Localizing weak microseismic events using transfer learning with a deep neural network

Vinard, Nicolas André; Drijkoningen, Guy Gérard; Verschuur, Dirk Jacob; Alexandrov, Dmitry; Eisner, Leo

DOI
10.1111/1365-2478.13238

Publication date
2022

Document Version
Final published version

Published in
Geophysical Prospecting

Geophysical Prospecting

EAGE EUROPEAN ASSOCIATION OF GEOSCIENTISTS & ENGINEERS

Geophysical Prospecting, 2022 doi: 10.1111/1365-2478.13238

Localizing weak microseismic events using transfer learning with a deep neural network

Nicolas André Vinard[1]*, Guy Gérard Drijkoningen[1], Dirk Jacob Verschuur[1], Dmitry Alexandrov[2] and Leo Eisner[2]

[1]*Delft University of Technology, 2628 CJ Delft, The Netherlands, and* [2]*Seismik s.r.o, Prague, the Czech Republic*

Received September 2021, revision accepted May 2022

ABSTRACT

Retrieving accurate microseismic source locations induced by hydraulic-fracturing operations is an important step to gain insights into the hydraulically stimulated reservoir volume. Recently, deep neural networks have been proposed that directly recover source locations from the seismic waveforms. The optimal performance of the proposed deep neural networks usually requires large training sets. The need for a large training set can be circumvented if a previously trained deep neural network can be used to start the training process with its weights instead of randomly initialized weights. These weights can then be fine-tuned using a smaller training set, which is also known as transfer learning . In this work, we implement a transfer learning workflow to update the weights of a deep neural network that was initially trained on a large synthetic dataset to localize microseismic events. We present two methods of processing, namely one post-monitoring mode and one continuous mode where the processing takes place during the monitoring period. We apply the methods to field data from a hydraulic fracturing site in Texas, USA. In the first scenario, a subset of the field data from the entire monitoring period is used to update the weights of the deep neural network, which is then applied to the remaining data resulting in mean and median distances of 227 and 182 m, respectively, compared to the results of a good localization method. In the second scenario, the deep neural network is updated daily with previously detected and located events and applied to the events detected the following day. Since the observed data used for training generally do not cover a wide range of source locations, we enrich the training set with synthetic data. The addition of synthetics for transfer learning ensures that the updated deep neural network provides accurate source locations for events with locations far from locations used during transfer learning. Transfer learning combining synthetic and real data performs significantly better (more consistent) locations than transfer learning without synthetics.

Key words: 3D, Neural network, Microseismic monitoring.

INTRODUCTION

The increasing demand for underground-related energy resources, such as geothermal and unconventional oil and gas reservoirs, as well as the growing interest in CO_2 sequestration and hydrogen storage, requires reliable and fast methods to monitor the seismic activity around the reservoir to both optimize the underlying task and mitigate risks associated with induced earthquakes (Gaucher *et al.*, 2015; Li *et al.*, 2020). Most of the seismicity associated with these activities are weak in

*E-mail:n.a.vinard@tudelft.nl

moment magnitudes, M_w, that is around and below zero (Van Der Baan *et al.*, 2013), and are called microseismic events as they are not felt at the surface. Therefore, the signal-to-noise ratio (S/N) is also poor, especially when detected by sensors close to the surface (Li *et al.*, 2019). Microseismic monitoring systems are set in place to detect, localize and estimate the source mechanisms and magnitudes of the induced events. The hypocentre locations provide information about the hydraulically stimulated reservoir volume, or they can be used to identify pre-existing fault systems.

In recent years, several machine learning and deep learning (DL) approaches have been proposed to identify hypocentre locations. One area of application is focused on using machine learning and DL-based picking algorithms that are able to pick arrival times nearly as good or even better than an analyst at a fraction of the time required for manual picking (Ma *et al.*, 2020; Ross *et al.*, 2018b, 2018a; Zhou *et al.*, 2019; Zhu and Beroza, 2019; Zhu *et al.*, 2019; Zhang *et al.*, 2020a). Ross *et al.* (2018b) trained a convolutional neural network (CNN) on earthquakes with labelled P-wave picks and first-motion polarities to first detect the onset of the P-wave and then determine the polarity of the P-wave. Ross *et al.* (2018a) trained a CNN on millions of three-component hand-labelled seismic records, which were split into records containing only P-waves, S-waves and noise. The CNN was trained to classify the input as a P-wave, S-wave or noise. Zhou *et al.* (2019) use a CNN to first detect earthquakes and then pass the detected waveforms to a recurrent neural network to pick P- and S-wave arrivals. PhaseNet (Zhu and Beroza, 2019) is a modified U-Net architecture (Ronneberger *et al.*, 2015) applying one-dimensional (1D) convolutions over three-component seismic waveforms that return probabilities around the P-wave and S-wave arrivals and noise. Zhu *et al.* (2019) develop a CNN that can be trained on smaller training sets compared to previous works that can be applied for P- and S-wave picking of aftershocks. Zhang *et al.* (2020a) trained a CNN to classify waveforms and arrival time picking for microseismic data. To train the CNN, they first convert the signal into the time-frequency domain using the continuous wavelet transform. Ma *et al.* (2020) propose a U-Net architecture for P- and S-wave classification on microseismic three-component data. First, the data are preprocessed such that the waveforms show clearer arrival times. Next, waveforms are converted to grey-scale images and fed to the U-Net to pick the S- and P-phase arrivals.

Other DL algorithms directly return the source locations without picking wave arrivals and directly relate observed waveforms to locations (Kriegerowski *et al.*, 2019; Mousavi and Beroza, 2020; Van den Ende and Ampuero, 2020; Zhang *et al.*, 2020b). Kriegerowski *et al.* (2019) accomplished this by training a CNN taking three-component seismic waveforms from several stations as input and outputting the source locations in terms of their (x,y,z)-coordinates. Zhang *et al.* (2020b) trained a deep neural network (DNN) with three-component waveforms from multiple stations as input that returns the source locations in terms of a three-dimensional (3D) probability density function. Van den Ende and Ampuero (2020) propose the use of graph neural networks, which incorporate the spatial information of the seismic stations in addition to the seismic waveforms to determine the location of the earthquakes as well as their magnitudes. Mousavi and Beroza (2020) trained Bayesian neural networks to estimate the location of earthquakes from single stations.

A considerable drawback of DL methods is that large training datasets, which sample the model space well, are usually required to reach good performances. This limitation can be overcome by using a previously trained DNN and refining it using a much smaller dataset. This is known as transfer learning (TL) and is based on the idea that DNNs applied to similar tasks share common features (Pan and Yang, 2009). In the field of geophysics, TL has been applied to a variety of problems. El Zini *et al.* (2019) used TL to detect bright spots in seismic data by first pre-training a CNN on unlabelled seismic data (unlabelled meaning that the information on whether a bright spot is or is not present in the input is missing) and then fine-tuning the network on a much smaller labelled dataset. By pre-training their CNN on unlabelled data, they circumvent the constraint of labelled datasets required for supervised machine learning tasks. Chai *et al.* (2020) used a phase picker previously trained on 0.7 million local earthquakes (tens of kilometre distances between sources and receivers) (Zhu and Beroza, 2019) and refined it to get better picks for microseismic data recorded from a metre-scale project. This was achieved using a small training dataset of 3500 seismograms. In other works, TL was used by pre-training DNNs with large synthetic datasets and then fine-tuning the DNNs with field data. This has been done for the task of seismic trace interpolation using a convolutional denoising autoencoder (Wang *et al.*, 2020) and for seismic fault detection using a CNN (Cunha *et al.*, 2020).

In this work, we apply TL to localize weak microseismic events using waveforms as input. As a starting point, we use a DNN that was trained with a large synthetic dataset and applied to a small field dataset to retrieve the source locations of hydraulic-fracturing (HF) induced earthquakes (Vinard *et al.*, 2022). This was achieved using a modified version of a U-Net (Ronneberger *et al.*, 2015), a type of CNN originally

developed for image segmentation, which is composed of both an encoder and decoder. The encoder extracts useful features in the input (waveforms), and the decoder maps the extracted features into a 3D Gaussian distribution of location probability (Vinard *et al*., 2022). In the following, we refer to the DNN that was trained on synthetic data as QNetSynth, where Q stands for quake and Synth for synthetic. QNetSynth reliably localizes the higher magnitude events; however, it fails to accurately localize lower magnitude events. This is problematic for monitoring applications where the majority of the events are low in magnitude, such as those observed in HF monitoring. To improve QNetSynth's performance, we apply transfer learning by updating it with field data. We refer to this updated version as QNet. Furthermore, we aim for QNet to return more consistent locations compared to the diffraction stacking locations.

To train the DNN, we need labelled data, meaning that we need the input (waveforms) and the known output (source locations), also called the label. This type of learning with labelled data is called supervised learning. After training, the QNet can receive new waveforms as input and return source locations.

Training a DNN with synthetic data for the task of microseismic source localization can be useful in retrieving initial locations of seismic events on real datasets. However, for low S/N events such training is insufficient. To increase the source location accuracy of the DNN trained on synthetic data, TL using field data is investigated. Furthermore, we are interested in both modes of processing: (1) post-monitoring processing when data are processed after being acquired and (2) continuous processing when the data are acquired while being processed (e.g. near-real-time or real-time processing).

