Data-driven Microseismic Event Localization: an Application to the Oklahoma Arkoma Basin Hydraulic Fracturing Data

Hanchen Wang¹, Tariq Alkhalifah¹, Umair bin Waheed², and Claire Birnie¹

The microseismic monitoring technique is widely applied to petroleum reservoirs to understand the process of hydraulic fracturing. Geophones continuously record the microseismic events triggered by fluid injection on the Earth’s surface or in monitoring wells. The microseismic event localization precision has a large impact on the performance of the technique. Deep learning has achieved significant progress in computer vision and natural language processing in recent years. We propose to use a deep convolutional neural network (CNN) to directly map the field records to their event locations. The biggest advantage of deep learning methods over conventional methods is that they can efficiently predict the characteristics of a huge amount of recorded data without human intervention. Thus, we use a CNN to predict the event location of field microseismic data, which were recorded during a hydraulic fracturing process of a shale gas play in Oklahoma, the United States. We use synthetic data with extracted field noise from the records to train the CNN. The synthetic training data allow us to produce the corresponding labels, and the extracted noise from the field data reduces the difference between the field and synthetic data. We use a correlation pre-processing step to avoid the need for event detection and picking of arrivals. We demonstrate that the proposed approach provides accurate microseismic event locations at a much faster speed than traditional imaging methods, such as time-reversal imaging. Comparison with an existing study on the same data is presented to evaluate the performance of the trained neural network.

Index Terms—Geophysical Data, Surface and Subsurface Properties

I. INTRODUCTION

Hydraulic fracturing, or fracking, is a drilling production method used to extract petroleum (oil) or natural gas from the depths of the Earth. In the hydraulic fracturing process, injection of high-pressure water, chemicals, and sand will lead to cracks in and below the Earth’s surface being opened and widened. Some unconventional resources are extracted through fracturing because these reservoirs are tightly trapped in hard shale rock formations.

The fracturing process often ignites a huge number of microseismic events [1]. These microseismic event locations provide key information to help the engineers monitor the fracturing process and optimize the injection strategy [2]. Usually, a monitoring system with geophones is temporally or permanently set in a well or on the Earth’s surface to record the events continuously [3]. The locations of a tremendous number of microseismic events are usually estimated from the records. Hence, accurately locating microseismic events is important to the hydraulic fracturing process. However, detecting and locating all the events can be a challenging and time-consuming task with conventional methods. Thus, a lot of effort have been directed to automatically extract the microseismic waveforms to enhance the resolution of the imaging results. For example, an unsupervised deep learning scheme for waveform extraction is proposed to automatically image events from the noisy record to their source locations [4].

Conventional ways to locate the sources with high computational efficiency depend on inverting traveltime picks of both P-wave and S-wave arrivals [5]. However, traveltime-related methods are highly influenced by the picking precision, as they require manual efforts or automatic pickings by algorithms. Manual picking of arrivals, unfortunately, can hardly be done accurately and efficiently because the data suffer from low signal-to-noise ratio (SNR) [6]–[8]. Thus, some researchers proposed to use robust machine learning methods for denoising, which can enhance the picking performance [9], [10]. Time-reversed imaging (TRI) technique, utilizing the wavefields, are often used in the field [11]–[14]. By back-propagating the recorded data from their receiver locations to a given velocity model of the region of interest, guided by a wave equation, we are released from picking arrivals. An imaging condition is properly applied to extract the source locations from the back-propagated wavefield, such as the maximum energy imaging condition [15], [16], the geometric mean imaging condition [17], the cross-correlation imaging condition [18], and the maximum variance imaging condition [19]. Though TRI is free of manual picking, especially when dealing with poor SNR data, it suffers from other drawbacks. If only limited traces are available, TRI is not robust in accurately locating the sources. Many studies were made to mitigate the issue of irregular and sparse receivers [12], [20]–[26]. However, solving the wave equation is time-consuming and expensive, especially when a huge number of events are recorded.

