SEISMOLOGY

Convolutional neural network for earthquake detection and location

Thibaut Perol,[1,2]* Michaël Gharbi,[3] Marine Denolle[4]

The recent evolution of induced seismicity in Central United States calls for exhaustive catalogs to improve seismic hazard assessment. Over the last decades, the volume of seismic data has increased exponentially, creating a need for efficient algorithms to reliably detect and locate earthquakes. Today's most elaborate methods scan through the plethora of continuous seismic records, searching for repeating seismic signals. We leverage the recent advances in artificial intelligence and present ConvNetQuake, a highly scalable convolutional neural network for earthquake detection and location from a single waveform. We apply our technique to study the induced seismicity in Oklahoma, USA. We detect more than 17 times more earthquakes than previously cataloged by the Oklahoma Geological Survey. Our algorithm is orders of magnitude faster than established methods.

INTRODUCTION

The recent exploitation of natural resources and associated waste water injection in the subsurface have induced many small and moderate earthquakes in the tectonically quiet Central United States (*1*). Induced earthquakes contribute to seismic hazard. Between 2008 and 2017 only, nine earthquakes of magnitude greater than 5.0 might have been triggered by nearby disposal wells. Most earthquake detection methods are designed for moderate and large earthquakes. As a consequence, they tend to miss many of the low-magnitude earthquakes that are masked by seismic noise. Detecting and cataloging these earthquakes are key to understanding their causes (natural or human-induced) and, ultimately, to mitigating the seismic risk.

Traditional approaches to earthquake detection (*2*, *3*) fail to detect events buried in even the modest levels of seismic noise. Waveform similarity can be used to detect earthquakes that originate from a single region, with the same source mechanism ("repeating earthquakes"). Waveform autocorrelation is the most effective method to identify these repeating earthquakes from seismograms (*4*). Although particularly robust and reliable, the method is computationally intensive, scales quadratically with the number of windows, and thus is not practical for long time series. One approach to reduce the computation is to select a small set of representative waveforms as templates and correlate only these with the full-length continuous time series (*5*). The detection capability of template matching techniques depends directly on the number of templates used. When using a catalog of located earthquakes as a database of templates, the location of the detected event is restricted to that of the matching templates, and one can form families of events (*6*). Today's most elaborate template matching methods seek to represent the general features in the waveforms and reduce the number of templates by principal component analysis through subspace detection (*7*–*10*). However, the earthquake location information is lost during the decomposition of the database into representative eigen waveforms. Recently, an unsupervised earthquake detection method, referred to as Fingerprint And Similarity Thresholding (FAST), managed to reduce the complexity of the template matching approach. FAST extracts features, or fingerprints, from seismic waveforms, creates a bank of these fingerprints, and reduces the similarity search through locality-sensitive hashing (*11*). The scaling of FAST has shown promise with near-linear scaling to large data sets.

We cast earthquake detection as a supervised classification problem and propose the first convolutional neural network for earthquake detection and location (ConvNetQuake) from seismograms. Our algorithm builds on recent advances in deep learning (*12*–*15*). Previous studies have pioneered the use of artificial neural networks to classify seismograms from hand-engineered features (*16*, *17*, *18*) or compressed representations of the waveforms via neural autoencoders (*19*). ConvNetQuake is trained on a large data set of labeled raw seismic waveforms and learns a compact representation that can discriminate seismic noise from earthquake signals. The waveforms are no longer classified by their similarity to other waveforms, as in previous work. Instead, we analyze the waveforms with a collection of nonlinear local filters. During the training phase, the filters are optimized to select features in the waveforms that are most relevant to classification. This bypasses the need to store a perpetually growing library of template waveforms. Owing to this representation, our algorithm generalizes well to earthquake signals never seen during training. It is more accurate than state-of-the-art algorithms and runs orders of magnitude faster. In addition, ConvNetQuake outputs a probabilistic location of an earthquake's source from a single station. We evaluate the performances and limitations of our algorithm and apply it to induced earthquakes in Central Oklahoma, USA. We show that it uncovers new earthquakes absent from standard catalogs.

RESULTS

Data

The state of Oklahoma, USA has recently experienced a marked surge in seismic activity (*1*, *10*, *20*) that has been correlated with the intensification of waste water injection (*21*–*24*). Here, we focus on the particularly active area near Guthrie, OK. In this region, the Oklahoma Geological Survey (OGS) cataloged 2021 seismic events from 15 February 2014 to 16 November 2016 (see Fig. 1). Their seismic moment magnitudes range from M_w −0.2 to M_w 5.8. We use the continuous ground velocity records from two local stations GS.OK027 and GS.OK029 (see Fig. 1). GS.OK027 was active from 14 February 2014 to 3 March 2015. GS.OK029 was deployed on 15 February 2014 and has remained active

[1]John A. Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138, USA. [2]Gram Labs Inc., Arlington, VA 22201, USA. [3]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA. [4]Department of Earth and Planetary Sciences, Harvard University, Cambridge, MA 02138, USA.
*Corresponding author. Email: tperol@alumni.harvard.edu

