sensors are available in the form of Samsung Galaxy S7 smartphones and custom-designed IoT sensors, as depicted in Figure 2.[1] As its first phase, CrowdQuake was initially deployed with three-hundred smartphone sensors over the southern part of South Korea, and it has been in operation for about a year to ascertain the feasibility of using low-cost sensors to detect earthquakes. CrowdQuake is now being expanded to cover the whole country with eight-thousand IoT sensors.[2] In this paper, we focus on the first-phase deployment of CrowdQuake and its operational results.

The sensors continuously record three-component acceleration waveforms and send them to the gateways. The gateways then process the acceleration data and leverage a *deep learning* model, which shall be explained in detail later (Section 4), to detect the presence of an earthquake in real time. In the event of an earthquake, the event and its corresponding raw data are sent to the data and monitoring servers. If otherwise, i.e., in the normal operation, the acceleration records are stored in the databases (within the data server) for long-term data storage and postprocessing, and also to locate misbehaving sensors if any. For the system implementation of CrowdQuake, we employ several well-maintained open-source projects such as Apache Kafka [1], InfluxDB [4], Elasticsearch [2], and Grafana [3].

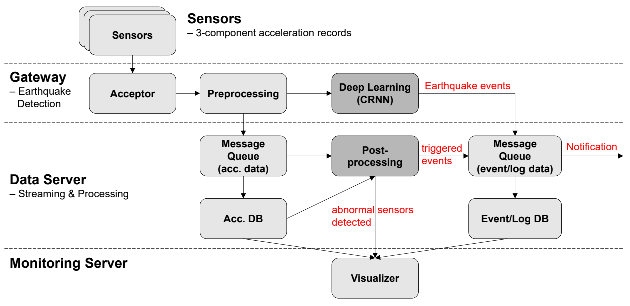


Figure 1: System architecture of CrowdQuake.

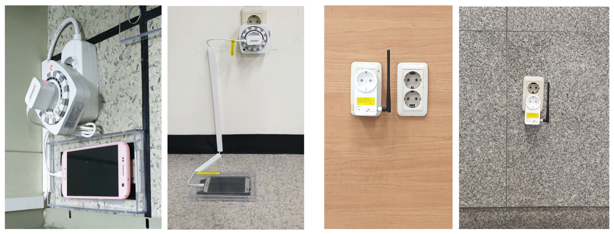


Figure 2: Snapshots of four actual sensors installed.

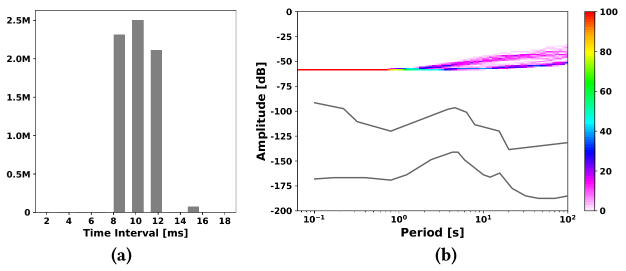


Figure 3: (a) Histogram of sampling intervals; (b) Noise floor of a sensor.

2.2 System Details

Acceleration Sensors. It is important to understand the quality of acceleration sensors for reliable operation of CrowdQuake. First, an important capability of each sensor is to maintain a constant measurement interval between two consecutive records over time, or to ensure a consistent sampling rate, i.e., the number of samples (data points) per second. We observe that our sensors exhibit reasonably accurate sampling performance at a sampling rate of 100 Hz, as shown in Figure 3(a). Note that the sampling rate of MyShake is 25 Hz [16, 18], which is deemed chosen for balancing a battery-performance tradeoff in participatory smartphone sensors. In our system environment, the sensors are regularly power supplied.[3]

[1]By low cost, we mean that the sensors used in CrowdQuake are cheap accelerometers, including the ones built in smartphones. Each of them costs less than \$10 US dollars.
[2]This implies that each sensor approximately covers an area of about 1 km^2 for most of the country.

[3]In fact, as a default mode, MyShake runs on a smartphone only when it is essentially stationary and also power supplied.

We next evaluate the sensitivity of our sensors against high- and low-earth background noise levels. The closer to the earth background noise levels, the better the sensor is. As can be seen from Figure 3(b), the noise floor of each sensor typically ranges from −58 dB to −60 dB. While it is still far from the earth background noise, we note that there has been an improvement in the sensitivity of (MEMS) acceleration sensors as compared to the ones for MyShake in 2016. It is shown in [16] that their noise floors are around −50 dB (See [16] for more details). Thus, we expect that our deep learning-driven CrowdQuake system equipped with improved sensors can go beyond the limitation set by MyShake that aims at detecting magnitude 5 or larger earthquakes at distances of 10 km or less. In other words, we expect that our system can detect smaller earthquakes, in addition to the ones with magnitude 5 and above, at the similar or even longer distances. We will show that this is indeed the case in Section 5.

Gateways. The gateways are responsible for collecting and processing the acceleration data from the sensors and detecting earthquakes via deep learning. In (the first phase of) CrowdQuake, to handle the data from 300 sensors simultaneously, we employ two gateways, each of which handles the data from 150 sensors. Each gateway assigns a 'worker' running on a separate CPU core to sensors in order to process their acceleration records and run our deep learning algorithm for earthquake detection, which will be explained in Section 4. Since the number of workers can be up to the number of CPU cores, we use two gateways to serve 150 sensors each. The gateway size thus needs to be increased properly, when the number of sensors increases.

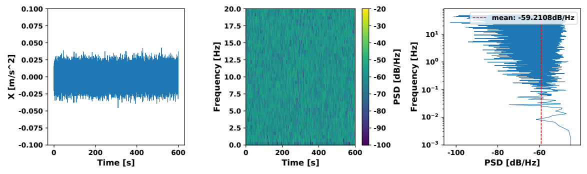
Each sensor first establishes a connection with the corresponding gateway, after which the gateway assigns an appropriate worker to the sensor. The sensor then sends acceleration data to the gateway every second, which is also transferred to the data server. In addition, the sensor periodically sends a 'heartbeat' message to inform the gateway of its status such as its remaining battery level and resource usages. A sensor can also be requested by a system administrator to send a heartbeat message to the gateway. Furthermore, each gateway generates the status logs of sensors under its management and the gateway itself, and send them to the data server.

Data Server. The data server streams the acceleration data and various events/logs to databases for long-term data storage and postprocessing, and also for visualization via the monitoring server. It has two message queues that are built using Apache Kafka, and two databases based on InfluxDB and Elasticsearch, respectively. The rationale behind this structural design is that there are various data types and a large volume of the acceleration data needs to be stored consistently. Note that the volume of data from each individual sensor still remains small.[4]

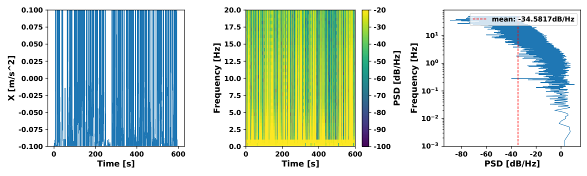
We first observe that the acceleration records are streaming time-series data and are of regular data type. It is mainly for long-term data preservation and also for on-demand postprocessing that is done infrequently. On the other hand, the events and logs are of irregular type and they are generated sporadically, with relatively smaller data volume. It is also mainly for visualization. Therefore, the former would need a database with faster data handling speed, while the latter would need a database that is suitable for data of irregular type and can be better integrated with the monitoring server for visualization. We have thus chosen InfluxDB for the former, which is a special database for time-series data, and Elasticsearch for the latter, which provides better indexing and searching functionalities.