Another popular point-source imaging method, the diffraction stack imaging (DSI), is based on computing a stacking function in a volume of interest, which assesses the coherency of waveforms in space. The event location and origin time can then be determined by the maximum value in the stack function [27].

Another category of microseismic imaging methods is based on waveform inversion [28]–[32]. The inversion is also responsible for velocity model building using a proper objective of data matching. However, similar to TRI methods, the inversion-based methods require an even higher computational cost because at least two wavefields are simulated for each

1Physical Science and Engineering Division, King Abdullah University of Science and Technology, KSA
2Department of Geosciences, King Fahd University of Petroleum and Minerals, KSA
event in every iteration, and hundreds of iterations are essential for convergence.

With the explosive development in computational resources, utilizing machine learning to detect, locate and interpret passive or microseismic events automatically is gaining much attention. Human experts are responsible for data preparation, network architecture designation, and hyper-parameter fine-tuning. Once the networks are well trained, they work instantly and with no need for human interactions. They vastly reduce human efforts and computational costs while maintaining high accuracy in locating the events. Due to most machine learning methods being purely data-driven, we no longer need waveform simulations or arrival pickings. Many methods of machine learning methods in passive seismic event detection were proposed recently [33]–[37]. These methods were aimed to detect the microseismic events from the raw field monitoring data by training different neural networks using synthetically generated data. After detection, the microseismic events were inserted into conventional locating algorithms, which suffered from the aforementioned challenges. Thus, several techniques were proposed to locate earthquakes automatically. A convolutional neural network (CNN) was trained to be a classifier using more than 2000 induced microseismic events recorded by two stations to determine the source locations [38] in six regions. Another study also proposed to train a CNN using more than 2000 events from West Bohemia, recorded on nine local stations, to locate clustered earthquakes. Recently, a convolutional network was used to predict the earthquakes in Oklahoma with only 30 stations [39]. A Bayesian-based deep learning algorithm was used to predict natural earthquakes, as well [40]. These approaches were designed for global scale earthquakes with endurance for significant location error.

In this study, we train a CNN using correlated P-wave synthetic data as input. The network predicts the event locations in space as a regression task. Once the CNN is well trained, it can be used for real-time microseismic events locations on the raw monitored record segments. We generate synthetic acoustic data with the given 3D P-wave velocity model as our training and validation data. The given 3D P-wave velocity model is generated from well-log information and calibrated by perforation shots. We add the recorded noise extracted from the field records to the training and validation data to mitigate the difference between the synthetic and field data. We also add weak white noise to the synthetic data as a data augmentation step. Once the network performs sufficiently on the synthetic testing data, we test the network performance on the field data. Finally, we compare our network predicted results with an existing study on the same data [41] and a traditional TRI-based method.

II. Multi-task Learning in Deep Convolutional Neural Networks

In machine learning (ML) tasks, we generally train a single model or an ensemble of models to perform our desired tasks. We then fine-tune the model(s) to obtain the best performance. We can achieve it by focusing on our single task and ignore information that may help us do better on our task. This information typically comes from the training data of related tasks. The model will be generalized better on our original task by sharing the characterizations of related tasks. Such an approach of sharing characterizations is called multi-task learning (MTL). MTL has been proven to be successful in many applications of machine learning, such as natural language processing [42], speech recognition [43] and computer vision [44]. In our application, we define the microseismic event location coordinates in space as three tasks \((X, Y, Z)\).

Hard parameter sharing is the most commonly used approach in MTL [45]. By sharing the hidden layers between all tasks while keeping specific output layers for each task, hard parameter sharing significantly reduces the risk of over-fitting [46]. In MTL, each task can be considered a regularizer for the other independent tasks because the shared layers are used to identify the general features of the input data that serve all the independent tasks. Thus, the more tasks we are learning simultaneously, the more our model has to find a representation that captures all of the tasks needed features, which in turn reduces the chance of over-fitting. The general architecture of hard parameter sharing is shown in Figure 1.

