Replies to critical reviews

Critiques by group 24.

From an overall point of view, the presentation is organized. It explains each part of the requirements in detail. Provided detailed background information and corresponding data, it can be seen that they did a good investigation before project begin. A detailed explanation of the applied Neural network was also made, and the application of the Neural network in earthquake detection was also mentioned before. The accuracy and confusion matrix are given in the result demonstration, and the optimization method is explained, which is great; because it allows people to see the changes clearly.

Some Improvement/ Unclear:

● It is recommended to indicate on the ppt slides or tell the topics when explaining, such as background, literature survey, so that it will be clear to the audience to understand the presentation.

Thanks. That is a good suggestion.

● Do you need to quote the figure on page 9?

Yes, we need to quote the figure. We did cite it in the report.

● The conclusion part on the last two pages of the ppt slides is a good idea, but it is not mentioned in the presentation. Do you need to mention it? Or you can put it before the reference page?
Due to the limited time, we cannot talk about it.

● Is it possible to show the results given in the slides when you do the code demonstration? It may complete the whole progress.

Yes. We demonstrated the final training result in the end. But due to the training time required is too long. We can only show some initial result during demonstration.

Critiques by group 28

● I'm interested to hear how and the context of picking this region for analysis. I understand it is a big contributor to seismic events in middle America, but why is that an important topic to research as opposed to other natural events? Are you concerned about an increase of activity in the area to rise to that similar to say, California?

All nature events are worth study scientifically. But the seismicity is very important to study since it could directly cause casualities and cause damage to infrastructure. The seismicity is high in California as well. The method is also applicable to California of course.

● More explanation of detection algorithms would be useful, even a simple sentence of what they entail and not just their issues.

In ConvNetQuake, the model can predict each window's label either as seismic noise or event with its goemetrical cluster. So the event/noise detection accuracy is defined as the number of event/noise detected correctly divided by the number of total windows. The localization accuracy is the percentage of windows already classified as events that have the correct geometrical cluster label.

● Is your data normalized by seismometer or site or globally? It may be interesting to use a clustering algorithm to separate data into further regions - or try additional clustering algorithms outside of kmeans.

The data is normalized for each channel. There are three channels corresponding to 3 different orientations. Indeed other clustering algorithms can be used. This will probably be our future works.

● How and why did you pick CovNetQuake over the other ML algorithms mentioned in the introduction? Which specific detection algorithm does it use?

There are several other ML algorithms. The ConvNetQuake developed by Perol is one of the earliest and it is open source program. That is why we picked it. The detection algorithm is based on CNN network. The training events were reported in Oklahoma Geological Survey.

● Did you try any other activation or loss functions?

Yes, we have tried activation function relu, tanh and leaky_relu.

● Your results are great so far! Your detection accuracy is much higher than the other models you mentioned. Confusion matrix results are showing obvious good work.

Thanks!

● I wonder if (lower) location accuracy results are due to the nature of seismic activities with residual disruptions spread far from the location site of the event.

It is possible. But the time period of testing data set is located between the time period of training data. So lower location accuracy is probably due to data we used for training was not enough.

● It looks like you tuned the hyperparameters for the specific dataset in CovNetQuake, did you change any additional parts of the original code?

Yes, the original code is for the version python2, tensorflow1. We updated the code to be python3 and tensorflow2 compatible.

● How many steps does it take to increase the detection accuracy from ~50% in the demo to the end result of >90%?

About 40k for detection and 160k for localization

● Great results so far but it is unclear how much code modification to the open source repo was performed.

We have modified the code to make it compatible to python 3 and TensorFlow 2. We have also tested adjusting several hyper parameters such as number of layers in order to obtain a better training result.

Critiques by group 27

The motivation was clear. Existing methods, literature review and available data were described well, though in the report making links between the previous literature and models a little more clear could help understand the current state of the field. The ConvNet Architecture Diagram is really nice and is a good visual for the architecture used. Results were explained thoroughly.

