geoconvention
Virtual Event
September 13 - 15
2021

Deblending using convolutional neural networks

Zhan Niu, Daniel O. Trad
CREWES/University of Calgary

Summary

Machine learning has been a booming subject in computer science and its applications have been made in various subjects including geophysics. Convolutional Neural Networks (CNNs) have great potential for solving image processing problems like denoising and interpolation. Deblending, considered as an underdetermined denoising problem, falls into this category. In this report, we use CNN to replace the deblending operator and its performance is analyzed. We use a 4-layer U-Net to perform deblending on synthetically blended shots from a wedge velocity model with point scatterers. We test out different hyper-parameters and the trained model could successfully remove the noise and preserve diffractions from the scatterers with some tolerance. The generality of the model is evaluated by testing the model on an easier 2-layer velocity model. The model can successfully identify and recover most parts of the primaries but fails to deal with some interferences and leaves them muted.

Introduction

Deblending is a technique to reduce the cost of acquisition. It enables us to fire several shots simultaneously, which not only reduces the time of recording but also reduces the cost of storing seismic data (Beasley et al., 1998). The reduction of recording time will also reduce the cost from labour and mitigate the exposure to some noise. The deblending process essentially separates overlapped shots, which is an under-determined problem to solve since it tries to produce several shots from each supershots. Therefore, additional constraints must be applied to get a unique solution. Pseudo-deblending is a technique that is commonly used which converts deblending into a denoising process.

Machine learning methods can be applied instead of signal-processing/inversion types of denoising operators. The problem can be defined in two ways, a classification problem or a regression problem. The classification generates a mask that indicates the position of the desired shots but leaves the interferences unsolved, while the regression problem tries to produce individual shots but requires more parameters to be determined. For example, Baardman et al. (2019) used a convolutional neural network (CNN, LeCun et al. 2015) for both problems. However, we think that CNNs may not be the best choice to capture the relationship between inputs and outputs due to the lack of skipping connections. Richardson and Feller (2019) chose a U-Net model with ResNet34 encoder pre-trained on ImageNet and trained with random velocity models that are harder than the reality. In this report, we will look at an easier case and discuss the suitability of U-Net on solving deblending problems.

Theory

The neural network architecture to solve the problem is the U-Net (Ronneberger et al.,2015). The U-Net was designed based on the CNNs and bridge connections were added so that it performs fast and well, especially for solving segmentation problems. The U-Net used contains 3 parts. The down-going/encoding part, the up-going/decoding part and the bridge connections.

A segmentation prediction produces a mask of a given picture indicating an area of interest. For example, this has applications on a brain MRI for finding the damaged area, or in our case in seismic, for targeting events in a noisy shot record. Essentially, it predicts the probability of a given pixel to be true. The probability on each pixel then can be converted to a true or false by applying a judging threshold. The reason why U-Net is more suitable to solve segmentation problems over traditional variations of CNNs is that U-Net has bridge connections that directly link the features with the same tier. Since the inputs and outputs of the segmentation usually correlate and share spatial similarities, the connections will greatly reduce the efforts to learn this relation by skipping unnecessary transforms, which helps to reduce the chance of vanishing gradients.