Convolutional Neural Networks are typical architectures of deep neural networks. In traditional neural networks consisting of fully connected layers, a high dimensionality with a large number of weights is involved in the training. A distinct feature of CNN is the concept of local neuron connectivity and shared weights. The connectivity is given by a convolution process, which significantly reduces the number of weights to be trained. The convolutional neural network is most commonly applied in analyzing visual imagery [47], [48]. The architectures of CNNs often differ, but typically it contains three types of layers: convolutional layers, down-pooling layers, and fully connected layers [49].

A. The CNN Architecture

Figure 2 shows the CNN architecture that predicts the source locations \((X, Y, Z)\) as three individual tasks in multi-task learning. It is a modified VGG-shaped network architecture. We input the records of the ten receiver arms (128 by 128 by 10) as ten-channel images, and the output labels correspond to the extracted features from these images. We have two pre-processing steps before feeding the input images to the network, which we will discuss in the next section. We use the convolutional layers to extract from the seismic record critical features, like the location of the apex and the curvature of the events, which correspond to the events’ horizontal locations \((X)\) and \((Y)\) and the events’ depth \((Z)\), respectively. A convolutional layer often consists of learnable filters or kernels, sliding over the input data and performing an element-wise multiplication [50].

In our MTL application, the convolutional layers are shared by all the tasks, which are shown as the blue-colored layers in Figure 2. In total, 14 convolutional layers are set to be the shared layers by all three tasks. The number of filters increased by 50 after each pooling operation, with the first convolutional layer beginning with 50 filters. A max-pooling layer is inserted after every two convolutional layers to downsample the input images. The down-sampling operation has
the effect of making the resulting down-sampled feature maps more robust to changes in the location of the feature in the image, which is often referred to as local translation invariance. The max-pooling layers used in the network have kernel sizes of $(2, 2)$ and strides of 2. After the shared CNN layers, the flattened matrix goes through fully connected layers to transit from feature maps to an output prediction for the model. Two specific fully connected layers are used to output the final prediction of each task, which are shown as the blue boxes on the right end of Figure 2. In our MTL application, the fully connected layers are task-specific, which means each task has its private fully connected layers to give the specific output labels. The numbers of neurons used for the first and second fully connected layers are 350 and 20, respectively, for all the tasks.

We built the network architecture based on the trade-off between the training efficiency and the network performance. With deeper (more CNN layers) and wider (more kernels in each CNN layer) networks, the training efficiency is lower but the performance can be better. However, it is also likely to face an over-fitting problem. We use this fourteen-layered CNN architecture because, from our experiments, there are enough neurons to identify the features in the data and yet the training cost is reasonable and we did not encounter an over-fitting problem.

A supervised learning scheme is performed in this study. All the layers pass through a LeakyReLU activation function. Rectifiers, such as ReLU and LeakyReLU, can effectively reduce the vanishing gradient problem because they only saturate in one direction [51]. The Mean Squared Error (MSE) loss for each task is calculated by:

\[ E_k = |l^k_p - l^k_t|^2, \]  

where \( k \) denotes the task index, \( i \) denotes the \( ith \) input data, \( l^k_p \) and \( l^k_t \) denote the predicted label and the ground truth, respectively. The total loss is calculated by summing over all the tasks and back-propagated to train the neural network by minimizing the total loss function. The learning process is iteratively run until the maximum epoch number is reached.

III. DATA PREPARATION AND TRAINING STRATEGY

The field data we use in this study correspond to passive seismic monitoring continuously for four days during a hydraulic fracturing stimulation of a shale gas reservoir in the Arkoma Basin in the United States. More detailed information can be found in previous work on the same dataset [41]. We use 17 previously detected and analyzed events as our testing data. The corresponding previously determined locations are provided and we use the locations as our reference labels in this case. We choose these events because they are the strongest events from all detected events in the raw data prior to any data processing.