● It would be interesting to see how similar their methods were to methods from the literature review. The data is coming from an Oklahoma seismic detection location but it was not clear how much data was available to work with. It would have been useful to know how much data was available and how much was used in this project (specific quantification).

We used the same data set as Perol et al.(2018). It includes three channels seismograms from 14 February 2014 to 16 November 2016. There are more data available up to date but it is not used since there is no comparison.

● Based on your confusion matrix it seems that all events were classified as noise but still had good accuracy because there is an imbalance in the instances of events vs. noise. Have you tried to weight your samples when training to avoid this potential issue?

206 out of 209 event windows are correctly classified as event. We have used 6 months' noise data and 33 months' event data to train the model. It is a great idea to increase the instances of events, but the data is not enough.

● We were slightly confused by the results given by the confusion matrix and the following slides which give accuracy for event detection. Some results showed an accuracy around 90% for event detection but from the confusion matrix it seemed as though there were no correct detections.

Two validate sets are used. 209 event windows in July 2014 and 131072 noise windows in July 2014 are separately input into the network. The inputs of the first matrix are all event windows while the inputs for the second matrix are all noise windows. The event detection accuracy in the first matrix is 206/209.

● We would suggest making it clear how your approach differs from ConvNetQuake - e.g. new/different data, which hyperparameters were tested and tuned, etc. It was hard to know from presentation exactly what your group did to build on current ConvNetQuake methods. Cool work and love the application you chose!

The data we used are the same with the paper. The hyperparamters we tested include acitvationfunction(tanh, leaky relu and relu), the number of convolutional layer(8,9,12), learning rate(1e-3,1e-4,1e-5) and batch size(16,32,64,256).

GROUP NUMBER 34
MACHINE LEARNING FOR EARTHQUAKE DETECTION IN OKLAHOMA

Xinyu Luo[⋆]*, Xiaoche Wang*[⋆]*, and I-Lin Yeh*[⋆]

[⋆]University of California San Diego, La Jolla, CA 92093-0238

ABSTRACT

The seismicity in Oklahoma region was low prior to 2009. But recent wastewater injection has significantly increased the number of earthquakes in Oklahoma. A complete earthquake catalog is important to understand the potential risk in the region. In this paper, We follow the method proposed by Perol et al.(2018) and use convolutional nueral network methods to detect small earthquakes in Oklahoma. The network consists of 9 layers. We tested the hyper parameter to obtain a better result of the test. The result shows improvement of training speed when using a different activation function. We also found that when introduction more noise record, the noise accuracy is higher.

***Index Terms*—** Seismicity, Earthquake detection, Convolutional Neural Network

1. INTRODUCTION

The seismicity in the Oklahoma region was low prior to 2009. But recent wastewater injection has significantly increased the number of earthquakes in Oklahoma. A complete earthquake catalog is important to understand the potential risk in the region. In this paper, We follow the method proposed by Perol et al.(2018) and use convolutional neural network methods to detect small earthquakes in Oklahoma. The network consists of 9 layers. We tested the hyper parameter to try to obtain a result of higher accurary. The result shows improvement of training speed when using a different activation function. We also found that when introducing more noise records, the noise accuracy is higher.

Here we follow the Perol et. al.(2018) [1] and uses ConvNetQuake program [2] to perform earthquake detection and location in the Oklahoma region. The input to the program is a three channel seismogram of 10s time window(100Hz). We then use a convolutional neural network to output a predicted result (earthquake or not). In the case of an earthquake, the location area code will be given. The program uses labeled seismic records to perform supervised machine learning. Our raw data is a three-channel continuous ground velocity waveform recorded at seismic stations GS.OK027 and GS.OK029 in Oklahoma from 15 February 2014 to 16 November 2016. A 9 layers convolutional network network is used to detect and locate the event. The outputs of the network are classes -1,0-5. The class 0-5 denotes the earthquake source of different regions and class -1 denotes the input is not earthquake signal. In our project we first try to reproduce the result in the Perl et al.(2018) and we next adjust the hyper parameters such as activation function, number of network layers, batch size and training rate in order to obtain a higher accuracy.

