Combining classical processing with Machine Learning

Daniel Trad

ABSTRACT

Machine learning (ML) is a powerful tool that has become very useful in many areas of science and seismic applications are not an exception. For example, the uses of ML in interpretation have proven to be very helpful. Also, in seismic processing, we see a significant effort for using ML techniques to help or replace traditional processing. Processing applications, however, present a more difficult challenge. In this report, we will discuss what some of those challenges are, and also we will discuss an environment to research this type of technique, where ML does not replace but rather cooperates with traditional signal processing and inversion. We will present this methodology with several examples: migration, multiple attenuation with Radon transform, near-surface noise (ground roll) suppression and interpolation. For the purpose of illustrating the flexibility of machine learning and the importance of data preprocessing, we will address all these problems by using the same deep learning network but changing only inputs and outputs. The goal of the report will be to emphasize the importance of combining conventional and neural network techniques.

INTRODUCTION

Practical experience developing processing tools for seismic data shows that typically it is more difficult to insert new tools within an existing dataflow than to create new tools. The reasons for that are not obvious unless you work with these dataflows. The final goal of the sequence of seismic data processing, which is the interpretation of migrated seismic sections, requires knowing the velocities, which are themselves also obtained during the processing sequence. This requires an iterative dataflow, where information is approximated but hopefully improved at each step. Different algorithms, which work in different multidimensional domains, work back to back in a dataflow where changing domains can be computationally very expensive. Errors in the outputs of one module affect the results of the following module. Some of these modules are computationally expensive and require dividing data into subsets, for example, in multidimensional windows. Overlapping of these windows introduces an additional mechanism to improve the certainty of the outcomes of the processing since incoherent information across windows becomes attenuated when their average is computed.

We are on the verge of a different and powerful technology that, at least for some problems, promises to be significantly more flexible than conventional approaches. While physics-based methods require us to write the specific rules that nature follows, ML approaches have the potential to extract and implement those rules directly from the data (Chollet, 2021). While in the first case we use physics to constrain the range of possible outputs that can be obtained, in the ML case we allow every possible output to occur at first, but prune them by training, eliminating in this way non-physical possibilities (Roberts, 2021). The ML approach, pruning by training, is highly dependent on the existence of abundant amounts of data in quantities never required before. Therefore, more than ever, we see that the abundance of training data is the enabler and limiting factor on what can be

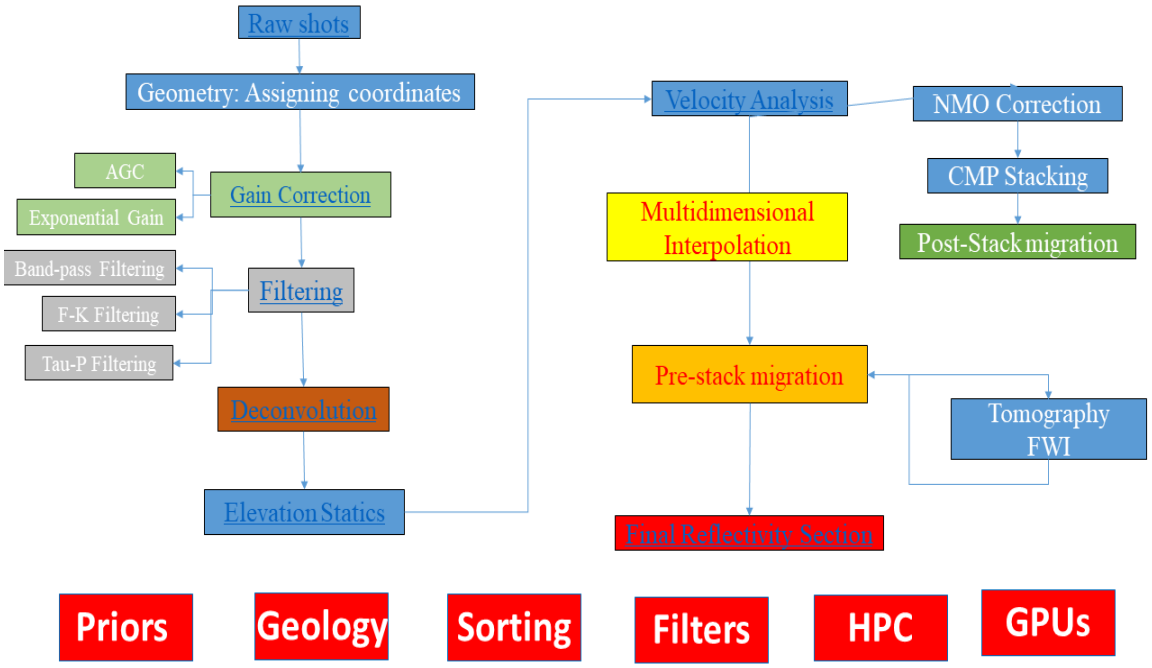


FIG. 1. A typical (simplified) seismic dataflow illustrating the complexities and enabling tools involved in seismic processing.

achieved.

The problem, therefore, resides in inserting this new technology into a classical dataflow. Much of the published work on deep learning (DL) tries to use an "end-to-end" machine learning dataflow, rather than an "end-to-end" hybrid dataflow as in this report. In my opinion, a pure ML approach does not work too well for seismic processing for the reasons delineated above. Before going into more detail about this, let us discuss the differences between classical and machine learning dataflows.

SEISMIC PROCESSING

Figure 1 shows a simplified version of a seismic processing dataflow. Each step is closely related to the steps before and after. Information is continuously extracted from the data, through the different algorithms, and inserted back into the dataflow to act as prior information for the following modules. Processing has to be consistent with reality, forcing processors to add geologic constraints which discard unrealistic results. Different processes work on different domains, and therefore data need to be sorted or divided into subsections. Noise from acquired data permeates across modules and requires filters. All these processes are computationally expensive and typically require high-performance computing techniques. In modern times, many of them are implemented with Graphic Processes Units (GPUs), which enable processes that were infeasible before.

There are many reasons why these flows are complex, but we can mention the most obvious two: a) seismic data are multidimensional (5 dimensions). b) These dimensions are irregularly and poorly sampled.

DEEP LEARNING AND COMPUTER VISION

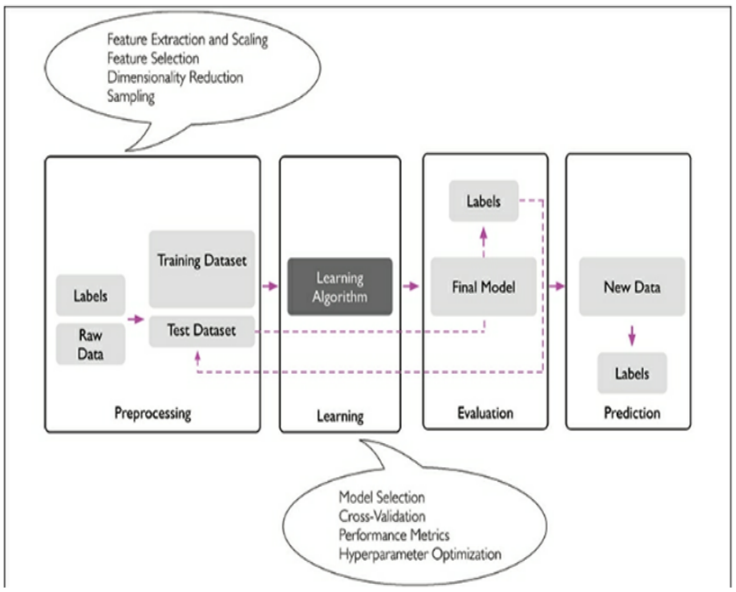
Machine learning typically contains four stages. To be more specific about their description, let us assume a neural network system, which is the main focus of this report. Figure 2 shows the following four stages (Raschka and Mirjalili, 2019):

- Preprocessing: data are converted into a format that can be read by later stages. This involves reading volumes of samples, reformatting them into regular shapes, and applying scales and normalizations. In addition, data are separated into training, testing and validation volumes. For supervised training, labels have to be assigned if not available.
- Learning: data are fed into neural networks to predict outcomes that match the labels. Predictions errors are used to recalculate the neural network weights until the prediction error (residuals) are minimized enough.
- Evaluation: data separated for testing are used to test the model. Also, an interpretation stage sometimes is added here, to understand why the network produces the results it does.
- Prediction: new data are input to the network and results are calculated. A monitoring phase is often required here to make sure the network does not drift or deteriorate with time.

Although not shown in Figure 2, often an additional stage of interpretation is added after evaluation. When possible, the results are interpreted in the sense of understanding why a particular input produces a particular output.

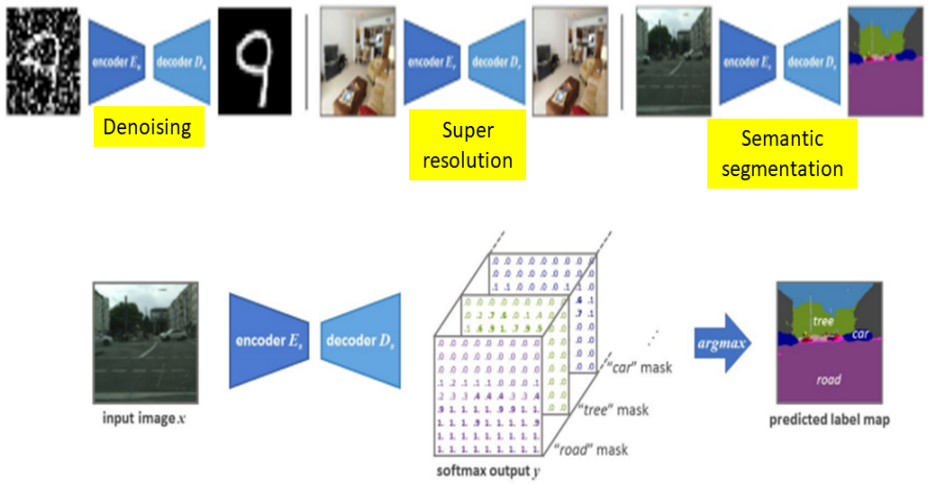
It is important to emphasize that a large part of the DL dataflow is data preparation. In that sense, we see a strong similarity with seismic processing, where migration of data requires significant preparation work. Essentially, the whole seismic processing flow is the preparation of the data for migration.