Data preparation

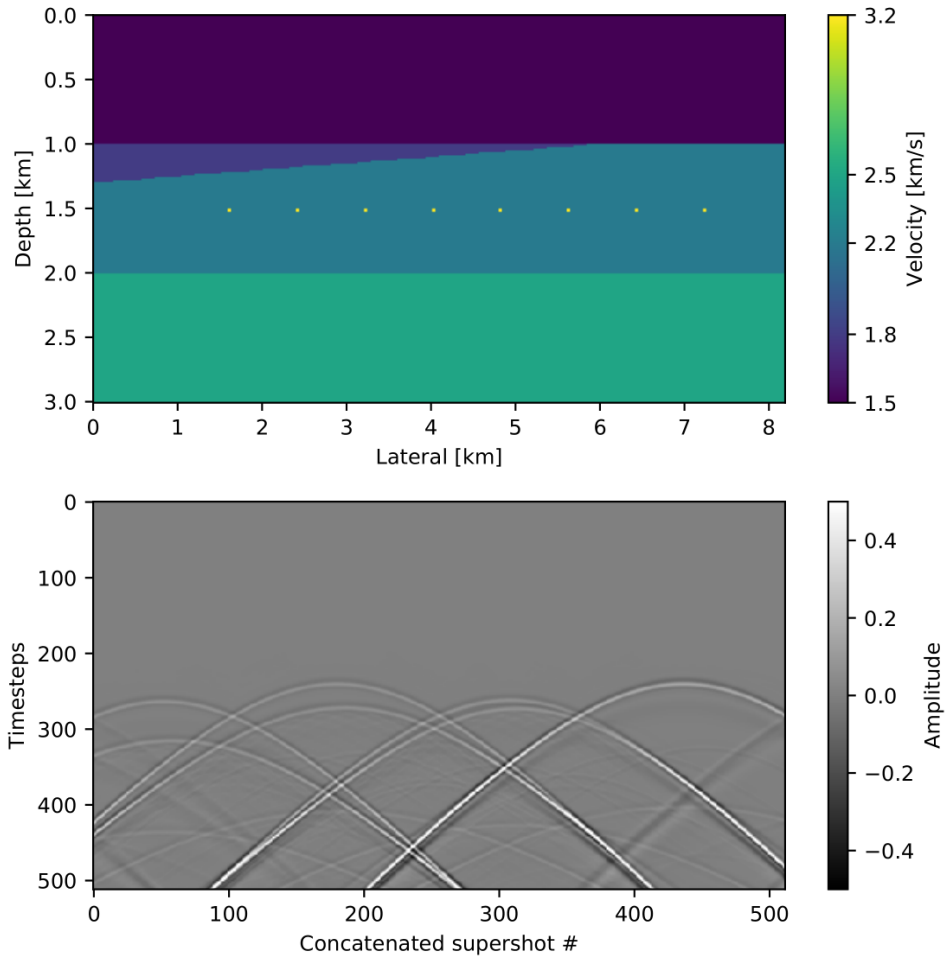

FIG. 1. The inputs fed to the U-Net model. The plots show the corresponding input (above) and label(bellow) pair at the 120th receiver, with 512 receiver slices in total.

All data used in this report were generated synthetically with the finite difference method. The blended data was created by injecting shots simultaneously with random delays and measuring the total wavefield in the receiver locations using the velocity model shown in Figure 1. This velocity model contains 3 layers and a wedge on the left, with several point scatterers under the

dipping layer. These scatterers are intended to test whether the deblending algorithm can honour data diffractions. The dipping layer of the wedge moves the apexes of reflections in the shot domain. In this model, 64 supershots were recorded with 4 shots blended in each and with 512 receivers. The data were resampled and the number of time samples was reduced from 3600 to 512 to reduce the computation cost. Both sources and receivers are evenly distributed at the near-surface. Also, we created for the training a regular data set without blending or time delays, which we call here "true data."

The blended data are first pseudo-deblended as follows: the supershots are repeated as many times as the number of blended shots per supershots, and the copies are concatenated in the shot axis with the time delays removed one shot at a time. After this pseudo-deblending, only those shots whose time delay was completely removed become coherent in the receiver domain. Since duplications of supershots were concatenated together, the blended data now has dimensions of $N_g \times N_t \times N_{shot}$, which refers to the number of receivers, timesteps, and shots, respectively. The number of shots here is the product of supershots and the number of blended shots. We treat each receiver gather as a picture which we feed to the network for training. The time and source axis becomes the height and width of the picture. Since the input is in grayscale, the input tensors have the format of $N_{sample} \times 1 \times N_t \times N_{shot}$ which is 512 × 1 × 512 × 256 where N_{sample} is the number of receiver gathers. The "true data", that is receiver gathers without blending, have the same dimensions as well. Both inputs and labels are normalized to be ranging from 0 to 1 for better generalization and a more stable model, as indicated by the scale bar.

The dataset was then separated into training and validation sets. In this project, the randomly chosen 20% of the entire dataset becomes the validation set and the rest becomes the training set which will be used for calculating the gradient.

Training

We used PyTorch (Paszke et al., 2017) for the machine learning framework and adapted the U-Net implementation described in Buda et al. (2019), which was designed for brain MRI. The U-Net has 4 tiers in depth with two 3 by 3 convolutional layers in each block with zeros padding of 3 at each boundary, which guarantees the inputs and outputs having the same dimension. The original model was designed to take inputs with 3 channels as RGB images and has 32 filters in the initial layer. In this report, however, since the receiver gathers only have one channel, the default 32 filters are more than needed. The U-Net uses ReLU as inter-layer activation functions and uses batch normalization layers. The output activation is sigmoid, which regularizes the outputs to a (0, 1) range.

After some testing and experimentation, we decided to train the model 300 epochs with an ADAM optimizer (Kingma and Ba, 2014), with an initial learning rate of 0.002. To mitigate large oscillations at later epochs, we decrease the learning rate every 100 iterations with a factor of 10%. This learning rate decay slows down the descent in later epochs but makes it tolerant to a bigger learning rate at the early stage (the default learning rate for the brain MRI problem was 0.0001). The validation loss reaches a plateau with small oscillations at 200th to 300th epochs. Models at those iterations can be considered to have the same confidence level. One could choose the model at the last epoch since it undergoes more training but we picked the model with the least validation error to avoid over-fitting.

