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. In this work, 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 is 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 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). While 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 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 unsuper-
vised earthquake detection method, referred to as FAST, managed to reduce the complexity of
the template matching approach. FAST extracts features, or fingerprints, from seismic wave-
forms, 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 (\textit{ConvNetQuake}) from seis-
mograms. Our algorithm builds on recent advances in deep learning \((12–15)\). Previous stud-
ies have pioneered the use of artificial neural networks to classify seismograms from hand-
engineered features \((16, 17, 19)\) or compressed representations of the waveforms via neural
autoencoders \((18)\). \textit{ConvNetQuake} is trained on a large dataset of labeled raw seismic wave-
forms 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. Thanks to this representation, our algorithm generalizes well to earth-
quake signals never seen during training. It is more accurate than state-of-the-art algorithms and
runs orders of magnitude faster. Additionally, \textit{ConvNetQuake} outputs a probabilistic location
of an earthquake’s source from a \textit{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 dramatic 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 (Oklahoma). In this region, the Oklahoma state 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 since. Signals from both stations are recorded at 100 Hz on 3 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 6 geographic clusters. For this we use the K-Means algorithm (25), with the Euclidean distance between epicenters as the metric. The centroïds of the clusters we obtain define 6 areas on the map (Fig. 1). Any point on the map is assigned to the cluster whose centroïd is the closest (i.e., each point is assigned to its Voronoï cell). We find that 6 clusters allow for a reasonable partition of the major earthquake sequences. Our classification thus contains 7 labels, or classes in the machine learning terminology: class 0 corresponds to seismic noise without any earthquake, 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 3 channels). We extract two types of 10 second long windows from these streams: windows containing events and
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 windows dataset 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) while the training set contains the remaining windows (2,709 event and 700,039 noise windows).

**Dataset augmentation** Deep classifiers like ours have many trainable parameters. While they require a large amount of examples of each class to avoid overfitting, they generalize correctly to unseen examples. To build a large enough dataset of events, we use streams recorded at two stations (GSOK029 and GSOK27, see Figure 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).
Our model is a deep convolutional network (Fig. 2) that takes as input a window of 3-channel waveform seismogram data 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 the Methods section for details).

Detection accuracy In a first experiment to assess the detection performance of our algorithm, we ignore the geographic label (i.e., labels 1–6 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. To summarize, 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. While 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 since, on
average, each class now only provides 40 training samples, which we attribute to insufficient
number of labels for proper training.

**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 Figure 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 approxi-
mately 99 % confidence. For the misclassified event, the probability distribution is more diffuse
and the location confidence drops to 40 %.

**Generalization to non-repeating 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 Figure
S3.

We choose two templates waveforms $T_1$ and $T_2$ (shown in Figure S4). Both waveforms
$T_1$ and $T_2$ exhibit opposite polarity (either due 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 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. ConvNetQuake however 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.

Such generalization of waveform recognition is also highlighted in our test set with real seismic waveforms. Some events in our test dataset from Oklahoma are non-repeating events (see highlighted events in Figure 1). While 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.

**Earthquake detection on continuous records** We run ConvNetQuake on 1 month of continuous waveform data recorded at GS.OK029 in July 2014. The 3-channel waveforms are cut into 10-second long, non-overlapping windows, with a 1 second 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 5 events per hour. Performing autocorrelation confirms 3949 of these are events (see supplementary materials for details). One set of these events that is repeated 479 times (either highly correlated or highly anti-correlated) is shown in Figure 5. For comparison, the subspace detection method ([10]) using 3-channel templates found 5737 events during this period.

**Comparison with other detection methods** We compare our detection performances to autocorrelation and FAST reported from 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, 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 one 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 dataset). While the earthquakes catalogs and locations are different, the comparison solely focuses on the algorithm performances given seismic data. Table 1 reports the classification accuracy of all three methods on the same duration of continuous time series (1 week). The main differences in the data sets is that the reported value of FAST and of the autocorrelation are for a single-channel seismograms (ConvNetQuake on 3 channels), one 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 (https://github.com/eqcorrscan, version 0.1.4). Given that eqcorrscan requires $\sim 128$ Gb of memory to detect events using only 100 3-channel 3 s-long templates sampled at 100 Hz, we limit the exercise to 6 continuous days of July 2014. Within this period, 10 events are cataloged by OGS, including the 2 non-repeating events highlighted in Figure 1. Using the 2709 templates to train both ConvNetQuake and eqcorrscan, we find that ConvNetQuake, correctly detects all of the events, while eqcorrscan only finds 4 out of 10 events.