2. RELATED RESEARCHES

Traditional method STA/LTA method is based on the ratio of short term to long term energy density (Allen, 1978)[3]. But the shortcoming of this method is that it is hard for it to detect small earthquake events. Because earthquakes usually happen on pre-existing faults, they have similar waveforms. Conventional template matching method uses waveform similarity to detect earthquakes.(Li et al., 2018)[4] It is capable of detection of very small events. However, the issue of this method is that it is computationally expensive and time consuming.

Therefore, recently, machine learning method has been applied to earthquake and non-conventional seismic source detection (Hilbert et al.[5], 2019; Perol et al., 2018[1]; [6], 2020; Wu et al., 2018[7]; Ross et al., 2018[8]; Zhu and Beroza, 2018[9]). Convolutional neural network(CNN) has sparse interactions feature, which permits faster processing. Perol(2018) reported that the ConvNetQuake only uses 1 minute to achieve a better detection accuracy than conventional template matching method. Ross et al. used 4 layer convolutional and 2 fully connected layers. The model was able to detect new events with different waveforms. Wu et al. used 7 fully connected layers. It can detect various sizes of earthquakes in Laboratory. Mousavi et al. used one very-deep encoder and three separate encoders. It can be used to both detect earthquakes and pick the P and S wave arrival. The model detected two times more earthquakes during the 2000 Tottori earthquake in Japan. Since those neural networks focus on different areas and have different data processing procedures, it is difficult to directly compare their results.

3. DATASET

We use the three-channel monthly ground velocity waveform from seismic stations GS.OK027 and GS.OK029 in Okla-

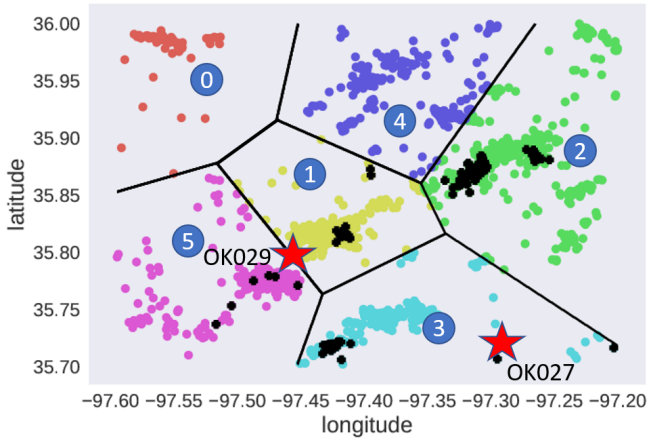

Fig. 1. Geometrical clusters and the earthquake event locations. The red stars are the seismic station GS.OK027 and GS.OK029. Each earthquake event is assigned into its geometrical cluster labeled from 0 to 5. The black dots are the earthquake events from July 2014 for the testing data. Other dots with 6 different colors are the earthquake events from February 2014 to November 2016 for the training data.

homa. The three channel corresponds to three orientation: vertical, north-south, and west-east. The monthly stream dataset can be found in the ConvNetQuake Github page [2]. The 2021 seismic events, i.e, earthquakes, from February 2014 to November 2016 are cataloged by Oklahoma Geological Survey (OGS).

The 2021 seismic events (earthquakes) are divided into 6 geometrical clusters using K-means algorithm. As shown in Fig.1, each (earthquake) event is assigned into a geometrical cluster. There are 7 classes or labels in total: Class -1 being seismic noise without earthquakes and class 0-5 being (earthquake) events with its geometrical cluster label.