A common application of DL is in computer vision. For example, autoencoders and similar network patterns (Figure 3) are essentially compression algorithms (Géron, 2022). By reducing information to a minimum number of degrees of freedom, that is mapping the data to a lower dimensional manifold (Chollet, 2021), these networks have the capability to reduce noise, improve the resolution or detect features or shapes in an image. The data format input to these algorithms resembles the format of a migrated seismic section. Both of these formats are essentially regular grids in space, where every pixel represents a numerical encoding of, for example, a physical property. Since seismic interpretation is typically done on migrated sections, it is easy to see why computer vision algorithms for DL are easy to use for seismic interpretation. For example, salt bodies, facies and faults identification, all can be done with semantic segmentation algorithms by assigning a probability to each pixel that it belongs to a particular target feature.



Raschka, Sebastian; Mirjalili, Vahid. Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2. Packt Publishing.

FIG. 2. Dataflow stages for a typical supervised deep learning flow. Most of the machine learning work is in the data preparation.



Géron, Aurélien. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow . O'Reilly Media.

FIG. 3. Applications of deep learning for processing images in computer vision. The format for the input data can be readily compared to a migrated seismic section.

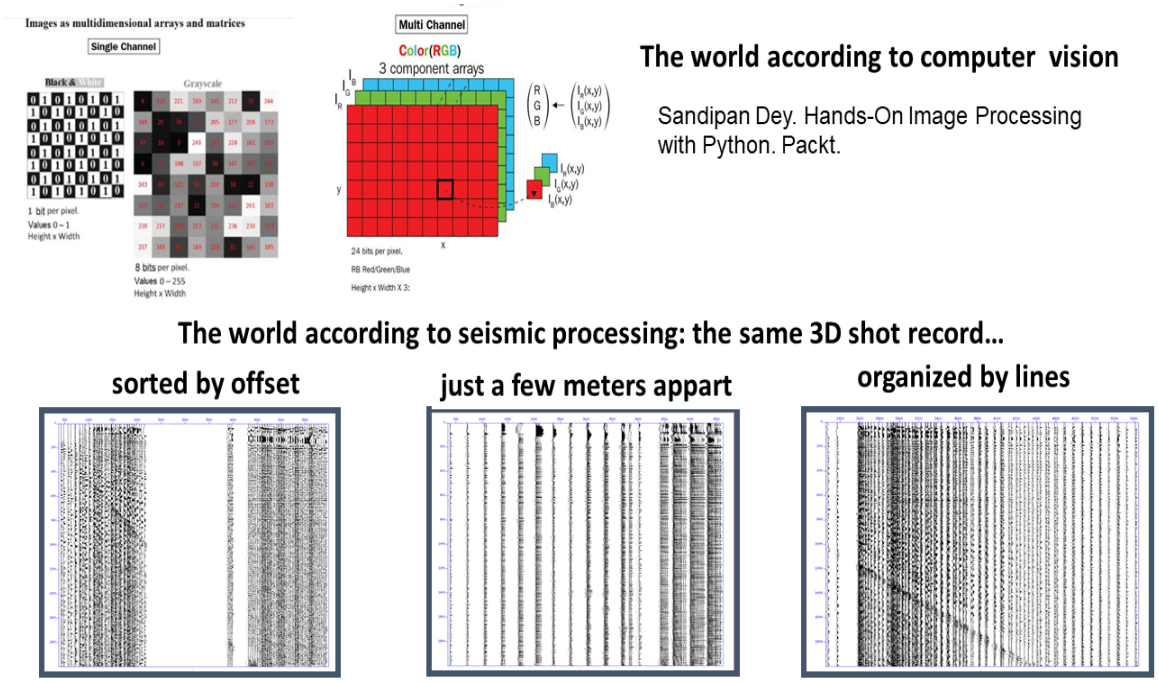


FIG. 4. Differences between data for computer vision and for seismic processing.

However, when we go from interpretation to processing, this similarity breaks. The world of computer vision (Figure 4a) typically is represented by multidimensional matrices or tensors (in the machine learning literature). These matrices are grouped in channels, where each channel represents a property, for example, Red, Blue and Green intensities (RBG), or Hue, Saturation and Value (HSV). Neural networks are very efficient in handling regular volumes like these by using a convolutional pattern algorithm for which GPUs excel at. The world of seismic processing (Figure 4b), on the other hand, is formed by irregularly sampled groups of seismic traces. Seismic data have 5 dimensions, but typically subsets of them are separated to facilitate their manipulation. These subsets contain large sampling gaps with irregular distributions, and these gaps change very rapidly from location to location or different acquisitions. The continuity of the seismic events can only be accounted for when true irregular sampling is taken into account. Minor variations on the irregularity, for example, introduced by binning, produce a major deterioration of high frequencies when algorithms stack across coherent events. In essence, patterns are seriously affected by sampling. These complications have led geophysicists to create complex dataflows, where data are manipulated on different types of groups from beginning to end. Sorting on these groups is key for the functionality of the processing modules, so data re-arrangement is carefully planned ahead of time with serious consequences for efficiency. In addition, algorithms need to be able to account for irregular sampling and only then, patterns or coherence can be detected and applied for further developments.

This discussion hints at the problem that processing images is not the same as processing seismic data so applications of ML to processing require a hybrid classical/ML combination. In previous research that we have done at CREWES on this area, typically we have used classical processing to create datasets and then moved into ML where we have tried different techniques. In practice, this approach was broken and made research

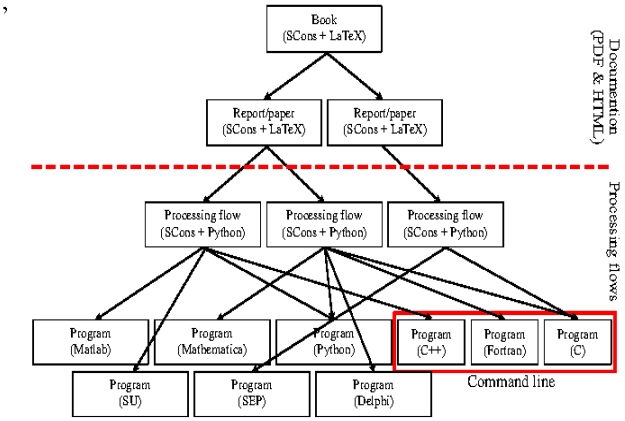


FIG. 5. The open-source Madagascar environment and its flexible setup (Fomel et al., 2012). $Scons$, a python-based building utility similar to $make$, permits the combination of tools written in different languages and environments in a Linux I/O pipeline system.

difficult because there was a disconnection between the two stages. More recently, we have explored a different approach where both stages are run simultaneously in one dataflow. This has been greatly facilitated by the Madagascar environment (Fomel et al., 2012),, a highly flexible open-source system that combines python as a "module-glue dataflow" with standard DL tools like Tensorflow (Figure 5). The hybrid set-up allows us to try different combinations of inputs and labels and the same time that different network designs.

An interesting outcome of these experiments is that solving processing problems with DL seems to be much more dependent on the data processing that the network architecture itself. In fact, it is possible to solve a wide variety range of problems using the same DL architecture but varying the processing part. In this report, I show this by addressing the following four problems: Least Squares Reverse Time Migration, multiple attenuation by Radon transform, Ground Roll attenuation, and interpolation of shot gathers, in all cases using the same network: a classical U-Net.

UNET

The U-Net (Ronneberger et al., 2015) is a very common type of auto-encoder, which is a type of network that maps the input to output with (typically) the same size (Géron, 2022; Chollet, 2021). The network contains two parts, one that encodes the information, creating a mapping of the data into a manifold that hopefully contains smooth variations of the input features, and a decoder, that maps the information back to the original space. The reason why U-Nets are so useful for computer vision and also in seismic processing and interpretation is that they are *self-supervised*. This means that the inputs are the labels, making it easy to gather data. Also, it means that we can use a more desirable version of the input as labels, leading to a wide range of applications. For example, the input can be a noisy gather and the output a clean gather. Even when it is a supervised method, it is often easy to produce the labels. For example, we can use a sophisticated algorithm to clean many gathers and then teach the network how to do the same but with a different setup.

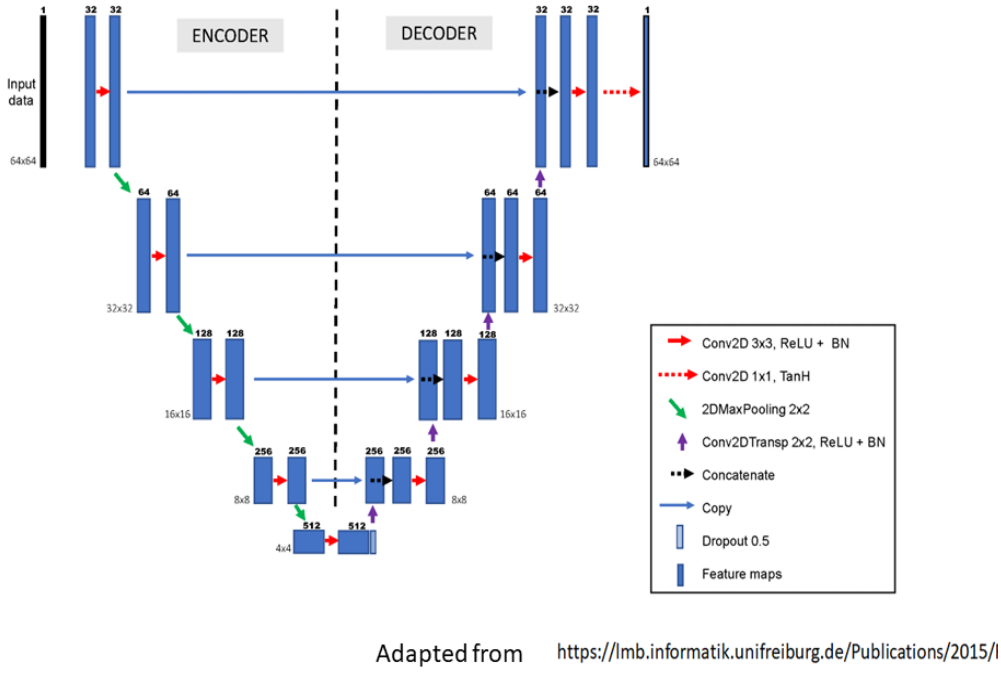


FIG. 6. A classical U-Net used in the examples of this report.