**Scalability to large datasets** ConvNetQuake is highly scalable and can easily handle large datasets. The runtimes of the autocorrelation method, FAST, and ConvNetQuake necessary to analyze one 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 GPU. For the test set, we ran ConvNetQuake on a dual core Intel i5 2.9 GHz CPU. 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.
Though not directly comparable because of the differences in sampling rates and number of channels involved, ConvNetQuake is clearly 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 one-month long continuous time series, ConvNetQuake run times is 4 minutes 51 seconds while that of FAST is 4 hours 20 minutes (see Fig. 6a).

Like other template matching techniques, FAST’s database grows as it creates and store 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 run time.

The limitations 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 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 as input a 3-channel window of waveform data 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 dataset are illustrated in Fig. 1. Our algorithm outputs a $M$-D vector of probabilities that the input window belongs to each of the $M$ classes. Fig. 2 illustrates our architecture.

Network architecture The network’s input is a 2-D 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 since we use 10 second-long windows sampled at 100 Hz, the time index is $t \in \{1, \ldots, 1000\}$. The core of our processing is carried out by a feed-forward stack of 8 convolutional layers ($Z^1$ to $Z^8$) followed by 1 fully connected layer $z$ that outputs class scores. All the layers contain multiple channels and are thus represented by 2-D tensors. Each channel
of the 8 convolutional layers is obtained by convolving the channels of the previous layer with a bank of linear 1-D filters, summing, adding a bias term, and applying a point-wise non-linearity as follows:

\[
Z_{c,t}^i = \sigma \left( b_{c}^{i} + \sum_{c'=1}^{C_i} \sum_{t'=1}^{3} Z_{c',st+t'-1}^{i-1} \cdot W_{cc}'^i \right) \text{ for } i \in \{1, \ldots, 8\}
\]  

(1)

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

Equation 1 shows that our formulation slightly differs from a standard convolution: we use *strided* convolutions with stride \( s = 2 \), i.e. the kernel slides horizontally in increments of 2 samples (instead of 1). This allows us to downsample the data by a factor 2 along the time axis after each layer. This is equivalent to performing a regular convolution followed by subsampling with a factor 2, albeit more efficiently.

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

After the 8th layer, we vectorize the tensor \( Z^8 \) with shape \((4, 32)\) into a 1-D tensor with 128 features \( \bar{Z}^8 \). This feature vector is processed by a linear, fully connected layer to compute class scores \( z_c \) with \( c = 0, 1, \ldots, M - 1 \) given by:

\[
z_c = \sum_{c'=1}^{128} \bar{Z}^8_{c'} \cdot W_{cc'}^9 + b_c^9
\]

(2)

Thanks to this fully connected layer, the network learns to combine multiple parts of the signal.
(e.g., P-waves, S-waves, seismic coda) to generate a class score and can detect events anywhere within the window.

Finally, we apply the softmax function to the class scores to obtain a properly normalized probability distribution which can be interpreted as a posterior distribution over the classes conditioned on the input $Z^0$ and the network parameters $W$ and $b$:

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

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

Compared to a fully-connected architecture like in Kong et al. (19) (where each layer would be fully connected as in Equation 2), convolutional architectures like ours are computationally more efficient. This efficiency gain is 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 $500 \times 32$ and $250 \times 32$ 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 4 orders of magnitude increase.

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

**Training the network** We optimize the network parameters by minimizing a $L_2$-regularized cross-entropy loss function on a dataset of $N$ windows indexed with $k$:

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

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:

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

To regularize the neural network, we add an $L_2$ penalty on the weights $W$, balanced with the cross-entropy loss via the parameter $\lambda = 10^{-3}$. Regularization favors network configurations with small weight magnitude. This reduces the potential for overfitting (30).

