Microseismic event or noise: Automatic classification with convolutional neural networks

Benjamin Consolvo, Michael Thornton, MicroSeismic, Inc*

Summary

Microseismic monitoring is a crucial element to understanding hydraulic fracturing operations prior to oil and gas production. One of the more tedious quality control (QC) measures that must often be performed following a microseismic processing workflow is a visual inspection of seismic data to determine whether the data contain microseismic events or only noise. The manual nature of these inspections can take many weeks, sometimes over a month, to perform for one geophysicist. Automated approaches usually use a short-term-average long-term-average (STA/LTA) ratio, but end up picking false positives on noisy data. We propose using a supervised deep learning algorithm, a convolutional neural network (CNN), to automatically classify microseismic events from noise. Using our deep learning approach, we show that the time for QC can be reduced from weeks to hours with high accuracy.

Introduction

Microseismic events can be identified visually (after some processing) on a seismic image as a high amplitude edge over background noise (top of Figure 1a-c). A panel of noise has no coherent edge structure (top of Figure 1d-f). The bottom portion of Figure 1 shows a heat map of the final gradient from the CNN overlaid on the seismic image. The classification of "event" or "noise" images makes it a suitable candidate for machine learning (ML). Some have proposed to predict the presence of certain features in seismic data using attribute-based ML (Shin et al., 2005; Qu et al., 2018). However, attribute-based studies can be very sensitive to noise, producing both false lows and false highs.

Another approach that has historically been used to identify microseismic events and earthquakes is a STA/LTA algorithm (Allen, 1978; Ruud and Husebye, 1992; Vaezi and Van der Baan, 2015; Kalkan, 2016). However, the STA/LTA algorithm often triggers on background noise, which varies during hydraulic fracturing operations (Song et al., 2010; Chen, 2018). The detection of STA/LTA depends on predetermined threshold parameters and will often pick up on noise spikes, rather than microseismic events.

With the rise in available computational power, deep learning and CNNs have proven to be both a very powerful and practical tool to solve computer vision problems, including classification (Krizhevsky et al., 2012), segmentation (Ronneberger et al., 2015), and object detection (Ren et al., 2017). One prolific example of the successes of CNN in computer vision is Facebook's *DeepFace* facial recognition technology (Taigman et al., 2014). The years 2012 onward brought an increasing number of convolutional layers in CNNs, from 8 layers in AlexNet (Krizhevsky et al., 2012) to 152 layers in ResNet (He et al., 2016).

In a limited sense, neural networks attempt to imitate the brain in how the brain processes, stores, learns, and retrieves information. Consider the analogy of an infant who processes new images and learns or "trains" on images to be able to make classifications in the future. In a similar way, a