In this dataset, four horizontal wells were drilled into the shale formation at true vertical depths of approximately 2100 m. A large aster-shape array is deployed as the microseismic monitoring system with 911 single-component vertical receivers. The receiver geometry is shown in Figure 3. The receivers are spread out along ten radial arms centered around the wellhead. The minimum number of receivers per line is 54 (line four), and the maximum number is 122 (lines two and ten). The single-component receivers record only vertical displacement on the Earth’s surface. As a result, P wave arrivals dominate the data, and we can hardly see S wave arrivals. Thus, we decide to ignore the S waves and focus on P waves only as P wave arrivals potentially contain all the key features needed to locate the events.

In microseismic experiments, the data recorded from the field are incredibly long, sometimes extending for days. The nature of microseismic monitoring is that the source origin time and location are unknown, and only the recording time is known. It is generally expensive to label the field data by conventional methods and use it to train the model. Also, field data are limited and can probably be used only in transfer learning. In this case, the accuracy of the resulting NN model is only as good as the computed labels from field data. Thus, we propose to use synthetically generated data as the training and validation dataset for the NN model and the field data as the testing dataset. It is much cheaper to generate labeled synthetic data as we fully control the synthetic computing steps.

The blue box area in Figure 3 shows our region of interest. It is decided by the prior information of the region, mainly by the labeled events. The blue box covers the area of \( x = [1463, 3048] \) m, \( y = [1463, 2743] \) m, \( z = [1524, 2286] \) m. We synthetically generate the training and validation data using a Ricker wavelet as the source wavelet. The source parameters are randomly selected within certain ranges. The event locations are within the blue box area; the peak frequencies of the wavelets are between 25 – 35 Hz (matching the field data spectrum); the maximum amplitudes (referred to as the event magnitudes) of the wavelets are between 0.5 – 1. The random selection is based on a uniform probability for all the numbers in the ranges. The chosen source parameters should be consistent with the real data. Otherwise, it will affect the application of the NN model. A prior analysis of a certain amount of real data is necessary for parameter selections. We use the velocity model shown in Figure 4 to generate the training data. The velocity is a 3D \( V_p \) model built by the data provider from well-log information of the region and then calibrated by perforation shots. The data provider also used the model to study the microseismicity of the area in a previous study [41]. In total, we have generated 5000 data with an acoustic finite-difference modeling code corresponding to 5000 randomly selected source parameters. We randomly divide the data into a training set and a validation set following an 80/20 split rule such that 4000 data samples act as the training set and 1000 data samples act as the validation set.

Conventional pre-processing of the data includes splitting the data into several segments corresponding to the detected events. However, the segment size varies from data to data and it usually is too large for training. We process the data by picking a reference trace from every segment and cross-correlate it with all the traces in the segment using:

\[ D(x, \tau) = d_{ref}(t) \ast d(x, t), \]
where \( d(x, t) \) is the data segment, \( d_{ref} \) is the chosen reference trace from the data segment, \( * \) is the correlation operation and \( \tau \) is the lag of the correlation operation. It can be summarised into the following steps: 1. pick a reference trace from each recorded data; 2. correlate the reference trace with the whole data. The data before and after the correlation are shown in Figure 5. Note that the single event in this segment is pulled to approximately zero lag after the correlation operation. Thus, we can represent a data segment with a smaller time window around the zero time lag regardless of the origin time of the event in the segment. Only cross-correlation cross-talk will have large time lags. In this way, we may significantly reduce the CNN input data size while retaining the key features in the data. Note that the whole pre-processing is purely automatic. We do the same pre-processing to the synthetic training data and the field data and feed the data with a short time lag window as the input to the network.