In the next section, we explain the transfer learning process in more detail and how the DNN is evaluated. We also discuss its application and illustrate how the methodology can be applied to a dataset from a monitored hydraulic fracturing site. The case study is investigated in both a post-monitoring processing mode as well as a continuous acquisition mode. Finally, we discuss limitations and potential of this approach for future applications. Note that we do not necessarily aim at improving the quality of the locations, but its automation and consistency.

TRANSFER LEARNING

In this section, we describe the transfer learning (TL) process used to predict the source locations using the synthetically trained deep neural network (DNN) and QNetSynth. The labelled training data consist of input-output pairs, which for QNetSynth consisted of synthetic seismic waveforms as input and their corresponding source locations represented as three-dimensional (3D) Gaussian distributions as output. The peak of the Gaussian distribution is taken at the source location, (x_s, y_s, z_s) of the event, and the standard deviation, σ, has a fixed (input) value in all directions independent of the input data. The Gaussian distribution is defined as

$$s(x, y, z) = \exp\left(-\left(\frac{(x-x_s)^2}{2\sigma^2} + \frac{(y-y_s)^2}{2\sigma^2} + \frac{(z-z_s)^2}{2\sigma^2}\right)\right). \quad (1)$$

The values in the output range from 0 to 1. In supervised learning, the weights of a DNN are optimized by minimizing a loss function that computes the difference between the label and the output generated by the DNN based on the current weights. After training, the DNN can be applied to new (unlabelled) data to return 3D Gaussian distributions. If the weights of the DNN can extract the relevant features from the input with ease, the returned Gaussian distribution will have a peak value of 1 at the location expected by the DNN.

The architecture of QNetSynth is shown in Figure 1. QNetSynth consists of convolutional layers in the encoder where the input is gradually down-sampled as it moves further down the convolutional layers. In the decoder, transposed convolutional layers gradually up-sample the previously down-sampled input. Additionally, the architecture contains a few skip connections that pass the output from layers in the encoder to the decoder by concatenating the encoder output with the decoder output. The Adam algorithm (Kingma and Ba, 2015) was used to train QNeSynth using a learning rate (step size) of 0.001, a batch size of 20 and the sigmoid cross-entropy loss function. The height and width of all filters were set to 3, and the rectified linear unit (Nair and Hinton, 2010) was used as the activation function. Furthermore, the standard deviation of the 3D Gaussian distribution was selected as 200 m. This is a hyperparameter that needs to be selected before training, and it represents a trade-off between the resolution and training convergence. For more details about QNetSynth, we refer to Vinard *et al*. (2022).

Transfer learning for post-monitoring and continuous acquisition mode

We investigate two possible applications, one that is suitable for post-monitoring processing and the other that represents a continuous acquisition mode. In both scenarios, alternative detection and localization methods are used to build a field training set that can be used to update the weights of

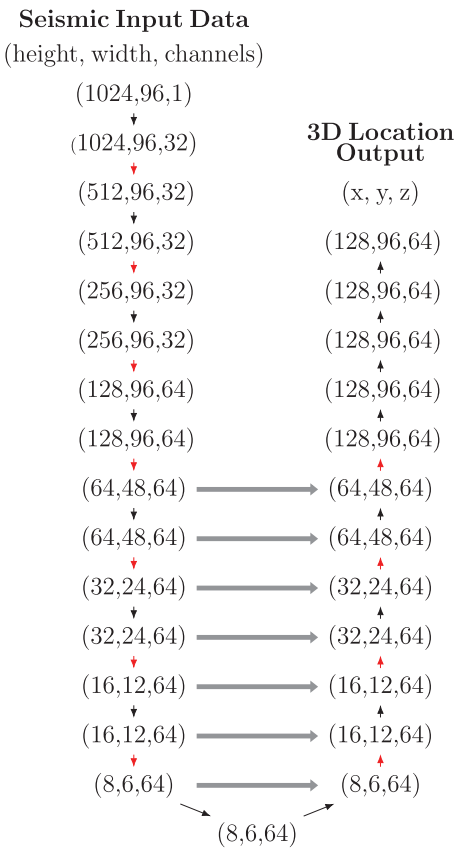

Figure 1 DNN architecture with seismic data as input and 3D location output. An encoder (left) consists of convolutional layers with red arrows denoting convolutions used for down-sampling. A decoder (right) consists of transposed convolutional layers with red arrows indicating transposed convolutions used for up-sampling. Grey horizontal arrows denote skip connections.

QNetSynth. Alexandrov *et al.* (2020) detected and localized the events used in this study. As the detection and localization algorithm, the diffraction stacking (DS) algorithm (Anikiev *et al.*, 2014) was used and some of the DS-localized events were further improved in a post-processing step by a relative location (RL) method (Grechka *et al.*, 2015) that requires a set of master events. We refer to the latter method as DSRL. For the post-monitoring scenario, we build a training set with a subset of the DS-detected events as input (waveforms) and we use the DS locations to create the corresponding labels. QNetSynth is then updated using that training set, and this updated network, QNet, is then applied to the remaining detected events. The QNet predicted locations are then compared to the DSRL locations. In the continuous acquisition mode scenario, the hydraulic-fracturing (HF) operations are still ongoing, and events are detected and localized by DS. After the first day of operations, a training set can be built from the DS-detected and DS-localized events, which is then used for TL to update QNetSynth. The updated DNN, QNet, can then be applied on the next day to retrieve locations of the DS-detected events. After each day, the previous QNet can be updated with new DS-detected and DS-localized events and applied on the next day together with DS.

For the QNet to return good source locations on new data, the training set needs to be similar to the new data. This is a major challenge as source locations vary and change over time, especially in HF operations. Since the field data used for training may not cover all possible locations, the DNN will be biased towards the locations in the field dataset used for training and fail to generalize to other locations. This would limit the applicability of the method. To overcome this issue, we enrich the training set with synthetic data that were used to train QNetSynth and cover the entire region of interest.

Transfer learning workflow

The general TL workflow is summarized by the flowchart in Figure 2. We start the learning process with QNetSynth, which will be equal to QNetPrev (where Prev stands for previous) in the total TL flow. The training set is a combination of labelled fields and labelled synthetic data. For the case of the labelled field data, the source locations were computed by another method (e.g. diffraction stacking). The synthetic database contains all events that were used to train QNetSynth. Instead of using all of the synthetic data, we randomly pick a subset of n synthetic events and apply data augmentations (random bulk time shifts, random muting of traces, adding field noise) to the synthetics (see also Vinard *et al.* (2022)). The combination of the augmented synthetic dataset with the field data forms the training set that is used to update the weights of QNetPrev in the TL process. Note that at each epoch (training set passed forward and backwards through the network to update the weights) a new subset of n random synthetics is selected from the synthetic database. The training set contains more labelled synthetic data than field data; however, the synthetic data used at each epoch are different due to random sampling of events from the larger synthetic database and random data augmentations, whereas the field data are always the same (we tested different values of n to determine its optimal value). If all of the synthetic data were used for TL at each epoch, the field data would be underrepresented in the training set. This would lead to a highly imbalanced training set (Chawla *et al.*, 2004), and the weights in the updated QNet would be biased towards the synthetic data with few changes in its weights in