We also note that the data needs to be retrieved from each message queue and stored into its corresponding database as fast as the data is pushed into the queue, thereby preventing from message overflows. Our micro-benchmark test indicates that each message queue based on Apache Kafka can handle up to about 200K messages per second, i.e., the acceleration records from 2K sensors per second, in its input and output, when the input and output data rates are the same. Furthermore, InfluxDB can store up to about 160K messages per second, i.e., the data from 1.6K sensors per second, while Elasticsearch can do up to about 70K messages per second, which is reasonable speed for events and logs. Therefore, the data server operates without any problem.

[4]Each sensor generates 100 three-component data points per second, which translates into 2K bytes per second, where the size of each three-component data point, along with its timestamp, is 20 bytes. By using bitwise operations, the data volume can reduce down to about 700 bytes per second.



(a) A normal sensor



(b) An anomalous sensor

Figure 4: Normal and anomalous sensor records.

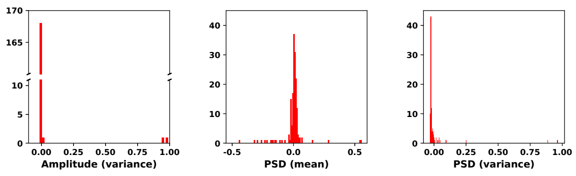


Figure 5: Histograms of 3 features in the x-axis component.

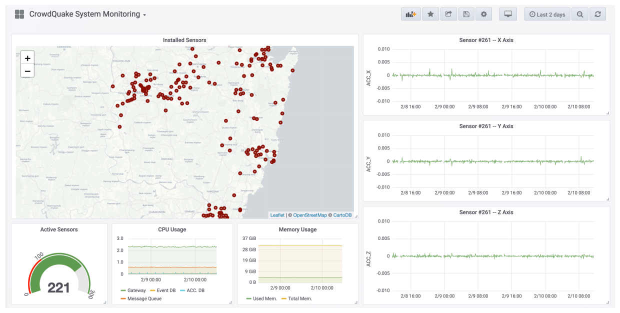


Figure 6: A screen snapshot of the monitoring server.

Anomaly Detection & Other Postprocessing. The data server also has an anomaly detection module to locate abnormal sensors, since sensors can possibly misbehave at any time for some reason and the abnormal sensors can negatively affect the capability of earthquake detection in CrowdQuake. As shown in Figure 4, the (time-domain) waveforms, spectrogram, and power spectral densities (PSDs) of the background noise signals of normal and abnormal sensors are generally quite different from each other. Note that the unit of acceleration data here in time domain is all m/s^2 and its DC offset (mean amplitude) is removed.[5] Overall, we observe that the amplitude range of the background noise of normal sensors is consistent, the spectrogram exhibits no dominant frequency component (or uniform power across the frequency band), and the average PSD is also consistent for all three components. However, the background noise measured on abnormal sensors deviates from this common behavior of normal sensors.

To identify such abnormal sensors, we compute (i) variance (mean squared deviation) of the amplitude of the noise waveform, and (ii) mean and (iii) variance of the PSD of the waveform for each component, thereby giving 9 features in total (for each sensor). Figure 5 shows the histograms of 3 normalized features. We then use the k-means clustering algorithm [5] for anomaly detection. (i) We first cluster the 300-sensor data of 9 normalized features into $k = 4$ clusters. (ii) We treat the biggest cluster that contains most of the data as a main cluster of normal sensors. (iii) We measure the Euclidean distance of the 9-feature data of each sensor to the centroid of the main cluster, and rank sensors in an increasing order of distances to the centroid. The longer the distance, the worse the sensor is. Thus, the bottom 10% to 20% sensors, whose size is tunable, are considered as a group of potential abnormal sensors and they are highlighted on a map, so that a system administrator can postprocess the acceleration data from the sensors and make a site visit for further examination, if necessary. CrowdQuake currently runs the anomaly detection module every 30 minutes.

The data server also allows us to perform post-hoc data analysis on the past acceleration records to identify undetected earthquakes, if any, and understand why they were missed, by providing the results of time-frequency domain analysis such as the waveform in time domain, spectrogram, and PSDs, along with their statistics. By the actual operation of CrowdQuake for about a year, we have noticed that most abnormal data are caused by man-made sources, e.g., regular inspections by employees or the operation of air conditioning for cooling the facilities. Thus, unexpected events and noises are always possible, thereby making the problem of detecting earthquakes correctly without false alarms *challenging*.

Monitoring Server. For easy maintenance and monitoring, we build a monitoring server using the Grafana platform [3], which enables a system administrator to monitor the current status of CrowdQuake. This provides various information visually, such as sensors' locations, the number of active sensors, system resource usages, acceleration records, abnormal sensors, and detected earthquakes. Figure 6 shows an exemplary screen snapshot of this monitoring server.

[5]The acceleration of gravity g is 9.80665 m/s^2 on Earth.

3 REVISITING THE STATE OF THE ART MODEL FOR REAL-TIME DETECTION

Before going into the details of our deep learning model used in CrowdQuake, we revisit the state-of-the-art machine learning model for real-time earthquake detection. It is a simple artificial neural network (ANN) model with three features and has been used as the core detection algorithm in the MyShake system [16, 18, 19]. In particular, we find out that the ANN model and its associated features have two main drawbacks, which lead to deficient detection/classification performance.

We first provide an overview of the ANN model with three associated features proposed in [16]. Each three-component acceleration record is divided into 2-second sliding windows with a 1-second step, i.e., every two consecutive windows have an overlap of 1 second. In other words, the basic data unit is a 2-second acceleration window (with an overlap of 1 second), which we hereafter call an "instance". It is worth noting that such a short duration of each instance is chosen for the purpose of 'real-time' earthquake detection. Then, for *each* instance, the following three features are extracted: (i) cumulative absolute velocity (CAV), (ii) interquartile range (IQR), and (iii) zero crossing count (ZC). As shown in Figure 7(a), the ANN model is a multi-layer perceptron (MLP) [13] that has a hidden layer of 5 neurons, and it takes the three features as an input for a binary classification, where earthquake and non-earthquake data instances are labeled as 1 and 0, respectively. The final output of the model, say $\hat{y}$, indicates the probability that a given input instance results from an earthquake. With a choice of the value of a probability threshold, say p, if $\hat{y} \geq p$, then the corresponding input instance is classified as an earthquake, otherwise a non-earthquake. See Appendix A for more details.

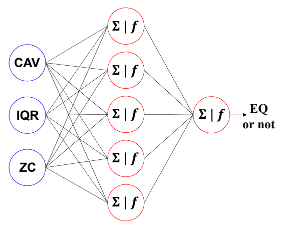
While the ANN model has been shown effective in [16, 18], we identify the following limitations of the ANN model through extensive experiments for various test cases.

1. The three features show *unsatisfactory* discriminative power.
2. The ANN model is *sensitive* to (i) the sampling rate of acceleration data, (ii) how earthquake 'instances' are extracted from the (publicly available) earthquake acceleration waveforms recorded on seismometers, and (iii) the value of the threshold p.

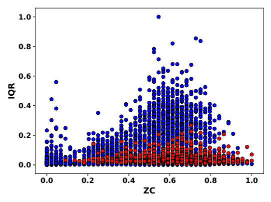
The overall performance of the ANN model also shows *a high false alarm rate* to detect earthquakes.

Model Performance. We train and evaluate the ANN model for various test cases, including the setting used in [16, 18], to investigate how the sampling rate of acceleration data affects the performance of the ANN model and also how the way that earthquake instances (used for mode training and testing) are extracted from the existing earthquake data influences the model performance.