This new ML implementation may be faster (after training is done) or may be more robust by using more information. Perhaps the labels obtained by conventional techniques do not work well outside the range of the training data so the network may be a way to extend the applicability of the conventional technique. Figure 6 illustrates a classical U-Net. From the left, we see data inputs arranged in the form of channels. For usual pictures, this could be an RGB encoding (Red-Green-Blue channels). For seismic, these channels can be anything that provides useful information to the network. As the input goes through different layers, two things happen:

- resolution decreases by decimating the images (resolution changes)
- more patterns are detected by increasing the number of feature channels (filters)

As the input moves to the centre of the U-Net, an increasing number of filters capture information that is important to reproduce the desired output. These filters are calculated from the residuals (the difference between what we want and what we produce) by back-propagation. This is essentially the same process we apply for inverting a chain of operators by steepest descent or conjugate gradients. Once the information is compressed to the minimum number of pixels but the maximum number of filters (or patterns) the process is reversed in the decoder to re-create the labels.

It is interesting to notice that the U-Net creates filters in a similar manner as we would typically create by parametric functions. However, in the U-Net there are usually more

filters than in conventional approaches and they are connected in a non-linear manner by activation functions. This makes the network more powerful and flexible than a similar conventional implementation but also significantly more computationally expensive. For the network chosen for this report, there are almost 4 million parameters which are far more than we would typically use in a classical algorithm. Other networks with fewer and more parameters were tested, but this was chosen because worked better on most of the problems. These parameters are calculated during training which is the most expensive part of DL. Data science publications often claim ML is faster than classical techniques but seldom do they clarify that it is true only if training is not taken into account. Also, often these comparisons are made with poor implementations, typically using high-level languages, rather than optimized implementations like the C++ -CUDA versions that DL uses when invoking the *cudnn* library.

Another important aspect of this network is that there are not many constraints when it comes to input and output. The only "fixed" parameter is the size of the input windows (in this case, 64×64 pixels). The number of input samples and the number of input channels are variable. In fact, the network effect is determined by the content of the inputs and the labels. That flexibility makes neural networks capable of solving all kinds of different problems without changing the network design. Illustrating this is one of the goals of this report.

SOLVING PROCESSING PROBLEMS WITH ML

In this report, I want to explore this flexibility by solving many different problems with the same U-Net. To achieve this, I combine a typical DL dataflow with different seismic dataflows. The problems addressed in this report are:

- Least Squares Reverse Time Migration
- Multiple attenuation by Hyperbolic and Parabolic Radon transforms
- Ground Roll attenuation
- Interpolation by Fourier transform

Example 1: approximating image-domain LSRTM

Figure 7 shows an acoustic RTM image obtained from the Marmousi model using only 8 shots. This image is unusually sharp because it was obtained from the exact velocity model without smoothing. Because RTM is a two-way wave equation (Baysal et al., 1983), the image condition of the source and receiver wavefields will contain contributions of all internal multiples. In real seismic data imaging, we only have smooth approximate velocity models, which lead to images with limited resolution, as we see in Figure 8.

The enhanced resolution in Figure 7 is the goal of many migration variations, for example, Least Squares Reverse Time Migration (LSRTM)(Dong et al., 2012). This more advanced version of migration increases the bandwidth by deconvolving the acquisition

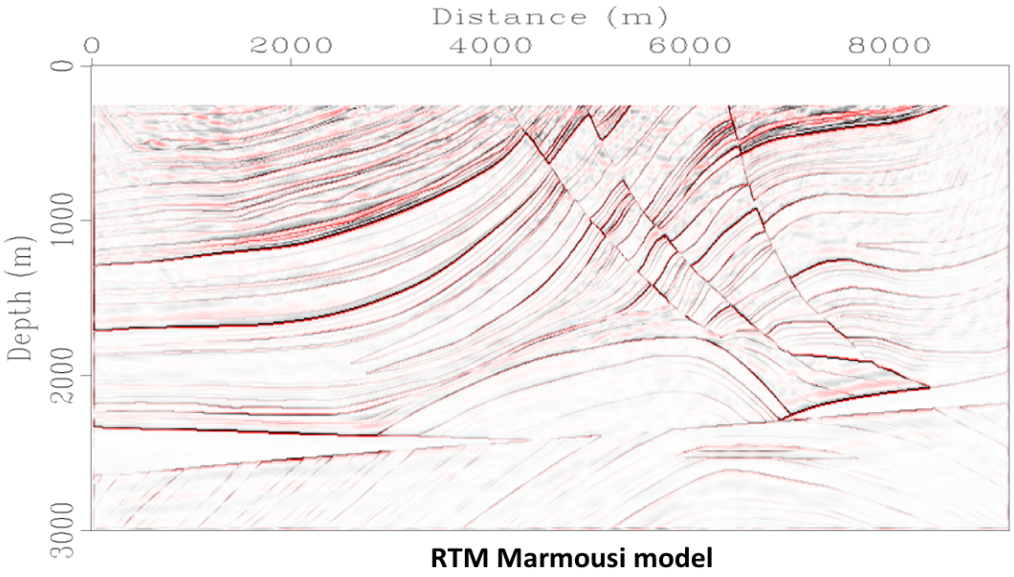


FIG. 7. RTM for Marmousi model with 8 shots, using an exact velocity model.

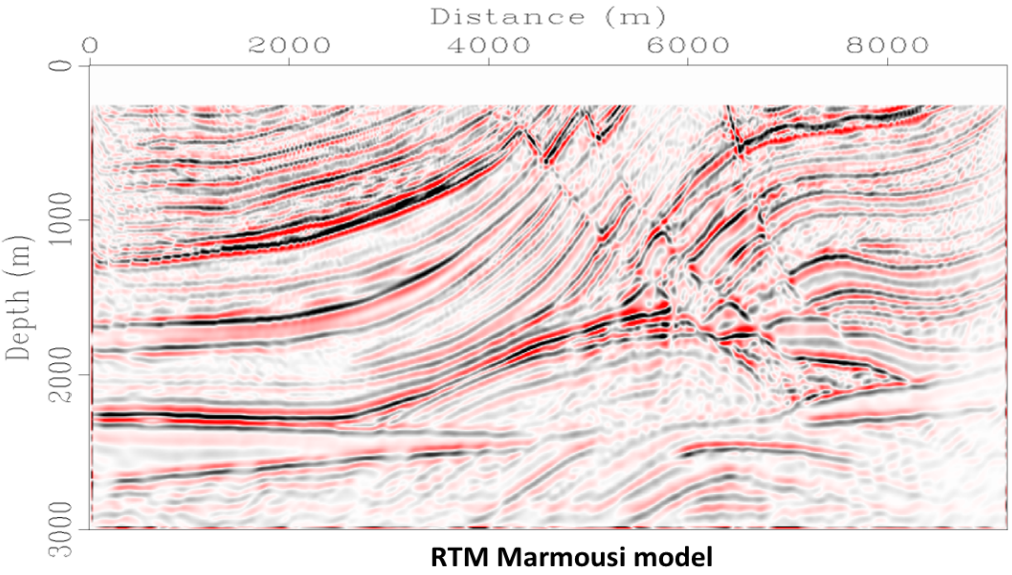


FIG. 8. RTM for Marmousi model with 8 shots, using a smooth velocity model.