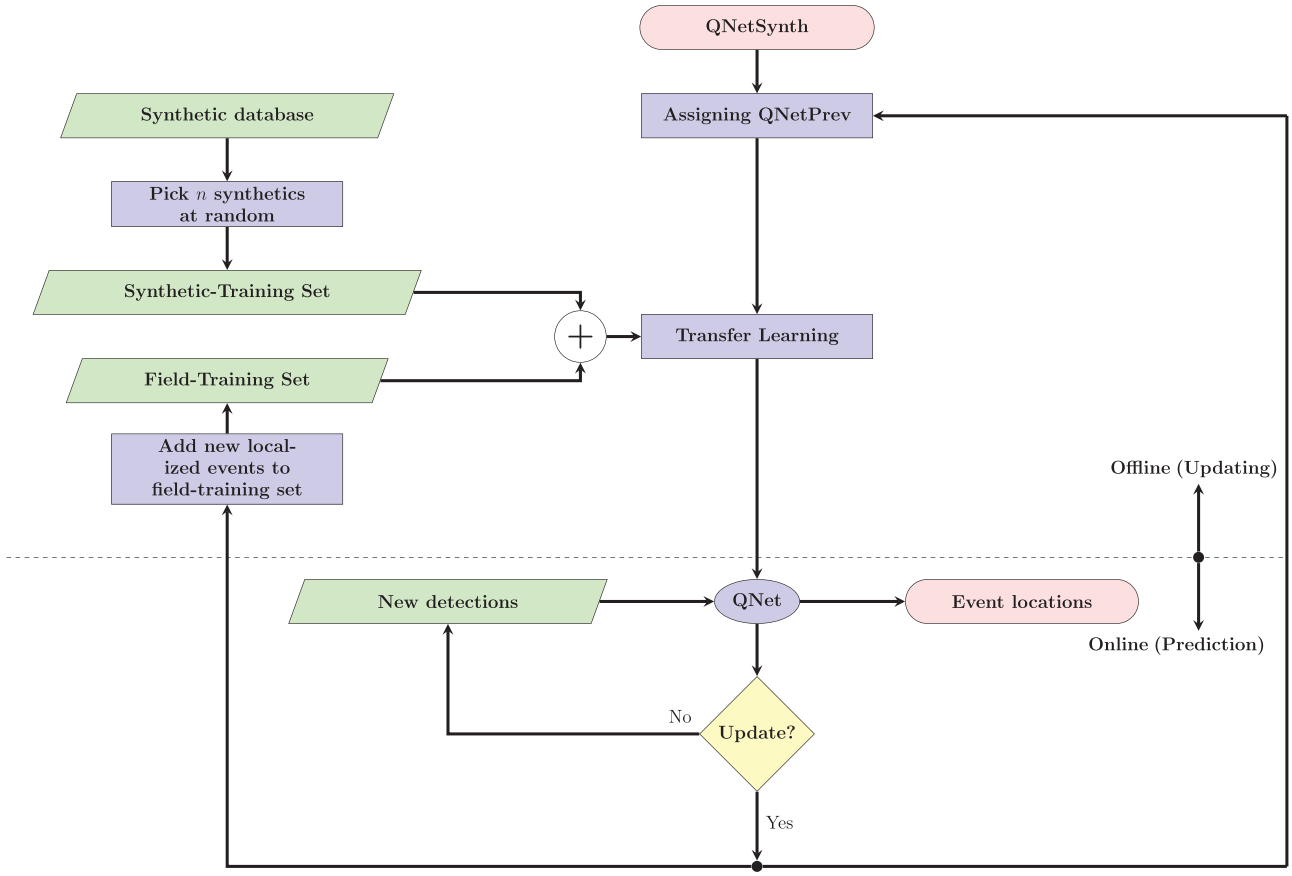

Figure 2 Workflow describing TL to update weights of QNetPrev, applying QNet on new field data and possible further updating of QNet using more field data.

favour of field data. After TL, QNet can be used to reconstruct 3D Gaussian distributions on new field data, where the waveforms are taken as input and the output is a 3D distribution as shown in Figure 3. The TL process can be repeated whenever an updating condition is met. In that case, the field dataset used in training is enlarged with new events and QNet is set as QNetPrev to repeat the TL process with the newest set of weights. Again, here we take the source locations determined by another method (e.g. DS) to create the labels for the new set of events.

Updating QNetPrev only with the field dataset could lead to overfitting, meaning that QNet can produce very good results on the training data but fail to accurately locate events in areas where there were no samples in the training data. Combining TL with both the field and the synthetic data helps to reduce overfitting.

The data augmentations applied to the synthetic data are used to increase the size of the dataset and also help in reducing overfitting. Random bulk time shifts and random muting of traces are augmentations that are easy to implement and also help the learning process by creating variability in the data. The addition of field noise on top of the synthetics was shown to be crucial for QNetSynth to localize field data events (Vinard *et al.*, 2022).

Transfer learning training and evaluation

During TL, we allow all the weights of QNetPrev to change. Freezing parts of the weights during TL, that is preventing those weights to be updated during TL, did not result in noticeable changes. Thus, we decided to allow all of the weights to change. We use Tensorflow software (Abadi *et al.*, 2015) for training using the Adam optimizer (Kingma and Ba, 2015). We set the learning rate (step size) to 0.001 and use a batch size of 20 (the number of training examples that are passed forward and backward through the network to update the weights). The same learning rate and batch size were used to train QNetSynth (Vinard *et al.*, 2022). Note that the

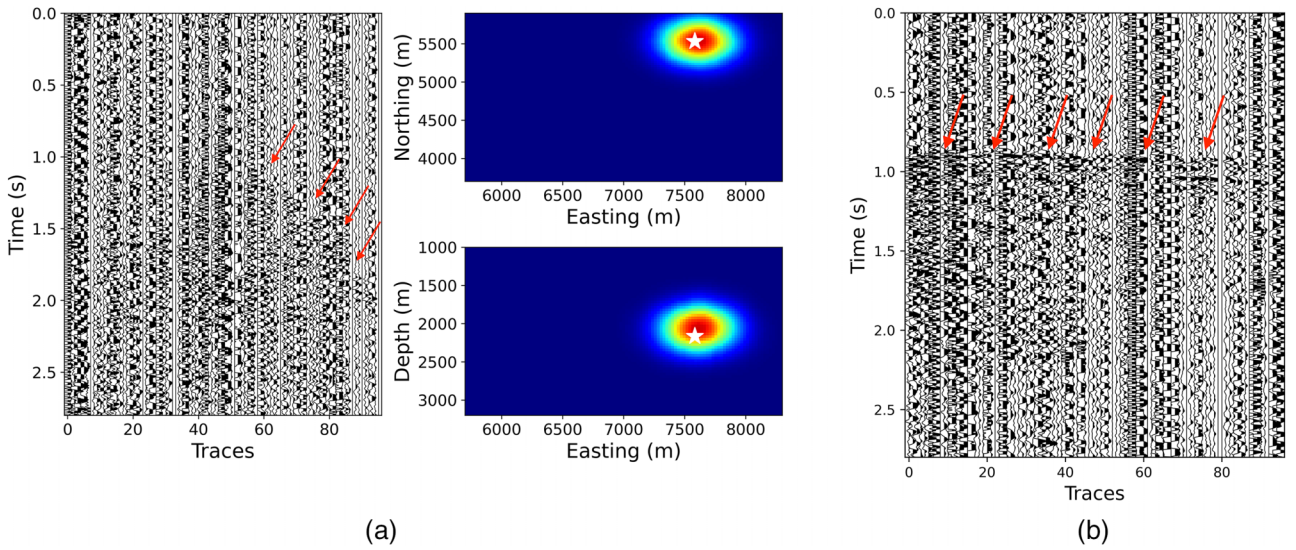

Figure 3 (a) Example of waveform input to QNet on the left and its output (cross sections taken at the maximum voxel in the map view and the view from the south) on the right. The white star denotes the location computed by DS. Clipping is applied for visualization purposes only. (b) A normal-moveout corrected version of the input waveform, created using the 1D velocity model used for DS to better visualize the signal.

input data (waveforms) are always (not only during training) normalized by their maximum amplitude value before being passed to the QNet.

During TL, we set a fixed number of epochs over which to update the weights of QNetPrev. We compute a metric between the output generated by the DNN and the expected 3D Gaussian distribution on the validation set at the end of every epoch, and after training we save the weights that maximized that metric over all epochs. As a metric, we use the Dice similarity coefficient (C_{Dice}) (Dice, 1945), which is defined as

$$C_{\text{Dice}}(t,\hat{t}) = 2\frac{t \cap \hat{t}}{t + \hat{t}}, \tag{2}$$

where t is the label, that is the 3D-Gaussian distribution defined with its peak at the location given by the DS, and $\hat{t}$ is the 3D output distribution produced by the QNet. As in Vinard *et al.* (2022), we clip the values in both t and $\hat{t}$ above 0.1 to 1 and the rest to 0 before computing the C_{Dice}. Thus, if there is a perfect overlap the C_{Dice} value equals one and if there is no overlap it is zero. As the loss function, we use the sigmoid cross-entropy loss that was also used to train QNetSynth.

DATA

The field data used in this study were acquired in Texas, USA, in 2010, during hydraulic-fracturing operations in the Barnett Shale Formation in the Fort Worth basin. The monitoring system used 543 vertical component geophones buried in shallow boreholes, where each borehole contained three geophones placed at 30, 45 and 60 m below the surface. The system covered an area of approximately 144 km^2. Alexandrov *et al.* (2020) generated a one-dimensional (1D) layered P-velocity model from the site from sonic logs and computed hypocentre locations of the events using a migration-type diffraction-stacking (DS) technique (Anikiev *et al.*, 2014) and further refined the location of some events using a relative location (RL) method (Grechka *et al.*, 2015) using a set of 27 master events. The diffraction stacking and relative location, DSRL, method improved the depth estimates of the events, relocating them closer to the injection wells located between 2000 and 2200 m depth. However, the relative locations can only be computed after the whole monitoring period since it requires a set of master events, which are usually only available for postprocessing. This is why the DS locations are used to create the labels during training. However, after training the deep neural network's (DNN) performance is compared to the DSRL locations.

The signal-to-noise ratio (S/N) of the events in our dataset is very low with the majority of events having an S/N below 1 dB, as summarized in Table 1. As Table 1 reveals, 622 events have an S/N ratio below 0.77 dB, implying that on most traces the signal is below the noise level. The signal for such events is enhanced by diffraction stacking allowing those events to be detected and located.