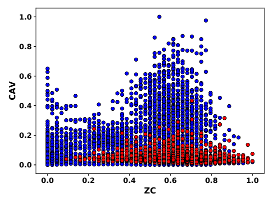
To this end, for earthquake data, we consider the K-NET records of Japan earthquakes available in National Research Institute for Earth Science and Disaster Resilience (NIED) [20]. Specifically, we use a set of earthquake records, each of whose x-axis component peak ground acceleration (PGA) is greater than 0.1 gravity (g). For non-earthquake data, we use a wide range of acceleration records that we have collected using the low-cost sensors. They are all recorded at a sampling rate of 100 data points per second. It is worth noting that there is a data imbalance problem for real-time earthquake detection, since non-earthquake data instances are far



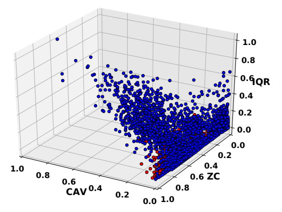
(a) The ANN model



(b) ZC vs. IQR

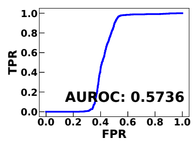


(c) ZC vs. CAV

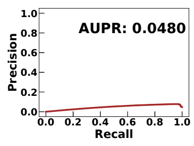


(d) ZC vs. CAV vs. IQR

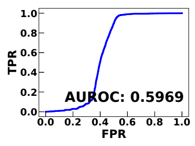
Figure 7: (a) The structure of the ANN model. (b)–(d) Comparing the three features from earthquake and non-earthquake instances for the case with the sampling rate of 25 Hz and the selected duration of 2 seconds. Red and blue dots indicate earthquake instances and non-earthquake instances (centroids), respectively.



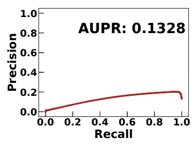
(a) ROC/25Hz/2s



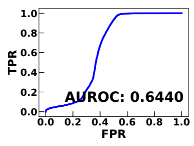
(b) PR/25Hz/2s



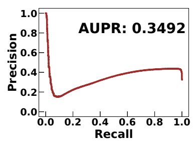
(c) ROC/25Hz/4s



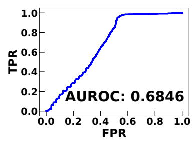
(d) PR/25Hz/4s



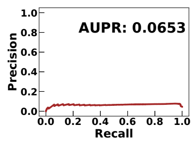
(e) ROC/25Hz/10s



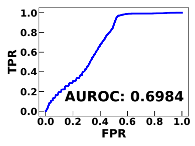
(f) PR/25Hz/10s



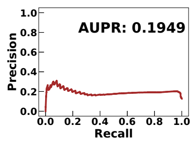
(g) ROC/50Hz/2s



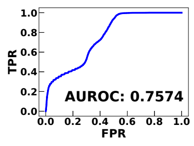
(h) PR/50Hz/2s



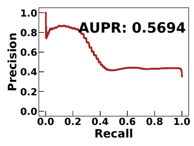
(i) ROC/50Hz/4s



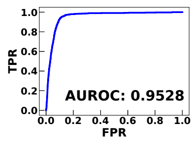
(j) PR/50Hz/4s



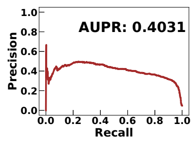
(k) ROC/50Hz/10s



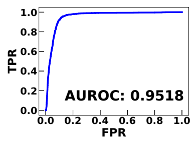
(l) PR/50Hz/10s



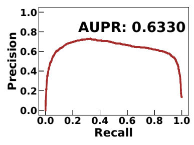
(m) ROC/100Hz/2s



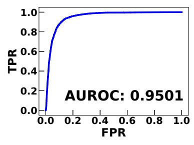
(n) PR/100Hz/2s



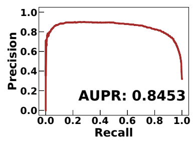
(o) ROC/100Hz/4s



(p) PR/100Hz/4s



(q) ROC/100Hz/10s



(r) PR/100Hz/10s

Figure 8: The ROC and PR curves of the ANN model.

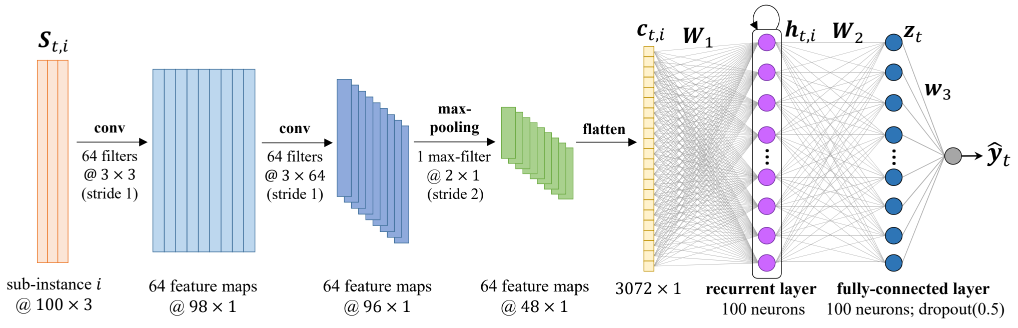
more than earthquake data instances. For the ANN model, the *k*-means clustering method is used to handle this problem [16].

We then consider three different 'sampling rates' such as 25 Hz (down-sampled), 50 Hz (down-sampled), and 100 Hz for both earthquake and non-earthquake data to obtain their corresponding instances, respectively. For each (existing) earthquake waveform, we also consider its three different ranges around the peak – three different 'selected durations' of the strongest portion of each waveform, to extract the corresponding earthquake instances, which are 2-second, 4-second, and 10-second ranges of the waveform around the peak, respectively. See Appendix B for more details.

We first observe that the discriminative power of the three features used in the ANN model turns out to be poor, as can be seen from Figure 7(b)–(d). That is, each subfigure indicates that it is hard to find a separating hyperplane of the data and the presence of such a hyperplane is even unclear. We further observe that the performance of the corresponding ANN model also becomes quite limited. In Figure 8, we present the receiver operating characteristic (ROC) and precision-recall (PR) curves of the ANN model for different test cases of sampling rates and selected durations.[6] We also provide the values of the area under ROC (AUROC) and the area under PR (AUPR) in each figure. Here, TPR stands for true positive rate, which is the recall. FPR stands for false positive rate, which is also called fall alarm rate (FAR). For the ROC curve, the closer to the upper *left* corner (a perfect classifier), the better the model is. As to the PR curve, if it is closer to the upper *right* corner, the model is better. In other words, the greater AUROC/AUPR, the better the model is. Also, for a given threshold p, the higher recall, higher precision and lower FAR, the better the model is. See Appendix B and Appendix C for more details on the performance metrics and detailed performance results with threshold $p = 0.5$, respectively.

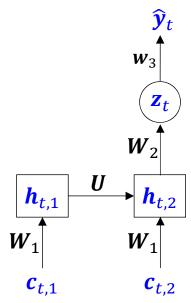
Figure 8 shows that the ANN model under all test cases achieves high TPR/recall at the cost of low precision and high FPR/FAR. It further implies that the ANN model is sensitive to the sampling rate, the selected duration of the strongest portion of each earthquake waveform, and also the choice of p. We notice that the performance of the ANN model can be improved when increasing the sampling rate to 100 Hz and the selected duration to 10 seconds, which is somewhat opposite to the setting in [16, 18] where they use the sampling rate of 25 Hz and consider a short (yet unknown) duration of the strongest portion. Nonetheless, even with the best case, the FPR/FAR is still above 10% to 20% to ensure high recall/TPR, which means that the system would lead to a large number of false alarms while detecting earthquakes in real time. We further conduct the

[6]The ROC and PR curves capture a tradeoff between TPR and FPR (recall vs. FAR) and the one between precision and recall, respectively, with varying values of p [12]. It is worth noting that when dealing with the data/class imbalance problem as is the case here, the PR curve is generally preferable to the ROC curve [9].