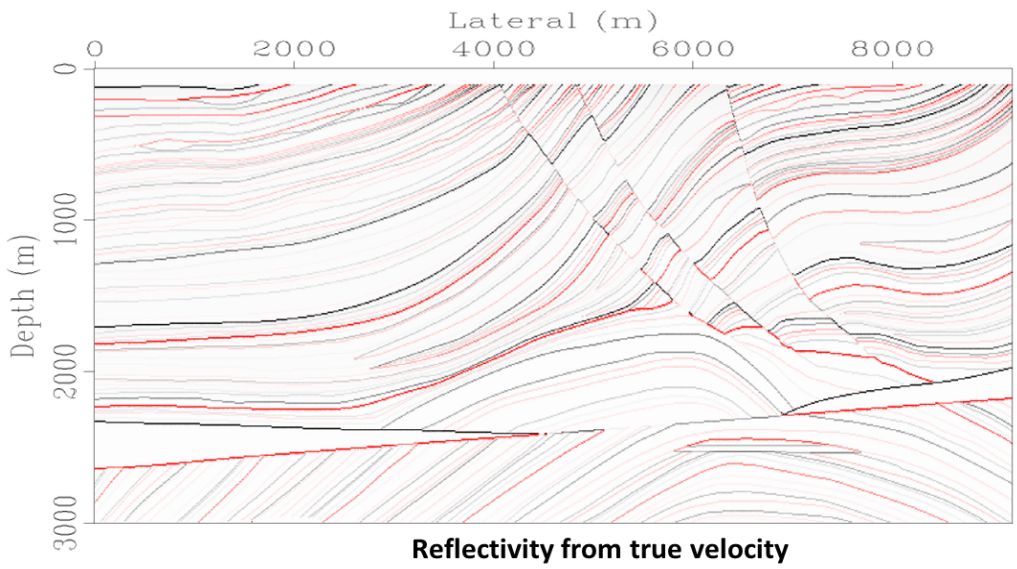


FIG. 9. True reflectivity for the Marmousi model obtained from the velocities.

footprint and the source wavelet from the migration obtained directly by the imaging condition (Schuster, 2017; Trad, 2020)

These techniques are usually computationally expensive and, in addition, require careful data manipulation to make sure amplitudes are not distorted. This is a key requirement since the reflectivity will be modified until it models the amplitudes in the data with a minimum error. A variation on this approach is LSRTM in the image space (Yu et al., 2006), where the problem is posed in a way that the regular RTM image is the input and the Hessian is approximately deconvolved from it by using pre-designed filters. There are some advantages and disadvantages of this approach, outside the scope of this report, but here we will use this technique. The goal will be to train a network using as inputs the regular RTMs and the labels will be the LSRTM results. For practical reasons will replace the LSRTM for a theoretical reflectivity obtained from a synthetic velocity model, but with the understanding that for real data sets LSRTM would fill this role. This type of approach has been published recently by a few authors, for example, Torres and Sacchi (2022). Huang and Trad (2022) illustrates a similar application but extended to the addition of energy from multiples.

To keep the focus of this report, we will attempt this goal by using a U-Net (Figure 6). Many other networks are possible, but I choose a U-Net as a general Auto-Encoder type. However, in addition to the network architecture, there are still many choices to make: what do we use for input (channels) and outputs (labels)? This has to be partially decided by intuition and partly for experimentation. The first intuitive choice for the labels is the true reflectivity obtained from the velocity model (Figure 9). However, testing showed that a full-band true-reflectivity creates a too demanding goal for the network, leading to artifacts in frequencies outside the data bandwidth. A band-limited version (Figure 10) proved to be a better label.

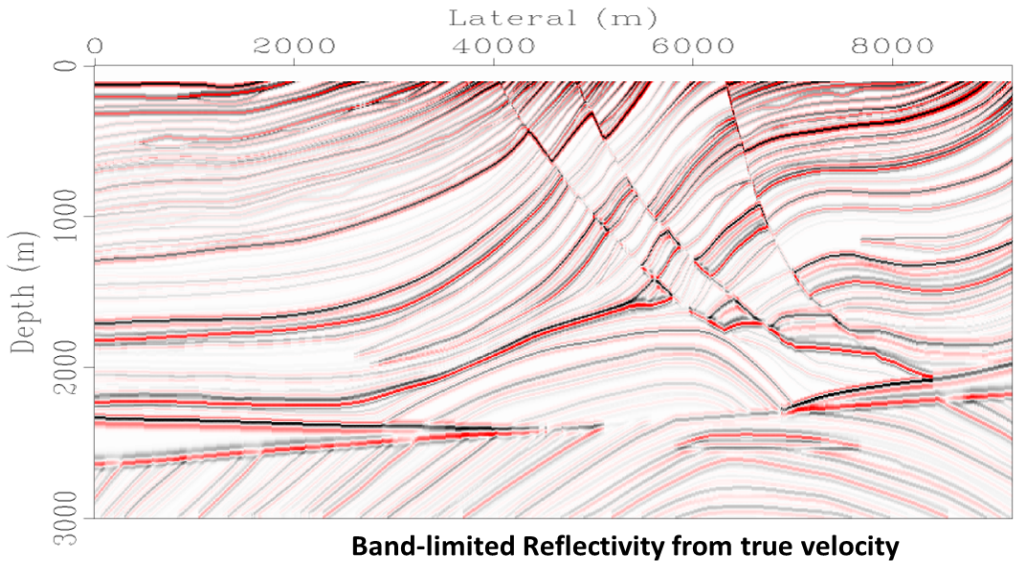


FIG. 10. Band-limited reflectivity for the Marmousi model obtained from the velocities.

The second problem to consider is what kind of information the network needs. This information is set in the form of channels. For the first channel, it makes sense to use the RTM, since this will be the information available for real problems. A first attempt appears in Figure 11. The output was not good, with many blank areas. It seems the network may need more information. A very practical feature of neural networks is their capability to combine information without the need to have physics (and mathematics) connecting this information (although if we can, it is better to do so). Let us put as a second channel the smooth velocity we used for the migration (Figure 12). We see that it was quite helpful, so we can continue this trend and add another channel with illumination (Figure 13). This setup seems not much better than before, and also shows the boundaries between windows (64×64).

As suggested by Sam Gray (personal communication), the network seems to be learning to take the derivative of the velocities and that may be a cause of accentuating the amplitude variations in the overlapping between windows. Therefore, it seems logical to replace velocities with reflectivities to fix the problem. That led to the last attempt in Figure 14, where the second and third channels are replaced by a band-limited smooth approximated reflectivity obtained from the same velocity model used for the migration. This is one of those cases where the combination of experimentation and intuition can tell us what the optimal setup for the network could be. It is also interesting to notice how all the tests were performed with exactly the same network, just by changing the pre-processing part.

We achieved these results on the trained data, so now the challenge is to see how it generalizes. In Figure 15 appears a prediction obtained for a completely different model. In this case, we have no labels, just a poorly defined migration obtained from a smooth velocity model. The enhanced migration shows a more detailed structure, suggesting that the information captured from the training with the Marmousi model generalizes to a different geology.

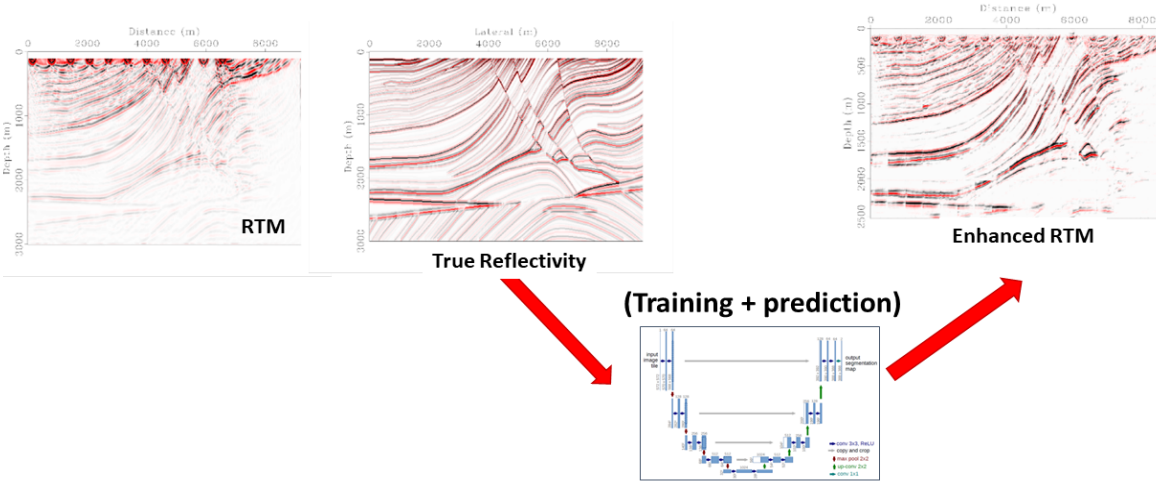


FIG. 11. A first (unsuccesful) attempt with one channel.

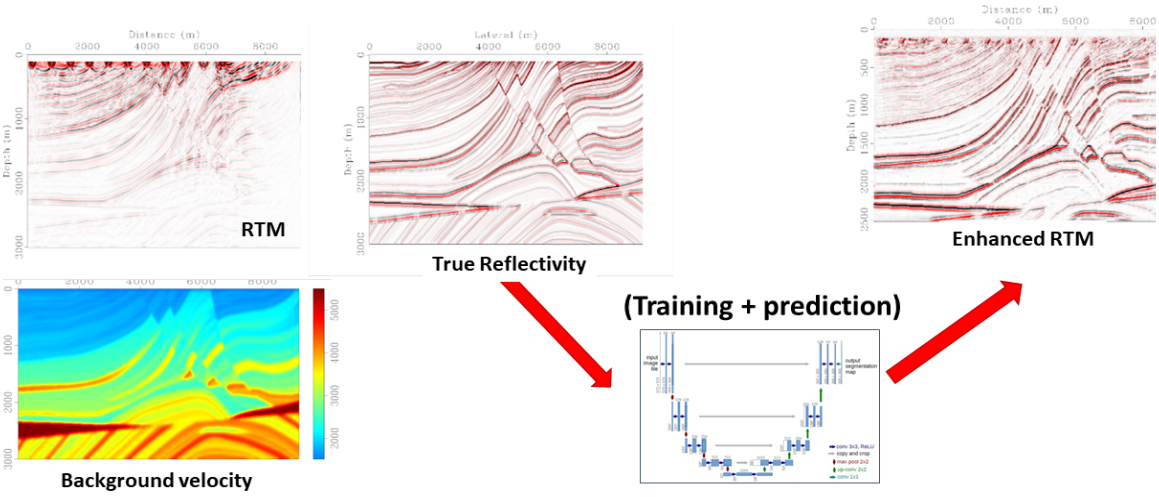


FIG. 12. A second (better) attempt with two channels.

