geoconvention
Calgary • Canada • May 13-17 2019

Deep Convolutional Neural Network with Transfer Learning for Automatic Velocity Analysis

Min Jun Park and Mauricio D. Sacchi

Department of Physics, University of Alberta

Summary

We propose using a Convolutional Neural Network (CNN) to perform velocity analysis automatically. We interpret velocity analysis as an image classification problem where we generate a training dataset by modeling synthetic data. After the training process, we test the trained model with another synthetic dataset that was not part of training. In general, the CNN model can predict a reasonable velocity model from unseen input data. However, if the new input data is significantly different from the one used for training, the model will hardly pick the correct velocity. In this case, we use transfer learning to the update the CNN base model with a small portion of the target data. We exemplify our algorithm with a marine dataset from the Gulf of Mexico.

Introduction

Building a velocity model is a critical step in seismic data processing. Traditionally, a processor picks a geologically consistent stacking velocity from the semblance (Taner and Koehler, 1969; Neidell and Taner, 1971). In general, velocity analysis is a time-consuming task because it requires the visual examination of a large number of semblance panels by a processor. Recently, there has been a significant improvement in the field of deep learning due to progress in computer speed and memory (LeCun et al., 2015). CNN is one of the most powerful algorithms to classify images among various approaches of deep learning (LeCun et al., 1989; Simard et al., 2003; Krizhevsky et al., 2012; He et al., 2016). It is superior in extracting features because it uses spatially invariant filters capable of considering the local connectivity of input images (LeCun et al., 1995). In this study, we propose an algorithm to perform automatic velocity analysis using a CNN.

CNN structure

The CNN architecture we use consists of two convolution layers and two fully-connected layers followed by softmax-classification layer. The output feature maps I_j of $(i+1)th$ the convolutional layer is represented by

$$I_j^{i+1} = MP(ReLU(f(K_j^{i+1}, I^i))) \tag{1}$$

where K is the filter matrix in the current layer $i+1$. Similarly, j represents the number of filters (or the number of output feature maps) and $f(\bullet)$ indicates the convolutional operator. A `ReLU` (Rectified Linear Unit) is the activation function and MP is indicates maximum pooling operator. The final output I is flattened through fully-connected layers. We apply `ReLU` and `Dropout` to each fully-connected layer (Srivastava et al., 2014). We also use the `softmax classifier` to estimate the probability of each class. The cost function to train the CNN is given by the cross-entropy. Finally, we use a stochastic gradient descent algorithm implemented in `Tensorflow` (Abadi et al., 2016).

Training data and transfer learning

We define the input data by using two images: the guide image (G) and the target image (T). While G is the entire semblance, T includes only the values in specific τ interval where τ is two-way zero offset traveltime. To define the labels, we divide the semblance velocity axis into 40 compartments and define each compartment as one class. That is, one class represents a specific velocity bin. We

use five synthetic custom velocity models to generate the training data. First, we simulate data from the velocity models by solving the acoustic wave equation with the Finite Difference Method. Then we use analytically computed RMS velocities to assign the target labels to predict. We also conduct CMP sorting and apply automatic gain control to the data to equalize amplitudes. Then, we compute the semblance panels for each model. We also normalize the semblance panels to a common to a predefined scale. After the training, our model can predict stacking velocities from unseen data. However, if the target data and the training data are significantly different, the base model can barely output the correct labels. To overcome this limitation, we use transfer learning (Pan et al., 2010; Yosinski et al., 2014), which is an additional training process with a small portion of target data that permits to update the CNN.

Examples

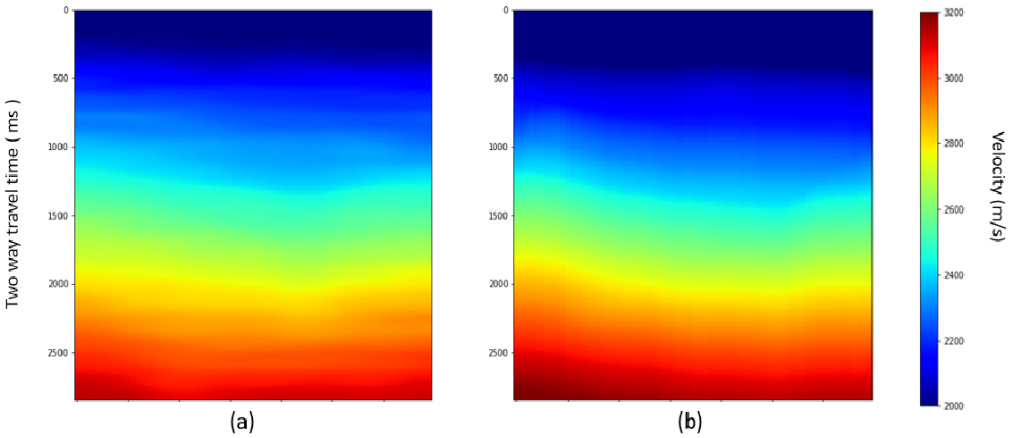


Figure 1: (a) Velocity field predicted via CNN and (b) the label velocity model.