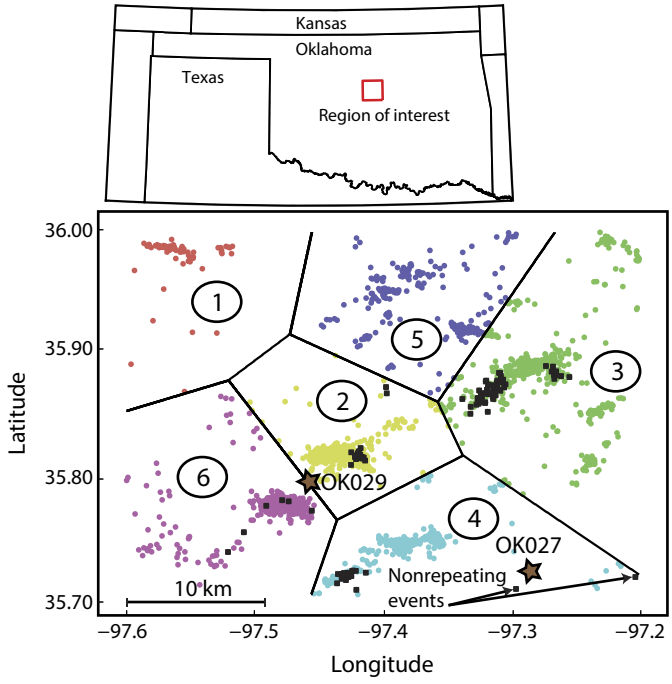

Fig. 1. Earthquakes and seismic station in the region of interest (near Guthrie, OK) from 14 February 2014 to 16 November 2016. GS.OK029 and GS.OK027 are the two stations that record continuously the ground motion velocity. The colored circles are the events in the training data set. Each event is labeled with its corresponding area. The thick black lines delimit the six areas. The black squares are the events in the test data set. Two events from the test set are highlighted because they do not belong to the same earthquake sequences and are nonrepeating events.

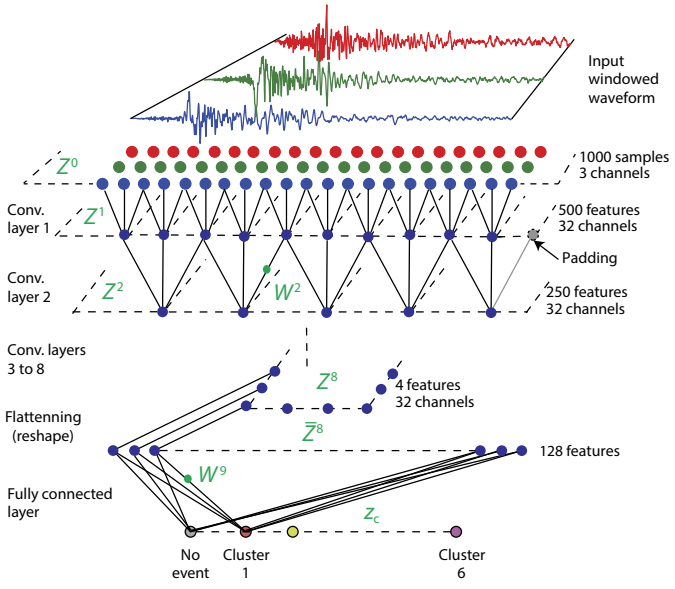

Fig. 2. ConvNetQuake architecture. The input is a waveform of 1000 samples on three channels. Each convolutional layer consists in 32 filters that downsample the data by a factor of 2 (see Eq. 1). After the eighth convolution, the features are flattened into a 1D vector of 128 features. A fully connected layer outputs the class scores (see Eq. 2).

since. Signals from both stations are recorded at 100 Hz on three channels corresponding to the three spatial dimensions: HHZ oriented vertically, HHN oriented north-south, and HHE oriented west-east.

Generating location labels

We partition the 2021 earthquakes into six geographic clusters. For this, we use the *K*-means algorithm (*25*), with the Euclidean distance between epicenters as the metric. The centroids of the clusters we obtain define six areas on the map (Fig. 1). Any point on the map is assigned to the cluster whose centroid is the closest (that is, each point is assigned to its Voronoi cell). We find that six clusters allow for a reasonable partition of the major earthquake sequences. Our classification thus contains seven labels or classes in the machine learning terminology: Class 0 corresponds to seismic noise without any earthquake, and classes 1 to 6 correspond to earthquakes originating from the corresponding geographic area.

Extracting windows for classification

We divide the continuous waveform data into monthly streams. We normalize each stream individually by subtracting the mean over the month and dividing by the absolute peak amplitude (independently for each of the three channels). We extract two types of 10-s-long windows from these streams: windows containing events and windows free of events (that is, containing only seismic noise).

To select the event windows and attribute their geographic cluster, we use the catalogs from the OGS. Together, GS.OK027 and GS.OK029 yield 2918 windows of labeled earthquakes for the period between 15 February 2014 and 16 November 2016. Benz *et al.* (*10*) built a new comprehensive catalog of events between 15 February and 31 August 2014 but have not yet provided earthquake location, which are needed for training.

We look for windows of seismic noise in between the cataloged events. Because some of the low-magnitude earthquakes may be buried in seismic noise, it is important that we reduce the chance of mislabeling these events as noise. We thus use a more exhaustive catalog created by Benz *et al.* (*10*) to select our noise samples. Our process yields 831,111 windows of seismic noise.

Training/testing sets

We split the window data set into two independent sets: a test set and a training set. The test set contains all the windows for July 2014 (209 events and 131,072 windows of noise), whereas the training set contains the remaining windows (2709 events and 700,039 noise windows).

Data set augmentation

Deep classifiers, such as ours, have many trainable parameters. Although they require a large amount of examples of each class to avoid overfitting, they generalize correctly to unseen examples. To build a large enough data set of events, we use streams recorded at two stations (GS.OK029 and GS.OK27; see fig. S1), thus roughly doubling the number of windows in the original training set. The input of our network is the three-channel raw waveform (seen as a single stream) recorded at either of these stations. Furthermore, we generate additional event windows by perturbing existing ones with zero-mean Gaussian noise. This balances the number of event and noise windows during training, a strategy to regularize the network and prevent overfitting (*26*–*29*).

ConvNetQuake

Our model is a deep convolutional network (Fig. 2) that takes a window of three-channel waveform seismogram data as input and predicts its label either as seismic noise or as an event with its geographic cluster.