For the preprocessing, we normalize the monthly velocity waveform for each channel independently. Using the OGS catalog, we extract the 10-second long event windows labeled with its geometrical cluster from 0 to 5. For the seismic noise windows, we use the catalog created by Benz et al. [10] to create the 10-second long noise windows labeled with -1.

We then split all the event and noise windows into training and testing sets. The testing dataset contains 209 event and 131072 noise windows only from July 2014. The training dataset contains 2709 event windows from the remaining months from February 2014 to November 2016 and 700039 noise windows from February to August except July in 2014. We will train the model and use the model to predict the probability of each class for the input window. Then the class with the highest probability will be the input window's predicted class either as seismic noise or event with its geometrical cluster.

4. NEURAL NETWORK ARCHITECTURE

We use ConvNetQuake [2], an open-source CNN code in tensorflow, for earthquake detection and location. ConvNetQuake takes 3-channel velocity waveform data as input and outputs the probability over 7 classes. Class -1 is the seismic noise and class 0-5 correspond to the earthquake event with geometrical clusters as defined in previous section.

The input of the neural network is the 3-channel waveform over 10-second window. Since the data is sampled at 100 Hz, there are 1000 samples per channel. The network structure contains eight convolutional layers and one fully connected layers to output the score of each class (see Fig. 2).

In each convolutional layer, we apply a one-dimensional filter with bias and nonlinear ReLU (rectified linear unit) activation function. The number of output filter in the convolution is 32. The kernel size 3. We use the convolution with stride equal to 2. So the feature size becomes half after each convolutional layer. The input has 1000 features per channel. After 8 convolutional layers, we have 32 channels with 4 features per channel.

After eight convolutional layers, we flatten it into a 1D array with 128 features. Then we use a fully connected layer to transform it from 128 features to 7 features, which give the scores for each of the 7 classes.

The last step in the network is to apply the softmax function to obtain the probability over 7 classes. The softmax operation is given as

$$p_c = \frac{\exp(z_c)}{\sum_{i=-1}^{5} \exp(z_i)} \quad c = \{-1, 0, ..., 5\}, \tag{1}$$

where p_c is the probability of class c and z is the tensor after the fully connected layer.

In the network, we do the training by minimizing the cross-entropy loss function with L2 regularization. The loss function is defined as

$$L = \frac{1}{N} \sum_{k=1}^{N} \sum_{c=-1}^{5} q_c^{(k)} \log(p_c^{(k)}) + \lambda \sum_{i=1}^{9} ||W^i||_2^2, \tag{2}$$

where N is the number of training windows, i.e., 2709 event + 700039 noise windows. And $q_c^{(k)}$ is the one-hot encoding, which is equal to 1 if class(k)=c and equal to 0 otherwise. W^i is the weighting matrix for ith layer. There are 9 layers in total, so W^i is summed to 9. The L2 regularization with parameter $\lambda = 10^{-3}$ is added to prevent overfitting.

We minimize the loss function using the batched stochastic gradient descent. The default batch size is 64, which means each batch contains 64 windows of noise and 64 windows of events. In each training step, we feed the batch as input and do the backward propagation to get the gradient, then update the weighting function with learning rate of 10^{-4}. The optimization method is ADAM.