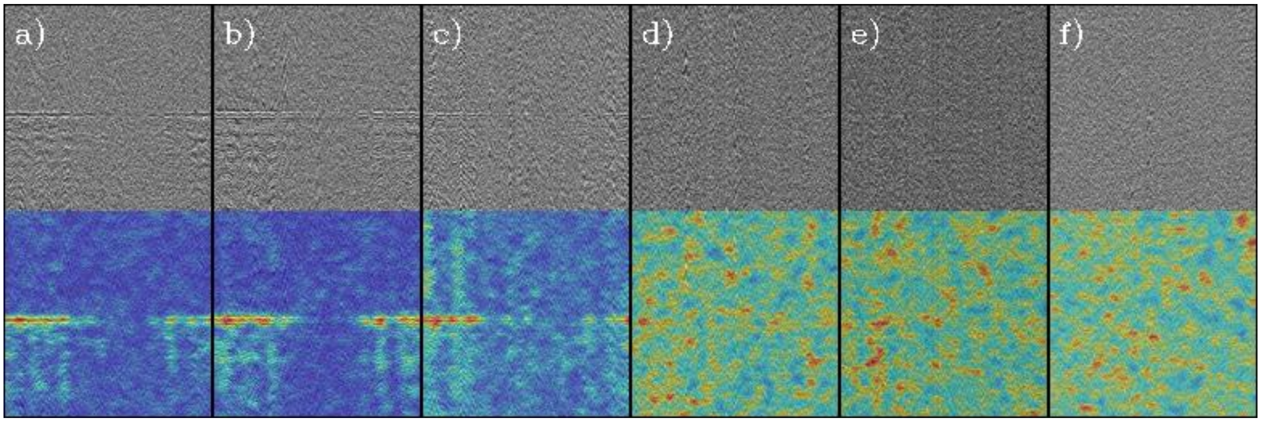

Figure 1: Panels a) – c) are input data labelled as "event", and panels d) – f) are input data labelled as "noise". The top portion of each panel is an example of seismic data for training, and the bottom half of each panel is the same seismic data with a heatmap of the average gradient from the convolutional neural network overlaid. Warmer colors (red) show a higher weighting of the gradient is placed in these pixel regions during training, and colder colors (blue) indicate a lower weighting.

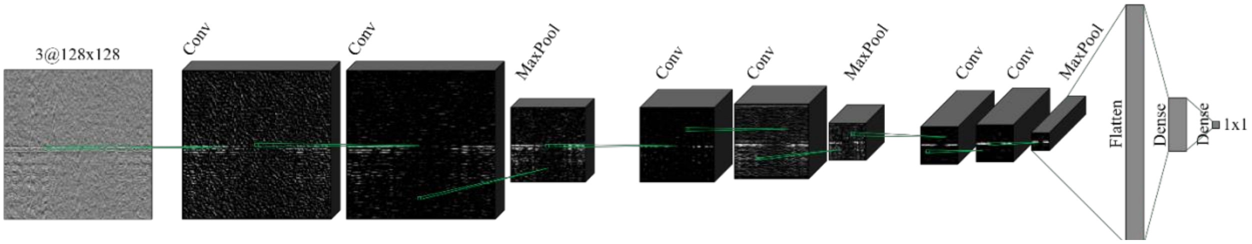

Figure 2: The convolutional neural network (CNN) architecture used during model training has 3 main convolution blocks, each ending with a max pooling operation. *Conv* is short for *Convolution*, and *MaxPool* is short for Max Pooling. The input is an image with 3 channels, and 128 × 128 pixels. The output is a single value as a categorical predictor, event or noise. The images on the front of the cubes are sample activation outputs. The *Keras* (Chollet, 2015) Python machine learning package was used for the implementation of the architecture.

CNN attempts to train using layers or "neurons" to make predictions on images.

In this paper, we approach the microseismic event vs. noise classification problem purely from an image-based approach with CNNs without using generated attributes or STA/LTA algorithms. The main goal of this study is to automate the task of microseismic event QC, with greater speed while retaining accuracy.

Field Data

In 2018, seismic data were collected from surface geophones in a star pattern (Duncan and Eisner, 2010) radiating out from a hydraulic fracturing site in west Texas, USA. There were 12 arms in the array, and a total of approximately 3,000 stations. Each station had 6 geophones per channel, giving a total geophone count of approximately 18,000. The seismic data were recorded at a 2 ms sample rate with a Sercel 428 recording system. The average target formation depth was approximately 7,000 ft below the well kelly bushing.

Preprocessing

Beginning from the raw field data, the following preprocessing steps were performed: debias, traveltime corrections, noise attenuation, spectral shaping, resampling to 4 ms, event triggering, mechanism corrections, and normal moveout (NMO) velocity corrections. The steps described so far are common in most microseismic workflows prior to event QC. Each seismic data panel representing a potential microseismic event contained approximately 3,000 traces and 249 samples (1 s of data sampled at 4 ms). The event triggering stage mentioned above is designed to locate microseismic events. However, the traditional triggering algorithms often trigger on noise, presenting a need for manual QC. The manual event QC that follows the preprocessing workflow usually takes many weeks to complete for one dataset, as there are often tens of thousands of images that need to be classified, either as noise or as events.