The parameters of the network are optimized to minimize the discrepancy between the predicted labels and the true labels on the training set (see Methods for details).

Detection accuracy

In a first experiment to assess the detection performance of our algorithm, we ignore the geographic label (that is, labels 1 to 6 are considered as a single "earthquake" class). The detection accuracy is the percentage of windows correctly classified as earthquake or noise. Our algorithm successfully detects all the 209 events cataloged by the OGS. Among the 131,972 noise windows of our test set of July 2014, ConvNetQuake correctly classifies 129,954 noise windows and misclassifies 2018 of the noise windows as events. Among those windows, 1902 windows were confirmed as events by the autocorrelation method (detailed in the Supplementary Materials). That is, our algorithm made 116 false detections. In summary, our algorithm predicts 129,954 true negatives, 116 false positives, 0 false negative, and 2111 true positives. Therefore, the precision (fraction of detected events that are true events) is 94.8%, and the recall (fraction of true events correctly detected) is 100%.

Location accuracy

We then evaluate the location performance. For each of the detected events, we compare the predicted class (*1*–*6*) with the geographic label chosen from the OGS catalog. We obtain 74.5% location accuracy (fraction of correctly labeled class) on the test set (see Table 1). For comparison with a "chance" baseline, selecting a class at random would give 1/6 = 16.7% accuracy.

We also experimented with a larger number of clusters (50; see fig. S2) and obtained 22.5% in location accuracy. Although the accuracy for higher-resolution location is lower, it remains 10 times better than a chance at 1/50 = 2%. The performance drop is not surprising because, on average, each class now only provides 40 training samples, which we attribute to an insufficient number of labels for proper training.

Table 1. Performances of three detection methods, excluding the overhead runtimes (1.5 hours of offline training for ConvNetQuake and 47 min of feature extraction and database generation for FAST). Autocorrelation and FAST results are as reported by Yoon *et al.* (*11*). The computational runtimes are for the analysis of 1 week of continuous waveform data. NA, not applicable.

	Autocorrelation	FAST	ConvNetQuake (this study)
Precision	100%	88.1%	94.8%
Recall	77.5%	80.1%	100%
Event location accuracy	NA	NA	74.6%
Reported runtime	9 days, 13 hours	48 min	1 min, 1 s

Probabilistic location map

Our network computes a probability distribution over the classes. This allows us to create a probabilistic map of earthquake location. We show in Fig. 3 the maps for a correctly located event and an erroneous classification. For the correctly classified event, most of the probability mass is on the correct class. This event is classified with approximately 99% confidence. For the misclassified event, the probability distribution is more diffuse, and the location confidence drops to 40%.

Generalization to nonrepeating events

Our algorithm generalizes well to waveforms that are dissimilar from those in the training set. We quantify this using synthetic seismograms and compare our method to template matching (*5*). We generate day-long synthetic waveforms by inserting multiple 45 copies of a given template over a Gaussian noise floor, varying the signal-to-noise ratio (SNR) from −1 to 8 dB. An example of synthetic seismogram is shown in fig. S3.

We choose two templates waveforms T_1 and T_2 (shown in fig. S4). Both waveforms T_1 and T_2 exhibit opposite polarity (either due to a different location or source focal mechanism) and pulse duration (event size). Using the procedure described above, we generate a training set using T_1 and two testing sets using either T_1 or T_2. We train both ConvNetQuake and the template matching method (see the Supplementary Materials) on the training set (generated with T_1).

On the T_1 testing set, both methods successfully detect all the events. On the other testing set where only T_2 is present, the template matching method fails to detect the inserted events, even at high SNR. However, ConvNetQuake recognizes the new (unknown) events. The accuracy of our model increases with SNR (see Fig. 4). For SNRs higher than 7 dB, ConvNetQuake detects all the inserted seismic events.

This generalization of waveform recognition is also highlighted in our test set with real seismic waveforms. Some events in our test data

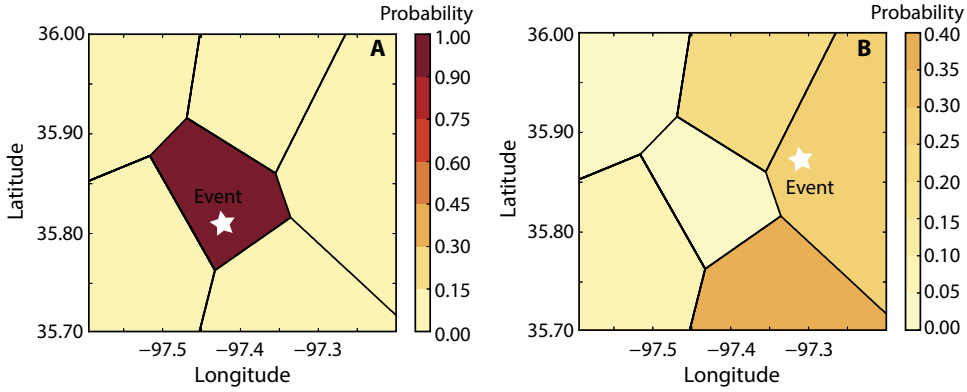

Fig. 3. Probabilistic location map of two events. (**A**) The event is correctly located, and the maximum of the probability distribution corresponds to the area in which the event is located. (**B**) The event is not located correctly, and the maximum of the probability distribution corresponds to an area different from the true location of the event.

set from Oklahoma are nonrepeating events (see highlighted events in Fig. 1). Although a template matching method using the waveforms from our training set cannot detect them, ConvNetQuake detected them using its compact and general representation of waveform features.