We input the ten arms as ten channels to the first convolutional layer, similar to how we usually treat RGB images. However, the trace numbers in the ten arms are not consistent. Thus, we conduct an interpolation with a \( \text{sinc} \) function along the receiver axis to make all the ten input channels consistent in size. The interpolated arms have 128 traces for all arms. Figure 6 shows examples for synthetic and field recorded data. We pick the first trace of each gather as the reference trace and implement the correlation operation in equation 2. Then we cut a short time-lag window around the zero-lag position, and the outcome field and synthetic data are shown in Figure 7.

Although the two images in Figure 7 share some similarities in the key features, such as the microseismic event curvature, they also are reasonably different, mainly due to the strong field noise in the field data. Thus, we use a semi-synthetic strategy to mitigate the gap between the synthetic and field data. We extract in total 10000 pure field noise data from the large time-lag positions in the correlated field data, shown in Figure 8. Such extracted noise data contains only noise and no events. We add the field noise to the training and validation data with a random selection approach to train the model to ignore the field noise and focus only on the main event. The noise added to the training data starts from lower levels (100% of the signal energy) to higher levels (1000% of the signal energy) with respect to training epochs. The training data with different noise levels are shown in Figure 9. The workflow of the proposed method is shown in Figure 10.

The output labels of the network are the event locations in 3D \((X, Y \& Z)\) as three related tasks. Each task output has its unique feature in the input data. The event apex horizontal position in the different arms determines the \(X\) and \(Y\) coordinates; the moveout mainly defines the \(Z\) label (the depth of the source influences the curvature of the events). We independently normalize the labels of \(X\), \(Y\), and \(Z\), to the range of \([0, 1]\) to stabilize the training. In addition, we apply a Batch Normalization to reduce internal covariate shift and accelerate the training [52].

Thus, the network is used to learn the nonlinear relationships between the input data and output labels. Increasing the number of convolutional layers or the number of neurons in each layer may improve the performance of some training sets. However, too many neurons in the network may lead to overfitting, which means the network memorizes the data instead of learning them. Thus, the network, in this case, performs poorly on test data. Therefore, the choice of network parameters is quite crucial for optimal performance. One way to evaluate the over-fitting problem is to include regularization terms in training. Thus, we apply a 5% dropout [48] to each convolutional layer and use a weight decay regularizer \((\epsilon = 10^{-5})\) [53] to mitigate the over-fitting problem. We start the training with a learning rate of 0.01 and use the adaptive learning rate scheduler to automatically reduce the learning rate by four when the validation performance does not improve. We also add random noise of 10% signal energy level to the training data, which works like a Tikhonov regularization [54], to help us obtain robust predictions.

IV. EXPERIMENT RESULTS ANALYSIS

We first test this approach on synthetic (training and validation) data to demonstrate the applicability of the method. We then show the field data test results.

A. Synthetic Test Results

The training and validation datasets are generated synthetically, as described in the previous section. 5000 random events are simulated and divided into a training set and a validation set based on an 80/20 split. The velocity model and the receivers we use are consistent with the field data. The pre-processing steps of interpolation and correlation with a reference trace are implemented. We choose a time-lag window of 0.512 s around the zero-lag position. The input data size is \(128 \times 128\) samples in both space and time. We keep the field noise level at 100% through the training. The logarithmic scale training and validation loss values with respect to the training epochs are shown in Figure 11 (a) and (b), respectively. The validation loss not curling upward is a good sign that we are not facing an over-fitting problem. We stop the training at epoch 200 as the losses do not decrease anymore and the learning rate is sufficiently low. From single task loss curves, we find the \(Z\) loss values are the largest of the three, and \(X\) and \(Y\) losses are almost at the same level. Among the three tasks, \(Z\) is the most difficult label to predict. \(X\) and \(Y\) mainly depend on the arrival apex horizontal position, while \(Z\) depends on the moveout. The receivers are set on the Earth’s surface, making the moveout a smaller feature to capture than the apex location because shifts of an event in space introduce an obvious difference in the apex horizontal location and a relatively small difference in the moveout. Moreover, \(X\) and \(Y\) rely on similar input data features, which are different from those that \(Z\) relies on. In MTL, similar tasks will help each other in the convergence.