Table 1 S/N statistics of the 1245 field data events in dB, as taken from the S/N computed by (Alexandrov *et al*., 2020)

Mean	Std.	Min.	25%	50%	75%	Max.
0.87	0.55	0.40	0.64	0.77	0.96	9.31

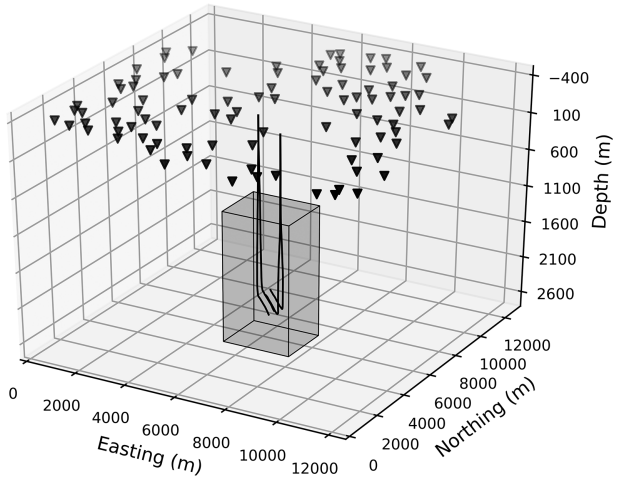

Figure 4 Receiver locations (black triangles) and regions where events can occur (shaded cuboid) extending from 5700 to 8300 m in Easting, 3700 to 5900 m in Northing and 1200 to 3000 m in depth. For the simulation, this space is increased by 200 m in all directions. Black lines represent the orientation of wells.

Synthetic data

QNetSynth was trained with synthetic data modelled with the reflectivity method (Kennett and Kerry, 1979) with the software ERZSOL3 (Kennett, 2005) using the layered P-velocity model generated by (Alexandrov *et al*., 2020) and with the same geophone locations defined in (Vinard *et al*., 2022). For QNetSynth only 96 geophone locations were used mainly to reduce computational and memory costs. QNetSynth was trained with 51,200 synthetic events that cover the entire event region. This region as well as the receiver locations and well locations are shown in Figure 4. Within the region of interest, random double-couple sources with centre frequencies ranging between 20 and 24 Hz were modelled. The synthetics were augmented with field noise, Gaussian noise varying in amplitude per trace, random bulk time shifts and random muting of traces during training. The size of the input data is $(1024 \times 96 \times 1)$, and the three-dimensional (3D) region where events can occur is discretized to a shape of $(128 \times 96 \times 64)$ with a grid size of (23 m, 27 m, 34 m) in Easting, Northing and depth, respectively. The label of each event is defined by its 3D Gaussian distribution with the peak equal to 1 at the source location and with a fixed standard deviation of 200 m in all directions. This fixed standard deviation was chosen for QNetSynth and found to be optimal for good convergence and resolution (Vinard *et al*., 2022). For more details about the synthetic data, we refer to Vinard *et al*. (2022).

Field data preprocessing

For the synthetic data, the label for the field data for training is created in the same way, that is as a 3D Gaussian distribution with a standard deviation of 200 m. For the field data, we use the locations retrieved by DS to create the Gaussian distribution. We apply a band-pass filter of 5–50 Hz to the detected field data (during both training and application of new data). No denoising steps are performed on the data. Finally, note that all of the data used to train the DNN and to make predictions were previously detected and confirmed as true detections by Alexandrov *et al*. (2020).

Field data for post-monitoring application

In order to apply transfer learning, we need labelled field data. Thus a preprocessing step that detects and localizes a number of events is needed. In our case, the preprocessing step was carried out by Alexandrov *et al*. (2020) and 1245 events were detected and located after 9 days of monitoring. We divide these events into field training, validation, and test sets. The labels are created using the DS locations. The field-training and validation sets are used during transfer learning (TL) to update the weights of QNetSynth and to determine the weights that maximized the C_{Dice} (equation 2) on the validation set over all epochs. Finally, the test set is used to apply the updated QNet to data not used during TL.

We randomly split the 1245 events using 60% for training, 20% for validation and 20% for testing. Thus 747 events serve as a field-training set and 249 events each serve as validation and test sets. This partitioning of the data is used for the post-monitoring application. The DSRL-epicentre locations of the 1245 events are shown in Figure 5 together with the well locations.

Field data for continuous acquisition modes

The methodology for continuous acquisition modes is based on dividing the time into intervals (in our case study into days) and using the labelled data from past intervals in TL for the newest time intervals. In this case study, as new events are DS detected and DS localized, we use those to update the

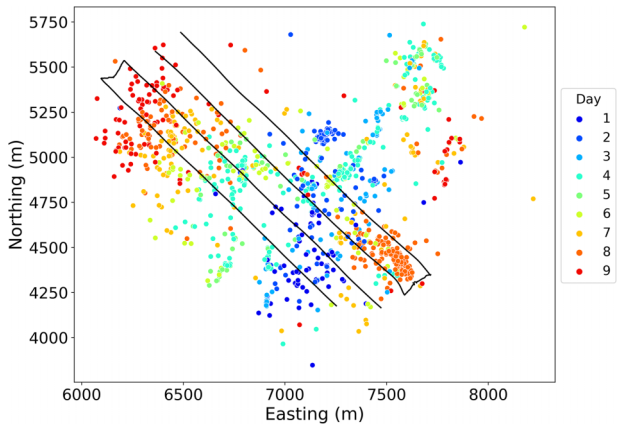

Figure 5 Epicentre locations of 1245 DS-detected and DSRL-localized events from the first to ninth day of monitoring. Each colour represents an event recorded on a particular day. Well positions are shown by black lines.

weights of the current model and apply it to events that are DS detected the following day. Thus, after the first day of monitoring we update QNetSynth with the DS-detected and DS-localized data from the first day and apply the updated QNet to localize the DS-detected events from the second day of monitoring. Next, we updated the QNet after the second day of monitoring with field data events that were DS detected and DS localized during the first two days of monitoring. This process was repeated until the last day of monitoring. Thus, the TL process summarized in Figure 2 is looped once per day. The events detected each day are randomly split into a field training and validation set with an 80/20% ratio. The field-training dataset is used to update the weights of the network in combination with the synthetic data for a fixed number of epochs. At the end of each epoch, we compute the C_{Dice} over the entire validation set and after training we select the weights from the epoch that maximizes the C_{Dice}. It is important to regularly update the DNN due to changing event locations that can affect the performance of the DNN. The changes in DSRL-epicentre locations from the first to the ninth day of monitoring (indicated by colours) are shown in Figure 5. It can be observed that the event locations change over time.

RESULTS

We present the results of both the post-monitoring and the continuous acquisition mode source-localization applications. Starting with QNetSynth, we apply TL to update its weights using a combination of field and synthetic data.

Post-monitoring application

During transfer learning (TL), we use the 747 field data events and synthetic data as training data. As mentioned above, the labels are created using the locations recovered by diffraction stacking (DS). However, we compare the QNet predicted locations to the DSRL locations as those are the more accurate locations. We use 100 epochs for training and choose the weights from the epoch that maximizes the C_{Dice} (equation 2) on the validation set and named the updated deep neural network (DNN), QNet. Increasing the number of epochs did not bring any significant improvements. To create the training set, a limited number of random synthetics were selected from the synthetic database at each epoch. The synthetic database contains 51,200 events, from which we randomly select a subset at each epoch. Thus, at every epoch a new set of synthetic samples is picked and added to the field-data events used for training. We experimented with different numbers and obtained the best results by randomly sampling 2000 events from the synthetic database at each epoch.

After TL, QNet is ready to be applied to the test set. The test set contains 249 events with moment magnitudes between −0.59 and 1.52. In Figure 6, we show the locations retrieved by QNet from the peak of the reconstructed distribution for all 249 events in the test set compared to the DSRL locations. Note that a grid pattern emerges in the DNN-predicted locations, which is due to the discrete 3D output space. In general, the hypocentre locations returned by QNet match well with the DSRL-localized events. The mean distances in the hypocentre, epicentre and depth between the locations provided by QNet and DSRL are 227 m, 148 m, and 141 m, respectively, with a median hypocentre distance of 182 m. The depth locations returned by QNet are concentrated at depths between 2000 and 2200 m for most events. This is also the depth of the fractured interval.