Since both the parameter set and the training data set are too large to fit in memory, we minimize Equation 4 using a batched stochastic gradient descent algorithm. We first randomly shuffle the $N = 702,748$ windows from the dataset. We then form a sequence of batches containing 128 windows each. At each training step we feed a batch to the network, evaluate the expected loss on the batch, and update the network parameters accordingly using backpropagation (13). We repeatedly cycle through the sequence until the expected loss stops improving. Since our dataset is unbalanced (we have many more noise windows than events), each batch is composed of 64 windows of noise and 64 event windows.

For optimization we use the ADAM (31) algorithm, which keeps track of first and second order moments of the gradients, and is invariant to any diagonal rescaling of the gradients. We use a learning rate of $10^{-4}$ and keep all other parameters to the default value recommended by the authors. We implemented ConvNetQuake in TensorFlow (32) and performed all our trainings on a NVIDIA Tesla K20Xm Graphics Processing Unit. We train for 32,000 iterations which takes approximately 1.5 h.

**Evaluation on an independent testing set**  After training, we test 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 is the one whose posterior probability $p_c$ is the highest. We evaluate 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.

Acknowledgments

T.P. thanks Jim Rice for his continuous support during his PhD and Loïc Viens for insightful discussions about seismology. Competing Interests: The authors declare that they have no competing interests. Funding: This research was supported by the National Science Foundation grant Division for Materials Research 14-20570 to Harvard University with supplemental support by the Southern California Earthquake Center (SCEC), funded by NSF cooperative agreement EAR-1033462 and USGS cooperative agreement G12AC20038. Authors contribution: T.P. and M.D designed the projet. T.P. implemented the software, performed the training and test 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. Data and material 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 (IRIS) Data Management Center and the network GS is available at doi:10.7914/SN/GS. The earthquake catalog used is provided by the Oklahoma Geological Survey. 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 FAS Division of Science, Research Computing Group at Harvard University. Competing interests: The authors declare that they have no competing interests.
References and Notes

1. W. L. Ellsworth, *Science* **341** (2013).

2. R. Allen, *Bulletin of the Seismological Society of America* **72**, S225 (1982).

3. M. Withers, *et al.*, *Bulletin of the Seismological Society of America* **88**, 95 (1998).

4. S. J. Gibbons, F. Ringdal, *Geophysical Journal International* **165**, 149 (2006).

5. R. J. Skoumal, M. R. Brudzinski, B. S. Currie, J. Levy, *Earth and Planetary Science Letters* **405**, 274 (2014).

6. D. R. Shelly, *Journal of Geophysical Research* **122**, 3739 (2017).

7. D. Harris, Subspace detectors: Theory, lawrence livermore national laboratory, *Tech. rep.*, internal report, UCRL-TR-222758 (2006).

8. D. Harris, D. Dodge, *Bulletin of the Seismological Society of America* **101**, 763 (2011).

9. S. A. Barrett, G. C. Beroza, *Seismological Research Letters* **85**, 594 (2014).

10. H. M. Benz, N. D. McMahon, R. C. Aster, D. E. McNamara, D. B. Harris, *Seismological Research Letters* **86**, 1318 (2015).

11. C. E. Yoon, O. O'Reilly, K. J. Bergen, G. C. Beroza, *Science advances* **1**, e1501057 (2015).

12. A. Krizhevsky, I. Sutskever, G. E. Hinton, *Advances in neural information processing systems* (2012), pp. 1097–1105.

13. Y. LeCun, Y. Bengio, G. Hinton, *Nature* **521**, 436 (2015).

14. A. van den Oord, *et al.*, *arXiv preprint arXiv:1609.03499* (2016).
15. W. Xiong, et al., *arXiv preprint arXiv:1609.03528* (2016).