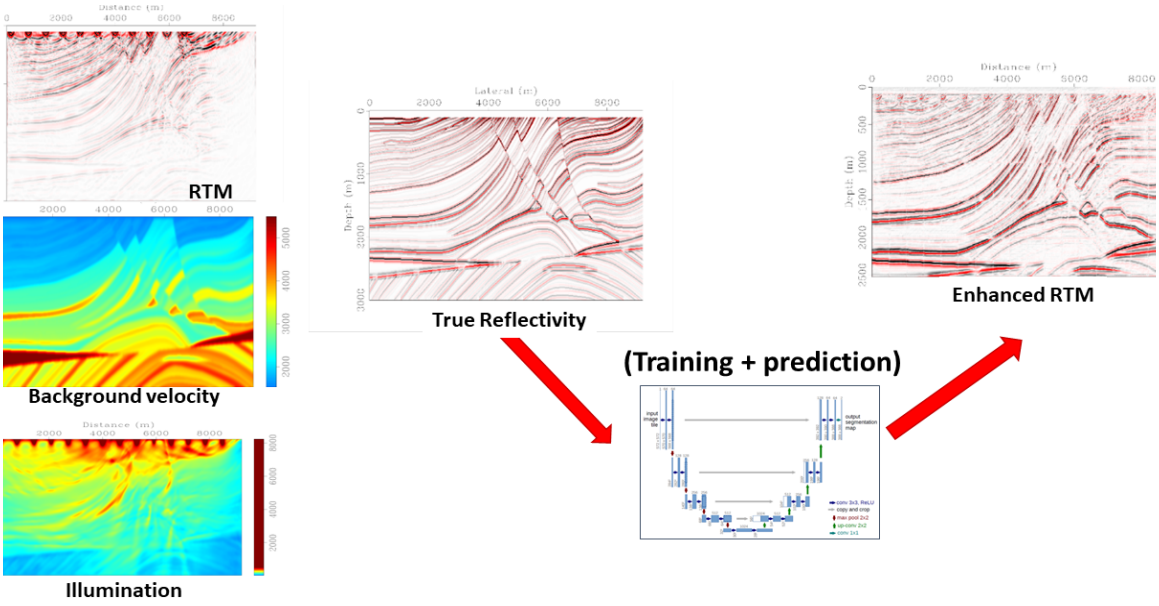


FIG. 13. A third attempt with three channels. Footprint artifacts from the windowing (64×64) are visible on the prediction as a consequence of the network learning to calculate the derivative from the second channel.

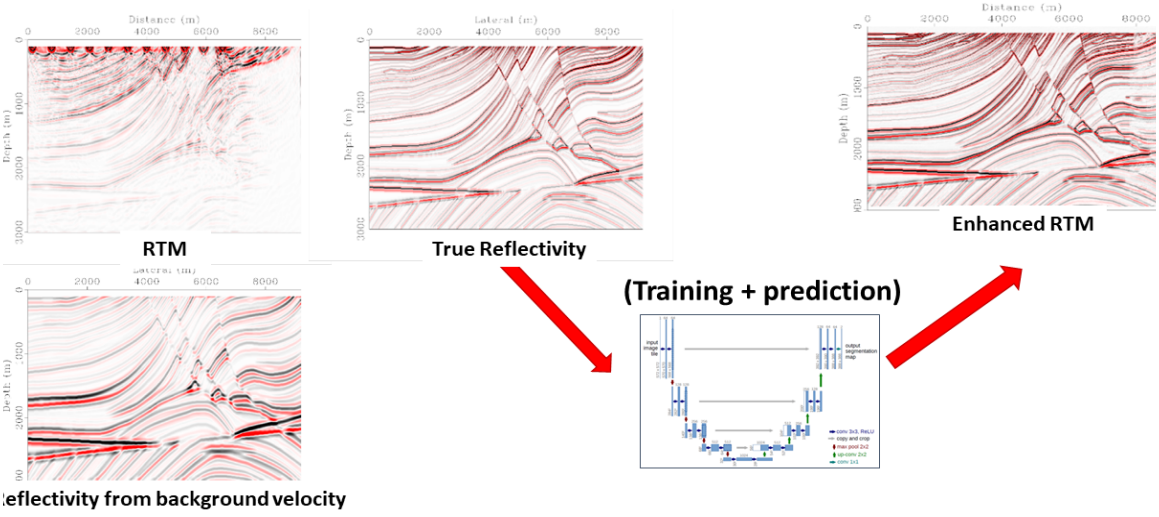


FIG. 14. A fourth and final attempt of changing the input channels. The artifacts on the window overlapping disappeared after replacing the background velocity channel with its derivative.

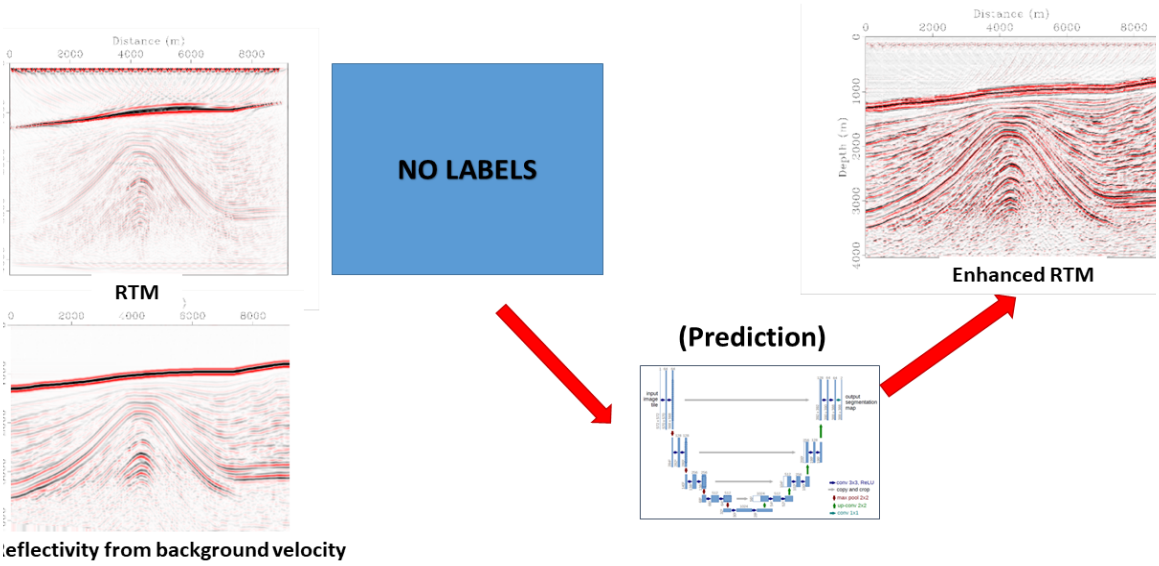


FIG. 15. Prediction on testing data obtained from a smooth velocity model. In this case, there are no labels and the second channel (reflectivity from background velocity) is poorly defined.

Multiple attenuation by Hyperbolic and Parabolic Radon transforms

In the second example, let us use the same network but change the inputs and labels to address a completely different challenge: the separation of primaries and multiples by Radon transforms.

For this work, I use a convolutional model instead of finite differences because it allows me to have control over the order of multiples and produces examples with fewer artifacts. However, the same dataflow works (and has been tested) with finite difference acoustic data from more complex velocity models. In Figure 16 we see one shot gather extracted from a data set of 60 shots with 380 receivers each. This data set was calculated from the flat model layer in d) by adding the primaries in b) and first-order multiples in c) to obtain the data in a). The problem we want to solve is, given the full data set in a) separate the primaries b) and multiples c).

Two possible approaches are 1) train a network with the input data a) and labels b) or c). 2) convert the data to a different domain and input the transforms of the data a) and the transform of the labels b) or c). In principle, assuming enough degree of freedom for the network, both would be possible but not equal in complexity. Working with transforms has an additional advantage in that it can be used to regularize the input to the network. Irregular sampling is very difficult to deal with for convolutional neural networks since the cross-correlations that the networks perform assume samples are equally spaced in all dimensions.

From our knowledge of seismic processing, we know that a good transform to separate primaries and multiples is the Radon transform (Beylkin, 1987; Hampson, 1986). Rather than try to teach the network how to calculate a Radon transform, we can provide the network with the transforms which we already know how to calculate efficiently. Also,

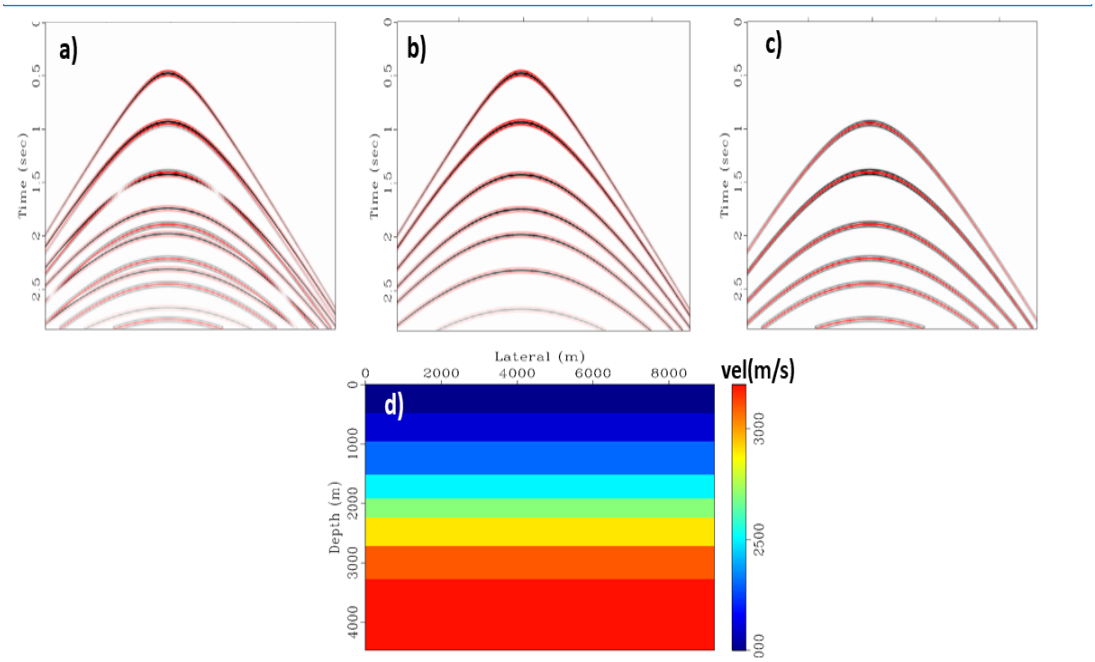


FIG. 16. Building the inputs and desired outputs for the problem. A geometry with 60 shots and 380 receivers was used to create data a), primaries only b), and first order multiples only c) from the model in d)