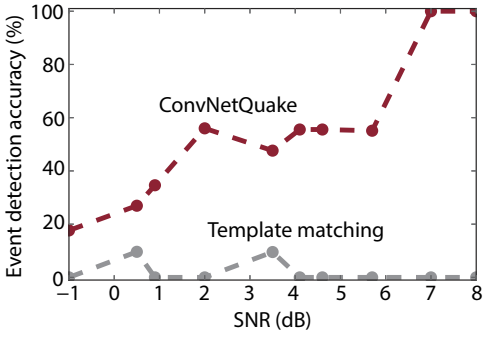

Fig. 4. Detection accuracy between ConvNetQuake and template matching. Percentage of the inserted events detected by the two methods in the synthetic data constructed by inserting an event template T_2 unseen during training as a function of the SNR. The training set consists of 4-day-long seismograms (SNR ranging from −5 to 1 dB) containing 45 inserted event templates T_1. The test set consists of 10-day-long seismograms (SNR ranging from −1 to 8 dB) containing 45 event templates T_2.

Earthquake detection on continuous records

We run ConvNetQuake on 1 month of continuous waveform data recorded at GS.OK029 in July 2014. The three-channel waveforms are cut into 10-s-long, nonoverlapping windows, with a 1-s offset between consecutive windows to avoid potential redundant detections. Our algorithm detects 4225 events in addition to those from the OGS catalog. This is more than five events per hour. Performing autocorrelation confirms 3949 of these are events (see the Supplementary Materials for details). One set of these events that is repeated 479 times (either highly correlated or highly anticorrelated) is shown in Fig. 5. For comparison, the subspace detection method (*10*) using three-channel templates found 5737 events during this period.

Comparison with other detection methods

We compare our detection performances to autocorrelation and FAST reported by Yoon *et al.* (*11*). Autocorrelation relies on waveform similarity (repeating earthquakes), and FAST relies on fingerprints similarity. Both techniques do not require a priori knowledge on templates,

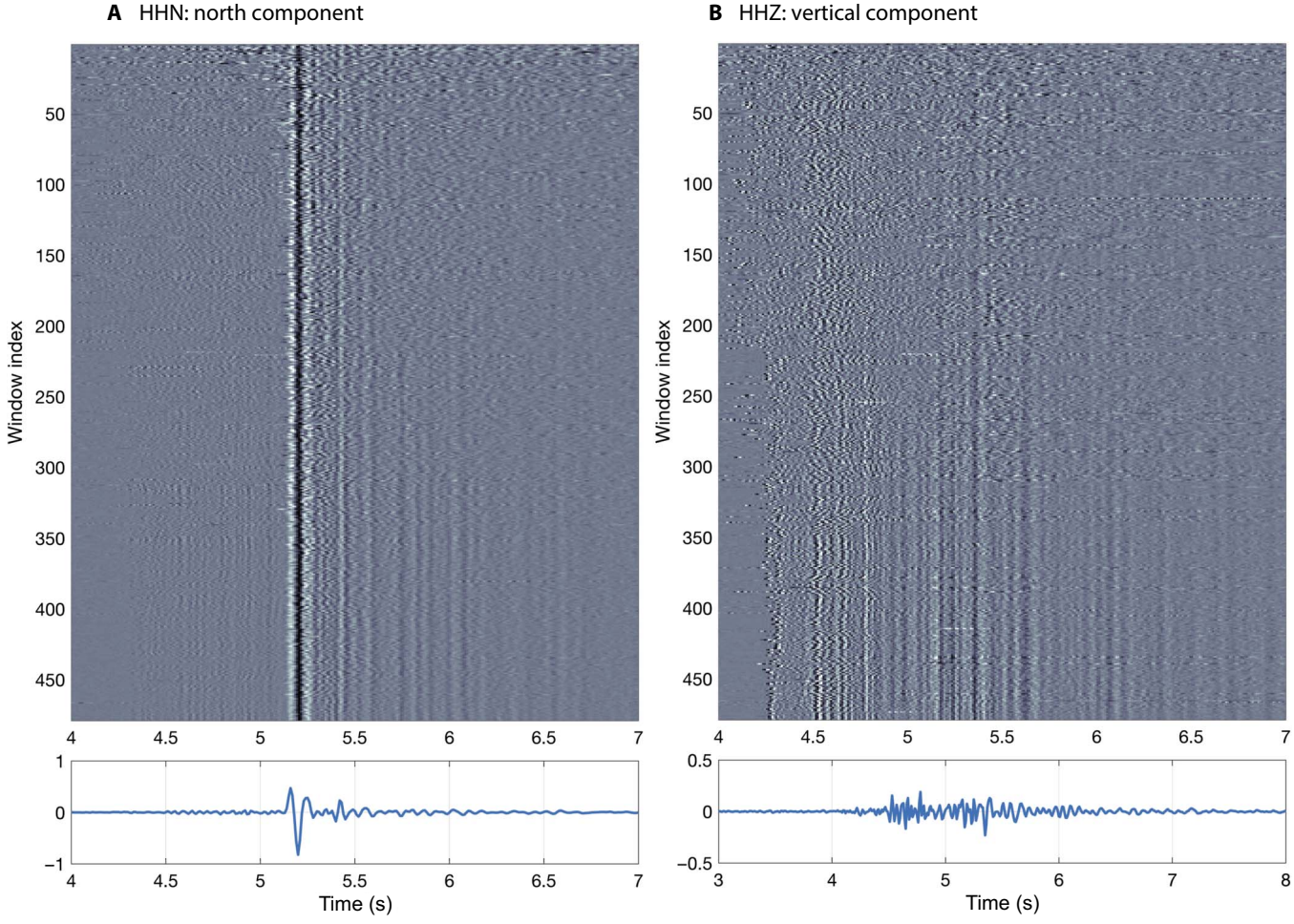

Fig. 5. Event waveforms detected by ConvNetQuake that are similar to an event that occurred on 7 July 2014 at 16:29:11. (**A**) North component and (**B**) vertical component. Top: 479 waveforms organized by increasing absolute correlation coefficient and aligned to the *S*-wave arrival. Waveforms are flipped when anticorrelated with the reference event window. Bottom: Stack of the 479 events.

and they provide a detection but no location. Because FAST generates a new database of template through feature extraction and is more computationally efficient than autocorrelation, it is particularly suited for regions where earthquakes are poorly cataloged.