The histogram (Fig. 7) of the location errors, computed as the distances between QNet-located events (from the peak of the distribution) and the DSRL locations shows that a majority of the events are located within 300 m from each other with a sharp decrease in events with distances greater than 300 m away from the DSRL locations in the validation and test sets. We take a closer look at those later events. We compare their magnitudes and signal-to-noise ratios (S/Ns) with the events located less than 300 m from the DSRL locations in the test set as well as to the magnitudes and S/Ns present in the field data used for training. We only plot the magnitudes between –0.6 and 1.0 and S/N between 0.4 and 1.9 in order to better observe the events that were located at greater

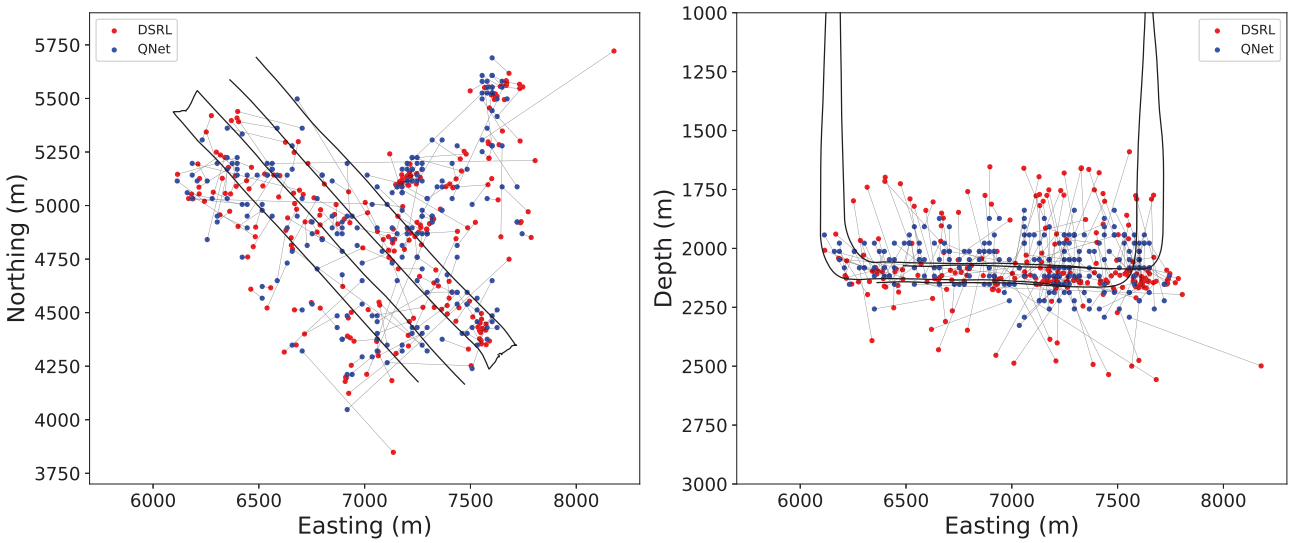

Figure 6 Locations predicted by QNet on the test set for the post-monitoring processing application plotted together with DSRL locations. Lines connect locations of the same event predicted by QNet and DSRL. The left-side plot is the map view, and the right-side plot is the view from the south.

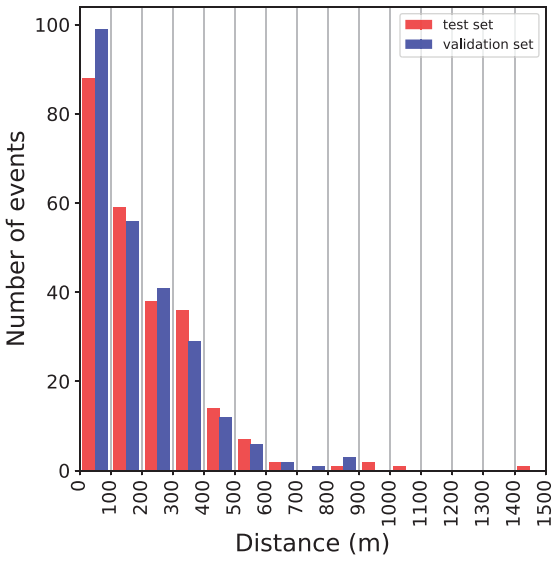

Figure 7 Histogram showing the number of events located in different distance bins of 100 m width as a function of the distances to the DSRL locations in the test and validation sets.

distances from the DSRL locations; see Figure 8. We observe that a majority of the events that are located more than 300 m from each other have low magnitudes and S/N. This is to be expected since there are fewer events within that moment magnitude and S/N range in the field-training set, and thus fewer of those examples that the DNN can learn from.

Continuous acquisition mode

For the continuous acquisition mode, we update the DNN on a daily basis, starting with the DNN that was trained on synthetic data, QNetSynth. For labelling of the detected events, we use the DS locations.

To investigate the added value of the synthetic data, we apply TL excluding and including synthetic data. Starting with QNetSynth and the events detected and located after the first day of monitoring, we follow the scheme described in Figure 2: QNetSynth becomes QNetPrev and the field-data events detected and localized on the preceding day by DS are used to update the weights of QNetPrev. For the situation where synthetic data are also used during TL, we randomly select a new set of 2000 synthetic events at each epoch, as we did for the post-monitoring application. We randomly create splits of 80/20% of the field data events to serve as field training and validation sets, respectively. After training for 100 epochs, we again keep those weights that maximized the C_{Dice} on the validation set. This updated model, QNet, is next applied to data detected on the next day of monitoring. This process is repeated until the last day of monitoring. We refer to the QNets obtained with this iterative TL workflow without synthetics as QNet1 and those updated with synthetics as QNet2. The C_{Dice} and loss curves over the training and validation sets using data after the first day of operations are shown in Figure 9. The curves look similar for the remaining TL iterations. The vertical dashed line shows the epoch at which the C_{Dice}

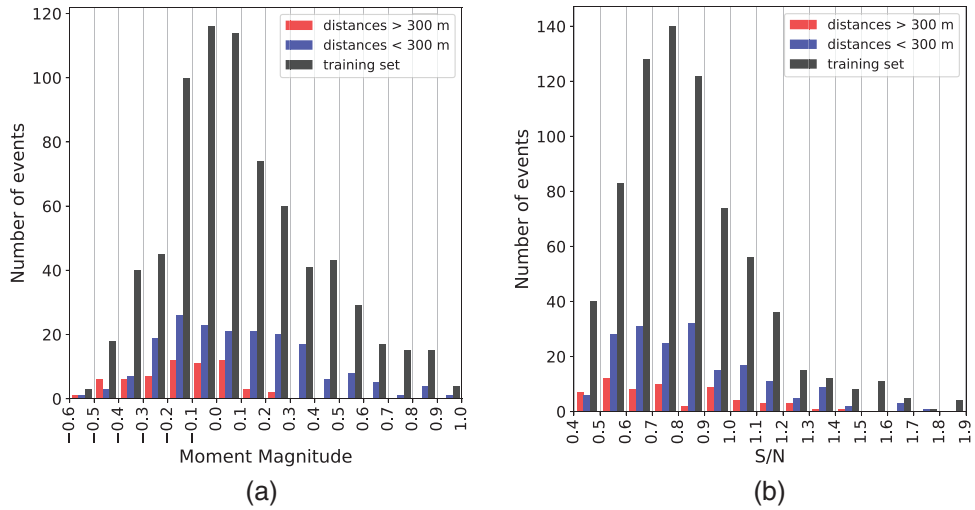

Figure 8 Histograms of moment magnitudes (left) and S/N (right) of events predicted at distances greater than 300 m (red) and less than 300 m (blue) from DSRL locations in the test set and distributions in the training set (black).

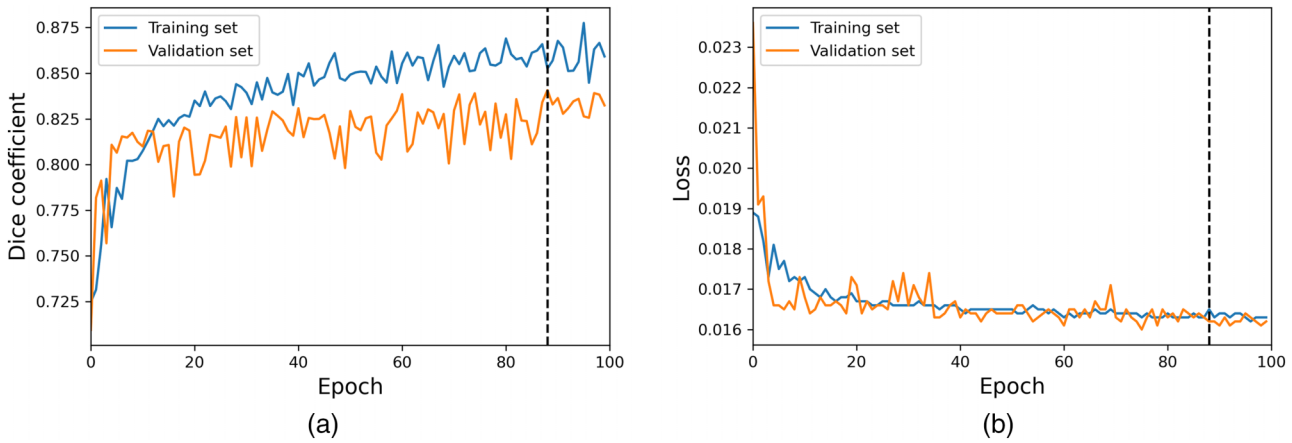

Figure 9 C_{Dice}-curves (left) and loss curves (right) of training and validation sets during the first TL iteration in continuous acquisition mode.

value over the validation set reached its maximum value. The training- and validation-loss curves are close to each other, indicating that the model is not heavily overfitting, and the validation loss in Figure 9 is steadier after 70 epochs.

After each TL iteration, we apply the updated QNets, QNet1 and QNet2, to the same field data detected the following day and compare the results to the DSRL locations of the events. Figure 10 shows the epicentre locations from the second to the ninth day of monitoring separately for QNet1 and QNet2. The red dots are DSRL localizations, and the blue dots represent locations retrieved by the QNets (QNet1 in the first and the third columns and QNet2 in the second and the fourth columns). The lines connect the DS localizations to the locations predicted by the QNets.