Figure 12 shows the label prediction errors for each task. The events are plotted in the order of error values from small to large. The three tasks are all well predicted on the synthetic validation set by the network. As we can see, the blue predicted curves and the green ground truth curves overlap. In Figure 12 (a), the relatively large errors on the
right end come from the smallest and largest \(X\) labels due to reduced illumination on the edges of the source region. Similar observation can be made in Figure 12 (b) for \(Y\). In Figure 12 (c), the large \(Z\) errors come from the deeper events because the moveout difference is smaller when the events are deep. We randomly select 20 events from the testing dataset and show the comparison of the predicted labels by the network and the corresponding ground truth in Figure 13. The average spatial distance between the blue predicted locations and the red ground truth ones is small, less than 10 m.

Figures 14 and 15 show the histograms of the misfit in location, \(X\), \(Y\), and \(Z\), respectively. The average location error is about 16 m, and the single label average error is also less than 20 m. We then calculate the error maps, shown in Figure 16, weighted by the inverse distance between each grid point and the validation set ground truth. We find that the large errors (indicated by light colors) are mostly located at larger \(Z\) labels (deeper events) and \(X\), \(Y\) edge labels, which further validates our analysis.

\[ <a(x,t),b(x,t)> = \frac{\sqrt{a(x,t),a(x,t)}> <b(x,t),b(x,t)>}{s} \]  
\[ \gamma = \frac{<a(x,t),b(x,t)>}{\sqrt{a(x,t),a(x,t)}> <b(x,t),b(x,t)}> s } \]

Then we plot the coefficients values and the label differences of the 17 events. If the two events have the same curvatures, the coefficients will be large regardless of the field noise (close to one). Otherwise, the correlation coefficients will be small (nearly zero). The correlation coefficient curve (blue) and the label difference curve (purple), shown in Figure 19, respectively, have an inverse correlation relationship between them. It hints that once we can synthetically reproduce the same data at the same labeled locations as the field data, we have a good prediction. Meaning that if the determination of the reference labels is consistent with the acoustic modeling (including the velocity model) used to generate the synthetic training data, represented by the high correlation, the prediction of the network is good. Otherwise, the provided labels may be related to the velocity model inaccuracy, the simplified forward modeling engine, or the reference label itself. We may further study the influence of these factors in our future work. Nevertheless, the predicted results on the field data for those reference labels satisfying our modeling setup are reasonably good.

\[ \gamma = \frac{<a(x,t),b(x,t)>}{\sqrt{a(x,t),a(x,t)}> <b(x,t),b(x,t)> s } \]

\[ <a(x,t),b(x,t)> = \int a(x,t)b(x,t)dxdt \]  

\[ \gamma = \frac{<a(x,t),b(x,t)>}{\sqrt{a(x,t),a(x,t)}> <b(x,t),b(x,t)> s } \]  

\[ <a(x,t),b(x,t)> = \int a(x,t)b(x,t)dxdt \]  

The training process is not only accurate but also efficient. It utilizes four paralleled GPUs (Nvidia RTX 2080 Ti) and only takes twelve hours. The trained network can predict all the testing set labels in a few seconds. This feature will be essential when using the network to monitor a field, especially when comparing it with conventional methods of either manual pickings or solving partial differential equations. Thus, the proposed method can provide a fully automatic approach that could locate microseismic events instantly. Also, in the process of monitoring in a field, even if the velocity model is updated through utilizing new data, the training of the network considering the new velocity can be cheap through transfer learning. It is given by using the current NN model as the initial model for the new training and freezing certain convolutional layers during the training.