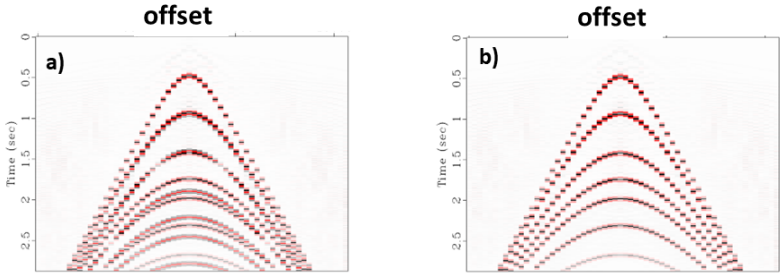


FIG. 17. The data with multiples (a) and without multiples (b), shown in the CMP domain.

Radon transforms are calculated in the CMP domain because they require events to have their apexes at zero offsets, a condition approximately satisfied for common midpoints (CMPs) but not for shot or receiver gathers. Figure 17 shows one CMP chosen among the 675 CMPs that this data set will generate (a) reflections with first order multiples, b) reflections only. As seen in Figure 17, the sampling on the CMP domain is usually coarser than on the shot and receiver domains unless the geometry is perfectly symmetric in terms of shot and receiver intervals. Here I have chosen for illustration a typical case where shots are far apart, producing much better shots than CMPs. Because this structure is flat, we could get better results directly on the shot domain, but we will keep the flow general to change the geology later.

Figure 18 shows the hybrid dataflow. The beginning and end of the flow involve efficient classical calculations using conventional signal processing techniques. The ML part is included in the dataflow. The process is an end-to-end dataflow, which makes it easier

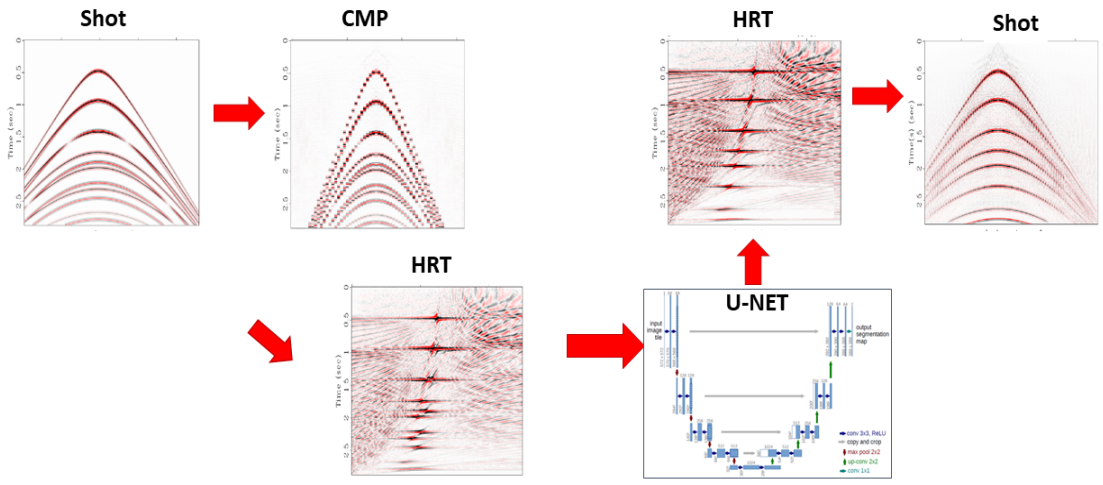


FIG. 18. Hybrid classical-ML dataflow. Shot gathers are converted to CMP gathers, transformed into Radon panels, introduced into the U-NET to predict new panels without multiples, which are converted back to CMPs and shot gathers.

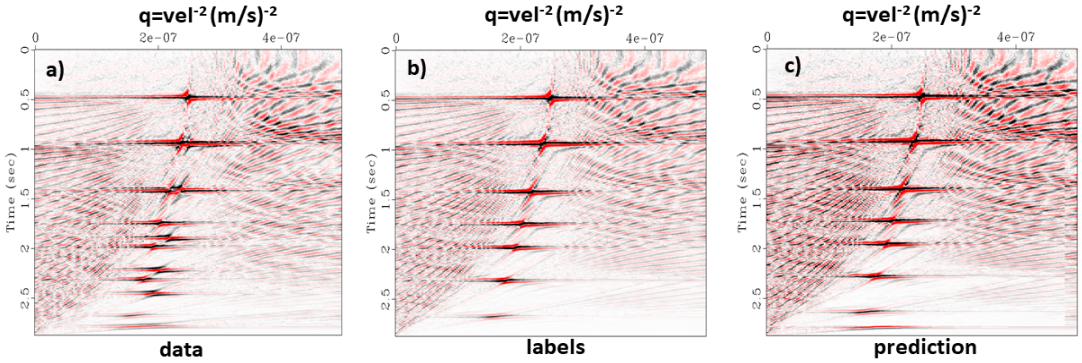


FIG. 19. Hyperbolic Radon transforms calculated from CMP gathers. There is one of these panels for each CMP and for each dataset. The horizontal axis represent $1/velocity^2$, and therefore moveout.

to experiment by changing simultaneously both types of parameters: inputs and outputs to the network and the network itself. It is interesting to note that 1) compared to the previous example, the network is the same but the inputs and outputs are different. 2) the computational cost of training the U-Net in this example was almost one order of magnitude larger than the conventional part of the dataflow (9 minutes versus 1 minute). Therefore, the training effort should be always taken into account, although a good generalization power from the network would make training required only once.

A couple of things are worth mentioning about Radon transforms: First, time-domain HRTs are computationally expensive in 3D but can be efficiently computed with Graphic Processing Units (GPUs). In this simple 2D example, I calculate 675 CMPs in 70 seconds by using a single GeForce GPU (RTX3060). Although a U-Net could be trained to produce something like a Radon transform (also using GPUs), it would be computationally less efficient because it would have to learn how to calculate the transform first.

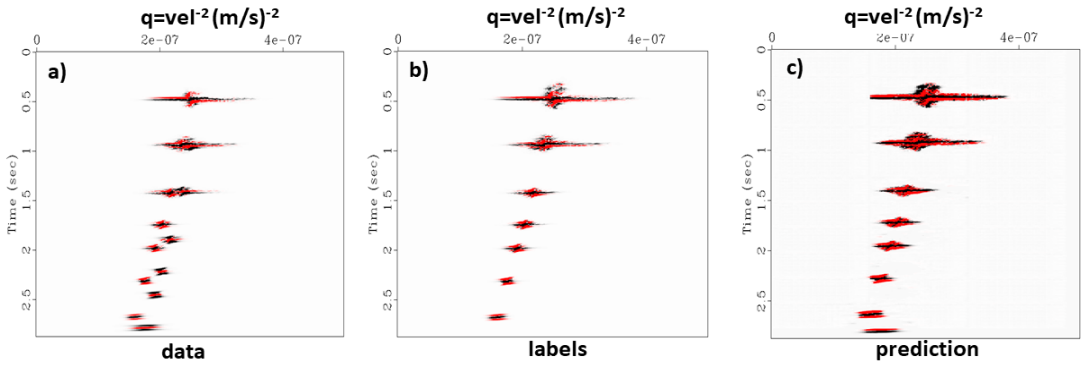


FIG. 20. Sparse HRT calculated from the same CMP gathers. The sparseness constraint eliminates many of the sampling artifacts.

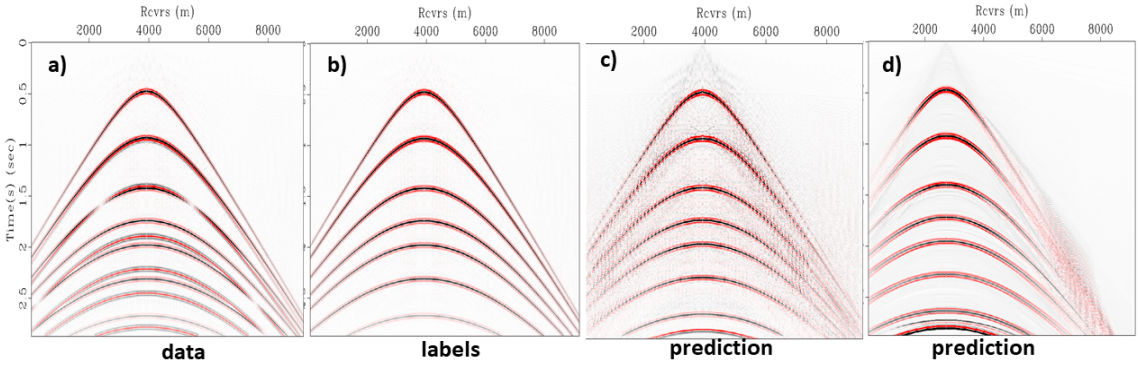


FIG. 21. Reconstruction results by doubling the number of shots. inputs a), labels b) and predictions c) for 60 input shots and d) 120 input shots. Clearly, the shot interval plays a key role as it affects the CMP sampling.

