MLReal: Bridging the gap between training on synthetic data and real data applications in machine learning

Tariq Alkhalifah, Hanchen Wang, and Oleg Ovcharenko, KAUST

ABSTRACT
Among the biggest challenges we face in utilizing neural networks for seismic data is the application on field data. The requirement for accurate labels often forces us to train our networks using synthetic data, where such information is readily available. However, the synthetic experiments often do not reflect the reality of the field experiment, and we end up with poor performance of the trained neural network (NN) models at the inference stage. This is because synthetic data lack many of the realistic features embedded in the real data. In other words, the real data set is far from being a sample from the distribution of the synthetic training set. Thus, we describe a novel approach to enhance our neural network model training on synthetic data with real data features (domain adaptation). This is accomplished by applying two operations on the input data to the NN model, whether they are from the synthetic or real data subset class: 1) The crosscorrelation of the input data section (i.e. shot gather or seismic image) with a fixed reference trace from that section. 2) The convolution of the resulting data with a randomly chosen auto correlated section of the other subset class. In the training stage, as expected, the input data are from the synthetic subset class and the auto-corrected sections are from the real subset class, and the random selection of sections from the real data is implemented at every epoch of the training. In the inference/application stage, the input data are the real subset class and the auto-corrected sections are from the synthetic data subset class. An example application on passive seismic data for microseismic event source location determination is used to demonstrate the power of this approach in improving the applicability of our trained models on real data.
Introduction

Machine learning (ML) is gaining a lot of traction as a tool to help us solve outstanding problems in seismic processing and interpretation. Most of the applications in our field have relied on supervised training of neural network (NN) models, where the labels (answers) are available (Wrona et al., 2018; Araya-Polo et al., 2018; Ovcharenko et al., 2019). These answers are often available for synthetic data as we numerically control the experiment, or they are determined using human interpretation or human crafted algorithms applied to real data. The challenge in training our NN models on synthetic data is the generalization of the trained models to real data, as that process requires careful identification of a training set and the inclusion of realistic noise and other variables between synthetic and real data. In other words, the synthetic and real data are usually far from being drawn from the same distribution, which is essential for the success of a trained NN model (Kouw, 2018). It requires that we accurately reproduce the correlated and uncorrelated noise \((n(t))\) present in the field data, where \(t\) here is the time (or any vertical axis unit like depth). More importantly, it also requires that we properly represent the source wavelet \((s(t))\) and the reflectivity \((r(t))\). In other words, for the synthetic data \(d_s = r_s(t) * s_s(t) + n_s(t)\), the distributions of these three functions should cover those for the real data, and this is very hard to accomplish. Here \(\ast\) stands for the convolution process. Thus, many synthetically trained NN models have not performed well on real data.

The concept of trying to bridge the gap between the training and application data in machine learning is referred to as domain adaptation (Kouw, 2018). The classic theory of machine learning assumes that the application data of a trained model come from the same population (sampled from the same distribution) as the training set. So we need the probability distribution of the synthetic dataset, \(p_s(y|x)\), where \(x\) are the inputs (i.e. seismic data), and \(y\) are the labels (i.e. traveltime picks), to equal the probability distribution of the real dataset, \(p_r(y|x)\). One category of data adaptation is referred to as subspace mapping in which we find a transformation, \(T\), that results in the distribution of the training input data to equal that of the testing data. Specifically, \(p_s(T(x)) = p_r(x)\). There are many ways to find the transformation or weights to make the distributions similar including the use of optimal transport. The method proposed in this abstract shares the general concept of this objective implemented in a systematic way. Thus, a trained NN model generalizes well when the testing data are represented, as much as possible, in the training data set. To help accomplish that when the application (testing) data are field seismic data, we propose, here, to inject as much of the field data features into the synthetic data training as possible. This can be accomplished by utilizing a combination of linear operations including crosscorrelation, autocorrelation and convolution between the synthetic and field data. These operations will bring the distributions of the training synthetic dataset, and the testing real dataset closer to each other, which will help the trained model generalize better to real data. The real data generalization example we use here is for an NN model dedicated to locate microseismic sources directly from recorded waveform data. We will share an application on active seismic data for predicting low frequencies at the meeting.