Before feeding the seismic trace data into the CNN, the data were converted and compressed to 128 × 128-pixel images. The trace amplitude values were normalized to a number between 0 and 255. An antialias filter was also applied before input to the CNN model. A total of 4109 images were split 60%-20%-20% into training, validation, and test, respectively. The images were either labelled as an event or as noise based on visual QC (see Figure 1). A single Tesla P4 8 GB GPU was used during training. To prevent overfitting, three kinds of data augmentation were used on the images in our study: rotation (shear), zoom, and horizontal flip. These methods of augmentation were fruitful toward building a predictive model that could accurately classify unseen images in this survey.

CNN Architecture

Figure 2 shows the relative shapes of the CNN architecture used during model training. The first and second convolutional layers ("Conv") have 32 filters of size 3 × 3 pixels with a stride of 1. Akin to a flashlight shining over a pixel area of 3 × 3, the convolution slides through the image, multiplying filter values with pixel values. The sum of the multiplication is the output of convolution. The concept behind applying a convolution is to abstract the features in the image. The initial filter values are random, but after a complete forward pass through the entire network, the filters are updated by calculating a gradient in the backward pass.

Following each convolutional layer, the result is passed to a nonlinear Rectified Linear Unit (ReLU) activation function, written as $f(x) = \max(0, x)$. The ReLU activation function is a successful and widely used activation function in deep learning (Jarrett et al., 2009; Nair and Hinton, 2010; Wu et al., 2018). The output of the convolution layer is a stack of these feature maps. Figure 3 shows the activation outputs for a sample image for the first three layers of the CNN. A higher value in the activation map indicates that at that location the feature is more likely present.