Secondly, these transforms suffer from a large number of artifacts that appear as noise, mostly on the left and right of the panels. This noise is a result of poor sampling and limited aperture on the CMPs. An approach to address this is the *sparse* Radon transform (Thorson and Claerbout, 1985; Sacchi and Ulrych, 1995; Trad et al., 2003). Figure 19 shows the Radon panels from the standard HRT, and Figure 20 the panels from the sparse HRT (a) input, b) label, c) prediction). In this dataflow, changing between these two variants of RT amounts to changing just one parameter, the number of external iterations (from 1 to 2). In this case, the sparse HRT did not improve the final results, but it is an example of how a hybrid classical-ML dataflow becomes essential for experimentation. Also, it is possible to apply an ℓ_1 constraint to the network directly, but the effect is different since sparseness on the neurons does not easily translate into the transform.

The final output of the dataflow, i.e. the shot gathers, shows significant noise and artifacts. Poor sampling is a possible reason for this problem. Taking advantage of the end-to-end dataflow, we can corroborate this by increasing the number of shots during modelling (changing one parameter in the dataflow). Figure 21 shows the inputs, labels and predictions when the number of shots is doubled from 60 to 120. In practice, a real data set would have to be interpolated before multiple attenuation, or perhaps a denser acquisition may be required. The numerical experiment just shows which are the factors we should take into account.

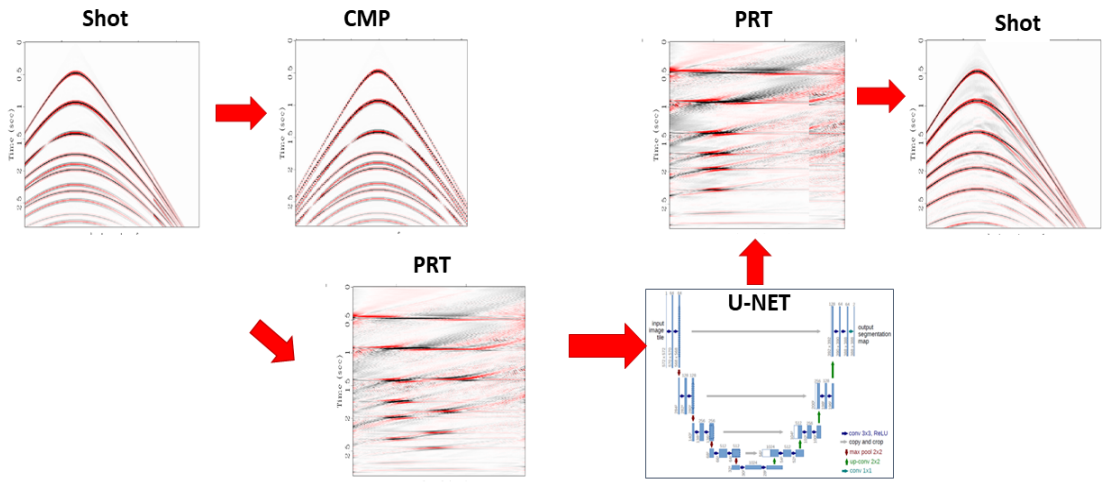


FIG. 22. Hybrid classical-ML dataflow using the PRT.

There are many other tests that can be performed. For example, the HRT can be replaced by a parabolic RT (PRT), which has some advantages in reconstruction but disadvantages in sparseness. Figure 22 illustrates the dataflow for this case. This brings also the idea of combining both as inputs for the network. Since the PRT and the HRT use different curvature parameters, combining both of them is not straightforward, but it can be done by using an additional network to map one to the other. Once a proper mapping is achieved, it is possible to incorporate both of them into the same U-Net by using different channels. Also, in practice, prediction of multiples instead of primaries followed by a filter and subtraction would be the standard multiple attenuation dataflow. This is the topic of a different report (Fontes and Trad, 2022)

Example 3: eliminating Ground Roll

The third example of hybrid conventional-ML dataflow is to train a network to remove ground roll. Just as in the previous case, it is possible to work directly with seismic gathers or apply some kind of transformation. Figure 23 illustrates a set of a) simulated data, b) ground roll only, and c) reflections only, all obtained by elastic finite difference modeling from topography with the SEAM2D dataset shown on (f). The full data a) was obtained by full modelling d), the ground roll b) from near surface modelling e) in model g) and the reflections by subtraction of a) and b). This methodology is explained in Sanchez et al. (2022a).

Figure 24 shows the hybrid dataflow and a result obtained by feeding the full data to the network and the reflections as a label. In this case, the seismic gathers were used directly, but the blue squares indicate a possible transformation that would make the process simpler for the U-Net. In Sanchez et al. (2022b) we illustrate a transformation based on a combination of linear Radon with astronomical parameters.

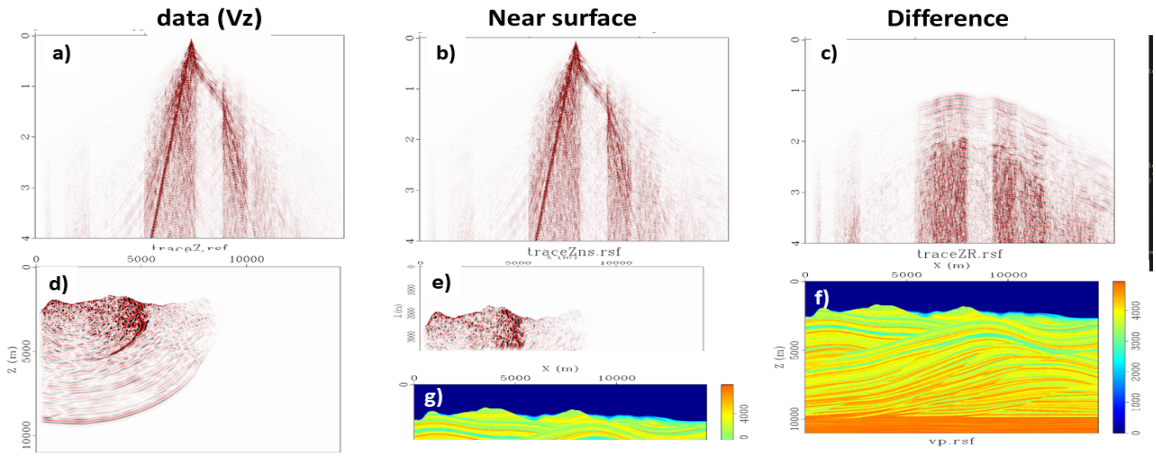


FIG. 23. Elastic Vz components of data with ground roll, ground roll alone, and data alone, generated by elastic finite difference modelling from topography using the SEAM2D model.

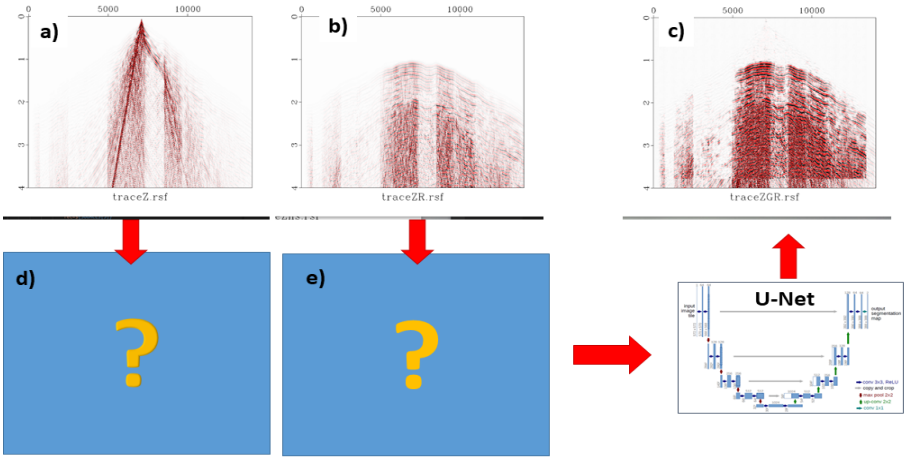


FIG. 24. Hybrid data flow, using as input the full data set, and labels the clean data set. In this example, no transform was applied. A better approach would involve transforms (indicated here by the blue squares).

Example 4: Interpolation

The last example involves the interpolation of seismic traces. For simplicity, we will consider only 2D interpolation. This example is just an illustration of the flexibility of DL, but a more powerful approach is to use multidimensional interpolation across all domains using coherence methods like Fourier, Low rank or Radon transforms. The extra dimensions are essential to overcome aliasing and missing information (Trad, 2009). The extension of this ML method to 5-dimensional interpolation would be computationally more expensive than for conventional techniques and the irregularity in space coordinates would adversely affect the convolutional operations. Here we will just explore the generalization capability for this problem. Notice that this discussion on generalization should be applied to the previous examples as well.

Using a data set to train a network and then predicting missing traces contained on the training labels is guaranteed to work well because of the large number of parameters that neural networks use. The network can memorize every missing sample from the labels. What is difficult is to train the network on a set of different data sets and apply it to interpolate on a completely different dataset, that is train a network with generalization power. A way to improve on generalization is to train the network with many different data sets. The more diverse the training data, and the more abundant, the better the generalization power. The ideal approach would be to continue the training each time a new data set (or computer resources) are available. This approach can be challenging since re-training with new data sets tends to deteriorate the previously calculated weights. A possible fix could be freezing the layers closer to the input and retraining the final layers. That could make the network learn new structural patterns but reuse details like the nature of seismic data (amplitude and phase variations from trace to trace). In this example, I opted by training all the datasets together. This makes the problem much more dependent on I/O constraints but it did achieve the goal of improving generalization.