$S_{t,i}$
conv
64 filters @ 3 × 3 (stride 1)
conv
64 filters @ 3 × 64 (stride 1)
max-pooling
1 max-filter @ 2 × 1 (stride 2)
flatten
$c_{t,i}$
W_1
$h_{t,i}$
W_2
z_t
w_3
$\hat{y}_t$
sub-instance i @ 100 × 3
64 feature maps @ 98 × 1
64 feature maps @ 96 × 1
64 feature maps @ 48 × 1
3072 × 1
recurrent layer 100 neurons
fully-connected layer 100 neurons; dropout(0.5)

(a) The overall architecture of our CRNN model



(b) Unfolded RNN structure

Figure 9: The architecture of our CRNN model. The data size here at each layer is based on a sampling rate of 100 Hz.

10-fold cross-validation and confirm the results, but here omit the details due to space constraint.

4 THE CRNN MODEL

Our observations thus far call for a novel learning model that would achieve high earthquake detection accuracy while maintaining low FAR. We thus propose *a convolutional-recurrent neural network (CRNN) model* that is a combination of convolutional neural network (CNN) and recurrent neural network (RNN) for earthquake detection as a binary classification. Here earthquake and non-earthquake instances are again labeled as 1 and 0, respectively. This model stems from an observation that time-series acceleration data at each component can be treated as one-dimensional (1D) images to be fed into the CNN (and also into the CRNN). That is, each 3-component acceleration record becomes a 1D image with 3 channels (like RGB colors), or a 2D image with 1 channel, in the context of CNN.

Specifically, we use the same basic data unit as the one for the ANN model, which is the 2-second sliding window (an instance). Our CRNN model takes each instance *directly* as an input, which is in contrast to the ANN model that takes the three features extracted from each instance as an input. Thus, assuming the sampling rate of 100 Hz (i.e., 100 data points per second), each input instance becomes a 200 × 1 image with 3 channels, or a 200 × 3 image with 1 channel. We here use the latter representation for brevity.

Model Structure. Our proposed CRNN model is depicted in Figure 9. It is the optimized one, starting from a simple CRNN model, by gradually modifying its structure with adding more layers and tuning hyperparameters. Our model benefits from both CNN and RNN. It no longer needs the handcrafted features like the ones for the ANN model, which have been shown to have *low* predictive power. Instead, the CNN component of our model automates feature engineering to extract informative features after filtering out noise via convolution filters. In addition, its RNN component introduces the notion of 'memory' to the network, capturing time dependency across the input instances.

Let $S_t \in \mathbb{R}^{n\times 3}$ be the t-th input instance, where n is the number of data points over a window of 2 seconds. We then divide S_t into two sub-instances of size $\frac{n}{2}\times 3$ (1-second window data each). Letting $S_{t,1}$ and $S_{t,2}$ be the corresponding sub-instances, they are sequentially fed into the model. We here take the case of 100-Hz sampling rate as an example, so $n = 200$.

- The first layer is a 1D convolutional layer that convolves each of 64 filters of size 3 × 3 across the input sub-instance $S_{t,i} \in \mathbb{R}^{100\times 3}$ ($i = 1, 2$), with a stride of 1 and without padding.[7] It then applies the rectified linear unit (ReLU) activation function, i.e., $\text{ReLU}(x) = \max\{0, x\}$, to the convolution outputs individually, leading to 64 feature maps, each of which is a column vector of length 98. That is, the output of this layer is a 98 × 64 matrix.
- The second one is another 1D convolutional layer having another 64 filters of size 3×64, which are individually applied to the output of the first layer, with a stride of 1 and without padding. It also applies the ReLU function to each convolution output and finally creates 64 feature maps, each of which is a 96 × 1 vector.
- The third one is a max-pooling layer for dimensionality reduction. It applies a max filter (with a stride of 2) to two non-overlapping consecutive elements to keep a bigger one only, for each feature map. Thus, the output of this layer is still 64 feature maps, yet each with a 48 × 1 vector.
- The fourth one is a flatten layer that is to merge and reshape the feature maps to be fitted to the following layer. The 64 feature maps obtained by the max-pooling layer are linked in series into a vector $c_{t,i} \in \mathbb{R}^{3072\times 1}$.
- The fifth one is a simple recurrent layer with 100 neurons and the hyperbolic tangent (TanH) activation function, i.e., $\text{TanH}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, element-wise. The outputs of this layer to the inputs $c_{t,1}$ and $c_{t,2}$ are obtained by

$$h_{t,1} = \text{TanH}(W_1 c_{t,1} + b_1) \in \mathbb{R}^{100\times 1},$$
$$h_{t,2} = \text{TanH}(W_1 c_{t,2} + U h_{t,1} + b_1) \in \mathbb{R}^{100\times 1},$$

  respectively, where $W_1 \in \mathbb{R}^{100\times 3072}$ is the input-to-output weight matrix, $U \in \mathbb{R}^{100\times 100}$ is the output-to-output weight matrix, and $b_1 \in \mathbb{R}^{100\times 1}$ is the recurrent bias vector. Note that the output $h_{t,1}$ that originally comes from $S_{t,1}$ is fed back along with the input $c_{t,2}$ to this recurrent layer, which in turn gives the output $h_{t,2}$. Thus, our CRNN network has a memory of 1 sub-instance due to the feedback.
- The sixth one is a fully-connected layer with 100 neurons and the ReLU activation function applied element-wise, leading to the following output z_t:

$$z_t = \text{ReLU}(W_2 h_{t,2} + b_2) \in \mathbb{R}^{100\times 1},$$

[7]It is still a 1D convolution, as the convolution is done over 1 dimension.

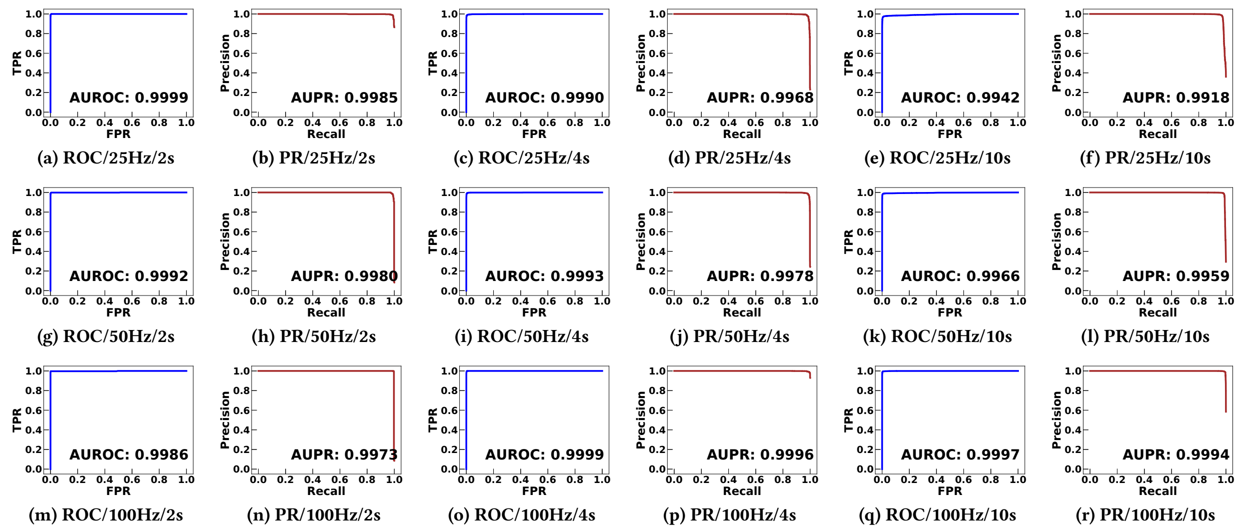


Figure 10: The ROC and PR curves of the CRNN model.

where $\boldsymbol{W}_2 \in \mathbb{R}^{100\times100}$ and $\boldsymbol{b}_2 \in \mathbb{R}^{100\times1}$ are the weight matrix and bias vector, respectively. Note that the output $\boldsymbol{z}_t$ is obtained based only on the input $\boldsymbol{h}_{t,2}$ (not $\boldsymbol{h}_{t,1}$), although $\boldsymbol{h}_{t,2}$ depends on both $\boldsymbol{h}_{t,1}$ and $\boldsymbol{c}_{t,2}$. To see this, refer to Figure 9(b) for the unfolded RNN structure.