Yoon *et al.* (*11*) applied FAST to detect new events during 1 week of continuous waveform data recorded at a single station with a single channel from 8 January 2011 to 15 January 2011 in northern California and compared with the autocorrelation method. Twenty-four earthquakes occurred during this period: a M_w 4.1 that occurred on 8 January 2011 on the Calaveras Fault (North California) and 23 of its aftershocks (M_w 0.84 to M_w 4.10, a range similar to our data set). Although the earthquake catalogs and locations are different, the comparison solely focuses on the algorithm performances given by the seismic data. Table 1 reports the classification accuracy of all three methods on the same duration of continuous time series (1 week). The main difference in the data sets is that the reported values of FAST and of the autocorrelation are for single-channel seismograms (ConvNetQuake on three channels), 1 week of continuous time series sampled at 20 Hz (ConvNetQuake uses 100-Hz data). The values of precision and recall are reported.

We further test the performance of our algorithm against a highly optimized template matching algorithm, EQcorrscan (version 0.1.4; https://github.com/eqcorrscan). Given that EQcorrscan requires ~128 GB of memory to detect events using only 100 three-channel, 3-s-long templates sampled at 100 Hz, we limit the exercise to six continuous days of July 2014. Within this period, 10 events are cataloged by OGS, including the two nonrepeating events highlighted in Fig. 1. Using the 2709 templates to train both ConvNetQuake and EQcorrscan, we find that ConvNetQuake correctly detects all of the events, whereas EQcorrscan only finds 4 of 10 events.

Scalability to large data sets

ConvNetQuake is highly scalable and can easily handle large data sets. The runtimes of the autocorrelation method, FAST, and ConvNetQuake necessary to analyze 1 week of continuous waveform data are also reported in Table 1. Our reported runtime excludes the offline training phase. This overhead is performed only once and took 1.5 hour on a NVIDIA Tesla K20Xm graphics processing unit. For the test set, we ran ConvNetQuake on a dual core Intel i5 2.9 GHz central processing unit. Similarly, FAST's runtime reported in Table 1 excludes the time required to build the database of templates (feature extraction) and only includes the similarity search. Although not directly comparable because of the differences in sampling rates and number of channels involved, ConvNetQuake is a fast algorithm. Excluding the training phase, it is approximately 13,500 times faster than autocorrelation and 48 times faster than FAST (Table 1). The runtimes for long time series also indicate that ConvNetQuake presents an almost-linear scaling between runtime and duration of the time series to analyze (similar to FAST). For the 1-month-long continuous time series, ConvNetQuake runtime is 4 min and 51 s, whereas that of FAST is 4 hours and 20 min (see Fig. 6A).

Like other template matching techniques, FAST's database grows as it creates and stores new fingerprints during detection. For 2 days of continuous recording, FAST's database is approximately 1 GB (see Fig. 6B). Methodologies that require growing databases of templates eventually see their performance decreasing with data size. Our network only needs to store a compact set of parameters (the weights of the filters), which entails a constant memory usage (500 kB; see Fig. 6B).

DISCUSSION

ConvNetQuake achieves state-of-the-art performances in probabilistic event detection and location using a single signal. This neural network outperforms other detection methods in computational runtime.

The limitation of the methodology is the size of the training set required for good performances for earthquake detection and location. Data augmentation has enabled great performance for earthquake detection, but larger catalogs of located events are needed to improve the performance of our probabilistic earthquake location approach. This makes the approach ill-suited to areas of low seismicity or areas where instrumentation is recent but well-suited to areas of high seismicity rates and well-instrumented.

The overhead on ConvNetQuake is limited to the training, which is performed once. After appropriate training, ConvNetQuake provides best performances in computational runtimes and memory usage, and comparable detection performances to most established detection methods. Because of its generalization to unseen events and its probabilistic location potential, ConvNetQuake will be well-suited to larger scale data sets and the inclusion of entire seismic networks in the approach. With improved performances in earthquake location, Gaussian mixture models can be used to produce continuous probabilistic location maps. Once deployed, ConvNetQuake can potentially provide very rapid

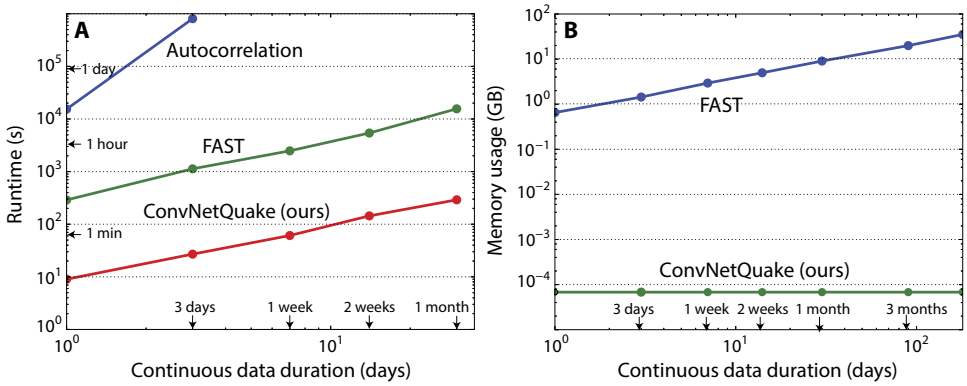

Fig. 6. Scaling properties of ConvNetQuake and other detection methods as a function of continuous data duration. (**A**) Runtime of the three methods where 1.5-hour one-time training is excluded for ConvNetQuake and where FAST's runtimes include the feature extraction (38%) and database generation phases (11%). (**B**) Memory usage of FAST and ConvNetQuake.

earthquake detection and location, which is useful for earthquake early warning. Finally, our approach is ideal to monitor geothermal systems, natural resource reservoirs, volcanoes, and seismically active and well-instrumented plate boundaries such as the subduction zones in Japan or the San Andreas Fault system in California.