With the exception of a few events, the epicentre locations recovered by QNet2 on the second day of monitoring were better compared to the DSRL locations compared to QNet1. The DNN updated without synthetic data during TL, QNet1, mislocates the small cluster of events located in the upper part of the plot (the black circle in Fig. 10). A similar observation can be made for the epicentres from the third day of monitoring, where QNet1 does worse at localizing the small clusters on the upper and lower parts of the plot whereas QNet2 does much better with the exception of a few outliers. These problems can be explained by the lack of training samples in those areas. This can be clearly observed on the fourth day of monitoring when comparing the locations predicted by QNet1 to those of QNet2. As the event locations up to the day used to

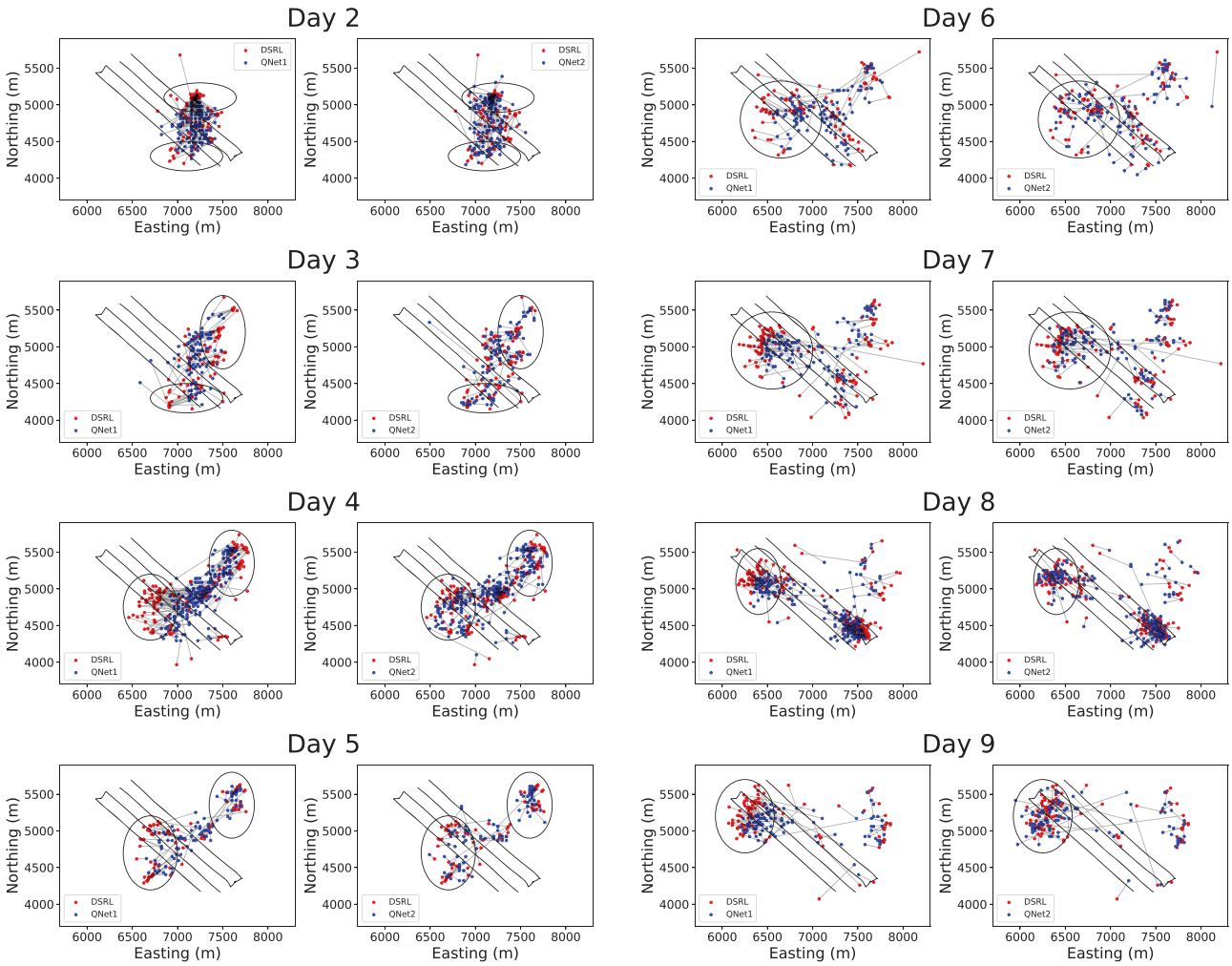

Figure 10 Epicentres returned by QNet1 updated without synthetics (columns 1 and 3) and QNet2 updated with synthetics (columns 2 and 4) from the second to the ninth day of monitoring. DSRL and respective DNN locations (QNet1 and QNet2) are connected by black lines.

update QNetPrev are always slightly different from the event locations from the following day, QNet1 seems to always be lagging behind slightly, as also observed throughout days 5 to 9 of monitoring. QNet2, however, can overcome this issue due to the use of synthetic data, which well sample the locations of interest.

The mean distance between all locations predicted by QNet1 and the DSRL locations is 282 m. The mean epicentre distance is 226 m. For QNet2, the mean hypocentre distance is 249 m and the mean epicentre distance is 167 m. These differences are large; however, note that DS locations and not DSRL locations were used for training, and, furthermore, we cannot be sure that the DSRL locations are true locations. Thus, updating QNetPrev with the synthetic data that cover the entire event region is more consistent with the DSRL-localized events. This is especially important with continuous processing where new events occur in regions where past events did not occur and were therefore not part of the training set used for TL. Figure 5 shows how the events migrate on a daily basis. The depth differences between the DSRL locations and the predictions by QNet1 and QNet2 are similar at 146 and 151 m, respectively.

DISCUSSION

In the introduction, we mentioned that QNetSynth (trained purely with synthetics) failed to localize many of the lower magnitude field-data events. In order to show how transfer learning (TL) helps to improve the localization of field data events, we compare the locations of QNetSynth with the daily

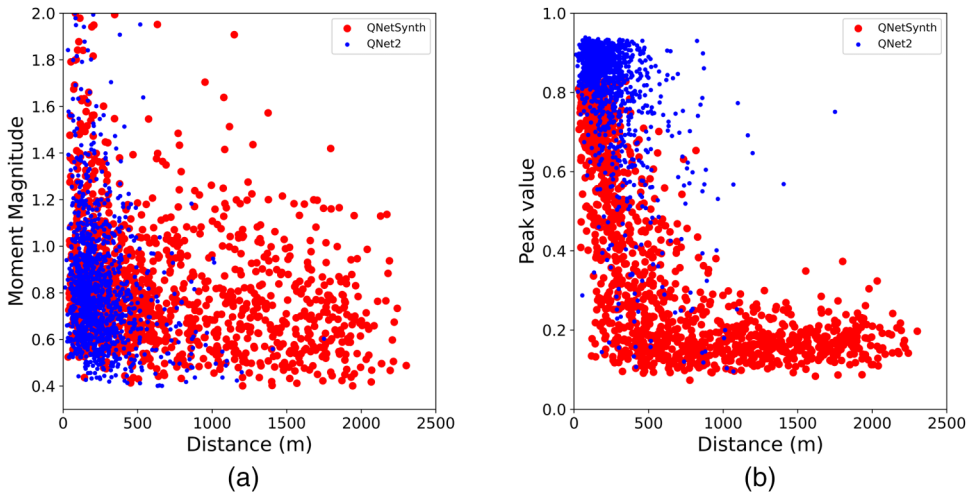

Figure 11 Comparing QNetSynth with QNet2. (a) Moment magnitude versus distance and (b) peak value versus Euclidean distance between the QNet locations and DSRL locations. QNetSynth is displayed by large red dots and QNet2 by small blue dots.

updated versions of QNet2, applied to the events recorded from the second to the ninth day of monitoring. We plot the moment magnitude of the events versus the distance between the DSRL locations and the locations returned by both QNet2 and QNetSynth in Figure 11(a). We can see that QNet2 is able to localize many of the lower magnitude events more accurately than QNetSynth.