- The last layer is an output layer with the logistic activation function. The final output is then given by

$$\hat{y}_t = f(\boldsymbol{w}_3 \cdot \boldsymbol{z}_t + b_3) = \frac{1}{1+\exp(-(\boldsymbol{w}_3 \cdot \boldsymbol{z}_t + b_3))} \in (0,1),$$

where $\boldsymbol{w}_3 \in \mathbb{R}^{100\times1}$ and $b_3 \in \mathbb{R}$ are the weight vector and bias term, respectively. Here, '$\cdot$' indicates the dot product. Note that this final output is the output of the model to the input instance $\boldsymbol{S}_t$. As was the case with the ANN model, for a given value of p, if $\hat{y}_t \geq p$, then the corresponding instance is classified as an earthquake, otherwise a non-earthquake.

For training the model while handling the data imbalance, we consider the following weighted cross-entropy loss function ℓ to be minimized:

$$\ell = -\big[c_0(1-y_t)\ln(1-\hat{y}_t) + c_1 y_t \ln \hat{y}_t\big],$$

where y_t is the true label of the input instance $\boldsymbol{S}_t$. Here, c_0 and c_1 are weights for the classes of non-earthquakes ($y_t = 0$) and earthquakes ($y_t = 1$), which can also be viewed as costs for a false alarm and a miss, respectively. Due to the data imbalance, we set $c_0 = 1$ and c_1 to be the ratio of non-earthquake instances to the number of earthquake instances, so that the total cumulative weights to both classes are the same.

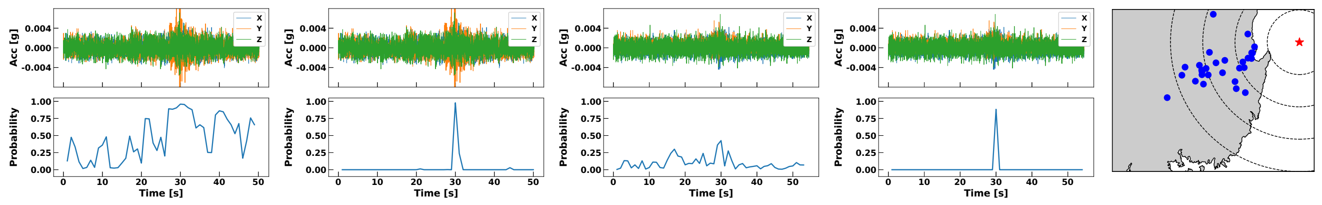
Model Performance. We evaluate our CRNN model through the same set of test cases as we used for the ANN model, and present the resulting ROC and PR curves of our model in Figure 10, where the corresponding AUROC and AUPR values are also reported in each subfigure. (See Appendix C for the detailed performance results with threshold $p = 0.5$.) We observe that our model demonstrates excellent overall performance *regardless of* the choices of sampling rate and selected earthquake duration, and also for an *almost entire range* of the values of the threshold p. Their detection accuracy is above 99% with the FAR less than 1% for all the cases. Overall, our CRNN model achieves remarkable performance improvement over the state-of-the-art ANN model. As was done for the ANN model, we employ the 10-fold cross-validation and confirm the results, but again omit the details due to space constraint.

Table 1: Runtime of the CRNN model [milliseconds]

# Sensors	100	300	500	1000	5000	10000
i7-7820X	18 ± 1	49 ± 1	81 ± 1	162 ± 1	795 ± 7	1600 ± 6
1080 Ti	9 ± 2	22 ± 3	34 ± 4	67 ± 4	344 ± 11	649 ± 28

Computational Cost. One may expect that our CRNN model would be computationally expensive and thus not suitable for the purpose of real-time detection. This is, however, not the case. With 300 sensors data, the runtime of executing the CRNN model per each 2-second input instance (so 300 instances in total) takes only about 49 ms. Note that other data processing time is negligible compared to this runtime. While the server of CrowdQuake currently uses an Intel Core i7 7820X processor, we observe that leveraging a GeForce GTX 1080 Ti GPU reduces the runtime to about 22 ms. We further observe that running the CRNN model at CrowdQuake is also scalable to thousands of sensors, as shown in Table 1. The runtime results are reported by taking the average over 20 iterations for each case. The value after ± indicates its standard deviation.

Additional Experiments. We further extend our study to the performance evaluation of the CRNN model built upon a different dataset, which contains a wider range of earthquake data that include the earthquake waveforms with *smaller* PGA values than before, while having the same set of non-earthquake data. Note that this newly built CRNN model has been used along with the previous one in CrowdQuake to see if it can better detect low magnitude earthquakes in practice. Specifically, we use a bigger set of Japan earthquakes, each of whose x-axis PGA is 0.05g or above, for model training and testing. We then evaluate the performance of the corresponding ANN and CRNN models for the same test cases as before, and we here summarize our results. See Appendix B and Appendix C for more details on the new dataset and


(a) Sensor-110; ANN (b) Sensor-110; CRNN (c) Sensor-218; ANN (d) Sensor-218; CRNN (e) A sensor map

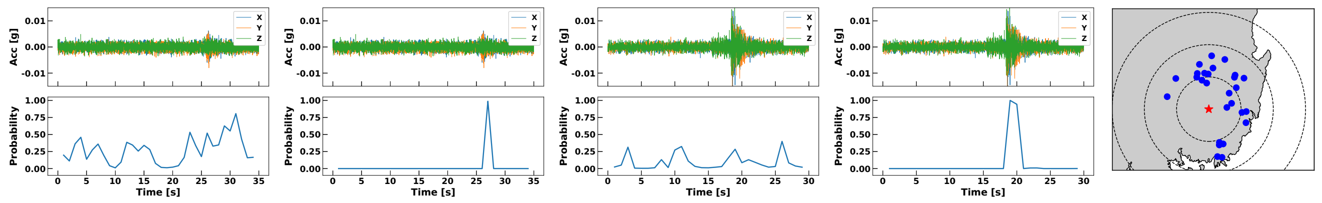
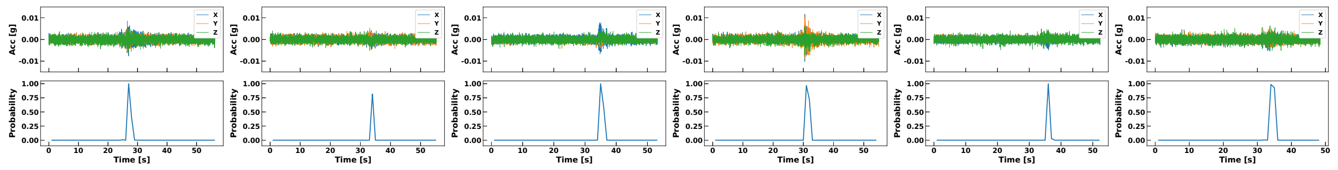

(f) Sensor-133; ANN (g) Sensor-133; CRNN (h) Sensor-184; ANN (i) Sensor-184; CRNN (j) A sensor map

Figure 11: The performance of the ANN and CRNN models when applied to (a)–(d) two 0210-earthquake waveforms and (f)–(i) two 1230-earthquake waveforms recorded by our low-cost sensors. Top: waveform; Bottom: prediction probability $\hat{y}$. Two maps of sensors whose records allowed the CRNN model to confirm (e) the 0210 earthquake and (j) the 1230 earthquake.