METHODS

ConvNetQuake takes a three-channel window of waveform data as input and predicts a discrete probability over M categories or classes in the machine learning terminology. Classes 1 to $M-1$ correspond to predefined geographic "clusters," and class 0 corresponds to event-free "seismic noise." The clusters for our data set are illustrated in Fig. 1. Our algorithm outputs an M-D vector of probabilities that the input window belongs to each of the M classes. Figure 2 illustrates our architecture.

Network architecture

The network's input is a two-dimensional (2D) tensor $Z^0_{c,t}$ representing the waveform data of a fixed-length window. The rows of Z^0 for $c \in \{1, 2, 3\}$ correspond to the channels of the waveform, and because we used 10-s-long windows sampled at 100 Hz, the time index is $t \in \{1, \ldots, 1000\}$. The core of our processing was carried out by a feed-forward stack of eight convolutional layers (Z^1 to Z^8), followed by one fully connected layer z that outputs class scores. All the layers contain multiple channels and are thus represented by 2D tensors. Each channel of the eight convolutional layers was obtained by convolving the channels of the previous layer with a bank of linear 1D filters, summing, adding a bias term, and applying a pointwise nonlinearity, as follows

$$Z^i_{c,t} = \sigma\left(b^i_c + \sum_{c'=1}^{C_i}\sum_{t'=1}^{3} Z^{i-1}_{c',st+t'-1}\cdot W^i_{cc't'}\right) \quad \text{for } i \in \{1, \ldots, 8\} \qquad (1)$$

where $\sigma(\cdot) = \max(0, \cdot)$ is the nonlinear ReLU (rectified linear unit) activation function. The output and input channels are indexed with c and c', respectively, and the time dimension is indexed with t and t'. C_i is the number of channels in layer i. We used 32 channels for layers 1 to 8, whereas the input waveform (layer 0) had three channels. We stored the filter weights for layer i in a 3D tensor W^i with dimensions $C_{i-1} \times C_i \times 3$. That is, we used three-tap filters. The biases were stored in a 1D tensor b^i. All convolutions used zero padding as the boundary condition.

Equation 1 shows that our formulation slightly differs from a standard convolution: We used strided convolutions with stride $s = 2$; that is, the kernel slides horizontally in increments of two samples (instead of one). This allowed us to downsample the data by a factor of 2 along the time axis after each layer. This was equivalent to performing a regular convolution, followed by subsampling with a factor of 2, albeit more efficiently.

Because we used small filters (the kernels have size 3), only the first few layers had a local view of the input signal and could only extract high-frequency features. Through progressive downsampling, the deeper layers had an exponentially increasing receptive field over the input signal (by indirect connections). This allowed them to extract low-frequency features (cf. Fig. 2).

After the eighth layer, we vectorized the tensor Z^8 with shape (*4*, *30*) into a 1D tensor with 128 features $\bar{Z}^8$. This feature vector was processed by a linear, fully connected layer to compute class scores z_c with $c = 0, 1, \ldots, M-1$ given by

$$z_c = \sum_{c'=1}^{128} \bar{Z}^8_{c'} \cdot W^9_{cc'} + b^9_c \qquad (2)$$

Owing to this fully connected layer, the network learned to combine multiple parts of the signal (for example, *P* waves, *S* waves, and seismic coda) to generate a class score and could detect events anywhere within the window.

Finally, we applied the softmax function to the class scores to obtain a properly normalized probability distribution, which could be interpreted as a posterior distribution over the classes conditioned on the input Z^0 and the network parameters $\mathbf{W}$ and $\mathbf{b}$

$$p_c = P(class = c|Z^0, \mathbf{W}, \mathbf{b}) = \frac{\exp(z_c)}{\sum_{k=0}^{M-1}\exp(z_k)} \quad c = \{0, 1, \ldots, M-1\} \qquad (3)$$

$\mathbf{W} = \{W^1, \ldots, W^9\}$ is the set of all the weights, and $\mathbf{b} = \{b^1, \ldots, b^9\}$ is the set of all the biases.

Compared to a fully connected architecture, such as that of Kong *et al.* (*18*) (where each layer would be fully connected as in Eq. 2), convolutional architectures, such as ours, are computationally more efficient. This efficiency gain was achieved by sharing a small set of weights across time indices. For instance, a connection between layers Z^1 and Z^2, which have dimensions of 500×32 and 250×32, respectively, requires $3072 = 32 \times 32 \times 3$ parameters in the convolutional case with a kernel of size 3. A fully connected connection between the same layers would entail $128,000,000 = 500 \times 32 \times 250 \times 32$ parameters, a four orders of magnitude increase.

Furthermore, models with many parameters require large data sets to avoid overfitting. Because labeled data sets for our problem are scarce and costly to assemble, a parsimonious model such as ours is desirable.