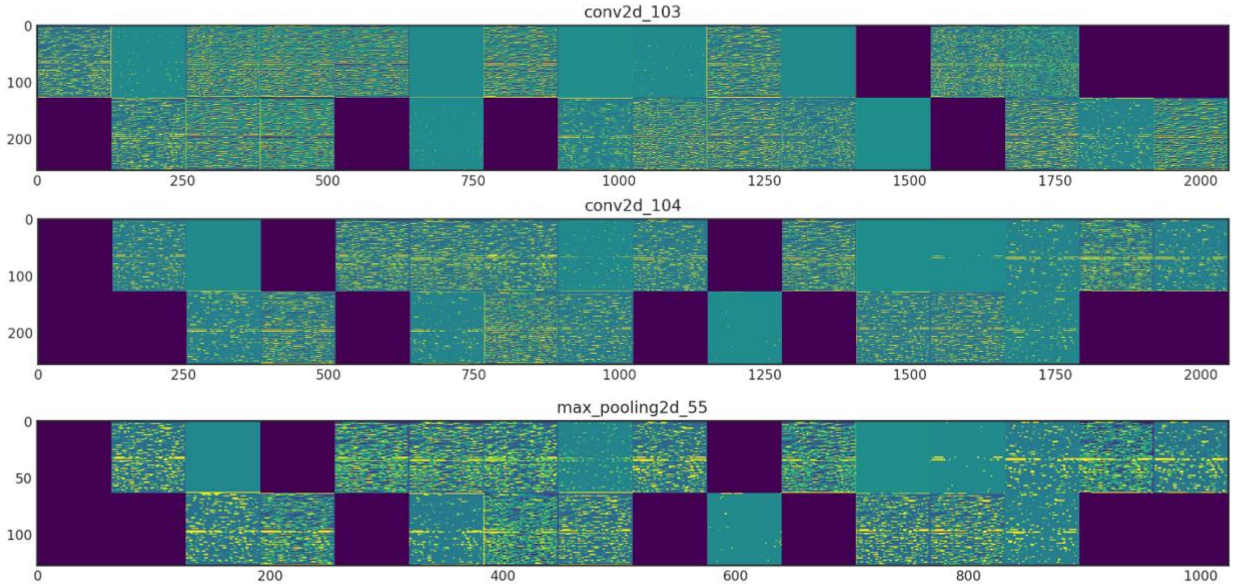

Figure 3: For a single sample image input, the activation function outputs are shown for the first three layers of the CNN: a 2D convolution (32 channels of 128 × 128-pixels), another 2D convolution (32 channels of 128 × 128-pixels), and a 2D max pooling (32 channels of 64 × 64-pixels). Some of the activations are blank, which indicates that the pattern that is present in the filter is not in the input image (Chollet, 2018).

A 2 × 2 max pooling function with a stride of 2 (“MaxPool” in Figure 2) is applied after the second convolutional layer in each block. The pooling layer is widely used to reduce the dimensionality and computation time, while keeping important information. Pooling also helps prevent overfitting by abstracting the features. The max pooling function receives as input 2 × 2 pixels and outputs the single maximum value of these pixels.

After the final max pooling layer, the CNN has a “Flatten” layer, which compresses the 2D feature maps into a long 1D vector. The last two steps in the architecture are two fully connected layers (“Dense”). The first fully connected layer is followed by a ReLU activation function, and the final fully connected layer is followed by a Sigmoid activation function.

After each forward pass of images through the network, a loss function is calculated. A low value for the loss function means that the model’s predictions are accurate. We used a binary cross-entropy loss function (Buja et al., 2005), which is well suited for binary classification schemes. In order to minimize the loss function of the model and calculate the gradient, we used the RMSprop optimization scheme (Tieleman and Hinton, 2012). Following the calculation of the gradient, the filters or ‘weights’ are calculated. The new weights are then used to begin the next forward pass of the CNN.

The CNN completes a full forward and backward pass for each batch. In our test, 40 images were used in a batch, until a total of 61 batches were reached (the full image training count). The completion of all batches marks the completion of one epoch. Other CNN architectures were also tested, including the 16-layer VGG Net (Simonyan and Zisserman, 2015), and another 2-layer CNN. However, the CNN shown in Figure 2 outperformed other tested architectures for the classification problem described here.

Results

Figure 4 shows the progression of model accuracy over 30 epochs for both the training and validation data. There was a close match in the training accuracy curve and validation accuracy curve, which is one indicator that the model was not overfitting on training data. Figure 5 shows the calculated loss function over the 30 epochs. The loss decreased by approximately 60% for both the training and validation data. The best model was selected from epoch 28 for the lowest validation loss value of 0.14, and a validation accuracy value of 0.94.

Average gradients for 6 sample images are displayed as heatmaps in the bottom of Figure 1, overlaying the seismic images. The images in 1 (a-c) were labelled as events and the images in 1 (d-f) were labelled as noise. The raw images along with the heatmaps of the gradient show that the CNN model learned a correlation between detected edges and microseismic events.

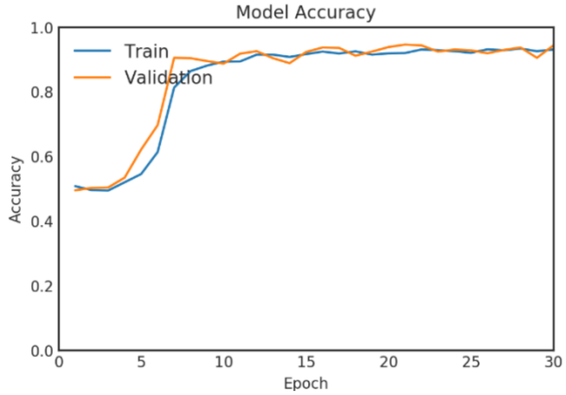

Figure 4: Over 30 epochs, the CNN training accuracy increases from 51% to 93%, and the validation accuracy increases from 50% to 95%.