(a) Sensor-81 (b) Sensor-251 (c) Sensor-254 (d) Sensor-262 (e) Sensor-285 (f) Sensor-288

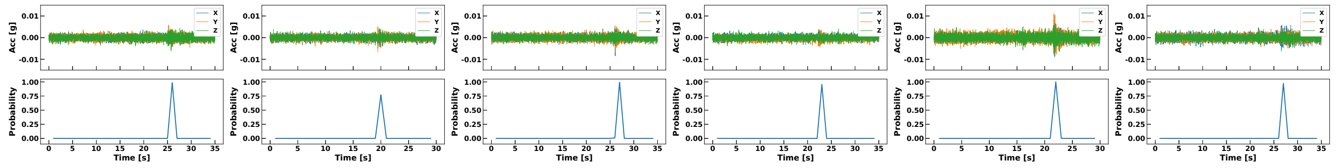

(g) Sensor-42 (h) Sensor-93 (i) Sensor-141 (j) Sensor-166 (k) Sensor-219 (l) Sensor-277

Figure 12: The performance of the CRNN model over (a)–(f) 0210-earthquake waveforms and (g)–(l) 1230-earthquake waveforms recorded on 6 different sensors.

the results, respectively. Our results indicate that the CRNN model still significantly outperforms the ANN model in all performance metrics. While the performance of the ANN model becomes slightly improved with this new dataset, its FAR remains non-negligible.

5 CASE STUDIES

It is clearly hard to validate CrowdQuake in practice, since earthquakes – the target for detection – are completely beyond our control and they happen irregularly and unpredictably. Nonetheless, there were five small earthquakes near the area where CrowdQuake is deployed, and CrowdQuake was able to detect all of them, albeit their small intensity measured. We here present two earthquakes as examples – one occurred on February 10, 2019 (named '0210 earthquake'), the other occurred on December 30, 2019 (named '1230 earthquake'). The PGA of the 0210 earthquake was 0.0045g and its corresponding intensity was III in the modified Mercalli intensity (MMI) scale [23]. The PGA of the 1230 earthquake was 0.0164g and its MMI intensity was IV. These earthquakes left the people near their epicenters shaken and feared.

The results are quite promising, since they somewhat demonstrate the feasibility of detecting earthquakes with low magnitudes, which have been considered too small to be detected using low-cost sensors, e.g., [16] and references therein. They also show the possibility of going beyond the limitation set by the recent MyShake system [16, 18, 19] that aims only at detecting magnitude 5 or larger earthquakes at distances of 10 km or less. We notice that the PGA values of earthquake waveforms used for training the ANN model in MyShake were all above 0.2g [16], which translates into the peak intensity VII in the MMI scale and is far bigger than 0.0045g and 0.0164g. While we use lower 'base' PGA values, i.e., 0.1g and 0.05g, to select earthquake waveforms (with PGA values greater than the base values) in order to train two different versions of our CRNN model in CrowdQuake, the base PGA values are still greater than 0.0045g and 0.0164g. However, we emphasize that we consider a relatively long portion of each earthquake waveform for model training, which is the 10-second selected duration, i.e., 1 second before the peak and 9 seconds after the peak (See Appendix B for more details.). Thus, the earthquake datasets for training actually contain low-amplitude portions of earthquake waveforms, so the two versions of our CRNN model are able to detect small earthquakes in practice. We also observe that the one with the base PGA value of 0.05g performs better than the other, so we here only report its results for brevity.[8]

We present a few representative results of CrowdQuake for detecting the 0210 and 1230 earthquakes. Several low-cost sensors in CrowdQuake were able to record the small earthquakes and its CRNN model confirmed them as earthquakes. For comparison, we also apply the ANN model, trained under the same condition (the 100-Hz sampling rate and 10-second selected duration) as the CRNN model was trained, to the same earthquake waveforms recorded, and find out that it is still not able to detect both earthquakes. Figure 11(a)–(d) shows the performance of the ANN and CRNN models in prediction probability $\hat{y}$ (classification output), when they are applied to the waveforms recorded on two sensors of CrowdQuake in the happening of the 0210 earthquake. Similarly, Figure 11(f)–(i) shows the performance comparison for the 1230 earthquake. Our CRNN model clearly captures the strongest portions of the earthquake waveforms even in the presence of relatively high background noise at the low-cost sensors. However, the ANN model is largely inaccurate and unreliable, or it is rather close to a random decision with the low-cost sensors for both earthquakes. This confirms the superiority of our CRNN model over the ANN model.

We further present the performance of CrowdQuake, or its CRNN model. Figure 12(a)–(f) shows that the CRNN model correctly detects the 0210 earthquake from the seismic waveforms recorded on 6 sensors. Similarly, Figure 12(g)–(l) indicates the detection of the 1230 earthquake with 6-sensor records. We also observe that the number of sensors that are able to pick up the seismic waveforms, which are then confirmed by the CRNN model, depends on the value of the probability threshold p. If we set $p = 0.5$, then the CRNN model enables about 20 sensors to detect both earthquakes, i.e., the prediction-probability outputs $\hat{y}$ of the CRNN model are greater than 0.5. When we lower down p to 0.2, it increases to 28 for the 0210 earthquake and 27 for the 1230 earthquake, respectively. The locations of the sensors are shown as the blue dots in Figure 11(e) and (j), respectively. The distance between each of the senors and its epicenter approximately ranges from 45 km to 140 km for the 0210 earthquake, and from 17 km to 53 km for the 1230 earthquake, respectively. Note that the distances are much longer than the 10 km considered in MyShake [16]. Therefore, our observations clearly indicate that CrowdQuake has great potential to detect a *wide* range of earthquakes using low-cost sensors in real time, including small earthquakes at long distances that would not have been considered possible in the current literature.

6 CONCLUSION

We have presented CrowdQuake, which is a deep learning-driven networked system of low-cost sensors for earthquake detection in real time, and its operational results. CrowdQuake is designed to manage streaming time-series acceleration data from hundreds to thousands of sensors and to be capable of real-time earthquake detection via our CRNN model. It also automates the process of identifying abnormal sensors, which can deteriorate the performance of earthquake detection. Finally, the CRNN model in CrowdQuake exhibits remarkable performance, i.e., 99% detection performance with a false alarm rate less than 1%, and enables CrowdQuake to have detected five small earthquakes over its one-year operation.

[8]It might be possible to further optimize the performance of our CRNN model by training it under the dataset of earthquakes with a properly chosen base-PGA value.

ACKNOWLEDGMENTS

This work was supported in part by the Korea Meteorological Administration under Grant No. KMA-135002988. We are grateful for the generous system support of SK Telecom. Y. Kwon and C. Lee are co-corresponding authors.

REFERENCES

[1] 2019. Apache Kafka. https://kafka.apache.org/

[2] 2019. ElasticSearch. https://www.elastic.co/

[3] 2019. Grafana. https://grafana.com/

[4] 2019. InfluxDB. https://kafka.apache.org/

[5] C. M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.

[6] R. W. Clayton, T. Heaton, M. Chandy, A. Krause, M. Kohler, J. Bunn, R. Guy, M. Olson, M. Faulkner, M. Cheng, L. Strand, R. Chandy, D. Obenshain, A. Liu, and M. Aivazis. 2011. Community seismic network. *Annals of Geophysics* 54, 6 (2011).