Training the network

We optimized the network parameters by minimizing an L_2-regularized cross-entropy loss function on a data set of N windows indexed with k

$$\mathcal{L} = \frac{1}{N}\sum_{k=1}^{N}\sum_{c=0}^{M-1} q^{(k)}_c \log(p^{(k)}_c) + \lambda\sum_{i=1}^{9}\| W^i \|_2^2 \qquad (4)$$

The cross-entropy loss measures the average discrepancy between our predicted distribution $p^{(k)}$ and the true class probability distribution $q^{(k)}$ for all the windows k in the training set. For each window, the true probability distribution $q^{(k)}$ has all of its mass on the window's true class

$$q^{(k)}_c = \begin{cases} 1 \text{ if } \text{class}(k) = c \\ 0 \text{ otherwise} \end{cases} \qquad (5)$$

To regularize the neural network, we added an L_2 penalty on the weights $\boldsymbol{W}$, balanced with the cross-entropy loss via the parameter $\lambda = 10^{-3}$. Regularization favored network configurations with small weight magnitude. This reduced the potential for overfitting (*31*).

Because both the parameter set and the training data set were too large to fit in memory, we minimized Eq. 4 using a batched stochastic gradient descent algorithm. We first randomly shuffled the $N = 702,748$ windows from the data set. We then formed a sequence of batches containing 128 windows each. At each training step, we fed a batch to the network, evaluated the expected loss on the batch, and updated the network parameters accordingly using backpropagation (*13*). We repeatedly cycled through the sequence until the expected loss stopped improving. Because our data set was unbalanced (we had many more noise windows than events), each batch was composed of 64 windows of noise and 64 event windows.

For optimization, we used the ADAM (*32*) algorithm, which kept track of first- and second-order moments of the gradients and was invariant to any diagonal rescaling of the gradients. We used a learning rate of 10^{-4} and kept all other parameters to the default value recommended by the authors. We implemented ConvNetQuake in TensorFlow (*30*) and performed all our trainings on a NVIDIA Tesla K20Xm graphics processing unit. We trained for 32,000 iterations, which took approximately 1.5 hours.

Evaluation on an independent testing set

After training, we tested the accuracy of our network on windows from July 2014 (209 windows of events and 131,972 windows of noise). The class predicted by our algorithm was the one whose posterior probability p_c was the highest. We evaluated our predictions using two metrics. The detection accuracy is the percentage of windows correctly classified as events or noise. The location accuracy is the percentage of windows already classified as events that have the correct cluster number.

SUPPLEMENTARY MATERIALS

Supplementary material for this article is available at http://advances.sciencemag.org/cgi/content/full/4/2/e1700578/DC1

section S1. Generalization ability of ConvNetQuake

section S2. Autocorrelation for detection of new events during July 2014

fig. S1. Waveform of an event recorded with station GS.OK029 and GS.OK027.

fig. S2. Earthquakes in the region of interest (near Guthrie, OK, USA) from 14 February 2014 to 16 November 2016 partitioned into 50 clusters.

fig. S3. Day-long synthetic seismic record constructed by inserting the waveform template shown in fig. S4A at random times into a time series of Gaussian noise.

fig. S4. Templates used to generate synthetic data.

fig. S5. Distribution of the correlation coefficients after autocorrelation of the windows classified as events by ConvNetQuake.

fig. S6. Waveforms of the events detected in cluster 3 using a correlation coefficient threshold of 0.1.

fig. S7. Waveforms of the events detected in cluster 3 using a correlation coefficient threshold of 0.2.

fig. S8. Waveforms of the events detected in cluster 3 using a correlation coefficient threshold of 0.3.