16. J. Wang, T.-L. Teng, *Bulletin of the Seismological Society of America* **85**, 308 (1995).

17. J. Wang, T.-I. Teng, *Bulletin of the Seismological Society of America* **87**, 1140 (1997).

18. A.P. Valentine, J. Trampert, *Geophysical Journal International* **189**, 2 (2012).

19. Q. Kong, R. M. Allen, L. Schreier, Y.-W. Kwon, *Science advances* **2**, e1501055 (2016).

20. A. A. Holland, *Bulletin of the Seismological Society of America* **103**, 1784 (2013).

21. K. M. Keranen, H. M. Savage, G. A. Abers, E. S. Cochran, *Geology* **41**, 699 (2013).

22. F. R. Walsh, M. D. Zoback, *Science advances* **1**, e1500195 (2015).

23. M. Weingarten, S. Ge, J. W. Godt, B. A. Bekins, J. L. Rubinstein, *Science* **348**, 1336 (2015).

24. M. Shirzaei, W. L. Ellsworth, K. F. Tiampo, P. J. González, M. Manga, *Science* **353**, 1416 (2016).

25. J. MacQueen, et al., *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability* (Oakland, CA, USA., 1967), vol. 1, pp. 281–297.

26. J. Sietsma, R. J. Dow, *Neural networks* **4**, 67 (1991).

27. N. Jaitly, G. E. Hinton, *Proc. ICML Workshop on Deep Learning for Audio, Speech and Language* (2013).

28. X. Cui, V. Goel, B. Kingsbury, *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)* **23**, 1469 (2015).

29. J. Salamon, J. P. Bello, *arXiv preprint arXiv:1608.04363* (2016).
30. A. Y. Ng, *Proceedings of the twenty-first international conference on Machine learning* (ACM, 2004), p. 78.

31. D. P. Kingma, J. Ba, *CoRR abs/1412.6980* (2014).

32. M. Abadi, *et al.*, TensorFlow: Large-scale machine learning on heterogeneous systems (2015). Software available from tensorflow.org.

33. D. E. McNamara, *et al.*, *Geophysical Research Letters* 42, 2742 (2015).
Table 1: **Performances of three detection methods**, excluding the overhead runtimes (1.5 hour of offline training for ConvNetQuake and 47 minutes of feature extraction and database generation for FAST). Autocorrelation and FAST results are as reported from *Yoon et al.* (11). The computational runtimes are for the analysis of one week of continuous waveform data.

![Earthquakes and seismic station in the region of interest](image1.png) (near Guthrie, OK) from 14 February 2014 to 16 November 2016. GS.0K029 and GS.0K027 are the two stations that record continuously the ground motion velocity. The colored circles are the events in the training dataset. Each event is labeled with its corresponding area. The thick black lines delimit the 6 areas. The black squares are the events in the test dataset. Two events from the test set are highlighted because they do not belong to the same earthquake sequences, they are non-repeating events.

![ConvNetQuake architecture](image2.png). The input is a waveform of 1000 samples on 3 channels. Each convolutional layer consists in 32 filters that downsample the data by a factor 2, see Equation 1. After the 8th convolution, the features are flattened into a 1-D vector of 128 features. A fully connected layer ouputs the class scores, see Equation 2.

![Probabilistic location map of two events](image3.png). (a) The event is correctly located, the maximum of the probability distribution corresponds to the area in which the event is located. (b) The event is not located correctly, the maximum of the probability distribution corresponds to a area different from the true location of the event.

![Detection accuracy between ConvNetQuake and template matching](image4.png). 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 signal to noise ratio. 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 $T_2$. 

|                      | Autocorrelation | FAST | ConvNetQuake (This study) |
|----------------------|-----------------|------|---------------------------|
| Precision            | 100 %           | 88.1 % | 94.8 %                   |
| Recall               | 77.5 %          | 80.1 % | 100 %                    |
| Event location accuracy | N/A            | N/A  | 74.6 %                    |
| Reported runtime     | 9 days 13 hours | 48 min | 1 min 1 sec              |
Figure 5: **Event waveforms detected by ConvNetQuake that are similar to an event that occurred on July 7 2014 at 16:29:11** (a) North component and (b) Vertical component. Top panels are the 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 panels are the stack of the 479 events.

Figure 6: **Scaling properties of ConvNetQuake and other detection methods as a function of continuous data duration.** (a) run time of the three methods where 1.5 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.