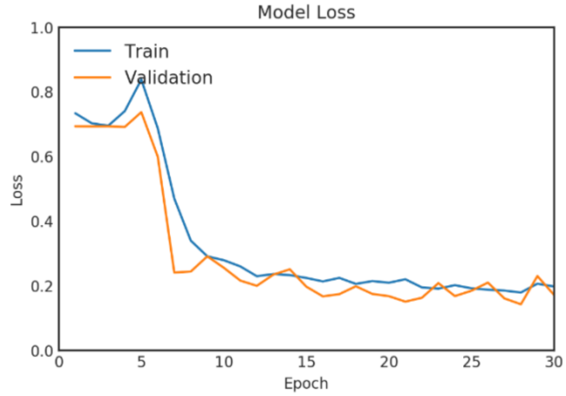

Figure 5: Binary cross-entropy loss function over 30 epochs shows an approximately 60% decrease in loss on the training and validation data.

Figure 6 shows a confusion matrix, comparing predicted and true labels for the test data (822 images). The matrix shows that 401 images were correctly classified as noise, 12 were misclassified as noise, 389 were correctly classified as events, and 20 were misclassified as events. The overall accuracy of the model on the test data was 96%. The predictions on 822 images only took 1.4 seconds.

Conclusions

We demonstrated the success of using an automated CNN workflow for classifying images of seismic data as either microseismic events or noise. The predictions on the test data were 96% accurate. One significant benefit of using an automated approach is the time savings. The predictions only took 1.4 s for 822 test images. Manual QC of these images might take 2 hours or more. Another benefit of using is the repeatability: the prediction could be completed on 10,000 images from the same survey, as the data distributions would be similar. Manual QC of 10,000 images would take weeks, whereas the CNN would take 3-4 hours for prediction. If the model needs to be re-trained with more data, the computation time is short at only 8 minutes for 2465 training images on one GPU.

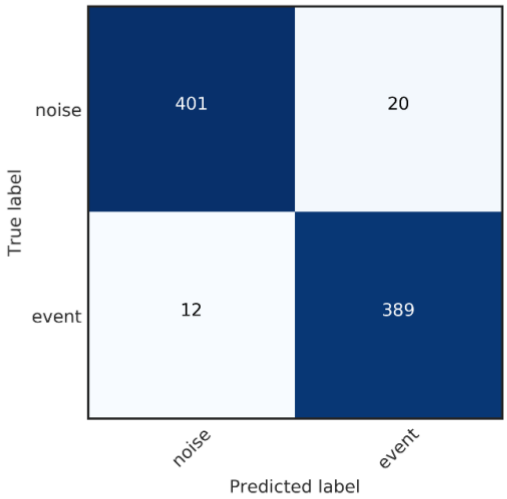

Figure 6: A confusion matrix to compare true labels with CNN-predicted labels for previously unused test data. 401 images were correctly classified as noise, and 389 images were correctly classified as events.

In the future, it may be useful to supplement the training data with other survey data to obtain a more generalized model. It would also be useful to generate probability distributions for each prediction output, to help determine which images need to be revisited for manual QC. The approach presented here could also be extended to include more categories, as there are practical cases where additional categories must be used to QC microseismic data.

Acknowledgements

We would like to thank MicroSeismic, Inc, for their support of our work. We also would like to acknowledge Google for providing use of a free GPU on Google Colab to run our tests.

REFERENCES

Allen, R. V., 1978, Automatic earthquake recognition and timing from single traces: Bulletin of the Seismological Society of America, **68**, 1521–1532.

Buja, A., W. Stuetzle, and Y. Shen, 2005, Loss functions for binary class probability estimation and classification: structure and applications: Working draft, November.

Chen, Y., 2018, Automatic microseismic event picking via unsupervised machine learning: Geophysical Journal International, **212**, 88–102, doi: https://doi.org/10.1093/gjihttps://doi.org//ggx420.

Chollet, F., 2018, Deep Learning with Python: Manning Publications Co.