[7] E. Cochran, J. Lawrence, C. Christensen, and A. Chung. 2009. A novel strong-motion seismic network for community participation in earthquake monitoring. *IEEE Instrumentation & Measurement Magazine* 12, 6 (2009).

[8] A. D'Alessandro. 2016. Tiny accelerometers create Europe's first urban seismic network. *Eos* 97, 10.1029 (2016).

[9] J. Davis and M. Goadrich. 2006. The relationship between precision-recall and ROC curves. In *Proceedings of ICML*. 233–240.

[10] M. Faulkner, R. Clayton, T. Heaton, K. M. Chandy, M. Kohler, J. J. Bunn, R. Guy, A. H. Liu, M. Olson, M. Cheng, and A. Krause. 2014. Community sense and response systems: Your phone as quake detector. *Commun. ACM* 57 (2014), 66–75.

[11] M. Faulkner, M. Olson, R. Chandy, J. Krause, K. M. Chandy, and A. Krause. 2011. The next big one: Detecting earthquakes and other rare events from community-based sensors. In *Proceedings of ACM/IEEE IPSN*. 13–24.

[12] T. Fawcett. 2006. An introduction to ROC analysis. *Pattern Recognition Letters* 27, 8 (2006), 861–874.

[13] I. Goodfellow, Y. Bengio, and A. Courville. 2016. *Deep Learning*. MIT press.

[14] S. Horiuchi, Y. Horiuchi, S. Yamamoto, H. Nakamura, C. Wu, P. A Rydelek, and M. Kachi. 2009. Home seismometer for earthquake early warning. *Geophysical Research Letters* 36, 5 (2009).

[15] D. P. Kingma and J. Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[16] Q. Kong, R. M. Allen, L. Schreier, and Y.-W. Kwon. 2016. MyShake: A smartphone seismic network for earthquake early warning and beyond. *Science Advances* 2, 2 (2016).

[17] Q. Kong, A. Inbal, R. M. Allen, Q. Lv, and A. Puder. 2018. Machine learning aspects of the MyShake global smartphone seismic network. *Seismological Research Letters* 90, 2A (December 2018), 546–552.

[18] Q. Kong, Y.-W. Kwon, L. Schreierz, S Allen, R. Allen, and J. Strauss. 2015. Smartphone-based networks for earthquake detection. In *Proceedings of 15th International Conference on Innovations for Community Services (I4CS)*. 1–8.

[19] Q. Kong, Q. Lv, and R. M. Allen. 2019. Earthquake early warning and beyond: Systems challenges in smartphone-based seismic network. In *Proceedings of ACM HotMobile*. 57–62.

[20] National Research Institute for Earth Science and Disaster Resilience (NIED). 2019. Strong-motion Seismograph Networks (K-NET, KiK-net). http://www.kyoshin.bosai.go.jp/kyoshin/data/index_en.html

[21] T. Perol, M. Gharbi, and M. Denolle. 2018. Convolutional neural network for earthquake detection and location. *Science Advances* 4, 2 (2018).

[22] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)* 15, 1 (January 2014), 1929–1958.

[23] D. J. Wald, V. Quitoriano, T. H. Heaton, and H. Kanamori. 1999. Relationships between peak ground acceleration, peak ground velocity, and modified Mercalli intensity in California. *Earthquake Spectra* 15, 3 (1999), 557–564.

[24] Y.-M. Wu. 2015. Progress on development of an earthquake early warning system using low-cost sensors. *Pure and Applied Geophysics* 172, 9 (2015), 2343–2351.

[25] W. Zhu and G. C. Beroza. 2018. PhaseNet: A deep-neural-network-based seismic arrival-time picking method. *Geophysical Journal International* 216, 1 (2018), 261–273.

A DETAILS OF THE ANN MODEL

The ANN model in [16] uses the following three features that are extracted from each instance (a 2-second window) as an input: (i) cumulative absolute velocity (CAV), (ii) interquartile range (IQR), and (iii) zero crossing count (ZC). The CAV is a cumulative measure of the amplitude of the acceleration data for 2 seconds, which is defined as CAV $= \int_0^2 \|a(s)\| ds$, where $\|a(s)\|$ is the length (l^2 norm) of the vector of three acceleration components at time s. In practice, it is computed as the sum of the lengths of the vectors of three acceleration components recorded over 2 seconds. The IQR is a measure of statistical dispersion that indicates the middle 50% range of the amplitudes $\|a(s)\|$ for 2 seconds. Note that the CAV, when properly normalized, and the IQR capture the notions of the 'average' and 'variance' of the vector length, respectively, for each 2-second window. The ZC is a frequency measure that counts how many times the (acceleration) signal changes its sign for the 2-second window. Its maximum value is used among the ZC values of three components. Each feature is scaled or normalized to a range of 0 to 1.

The ANN model uses the logistic sigmoid activation function at the hidden and output layers, which is given by $f(x) = \frac{1}{1+\exp(-x)}$.

Letting $\boldsymbol{s}_t \triangleq [s_t(1), s_t(2), s_t(3)]^T$ be the feature vector of the t-th instance, its output $\hat{y}(\boldsymbol{s}_t)$ is then obtained by

$$\hat{y}(\boldsymbol{s}_t) = f\left(w_0 + \sum_{k=1}^{5} w_k f\left(w_0^k + \sum_{i=1}^{3} w_i^k s_t(i)\right)\right) \in (0,1),$$

where $\{w_i^k\}$ be the weight vector for the k-th neuron at the hidden layer and $\{w_k\}$ be the weight vector at the output layer.

There is a data imbalance problem for real-time earthquake detection, since non-earthquake data instances are far more than earthquake data instances. To address this problem, the k-means clustering method is used in [16]. Letting m be the number of earthquake feature vectors, the non-earthquake feature vectors are grouped into m clusters. The centroids of clusters are used to represent the (transformed) non-earthquake feature vectors for training the ANN model. We use backpropagation and stochastic gradient descent [13] for training the ANN model, where a learning rate of 0.2 is chosen. We also apply a different training algorithm, which is the stochastic gradient descent with the Adam optimizer [15], and observe similar results.

B EXPERIMENTAL SETTINGS

B.1 Dataset

For earthquake data, we first use a total of 2299 K-NET records of Japan earthquakes from NIED [20], each of whose x-axis component PGA is greater than 0.1g. To better detect low-magnitude earthquakes in practice, we also consider a smaller value of the PGA, which is 0.05g, when selecting earthquake waveforms from NIED, and thus use 8973 K-NET records with x-axis PGA values greater than 0.05g. For non-earthquake data, we use a wide range of acceleration records that we have collected using the low-cost sensors, which include background noise of the low-cost sensors measured in several environments and the ones recorded on the low-cost sensors while under various human activities. They are all recorded at a sampling rate of 100 data points per second.

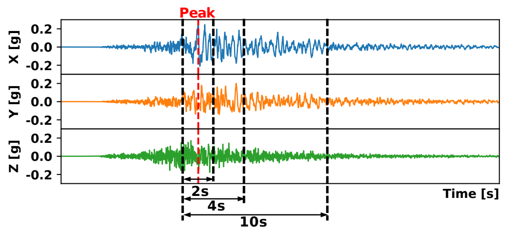


Figure 13: Three durations of an earthquake waveform.

B.2 Test Cases

We begin with the setting in [16, 18], where the sampling rate is 25 Hz and the strongest portion of each earthquake waveform is only selected for earthquake instances (2-second windows) to clearly separate (large-magnitude) earthquakes from non-earthquakes. For the former, the data is all down-sampled to 25 Hz. For the latter, we select the 2-second duration of each earthquake waveform around the peak acceleration in the x-axis component, i.e., the one with 1 second before the peak and after the peak. We further extend the test cases as follows. We consider sampling rates of 50 Hz and 100 Hz, in addition to 25 Hz. We also take two different durations – 4 seconds and 10 seconds, into consideration for choosing the strongest portion of each earthquake waveform. Specifically, the 4-second duration is the one with 1 second before the peak and 3 seconds after the peak, while the 10-second duration is the one with 1 second before the peak and 9 seconds after the peak. See Figure 13 for illustration. Thus, three durations lead to 1, 3, and 9 instances for each earthquake waveform, respectively. For non-earthquake data, we use the entire waveforms for obtaining instances.