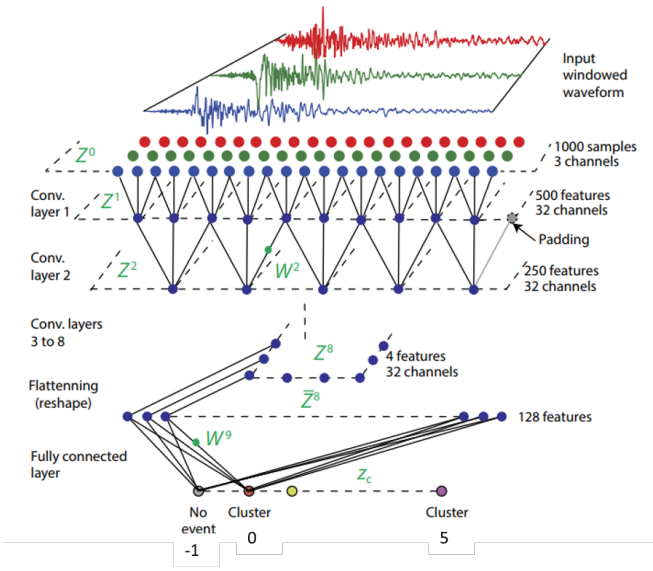

Fig. 2. The architecture of ConvNetQuake. [1]

5. RESULTS

The primary metrics to evaluate over results is accuracy, the percentage of windows correctly classified as events or noise.

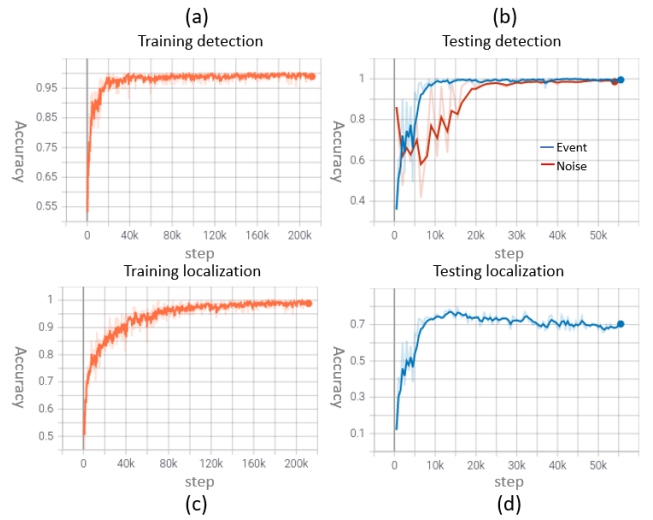

Fig. 3. Accuracy in training process (a) Training detection accuracy (b) Testing detection accuracy (c) Training localization accuracy (noise in red, event in blue) (d) Testing localization accuracy

Figure 3a and 3c shows the training accuracy in the training process, where the learning rate is set to 1e-4, the batch size is set to 64, the layer numbers is set to 9 and the activation function is set to 'relu'. 210k steps are trained in one hour 11 minutes. It takes approximately 20 minutes for the detection accuracy to reach 99%, while it takes 60 minutes for the localization accuracy to reach 99%. The training time for localization is far longer than detection.

Figure 3b and 3d shows the testing accuracy in the training process. Both of the event detection accuracy and the noise detection accuracy gradually get close to 1 and saturate after 40k steps, which is a sign of a perfect training. However the localization accuracy reaches its peak(0.78) at step 15k, then decreases with time, indicating the appearance of overfitting. To avoid overfitting and improve the performance, we suggest add dropouts and increase the training dataset.

Two testing sets are used. 209 event windows in July 2014 and 131072 noise windows in July 2014 are separately input into the network.

Actual\predicted	-1	0	1	2	3	4	5
-1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	0	0	37	0	1	0	2
2	3	1	0	78	7	27	3
3	0	2	0	9	24	3	0
4	0	0	0	0	0	0	0
5	0	0	3	0	1	0	8

(a)

Actual\predicted	Noise	Events
Noise	130173	899
Events	0	0

(b)

Fig. 4. Confusion matrices (a) Input is 209 event windows in July 2014. (b) Input is 130173 noise windows in July 2014

The confusion matrix shown in figure 4a is the result of the 209 input event windows, where label -1 corresponds to the noise and label 0 to 5 correspond to the six geographic clusters. As shown in the matrix, 206 out of 209 events are correctly detected, giving an event detection accuracy of 98.56%. And 147 event windows are correctly located, giving a localization accuracy of 70.33%. This value has great potential to be improved, as an overfitting might occur in the training process.