Figure 25 (first row) shows four different velocity models used for training, from which I generated four datasets with an acoustic finite difference method. Each data set consisted of 20 shots and 400 receivers, with ten random gaps along receivers on each shot gather. The gap sizes were variable between 20 and 80 meters. The full datasets (before introducing the gaps) were used as the labels to train the network. The testing model and one example of interpolation for data from this model are shown in the second row. In Figure 26 left-hand side, we see the training results for two of the four training data sets. On the right, we see the prediction for the testing data set. A comparison with a physics-based interpolation using Fourier showed both methods have similar quality but the physics-based approach did not require the training. A more fair comparison would have been with a full multidimensional approach (3D in this case, since data is only 2D).

CONCLUSIONS

In this report, we have seen four different seismic problems addressed by using a hybrid conventional+ML dataflow. Increasing the complexity of the network is not the best solution to address these types of problems. A better approach is to experiment with the dataflow by trying different inputs and labels. A combined dataflow that permits simultane-

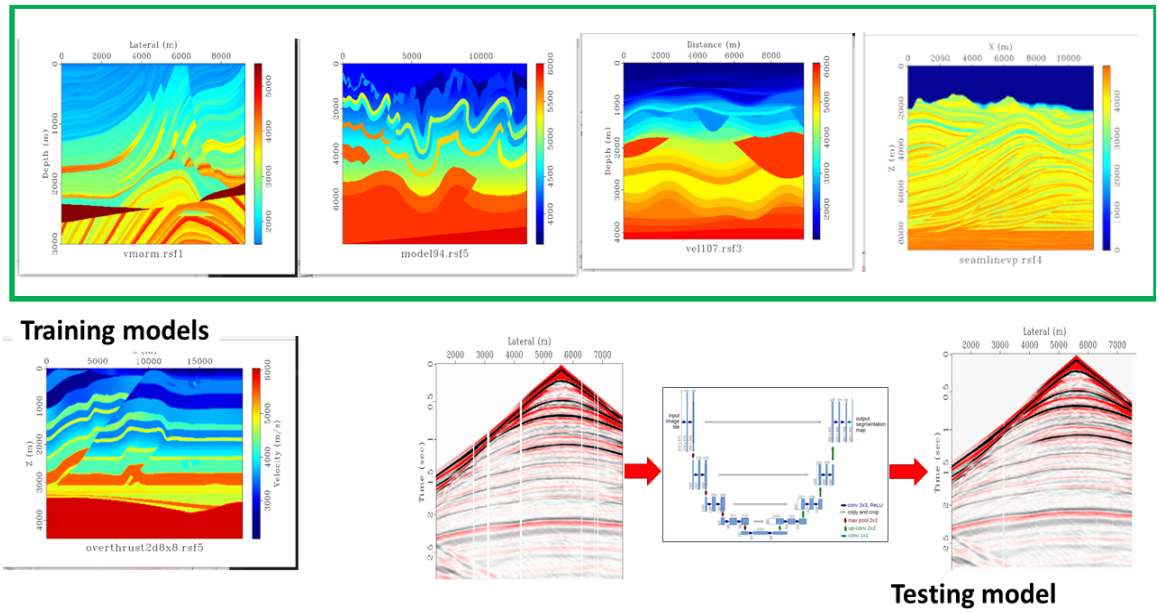


FIG. 25. Different velocity models used for training the network. The top row shows the training velocity models. The bottom row shows the testing model (overthrust model) and one prediction example. All the datasets were generated by using acoustic finite difference.

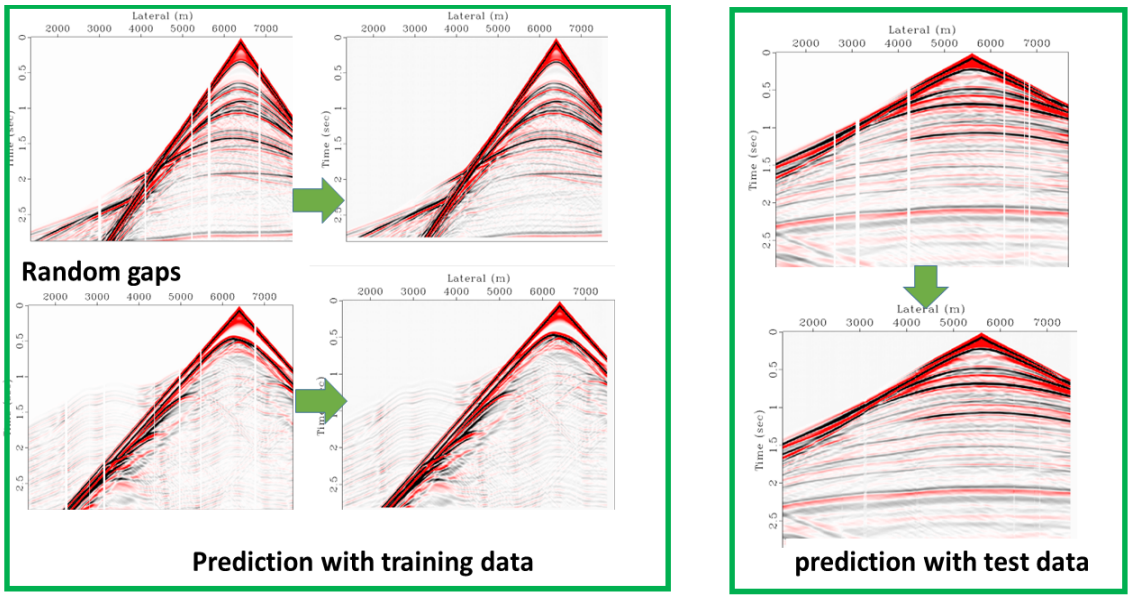


FIG. 26. Examples for interpolation for the training and testing data sets).

ous experimentation with both the ML and the conventional processing brings a significant speed up to the dataflow design. Another aspect that this report is trying to illustrate is the flexibility of neural networks to change their functionality completely without a need for architectural changes. The training process is sufficient to change the filters and therefore the output to any given particular goal. A major challenge in all problems involving seismic data processing and migration is to achieve a good generalization power, such that the computationally expensive data training does not need to be repeated for new types of data sets. I illustrated this in the first and last examples although many more questions require answers. For example, sensitivity to the geometry, to the source signatures and to the geological complexity. The goal of this report was, however, to illustrate a methodology that can be applied to different problems. The core of the methodology is to integrate the machine learning tools into the conventional dataflows such that rapid testing and development can be achieved in each case.

ACKNOWLEDGMENTS

I thank Sam Gray for many fruitful discussions, and review of this report and CREWES sponsors for contributing to this seismic research. I also gratefully acknowledge support from NSERC (Natural Science and Engineering Research Council of Canada) through the grants CRDPJ 461179-13, CRDPJ 543578-19 and NSERC Discovery Grant.

REFERENCES

Baysal, E., Kosloff, D. D., and Sherwood, J. W., 1983, Reverse time migration: Geophysics, **48**, No. 11, 1514–1524.

Beylkin, G., 1987, Discrete radon transform: IEEE transactions on acoustics, speech, and signal processing, **35**, No. 2, 162–172.

Chollet, F., 2021, Deep learning with Python: Simon and Schuster.

Dong, S., Cai, J., Guo, M., Suh, S., Zhang, Z., Wang, B., and Li, e. Z., 2012, Least-squares reverse time migration: Towards true amplitude imaging and improving the resolution, *in* SEG technical program expanded abstracts 2012, Society of Exploration Geophysicists, 1–5.

Fomel, S., Sava, P., Vlad, I., Liu, Y., Jennings, J., Browaeys, J., Bashk ardin, V., Godwin, J., Song, X., and Hennenfent, G., 2012, Madagascar software package and reproducible research.

Fontes, P., and Trad, D. O., 2022, Multiple attenuation by neural networks: CREWES Research Report, **34**.

Géron, A., 2022, Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: " O'Reilly Media, Inc.".

Hampson, D., 1986, Inverse velocity stacking for multiple elimination: Journal of the Canadian Society of Exploration Geophysicists.

Huang, S., and Trad, D. O., 2022, Convolutional neural network-based reverse time migration with multiple energy, *in* GeoConvention, Conference Abstracts.

Raschka, S., and Mirjalili, V., 2019, Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2: Packt Publishing Ltd.

Roberts, D. A., 2021, Why is ai hard and physics simple?: arXiv preprint arXiv:2104.00008.

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, 234–241.

Sacchi, M. D., and Ulrych, T. J., 1995, High-resolution velocity gathers and offset space reconstruction: Geophysics, **60**, No. 4, 1169–1177.

Sanchez, I., Agudelo, W. M., Sierra, D., and Trad, D. O., 2022a, Simulated removal of near-surface scattered waves by elasticwave modeling, *in* GeoConvention, Conference Abstracts.

Sanchez, I., Trad, D. O., Agudelo, W. M., and Sierra, D., 2022b, Ground Roll attenuation by Neural Networks: CREWES Research Report, **34**.

Schuster, G. T., 2017, Seismic inversion: Society of Exploration Geophysicists.

Thorson, J. R., and Claerbout, J. F., 1985, Velocity-stack and slant-stack stochastic inversion: Geophysics, **50**, No. 12, 2727–2741.

Torres, K., and Sacchi, M., 2022, Least-squares reverse time migration via deep learning-based updating operators: Geophysics, **87**, No. 6, S315–S333.

Trad, D., 2009, Five-dimensional interpolation: Recovering from acquisition constraints: Geophysics, **74**, No. 6, V123–V132.

Trad, D., 2020, Assumptions and goals for least squares migration: Geophysical Prospecting, **68**, No. 3, 802–814.

Trad, D., Ulrych, T., and Sacchi, M., 2003, Latest views of the sparse radon transform: GEOPHYSICS, **68**, No. 1, 386–399.

Yu, J., Hu, J., Schuster, G. T., and Estill, R., 2006, Prestack migration deconvolution: Geophysics, **71**, No. 2, S53–S62.