Duncan, P. M., and L. Eisner, 2010, Reservoir characterization using surface microseismic monitoring: Geophysics, **75**, no. 5, 75A139–75A146, doi: https://doi.org/10.1190/1.3467760.

He, K., X. Zhang, S. Ren, and J. Sun, 2016, Deep residual learning for image recognition: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770–778, doi: https://doi.org/10.1109/CVPR.2016.90.

Jarrett, K., K. Kavukcuoglu, M. Ranzato, and Y. LeCun, 2009, What is the best multi-stage architecture for object recognition?: IEEE 12th International Conference on Computer Vision, 2146–2153, doi: https://doi.org/10.1109/ICCV.2009.5459469.

Kalkan, E., 2016, An automatic P-phase arrival-time picker: Bulletin of the Seismological Society of America, **106**, 971–986, doi: https://doi.org/10.1785/0120150111.

Krizhevsky, A., I. Sutskever, and G. E. Hinton, 2012, ImageNet classification with deep convolutional neural networks: Advances In Neural Information Processing Systems 25, 1097–1105.

Nair, V., and G. E. Hinton, 2010, Rectified linear units improve restricted Boltzmann machines: Proceedings of the 27th International Conference on Machine Learning, 807–814, doi: https://doi.org/10.1.1.165.6419.

Qu, S., E. Verschuur, and Y. Chen, 2018, Automatic microseismic-event detection via supervised machine learning: 88th Annual International Meeting, SEG Technical Program, Expanded Abstracts, 2287–2291, doi: https://doi.org/10.1190/segam2018-2998279.1.

Ren, S., K. He, R. Girshick, and J. Sun, 2017, Faster R-CNN: Towards real-time object detection with region proposal networks: IEEE Transactions on Pattern Analysis and Machine Intelligence, **39**, 1137–1149, doi: https://doi.org/10.1109/TPAMI.2016.2577031.

Ronneberger, O., P. Fischer, and T. Brox, 2015, U-Net: Convolutional networks for biomedical image segmentation: International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI) **9351**, 234–241, doi: https://doi.org/10.1007/978-3-319-24574-4_28.

Ruud, B., and E. Husebye, 1992, A new three-component detector and automatic single-station bulletin production: Bulletin of the Seismological Society of America, **82**, 221–237.

Shin, H. J., D.-H. Eom, and S.-S. Kim, 2005, One-class support vector machines—An application in machine fault detection and classification: Computers and Industrial Engineering, **48**, 395–408, doi: https://doi.org/10.1016/j.cie.2005.01.009.

Simonyan, K., and A. Zisserman, 2015, Very deep convolutional networks for large-scale image recognition: International Conference on Learning Representations.

Song, F., H. S. Kuleli, M. N. Toksöz, E. Ay, and H. Zhang, 2010, An improved method for hydrofracture-induced microseismic event detection and phase picking: Geophysics, **75**, no. 6, A47–A52, doi: https://doi.org/10.1190/1.3484716.

Taigman, Y., M. Yang, M. Ranzato, and L. Wolf, 2014, DeepFace: Closing the gap to human-level performance in face verification: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1701–1708, doi: https://doi.org/10.1109/CVPR.2014.220.

Tieleman, T., and G. Hinton, 2012, Lecture 6.5–RMSProp: Divie the gradient by a running average of its recent magnitude, in Neural Networks for Machine Learning: Coursera.

Vaezi, Y., and M. Van der Baan, 2015, Comparison of the STA/LTA and power spectral density methods for microseismic event detection: Geophysical Journal International, **203**, 1896–1908, doi: https://doi.org/10.1093/gjihttps://doi.org//ggv419.

Wu, X.,Y. Shi, S. Fomel, and L. Liang, 2018, Convolutional neural networks for fault interpretation in seismic images: 88th Annual International Meeting, SEG Technical Program, Expanded Abstracts, 1946–1950, doi: https://doi.org/10.1190/segam2018-2995341.1.

10.1190/segam2020-3414896.1