For this ML implementation, we focus on the microseismic event location. We cannot identify the origin time of the
event within the cross-correlated segment using this approach. Though the origin time is important in earthquake seismology, it tends to be less important for reservoir monitoring and fracture mapping. To allow our method to map data directly with a reasonable size input to the network and less variance in the input dataset, we opted to sacrifice the origin time information using a correlation step. The passive event location accuracy is dependent on station locations, velocity model, and the physics involved in the training. In this study, we assume the given velocity model to be accurate enough. We focused on the acoustic assumption, but the same approach can be used for elastic media. In the field data experiment, we use only the events with the highest SNR, making our problem easier than the real monitoring case. If the noise level is even higher, we may need to further train the network, starting with the existing one, with stronger noise. The performance of the network on the weaker events may reduce accordingly.

C. Comparison with the traditional TRI methods

Besides the accuracy comparison between the proposed machine learning method and the previous study using a traveltime-based method, we also compare the accuracy and efficiency of the proposed method to traditional time-reversal imaging methods.

We choose to implement the TRI with the geometric mean imaging condition to determine the source location in space. The field data are separated into three groups of traces. Each group is reversed in time and injected as sources at the corresponding receiver locations using the given \( V_g \) model, which is also the one we relied on to implement the ML method. Then the back-propagated wavefields are multiplied and integrated over the time axis. The maximum energy positions after the integration are thus referred to as the migrated source locations by TRI. More details of the geometric mean TRI method can be found in [17].

An example of the calculated source images is shown in Figure 20, where the maximum energy point is the determined event location. After locating all the events, we compare the TRI event locations with the reference labels used in the previous experiment. The histograms of the events’ location mismatch of the proposed ML method and the TRI geometric mean method are shown in Figures 21(a) and (b), respectively. The average event location misfit using the TRI method is about 100m, which is larger than the proposed ML method misfit. Alongside this, the ML approach is far more efficient, which is an important feature for real-time monitoring. As a comparison, the TRI method required 4.7 minutes to calculate one back-propagated wavefield on an eight-thread workstation and 1.5 minutes to calculate the geometric mean imaging condition. Thus, the difference in computational cost will increase dramatically for realistic 3D applications.

V. Discussion

We generally have three ways to obtain data in seismic imaging: synthetically generated data, laboratory-generated data, and field-collected data. Machine learning, especially supervised learning, requires a sufficiently large amount of training data, which have the same features as the testing data. The training data have to be easy labeling. Otherwise, the data preparation will be expensive and time-consuming. Thus, in this study, we proposed to use a semi-synthetic strategy that the training data samples are synthetically generated, and pre-processing steps are applied to make the training data similar to the field data. Thus, the network’s performance on the field data is highly dependent on the quality of the synthetic training data and the similarity to the field data. When generating the data, we assumed that the forward modeling operator fully represents the physical process in the media, and the velocity model we use is accurate. However, the velocity model is generated by well-log information and calibrated by perforation shots. It is a simple estimation of the true Earth’s structure. Besides, we use an acoustic wave equation as the modeling operator and a point source as the source mechanism, which means we ignore the elastic effect and the source mechanism influence. These factors will directly influence the quality of the training data we generated, and the corresponding error in the field data performance is beyond the proposed method’s ability. The provided event locations are also estimated by the inaccurate velocity model and other assumptions. Thus, we use such locations as reference labels, which cannot be considered ground truths. The biggest advantage of the proposed ML-based method is its efficiency. Compared to traditional methods, especially TRI-based or waveform inversion based methods, once the network is trained, the prediction time is sufficiently short. However, the ML method, like any other approach, can face problems with data with very low SNR, which is common in passive seismic data. In this case, denoising processes, even those utilizing ML, might be needed. Further study on the influence of high SNR on the proposed approach will be covered in our future work.