Conditioning synthetic data for training

A trace in the seismic data can be represented by a combination of reflectively, source wavelet and noise, as follows:

\[
d^{ij}(t) = r^{ij}(t) * s^{ij}(t) + n^{ij}(t),
\]

where \(i\) is the index of the trace, and \(j\) is the index of the section in which the trace belongs to whether the section corresponds to a shot gather or a seismic image. Depending on the data, all three components \((r(t), s(t), n(t))\) can vary over traces and sections. For a shot gather for example, often \(r(t)\) changes with moveout, and of course, \(n(t)\) changes from any trace to another. In training a neural network model to work properly on \(d(t)\), we often generate synthetic data, \(d_s\) that hopefully includes a proper representation of these components. We can, however, migrate these components from the real data to the synthetic ones using linear operations, and thus, we define new training data as follows:

\[
d^*_s(t) = d^*_s(t) \otimes d^*_s(t) * d^{ij}(t) \otimes d^{ij}(t),
\]

where \(k\) is the index of a reference trace from the synthetic input data, and \(j\) is a randomly chosen index for a section from the real data whether the section corresponds to a shot gather or a seismic image. The
operator $\otimes$ represent crosscorrelation, and in this equation we have a crosscorrelation between the input synthetic section and a reference trace from that section convolved with a randomly drawn autocorrelated section from the real data. The reference trace can be a near offset trace in the case of a shot gather input, or it can be any trace from the input for a seismic image. The index of the reference trace should not change between sections to maintain the relative relation between sections. The random picked $j$ index varies in the training per epoch to allow for proper distribution of the real data imprint on the training set.

Conversely, when we apply the trained model onto the real data, we prepare the input to the model as follows:

$$d_i^j(t) = d^i(t) \otimes d_k^i(t) * d^i_j(s) \otimes d^i_j(s).$$ (3)

This includes the same operations as in equation 2 with the role of the real and synthetic data reversed. The idea of having two instances of each data in equations 2 and 3 is to balance their contribution into the new training and testing data sets. This way, we match properties of synthetic and field data used for training and inference, respectively. Meanwhile, Figure 1a demonstrates the process of applying equation 2, where the synthetic data were generated for a training of an NN model to predict low frequencies (Ovcharenko et al., 2019). The objective was to improve the full waveform inversion convergence for real marine data by adding low frequency content into the data. On other hand, Figure 1b demonstrates the process of applying equation 3 on real data for an input to the trained model. Note that the resulting shot gathers look similar for the two processes. They are not supposed to look the same as they should be uniquely related to the original synthetic or real data. However, they contain more energy and more features in which the NN model can utilize. We will share the performance of the trained NN model on this dataset at the meeting.

**Figure 1** The workflow chart for the MLReal applied to marine streamer data. a) The process used for producing the training data; b) the process used for producing the testing/application data. The circled cross symbol denotes a crosscorrelation operation, and the star symbol denotes a convolution operation.  

**A microseismic data example**

We will show the impact of the above operations on real data acquired as part of monitoring microseismic events. The passive seismic acquisition was performed using a star configuration of sensors as shown in Figure 2a. Figure 2c shows the real data for one microseismic event. The section includes 10 gathers from the various lines (azimuths) plotted side by side. We were provided a total of 75 of these sections, and the corresponding location of the events determined using alternative methods (we will use here only 10 of them, for better display). These labels (event locations) will serve to evaluate the accuracy of our trained NN model. For more details on the data, we refer the reader to Staněk and Eisner (2017). We were also given a velocity model for the area, which is shown in Figure 2b. Using this velocity model, we employ a finite difference approach to solve the acoustic wave equation and simulate wavefields from 5000 randomly placed seismic sources within the region of interest (the region we expect the real events to be located). The resulting 5000 synthetically recorded sections, using the layout in Figure 2a, and the corresponding event location (labels) are split in a random manner into a training set (4000 samples)
and a validation set (1000 samples). An example synthetic data section, for an event near the one for the real data section, is shown in Figure 2d. If we compare this synthetically generated section with the true one, we can appreciate the large difference between the two data in spite that they share similar general shapes as they originate from a nearby source. We do not have the source time information, which explains the shift between the events in the two sections.