The confusion matrix shown in figure 4b is the result of the input noise windows. Our model correctly classifies 130173 noise windows and misclassifies 899 of the noise windows as events. Thus the noise detection accuracy is 99.31%. The 'misclassified' noise windows are labeled as noise in the catalog, but some of them can be confirmed as events by the autocorrection method. Since we did not try

this, we cannot provide the precision and the recall.

To optimize our model, we tested different learning rates, batch sizes, number of layers and activation functions.

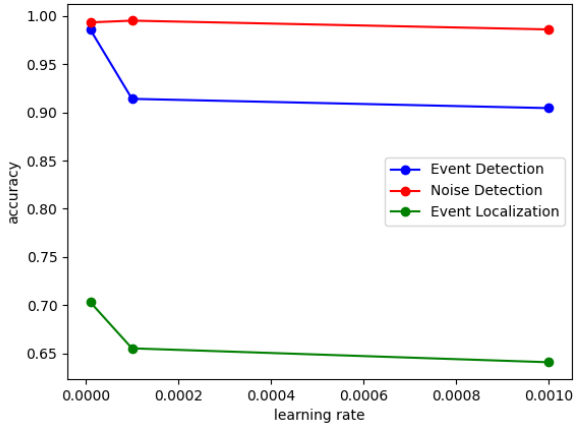

Fig. 5. Event detection accuracy(blue), noise detection accuracy(red) and localization accuracy(green) versus learning rate

Three different learning rates are tested: 1e-3, 1e-4 and 1e-5. Learning rate is the step size of gradient descent. A large learning rate may cause the learning jump over the minima. Figure 3 shows our test result. The event detection accuracy and the localization accuracy increases with the decreasing learning rate as expected. Especially when the learning rate changes from 1e-4 to 1e-5, the accuracies increase by 10%. However, the training time also dramatically increases from one hour to nine hours, since it takes longer to converge. Our learning rate is set to 1e-4 considering both the accuracy and the performance.

Batch sizes we tested are 16, 32, 64, 128 and 256, as shown in figure 4. When tuning the batch size, the accuracy does not have a clear trend like learning rate. The event detection accuracy gets smaller when the batch is small, while the localization accuracy gets larger. To balance this, we choose to use 64 as our batch size. Besides, the network trains faster with small batches, since the weights are updated after each propagation.

Finally we tested numbers of layers and activation functions relu, tanh and leaky_relu on another device, as shown in figure 7. With the same 9 layers, function relu and tanh takes 100k steps to reach 99% training accuracy while function leaky_relu took 150k steps. And all the accuracies of function relu stand out from others. So activation function relu is the best choice. Next, three convolutional layers are added to the initial nine-layer model. Comparing 9 layers and 12 layers, it is obvious that the localization accuracy decreased 50% even though the learning rate of the later one is 10 times smaller than the previous one. We can conclude

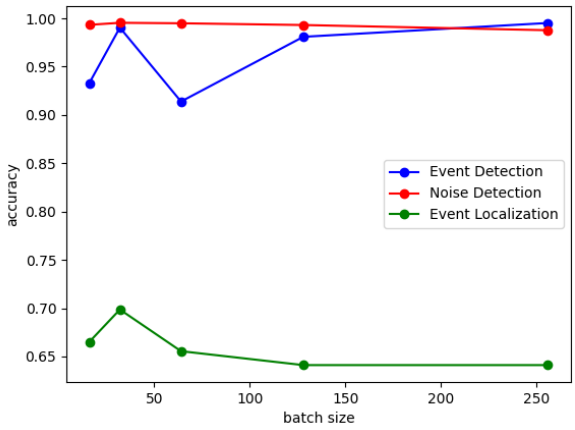

Fig. 6. Event detection accuracy(blue), noise detection accuracy(red) and localization accuracy(green) versus batch size

Layers	Steps	Learning rate	Activation functions	Events detection accuracy	Noise detection accuracy	Localization accuracy
12	350k	1e-5	tanh	0.90	0.88	0.27
9	240k	1e-4	tanh	0.86	0.78	0.50
9	160k	1e-4	Leaky_relu	0.79	0.58	0.11
9	500k	1e-4	relu	0.96	0.84	0.52

Fig. 7. Accuracy and training steps of tests of different numbers of layers and activation functions

that adding layers will cause the decrease of the localization accuracy.