We consider the maximum value of the distribution to be at the source location and refer to it as the peak value. We can see from Figure 11(b), where the peak value of the output is plotted with respect to the distance that the peak values are higher for QNet2. Thus by updating QNetSynth with field data (and synthetics) the number of confidently localized events increases. Based on Figure 11(b), we might consider setting a threshold on the peak value to accept only events that are above it. In practice, the threshold should be based on the validation set. For now, we set a threshold to 0.5 for both QNetSynth and QNet2 and compare the moment magnitude distribution of the events that pass the threshold (Fig. 12). We can see that most of the higher magnitude events for QNetSynth passed the threshold but that many of the lower magnitude events did not. Between moment magnitudes 0.6 to 1.6, the same number of events passes the threshold for QNetSynth and QNet2. For magnitudes below 0.6, increasingly more events pass the threshold in the case of QNet2 compared to QNetSynth and no events with M_w below 0.3 pass the threshold for QNetSynth whereas for QNet2 events down to M_w between -0.6 and -0.5 pass the threshold. The mean distance over all events recorded after the first day of monitoring is 688 m for QNetSynth and 249 m for QNet2. The

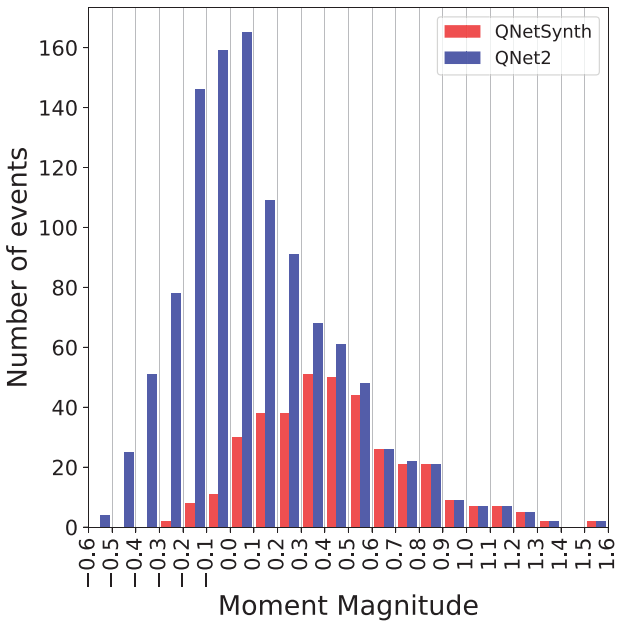

Figure 12 Magnitude distributions of events passing threshold 0.5 for QNetSynth (red) and QNet2 (blue).

localization improved similarly comparing QNetSynth to QNet: Over the test set of 249 events, the mean distance between QNetSynth locations and DSRL-locations is 747 m, whereas for QNet the mean distance is 227 m.

In this study, we focus our attention on the locations of detected microseismic events. For a practical situation, a detection algorithm needs to be employed to first detect an event. In this study, the diffraction stacking algorithm was used for

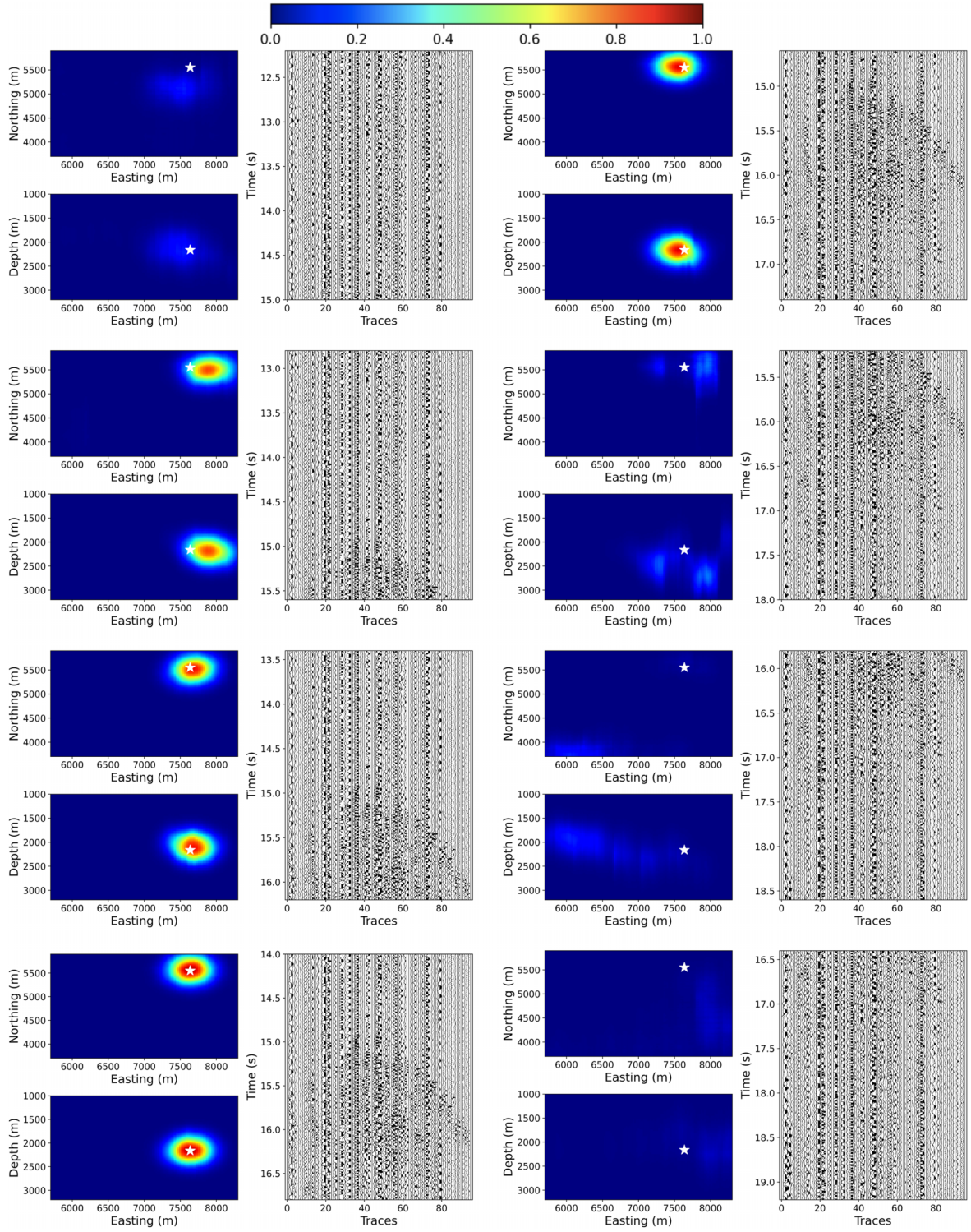

Figure 13 QNet2 trained on the first four days of field data applied to events recorded on thesixth day. QNet2 receives seismic data as input (records) and returns the 3D output (plan and section slice through maximum value). The white star denotes the DS location. Time increases from the left column downwards and continues from the upper left column. Clipping was applied to better visualize events in records (not applied during training and prediction).

detection (Anikiev *et al*., 2014). Alternatively, several machine learning based seismic event detection algorithms have been successfully applied in recent years (Dokht *et al*., 2019; Meier *et al*., 2019; Mousavi *et al*., 2019; Perol *et al*., 2018; Wu *et al*., 2018).

During the supervised learning phase, our deep neural network (DNN) was only trained with data containing events and it learned to extract features from that input and map it into a three-dimensional (3D) Gaussian distribution. Thus, the QNet was not fully trained to differentiate between noise and seismic events. However, if the input to the QNet contains noise only, we do not expect it to return a 3D Gaussian distribution with a high peak value. Therefore, the QNet might serve as an event detector. To investigate this possibility, we took a time window of approximately 7 s around an event recorded on the sixth day of monitoring. Next, we pass QNet2 (trained with iterative TL up to day 5) chunks of 2.8 s each shifted by 0.6 s from start to finish. The seismic data that are used as input to QNet2 as well as its output are shown in Figure 13. The predicted output is sliced horizontally and vertically through the maximum output voxel. The peak of the output distribution corresponding to noise is significantly below 1 before the signal enters the time window, and the output cannot be characterized as Gaussian. As soon as the signal appears on the first few receivers, the distribution's peak value is significantly higher and resembles a Gaussian distribution. However, the peak of the distribution does not yet match with the diffraction stacking (DS) location. In the two consecutive time windows, the peak of the distribution is on top of the DS location and the peak value is at its highest. Finally, at the later time steps, as the first signals start passing the receiver array, the distribution starts to change and eventually dissipates as no signals are present in the input. We believe this methodology can be extended to provide the detection of seismic events.

The synthetic data used to train QNetSynth and also used in the TL scheme were generated using the same velocity model as used to localize the events by diffraction stacking. The velocity model may not always be well known. If the velocity is complex then to create the synthetics, we may need to use a more computationally intensive method to compute seismograms. Therefore, analysing how accurate the velocity models need to be in order to train a DNN that is able to provide good locations for field data is a recommendation for further research. As is known from other methods, the accuracy of the locations depends on the velocity model and we expect this to be the case for this method as well.

In this study, we benefited from a well-known velocity model. However, in general, the velocity model, especially if used for simulating full waveform synthetic seismograms, may not be well known. Further investigation of the accuracy of the velocity model may help us to understand limitations of the proposed methodology, but this is beyond the scope of this study as we need to define the quality of the velocity model for better judgement.

While creating our training, validation and test sets from the field data, we randomly created the splits. Thus all three sets roughly cover the same moment magnitudes. It would be interesting to investigate the possibility of applying TL using high magnitude events in the first run and applying the updated DNN to low magnitude events to test if it can extrapolate its feature extraction and classification capacities to those events.

In order to get an idea about the computational effort needed for our approaches, generating 51,200 synthetics and running 100 simulations in parallel takes roughly 7 days on 2.3 GHz Intel Xeon CPUs. The training time depends on the size of the training set. A single epoch took approximately 60 s on a Tesla P100-PCIE-16GB GPU. Thus training for 100 epochs takes about 1.7 hours. Finally, applying the trained QNet on a single event to generate the output takes 0.28 s on a 3.1-GHz Dual-Core Intel Core i5 CPU. Hereby, we show that it is feasible to apply this method on a daily basis.