VI. Conclusion

We developed a CNN framework that can efficiently and accurately predict the microseismic event locations from microseismic monitoring data. To prepare the training and testing data, we interpolate data with a sinc function and cross-correlate a reference trace from a data segment with the rest of the traces. This process moves the events to the zero-lag position while retaining the key feature (event shape and curvature). We train the network on synthetic acoustic waveforms from a given field monitoring velocity model. We add field noise extracted from the correlated field data to the synthetic training data to close the gap between the training and field testing data. The training is based on a semi-synthetic and multi-stage strategy. We use certain regularization terms to avoid the over-fitting problem, such as weight decay and dropout. From our results, the proposed method can predict the event locations efficiently and accurately. The field data experiment demonstrated that the proposed ML-based approach could more efficiently predict microseismic sources’ location compared to conventional methods. Moreover, the accuracy of the ML-based method is similar to the TRI-based geometric-mean method, especially for those events in which
the reference location matches the physics used in the training of the network.

ACKNOWLEDGMENT

The authors would like to thank KAUST and KAUST Super Computing Lab for supporting the research. The authors would specially thank Dr. Frantisek Stanek and Dr. Leo Eisner for providing and discussing the field data. Finally, the authors would like to thank all Seismic Wave Analysis Group members for their helpful discussion and comments.

REFERENCES


Fig. 1. Multi-task learning hard sharing structure.

Fig. 2. The convolutional neural network architecture used in the proposed method. The pink block includes the process of adding the extracted field noise. Whereas, the dark purple block indicates the addition of random noise. The blue-colored layers are the shared convolutional layers by all the tasks. The blue boxes on the right are the independent fully-connected layers for the output of each specific task.
Fig. 3. The receiver (blue dots) location map view and the region of interest (sky blue).
Fig. 4. The 3D P-wave velocity model used for generating the synthetic training data. The velocity model is generated from well-log information of the region and then calibrated by perforation shots.
Fig. 5. Data comparison before and after the reference trace correlation operation, as an example. The recorded data on the left is correlated with the red reference trace along the time axis, converting the real recording time axis to a time lag axis.
Fig. 6. Examples of (a) raw field and (b) synthetic data.
Fig. 7. Correlated with the reference trace examples of (a) field and (b) synthetic data.

Fig. 8. Extracted noise example from field data after the correlation with a reference trace.
Fig. 9. Synthetic training data with different levels of field data noise added to them: (a) 100% (b) 300% (c) 500% (d) 1000%.
Fig. 10. The workflow of the proposed method based on a semi-synthetic multi-stage MTL strategy.
Fig. 11. The (a) training and (b) validation loss curves for synthetic data. Loss curves for the (blue) total, (orange) Z, (grey) Y, and (green) X labels.
Fig. 12. The label comparison between the ground truth and network prediction (a) X (b) Y and (c) Z. (blue) Predicted labels; (green) ground truth; (red) absolute label misfit. The curves are plotted in the order of errors from small to large.
Fig. 13. The comparison between (blue) the ML predicted synthetic testing event locations and (red) the corresponding ground truth. 20 random events from the synthetic testing dataset are shown.
Fig. 14. The prediction error histogram for spatial location accuracy.
Fig. 15. The prediction error histograms for (a) X, (b) Y, and (c) Z labels.
Fig. 16. The inverse distance weighted error maps for the (a) X-Y, (b) X-Z, and (c) Y-Z plane.
Fig. 17. The (a) training and (b) validation loss curves. Multi-stage strategy: from lower to higher noise levels. Loss curves for the (blue) total, (orange) Z, (grey) Y, and (green) X labels.
Fig. 18. Comparison between the (red) given event locations with traveltime picking and (blue) predicted event locations with our proposed machine learning method.

Fig. 19. Comparison of the (purple) global correlation coefficients and (blue) label differences between the given and predicted labels.
Fig. 20. An example source image generated by TRI with the geometric mean imaging condition.
Fig. 21. The histograms of the event location misfit between the reference labels and the results of (a) the ML-based method and (b) the TRI based method.