**Figure 2** a) The passive seismic acquisition lines. b) The velocity model estimated in the region, and used here to generate the synthetic data. c) The field recorded data for a single microseismic event along the 10 lines plotted side by side. d) The synthetic data along the same lines from a source near the field data one, which was provided for this event. The time of the source is unknown, which explains the shift.

Applying the operations in equations 2 and 3 on the synthetic and real data, we obtain the sections shown in Figures 3a and 3b, respectively. The reference trace from the input section corresponds to the first trace in each line. We window the part around zero lag to reduce the size of the input-to-the-network data. The two sections look much more alike than those in Figures 2c and 2d. Using sections like that shown in Figure 3a in which the location of the source (as label) is known from modelling, we train a 14-layer convolutional neural network to predict the location of the microseismic source \((x, y, \text{and } z)\). The training was executed over 5000 epochs, single batch, using an SGD optimizer.

**Figure 3** a) The training input data after injecting it with real data information using the proposed method. b) The testing application data after applying similar operations. Both data windowed to the size used for the neural network input.

Then we input 10 real data sections, after applying the operations in equation 3, into the trained model to evaluate the accuracy of the prediction. Figure 4a shows the predicted locations (in blue) and the provided ones (in red) in the region of investigation. The differences are generally small, and can be caused by many factors. For one, the NN model is known to have a bias toward smoothing the output. A very small network will levitate towards the mean of the training labels. Of course, a cure for that is to increase the network size. Another reason for the difference could be the simplified assumptions used in our data simulation for the training of our NN model compared to the modeling approach potentially used in determining the location of the events by the data providers. To further evaluate the improvements in data coherency between training and application using the proposed method, we compute the normalized correlation (cosine distance) between the new testing (real) sections and the corresponding synthetic ones. Figure 4b shows the normalized correlation for sections in which only the crosscorrelation with
a reference trace is used to mitigate the shift (dashed blue), and those for sections in which equations 2 and 3 were used (solid blue). Though the normalized correlation values can range between -1 to 1, the correlations here are positive and reasonably higher for our proposed method, compared to just crosscorrelating with a reference. We also plot in Figure 4b the distances between the predicted sources and provided ones for the same 10 events. These values do not exceed 40 meters. Considering the quality of the data and the layout of the sensors on the surface, the differences are reasonable. With only the crosscorrelation with a reference trace, which was developed earlier (Wang et al., 2020), the location difference averaged around 100 meters for these real data.

![Graph](image)

**Figure 4** a) A 3D plot of the locations of the predicted (blue) and provided (red) source locations for 10 field data events. b) The normalized correlation of the corresponding waveforms for the real and synthetic data of the events after applying the proposed linear operations (solid blue), and after applying only the crosscorrelation with a reference trace to mitigate the shift of the events between the two data (dashed blue). The distances between the provided and predicted source locations (red).

**Conclusions**

We proposed a novel technique to precondition the synthetic training data set for a supervised neural network optimization so that the trained model works better on real seismic data. The concept is based on incorporating as much information from the real data into the training without harming the synthetic data features crucial for the prediction. Considering the two data classes (synthetic and real), we specifically cross correlate an input section from one class of data with a reference trace from that data followed by a convolution with an autocorrelated section from the other class. For training the NN model, the input section is from the synthetic data class, and for the application (testing) of the NN model, the input section is from the real data class. A test of this approach on a microseismic source location task using input waveforms helped us improve the application of the NN model on real data.

**Acknowledgments**

We thank Umair bin Waheed from KFUPM, Frantisek Stanek from Seismik and Claire Birnie and Yuanyuan Li from KAUST for helpful discussions. We thank Microseismic, Inc. and Newfield Exploration Mid-Continent, Inc. for graciously supplying the data. We also thank KAUST for its support.

**REFERENCES**