6. CONCLUSION

In this project, We used the method proposed by Perol et al.(2018) and used convolutional neural network method to detect small earthquakes in Oklahoma. We tested the hyper parameter to obtain a better result of the test. The result shows improvement of training speed when using a different activation function. We also found that when introduction more noise data, the noise accuracy is higher. When increase the batch size, the detection accuracy gets larger and localization accuracy gets smaller. A smaller learning rate results in a higher accuracy but the training time is drastically increased. Our results haven't surpass the result by Perol. To increase the detection accuracy, we could try to augment the event data and this could potentially increase the training accuracy. It is also possible to further increase the accuracy by decreasing the learning rate.

We would like to thank the advises from Professor Peter Gerstoft and assistant Venkatesh Sathyanarayanan.

7. CONTRIBUTIONS

- Xinyu tested the number of layers and activation function and wrote Abstract, section 1, 2 and 6. Xinyu also modified the code to be tensorflow2 and python 3 compatible.
- Xiaoche tested the learning rate and wrote section 5. Xiaoche also modified the code to be tensorflow2 and python 3 compatible.
- I-Lin tested the batch size and wrote section 3 and 4. I-Lin ran the code in the original version in tensorflow 1 and python 2.

8. REFERENCES

[1] Thibaut Perol, Michaël Gharbi, and Marine Denolle. Convolutional neural network for earthquake detection and location. *Science Advances*, 4(2):e1700578, 2018.

[2] Convnetquake. https://github.com/tperol/ConvNetQuake.

[3] Rex V Allen. Automatic earthquake recognition and timing from single traces. *Bulletin of the Seismological Society of America*, 68(5):1521–1532, 1978.

[4] Zefeng Li and Zhongwen Zhan. Pushing the limit of earthquake detection with distributed acoustic sensing and template matching: a case study at the brady geothermal field. *Geophysical Journal International*, 215(3):1583–1593, 2018.

[5] C Hibert, D Michéa, F Provost, JP Malet, and M Geertsema. Exploration of continuous seismic recordings with a machine learning approach to document 20 yr of landslide activity in alaska. *Geophysical Journal International*, 219(2):1138–1147, 2019.

[6] S Mostafa Mousavi, William L Ellsworth, Weiqiang Zhu, Lindsay Y Chuang, and Gregory C Beroza. Earthquake transformer—an attentive deep-learning model for simultaneous earthquake detection and phase picking. *Nature communications*, 11(1):1–12, 2020.

[7] Yue Wu, Youzuo Lin, Zheng Zhou, David Chas Bolton, Ji Liu, and Paul Johnson. Deepdetect: A cascaded region-based densely connected network for seismic event detection. *IEEE Transactions on Geoscience and Remote Sensing*, 57(1):62–75, 2018.

[8] Zachary E Ross, Men-Andrin Meier, Egill Hauksson, and Thomas H Heaton. Generalized seismic phase detection with deep learning. *Bulletin of the Seismological Society of America*, 108(5A):2894–2901, 2018.

[9] Weiqiang Zhu and Gregory C Beroza. Phasenet: a deep-neural-network-based seismic arrival-time picking method. *Geophysical Journal International*, 216(1):261–273, 2019.

[10] Harley M Benz, Nicole D McMahon, Richard C Aster, Daniel E McNamara, and David B Harris. Hundreds of earthquakes per day: The 2014 guthrie, oklahoma, earthquake sequence. *Seismological Research Letters*, 86(5):1318–1325, 2015.