B.3 Data Splitting

We split the data for both classes of earthquakes (EQ) and non-earthquakes (non-EQ) into training set (70%) and testing set (30%). In particular, the EQ data is split based on earthquake records. For EQ dataset with PGA greater than 0.1g, 1610 and 689 out of 2299 records are randomly selected for training and testing, respectively. For EQ dataset with PGA greater than 0.05g, 6281 and 2692 out of 8973 records are used for training and testing, respectively. We then divide each earthquake record into instances (2-second windows with a 1-second step). On the other hand, the none-EQ dataset is randomly split based on the instances. Table 2 summarizes the numbers of instances for model training and testing.

Table 2: Summary of data for model training and testing

EQ dataset with PGA > 0.1g	Training set		Testing set	
Selected duration	# EQ	# Non-EQ	# EQ	# Non-EQ
2 seconds	1610	34810	689	14924
4 seconds	4830	34810	2067	14924
10 seconds	14490	34810	6201	14924
EQ dataset with PGA > 0.05g	**Training set**		**Testing set**	
Selected duration	# EQ	# Non-EQ	# EQ	# Non-EQ
2 seconds	6281	34810	2692	14924
4 seconds	18843	34810	8076	14924
10 seconds	56529	34810	24228	14924

B.4 Performance Metrics

To evaluate each model, we use TPR/recall, FPR/FAR, and precision that are obtained from its four outcomes, i.e., true positive (TP),

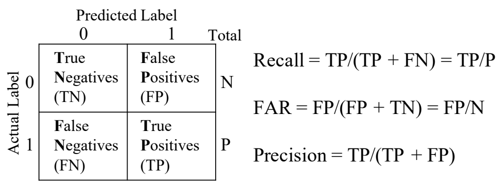


Figure 14: Confusion matrix.

false positive (FP), true negative (TN), and false negative (FN). Here, for each instance, positive means an earthquake, while negative means a non-earthquake. Note that *Recall* is the ratio of TPs to the overall number of positive instances. *FAR* is the ratio of FPs to the overall number of negative instances. See Figure 14 for their definitions. Recall that for a given threshold value p, the higher recall, higher precision and lower FAR, the better the model is.

We also use ROC and PR curves for performance evaluation. Recall that for the ROC curve, if it is closer to the upper left corner (a perfect classifier), which implies no FPs, the model is better. As to the PR curve, the closer to the upper right corner, having only TPs without any FP and FN, the better the model is.

B.5 Software

We use Python 3.7 and the following libraries for our implementations and experiments.[9]

- We use ObsPy 1.1.1 for earthquake data processing.
- We use TensorFlow 1.14 and Keras 2.2.4 to build our CRNN model. We use scikit-learn 0.21.2 for ANN and k-means clustering algorithm. We also use scikit-learn to compute the performance metrics.
- We use NumPy 1.16.1 and pandas 0.24.1 for data processing and manipulation. Matplotlib 3.0.3 is used to plot figures.

B.6 Hyperparameters and Configurations

ANN. The hidden-layer size is 5, and the activation function is 'logistic'. We use the stochastic gradient descent with a learning rate of 0.2 to train the ANN model. The regularization parameter is 0, and the maximum number of iterations is 10,000. We use the default values set in scikit-learn for all the other hyperparameters. The ANN model is trained on a CPU.

CRNN. For both convolutional layers, the number of filters is 64, the kernel size is 3, and the activation function is 'relu'. The pool size is 2 for the max-pooling layer. The simple RNN layer contains 100 neurons with 'tanh' activation function. The fully-connected layer uses 100 neurons with 'relu' function. The output layer uses 'sigmoid' function. We use mini-batch gradient descent with the Adam optimizer [15] to optimize the model parameters and weights, minimizing the loss function ℓ, where the mini-batch size of 256 is chosen empirically. We also use the dropout [22] at the fully-connected layer with probability 0.5 to avoid overfitting. We run the model training for 100 epoches, which is chosen as we have observed that the amount of reduction in the loss function ℓ after about 50 ~ 60 epoches becomes quite small. We use the default values set in TensorFlow for all the other hyperparameters. The CRNN model is trained on a GPU to reduce the training time.

[9]Code to reproduce our experiments is available at http://www.crowdquake.net.

C ADDITIONAL RESULTS

We below provide additional results of our experiments.

Results of EQ dataset with PGA > 0.1g. We present in Table 3 the performance of the ANN and CRNN models when we set the threshold p to be 0.5. Note that the ROC and PR curves of the models were given in Figure 8 and Figure 10, respectively.

Table 3: Performance comparison with a threshold of 0.5

		25 Hz			50 Hz			100 Hz		
		2 s	4 s	10 s	2 s	4 s	10 s	2 s	4 s	10 s
ANN	Recall	0.93	0.93	0.92	0.94	0.94	0.88	0.90	0.94	0.94
	Precision	0.07	0.20	0.43	0.07	0.20	0.43	0.31	0.52	0.69
	FAR	0.52	0.51	0.50	0.52	0.52	0.47	0.09	0.12	0.17
CRNN	Recall	0.99	0.98	0.97	0.98	0.99	0.98	0.99	0.99	0.99
	Precision	0.86	0.97	0.98	0.97	0.96	0.98	0.99	0.98	0.99
	FAR	0.01	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00

Results of EQ dataset with PGA > 0.05g. We here only provide the AUROC and AUPR values of the ANN and CRNN models in Table 4, instead of their ROC and PR curves due to space constraint. We also report the performance of the models with $p = 0.5$ in Table 5.

Table 4: Performance of the ANN and CRNN models

		25 Hz			50 Hz			100 Hz		
		2 s	4 s	10 s	2 s	4 s	10 s	2 s	4 s	10 s
ANN	AUROC	0.64	0.69	0.83	0.69	0.79	0.89	0.95	0.96	0.97
	AUPR	0.18	0.48	0.87	0.20	0.64	0.92	0.73	0.89	0.97
CRNN	AUROC	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	AUPR	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99

Table 5: Performance comparison with a threshold of 0.5

		25 Hz			50 Hz			100 Hz		
		2 s	4 s	10 s	2 s	4 s	10 s	2 s	4 s	10 s
ANN	Recall	0.93	0.94	0.98	0.92	0.92	0.92	0.92	0.94	0.96
	Precision	0.26	0.52	0.76	0.26	0.52	0.82	0.47	0.79	0.93
	FAR	0.47	0.47	0.50	0.47	0.43	0.31	0.18	0.13	0.11
CRNN	Recall	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	Precision	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	FAR	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

We observe that our CRNN model outperforms the ANN model in all performance metrics, regardless of the choices of the PGA value for selecting earthquake waveforms from NIED to build the set of earthquake instances for model training and testing. We also notice that the overall performance of the ANN model is slightly improved when we include more earthquake instances, which is the case with the PGA value of 0.05g. Nevertheless, the FAR of the ANN model is still higher than 10% even with the best case. Note that, for EQ dataset with PGA greater than 0.05g, we do not apply k-means clustering to the cases with the selected duration of 10 seconds, as the number of earthquake instances now becomes greater than that of non-earthquake instances, so the negative aspect of the data imbalance problem for detecting earthquakes no longer exists. For all the other test cases, k-means clustering is used to balance the dataset before training the ANN model.