REFERENCES AND NOTES

1. W. L. Ellsworth, Injection-induced earthquakes. *Science* **341**, 1225942 (2013).
2. R. Allen, Automatic phase pickers: Their present use and future prospects. *Bull. Seismol. Soc. Am.* **72**, S225–S242 (1982).
3. Mitchell Withers, Richard Aster, Christopher Young, Judy Beiriger, Mark Harris, Susan Moore, Julian Trujillo, A comparison of select trigger algorithms for automated global seismic phase and event detection. *Bull. Seismol. Soc. Am.* **88**, 95–106 (1998).
4. S. J. Gibbons, F. Ringdal, The detection of low magnitude seismic events using array-based waveform correlation. *Geophys. J. Int.* **165**, 149–166 (2006).
5. R. J. Skoumal, M. R. Brudzinski, B. S. Currie, J. Levy, Optimizing multi-station earthquake template matching through re-examination of the Youngstown, Ohio, sequence. *Earth Planet. Sci. Lett.* **405**, 274–280 (2014).
6. D. R. Shelly, A 15 year catalog of more than 1 million low-frequency earthquakes: Tracking tremor and slip along the deep San Andreas Fault. *J. Geophys. Res.* **122**, 3739–3753 (2017).
7. D. B. Harris, "Subspace detectors: Theory," *Tech. Rep.* (Internal Report UCRL-TR-222758, Lawrence Livermore National Laboratory, 2006).
8. D. B. Harris, D. A. Dodge, An autonomous system for grouping events in a developing aftershock sequence. *Bull. Seismol. Soc. Am.* **101**, 763–774 (2011).
9. S. A. Barrett, G. C. Beroza, An empirical approach to subspace detection. *Seismol. Res. Lett.* **85**, 594–600 (2014).
10. H. M. Benz, N. D. McMahon, R. C. Aster, D. E. McNamara, D. B. Harris, Hundreds of earthquakes per day: The 2014 Guthrie, Oklahoma, earthquake sequence. *Seismol. Res. Lett.* **86**, 1318–1325 (2015).
11. C. E. Yoon, O. O'Reilly, K. J. Bergen, G. C. Beroza, Earthquake detection through computationally efficient similarity search. *Sci. Adv.* **1**, e1501057 (2015).
12. A. Krizhevsky, I. Sutskever, G. E. Hinton, ImageNet classification with deep convolutional neural networks, in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, K. Q. Weinberger, Eds. (Curran Associates Inc., 2012), pp. 1097–1105.
13. Y. LeCun, Y. Bengio, G. Hinton, Deep learning. *Nature* **521**, 436–444 (2015).
14. A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, K. Kavukcuoglu, WaveNet: A generative model for raw audio. https://arxiv.org/abs/1609.03499 (2016).
15. W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, G. Zweig, The Microsoft 2016 conversational speech recognition system. https://arxiv.org/abs/1609.03528 (2016).
16. J. Wang, T.-L. Teng, Artificial neural network-based seismic detector. *Bull. Seismol. Soc. Am.* **85**, 308–319 (1995).
17. J. Wang, T.-l. Teng, Identification and picking of *S* phase using an artificial neural network. *Bull. Seismol. Soc. Am.* **87**, 1140–1149 (1997).
18. Q. Kong, R. M. Allen, L. Schreier, Y.-W. Kwon, MyShake: A smartphone seismic network for earthquake early warning and beyond. *Sci. Adv.* **2**, e1501055 (2016).
19. A. P. Valentine, J. Trampert, Data space reduction, quality assessment and searching of seismograms: Autoencoder networks for waveform data. *Geophys. J. Int.* **189**, 1183–1202 (2012).
20. A. A. Holland, Earthquakes triggered by hydraulic fracturing in south-central Oklahoma. *Bull. Seismol. Soc. Am.* **103**, 1784–1792 (2013).
21. K. M. Keranen, H. M. Savage, G. A. Abers, E. S. Cochran, Potentially induced earthquakes in Oklahoma, USA: Links between wastewater injection and the 2011 Mw 5.7 earthquake sequence. *Geology* **41**, 699–702 (2013).
22. F. R. Walsh III, M. D. Zoback, Oklahoma's recent earthquakes and saltwater disposal. *Sci. Adv.* **1**, e1500195 (2015).
23. M. Weingarten, S. Ge, J. W. Godt, B. A. Bekins, J. L. Rubinstein, High-rate injection is associated with the increase in U.S. mid-continent seismicity. *Science* **348**, 1336–1340 (2015).
24. M. Shirzaei, W. L. Ellsworth, K. F. Tiampo, P. J. González, M. Manga, Surface uplift and time-dependent seismic hazard due to fluid injection in eastern Texas. *Science* **353**, 1416–1419 (2016).
25. J. MacQueen, Some methods for classification and analysis of multivariate observations, in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability* (University of California Press, 1967), vol. 1, pp. 281–297.
26. J. Sietsma, R. J. F. Dow, Creating artificial neural networks that generalize. *Neural Networks* **4**, 67–79 (1991).
27. N. Jaitly, G. E. Hinton, Vocal Tract Length Perturbation (VTLP) improves speech recognition, in *Proceedings of the 30th ICML Workshop on Deep Learning for Audio, Speech and Language (ICML'13)*, Atlanta, GA, 16 to 21 June 2013.
28. X. Cui, V. Goel, B. Kingsbury, Data augmentation for deep neural network acoustic modeling. *IEEE/ACM Trans. Audio Speech Lang. Process.* **23**, 1469–1477 (2015).
29. J. Salamon, J. P. Bello, Deep convolutional neural networks and data augmentation for environmental sound classification. https://arxiv.org/abs/1608.04363 (2016).
30. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems https://arxiv.org/abs/1603.04467 (2015).
31. A. Y. Ng, Feature selection, L1 vs. L2 regularization, and rotational invariance, in *Proceedings of the Twenty-First International Conference on Machine learning (ICML'04)*, Banff, Alberta, Canada, 04 to 08 July 2004.
32. D. P. Kingma, J. Ba, Adam: A method for stochastic optimization. https://arxiv.org/abs/1412.6980 (2014).

Acknowledgments: T.P. thanks J. Rice for the continuous support during Ph.D. and L. Viens for the insightful discussions about seismology. **Funding:** This research was supported by the NSF grant Division for Materials Research 14-20570 to Harvard University, with supplemental support by the Southern California Earthquake Center, funded by NSF cooperative agreement EAR-1033462 and U.S. Geological Survey cooperative agreement G12AC20038. **Author contributions:** T.P. and M.D. designed the project. T.P. implemented the software, performed the training and testing of ConvNetQuake, performed the template matching for comparison, and wrote the manuscript. M.G. helped with the software development and with the manuscript. M.D. performed the autocorrelation to validate the new detections and helped with the manuscript. All authors contributed ideas to the project. **Competing interests:** The authors declare that they have no competing interests. **Data and materials availability:** The ConvNetQuake software is open-source. It can be downloaded at https://github.com/tperol/ConvNetQuake. The waveform data used in this paper can be obtained from the Incorporated Research Institutions for Seismology Data Management Center, and the network GS is available at http://www.fdsn.org/networks/detail/GS/. The earthquake catalog used is provided by the OGS. All data needed to evaluate the conclusions in the paper are present in the paper, in the Supplementary Materials, and/or in the GitHub repository. The computations in this paper were run on the Odyssey cluster supported by the Faculty of Arts and Sciences Division of Science, Research Computing Group at Harvard University.

Submitted 22 February 2017
Accepted 11 January 2018
Published 14 February 2018
10.1126/sciadv.1700578

Citation: T. Perol, M. Gharbi, M. Denolle, Convolutional neural network for earthquake detection and location. *Sci. Adv.* **4**, e1700578 (2018).

Convolutional neural network for earthquake detection and location

Thibaut Perol, Michaël Gharbi and Marine Denolle

Sci Adv **4** (2), e1700578.
DOI: 10.1126/sciadv.1700578