Results

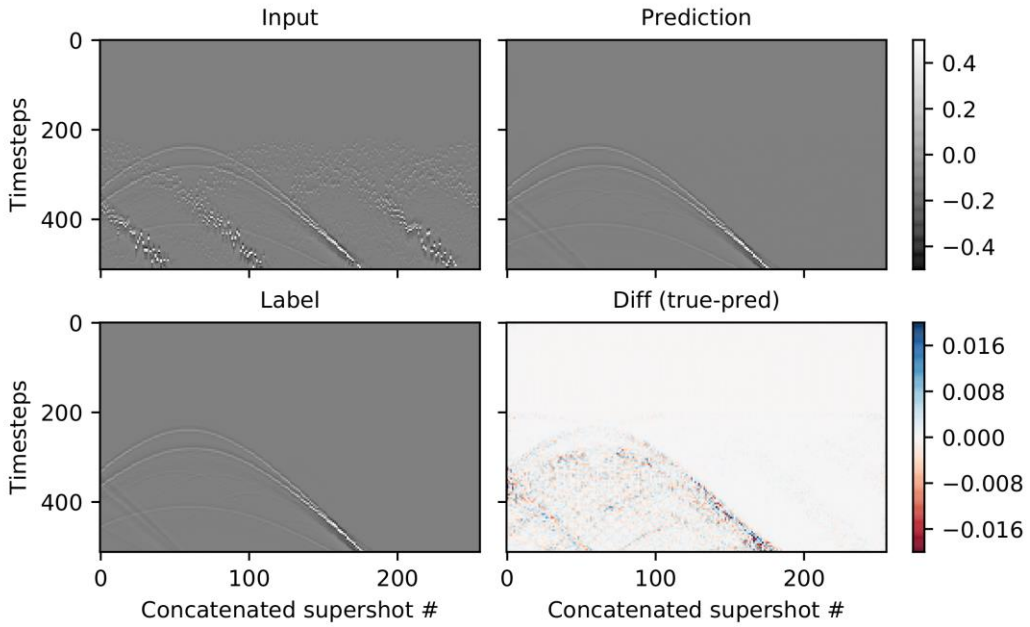

FIG. 2. The prediction and label for a sample in the validation set. All three grayscale images have the same scale. The picture in the bottom right is the residual of the prediction and the label, with a smaller colour scale.

Figure 2 shows an example for a prediction from the validation set. Most of the incoherent noise is removed. Furthermore, the diffractions from the point scatterers are mostly preserved although with some attenuation in their tail endings. The larger errors are concentrated in two regions. The first region is inside the major primary, where the reflections get complicated. Probably the identification of the reflections becomes difficult for the algorithm and the interference complicates this. It is important to keep in mind that the network does not know what reflections or diffractions are, but just percepts them as patterns. Furthermore, we can also see some meshed patterns at the bottom of the residual plot. These patterns could be multiples of the point scatterers' reflections or boundary artifacts. Likely the model behaves poorly for them because of their week amplitude and complexity. The second region is around the tails of the reflection. Something to mention is that the input data have some missing samples due to the removal of time delays, but the "true" data do not. To get the right prediction, the model tries not only to remove the incoherent noise but also to interpolate the missing samples, which itself can be considered as a complex problem to solve. Therefore, the prediction contains relatively larger errors after training on these points.

Conclusions

In this report, we trained a U-Net model to perform deblending, that is the separation of coherent and incoherent signal coming from blended shots. We tried several optimizations and network parameters and found the best combination. In the case where the training and test data come from the same velocity model, the network performs well by preserving small diffractions and correctly identifying primaries. It performs a bit worse for the shots at the edge of the model because of the lack of training pictures representative of this case. For the case where the test data comes from a different velocity model than the training data, the velocity model was made to be simpler than the training model, so it should not exceed the trained model's capability. In

this case, however, the network performs okay but not as well as the first case. More work is required to fully understand how to generalize the network to new problems. To address these issues, we plan to investigate generalizing the model by gradient boosting and provide several models for training.

Acknowledgements

We thank the sponsors of CREWES for continued support. This work was funded by CREWES industrial sponsors, China Scholarship Council (CSC), NSERC (Natural Science and Engineering Research Council of Canada) through the grants CRDPJ 461179-13 and CRDPJ 543578-19. Special thanks to Jian Sun at Penn State University and Marcelo Guarido for valuable discussions.

References

Baardman, R., Tsingas, C. et al., 2019, Classification and suppression of blending noise using convolutional neural networks, in SPE Middle East Oil and Gas Show and Conference, Society of Petroleum Engineers.

Beasley, C. J., Chambers, R. E., and Jiang, Z., 1998, A new look at simultaneous sources, in SEG Technical Program Expanded Abstracts 1998, Society of Exploration Geophysicists, 133–135.

Buda, M., Saha, A., and Mazurowski, M. A., 2019, Association of genomic subtypes of lower-grade gliomas with shape features automatically extracted by a deep learning algorithm: Computers in biology and medicine, 109, 218–225.

Kingma, D. P., and Ba, J., 2014, Adam: A method for stochastic optimization: arXiv preprint arXiv:1412.6980.

LeCun, Y., Bengio, Y., and Hinton, G., 2015, Deep learning: nature, 521, No. 7553, 436.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A., 2017, Automatic differentiation in PyTorch, in NIPS Autodiff Workshop.

Richardson, A., and Feller, C., 2019, Seismic data denoising and deblending using deep learning: arXiv preprint arXiv:1907.01497.

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.