CONCLUSIONS

In this work, we introduced a transfer learning (TL) scheme to update a deep neural network previously trained on synthetic data. The TL scheme can be either used a single time in a post-monitoring situation or iteratively for continuous monitoring. In the TL scheme, the QNet is updated using a combination of labelled fields and labelled synthetic data. By updating the QNet in this fashion, the number of confidently localized field-data events at low magnitudes drastically increased. Furthermore, we showed the importance of keeping the synthetic data during TL in order to provide accurate source locations in areas not yet covered by the field data used during training. Additionally, we provide a framework to regularly apply TL in a continuous data processing mode, which increases the localization performance of the QNet over time.

ACKNOWLEDGEMENTS

This research is supported by the Polish National Science Centre grant no UMO-2018/30/Q/ST10/00680.

DATA AVAILABILITY STATEMENT

The field data that support the findings of this study are available from Dmitry Alexandrov (dmitry.alexandrov@seismik.cz). The synthetic data that support the findings of this study are available from the corresponding author upon reasonable request. The trained deep neural network models and the codes that support the findings of this study are openly available at https://github.com/nvinard/QNet.

REFERENCES

Alexandrov, D., Eisner, L., Waheed, U.b., Kaka, S. I.E. and Greenhalgh, S.A. (2020) Normal faulting activated by hydraulic fracturing: a case study from the Barnett Shale, Fort Worth Basin. *The Leading Edge*, 39(3), 204–211.

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. and Zheng, X. (2015) TensorFlow: large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Anikiev, D., Valenta, J., Staněk, F. and Eisner, L. (2014) Joint location and source mechanism inversion of microseismic events: Benchmarking on seismicity induced by hydraulic fracturing. *Geophysical Journal International*, 198(1), 249–258.

Chai, C., Maceira, M., Santos-Villalobos, H.J., Venkatakrishnan, S.V., Schoenball, M., Zhu, W., Beroza, G.C., Thurber, C. and Team, E.C. (2020) Using a deep neural network and transfer learning to bridge scales for seismic phase picking. *Geophysical Research Letters*, 47(16), e2020GL088651.

Chawla, N.V., Japkowicz, N. and Kotcz, A. (2004) Special issue on learning from imbalanced data sets. *ACM SIGKDD Explorations Newsletter*, 6(1), 1–6.

Cunha, A., Pochet, A., Lopes, H. and Gattass, M. (2020) Seismic fault detection in real data using transfer learning from a convolutional neural network pre-trained with synthetic seismic data. *Computers and Geosciences*, 135, 104344.

Dice, L.R. (1945) Measures of the amount of ecologic association between species. *Ecology*, 26(3), 297–302.

Dokht, R.M., Kao, H., Visser, R. and Smith, B. (2019) Seismic event and phase detection using time–frequency representation and convolutional neural networks. *Seismological Research Letters*, 90(2A), 481–490.

El Zini, J., Rizk, Y. and Awad, M. (2019) A deep transfer learning framework for seismic data analysis: A case study on bright spot detection. *IEEE Transactions on Geoscience and Remote Sensing*, 58(5), 3202–3212.

Gaucher, E., Schoenball, M., Heidbach, O., Zang, A., Fokker, P.A., van Wees, J.-D. and Kohl, T. (2015) Induced seismicity in geothermal reservoirs: a review of forecasting approaches. *Renewable and Sustainable Energy Reviews*, 52, 1473–1490.

Grechka, V., De La Pena, A., Schisselé-Rebel, E., Auger, E. and Roux, P.-F. (2015) Relative location of microseismicity. *Geophysics*, 80(6), WC1–WC9.

Kennett, B.L. (2005) ERZSOL3 (reflectivity method). http://www.quest-itn.org/library/software/reflectivity-method.html [Accessed: 30 March 2020].

Kennett, B.L. and Kerry, N.J. (1979) Seismic waves in a stratified half space. *Geophysical Journal of the Royal Astronomical Society*, 57(3), 557–583.

Kingma, D.P. and Ba, J.L. (2015) Adam: A method for stochastic optimization. In: *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. ICLR, pp. 1–15

Kriegerowski, M., Petersen, G.M., Vasyura-Bathke, H. and Ohrnberger, M. (2019) A deep convolutional neural network for localization of clustered earthquakes based on multistation full waveforms. *Seismological Research Letters*, 90(2 A), 510–516.

Li, L., Tan, J., Schwarz, B., Staněk, F., Poiata, N., Shi, P., Diekmann, L., Eisner, L. and Gajewski, D. (2020) Recent advances and challenges of waveform-based seismic location methods at multiple scales. *Reviews of Geophysics*, 58(1), e2019RG000667.

Li, L., Tan, J., Wood, D.A., Zhao, Z., Becker, D., Lyu, Q., Shu, B. and Chen, H. (2019) A review of the current status of induced seismicity monitoring for hydraulic fracturing in unconventional tight oil and gas reservoirs. *Fuel*, 242, 195–210.

Ma, Y., Cao, S., Rector, J.W. and Zhang, Z. (2020) Automated arrival-time picking using a pixel-level network. *Geophysics*, 85(5), V415–V423.

Meier, M.-A., Ross, Z.E., Ramachandran, A., Balakrishna, A., Nair, S., Kundzicz, P., Li, Z., Andrews, J., Hauksson, E. and Yue, Y. (2019) Reliable real-time seismic signal/noise discrimination with machine learning. *Journal of Geophysical Research: Solid Earth*, 124(1), 788–800.

Mousavi, S.M. and Beroza, G.C. (2020) Bayesian-deep-learning estimation of earthquake location from single-station observations. *IEEE Transactions on Geoscience and Remote Sensing*, 58(11), 8211–8224.

Mousavi, S.M., Zhu, W., Sheng, Y. and Beroza, G.C. (2019) Cred: a deep residual network of convolutional and recurrent units for earthquake signal detection. *Scientific reports*, 9(1), 1–14.

Nair, V. and Hinton, G.E. (2010) Rectified linear units improve restricted Boltzmann machines. In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. Omnipress, pp. 807–814.

Pan, S.J. and Yang, Q. (2009) A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10), 1345–1359.

Perol, T., Gharbi, M. and Denolle, M. (2018) Convolutional neural network for earthquake detection and location. *Science Advances*, 4(2), 2–10.

Ronneberger, O., Fischer, P. and Brox, T. (2015) U-net: Convolutional networks for biomedical image segmentation. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, pp. 234–241.

Ross, Zachary E, M. M.-A., Hauksson, E. and Heaton, T.H. (2018a) Generalized seismic phase detection with deep learning. *Bulletin of the Seismological Society of America*, 108(5A), 2894–2901.

Ross, Z.E., Meier, M.-A. and Hauksson, E. (2018b) P wave arrival picking and first-motion polarity determination with deep learning. *Journal of Geophysical Research: Solid Earth*, 123(6), 5120–5129.

Van den Ende, M.P. and Ampuero, J.-P. (2020) Automated seismic source characterization using deep graph neural networks. *Geophysical Research Letters*, 47(17), e2020GL088690.

Van Der Baan, M., Eaton, D., Dusseault, M., et al. (2013) Microseismic monitoring developments in hydraulic fracture stimulation. In *ISRM International Conference for Effective and Sustainable Hydraulic Fracturing*. International Society for Rock Mechanics and Rock Engineering, 440–466.

Vinard, N., Drijkoningen, G. and Verschuur, D. (2022) Localizing microseismic events on field data using a U-Net-based convolutional neural network trained on synthetic data. *Geophysics*, 87(2), KS33–KS43.

Wang, Y., Wang, B., Tu, N. and Geng, J. (2020) Seismic trace interpolation for irregularly spatial sampled data using convolutional autoencoder. *Geophysics*, 85(2), V119–V130.

Wu, Y., Lin, Y., Zhou, Z., Bolton, D.C., Liu, J. and Johnson, P. (2018) DeepDetect: a cascaded region-based densely connected network for seismic event detection. *IEEE Transactions on Geoscience and Remote Sensing*, 57(1), 62–75.

Zhang, G., Lin, C. and Chen, Y. (2020a) Convolutional neural networks for microseismic waveform classification and arrival picking. *Geophysics*, 85(4), WA227–WA240.

Zhang, X., Zhang, J., Yuan, C., Liu, S., Chen, Z. and Li, W. (2020b) Locating induced earthquakes with a network of seismic stations in Oklahoma via a deep learning method. *Scientific Reports*, 10(1), 1–14.

Zhou, Y., Yue, H., Kong, Q. and Zhou, S. (2019) Hybrid event detection and phase-picking algorithm using convolutional and recurrent neural networks. *Seismological Research Letters*, 90(3), 1079–1087.

Zhu, L., Peng, Z., McClellan, J., Li, C., Yao, D., Li, Z. and Fang, L. (2019) Deep learning for seismic phase detection and picking in the aftershock zone of 2008 Mw7. 9 Wenchuan Earthquake. *Physics of the Earth and Planetary Interiors*, 293, 106261.

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