Figures 1 show the label model (b) and the predicted velocity model (a). In the real data case, however, given that the CNN was trained with synthetic data and that our real marine dataset is quite different from the synthetic data, we have to adopt transfer learning. In other words, we extracted nine velocity profiles (dashed lines in Figure 2) from the real data which were obtained via manual velocity analysis and use them for transfer learning. Figure 2 shows the predicted velocity field via CNN. White lines indicate the positions of the CMP locations where we adopted transfer learning.

To verify our result, we first perform the NMO correction. Figure 3 represents the results of the NMO correction performed using both the velocity from the manual analysis (Red lines) and the velocity from the CNN prediction (White lines). Finally, we compare stack sections for both the manual analysis (Figure 4 (a)) and the CNN prediction (Figure 4 (b)) We can observe the notable similarity of the two sections.

Conclusion

In this study, we have developed a method to perform velocity analysis automatically via a CNN. We expect the trained CNN model to act like a professional human processor who can consider the entire trend of the semblance and at the same time select focused energy at different times. Both synthetic and real examples are tested with our method, and the results are considered reasonable. Our approach has the potential to update the base model continuously as new data is incorporated into our analysis. As the training data accumulates, the CNN can continue to evolve. When the base

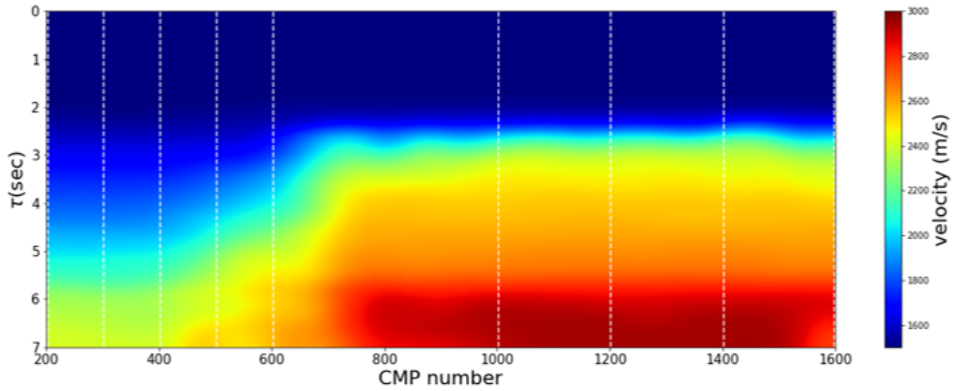
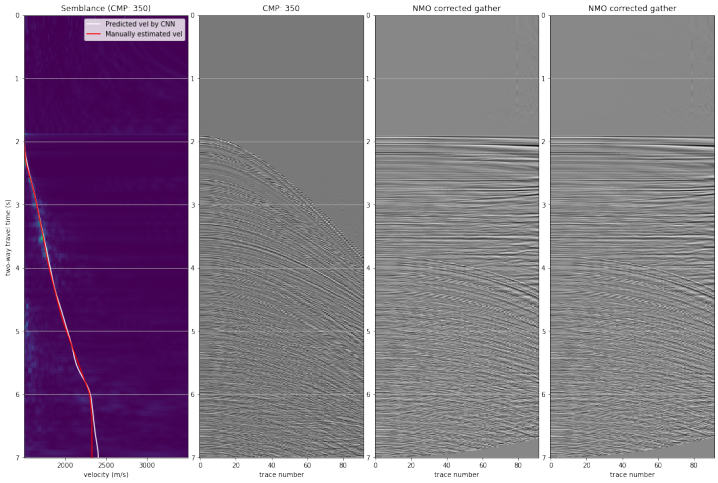
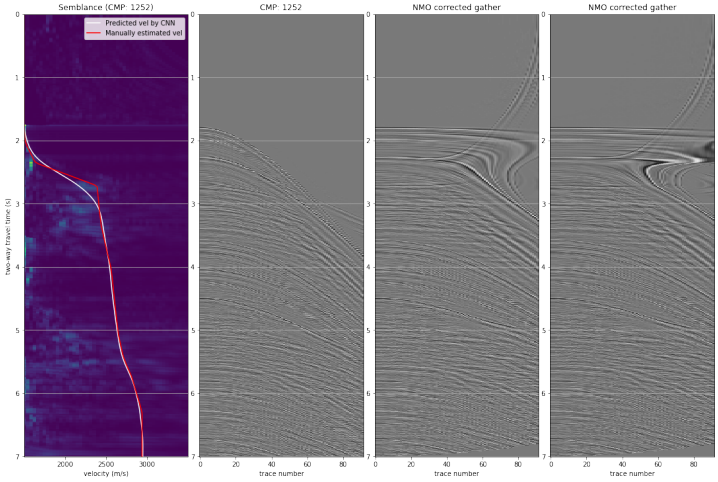


Figure 2: The predicted velocity model from CNN. We use 9 semblance panels from a total number of 1400 at CMP: 202,302,402,502,602,1002,1202,1402,1599 to perform transfer learning. Dashed lines indicate the location of them.



(a)



(b)

Figure 3: NMO corrected data for CMPs number (a) 350 and (b) 1252. From left to right: Semblance, CMP, CMP after NMO with manually picked velocity and automatically estimated velocties, respectively.

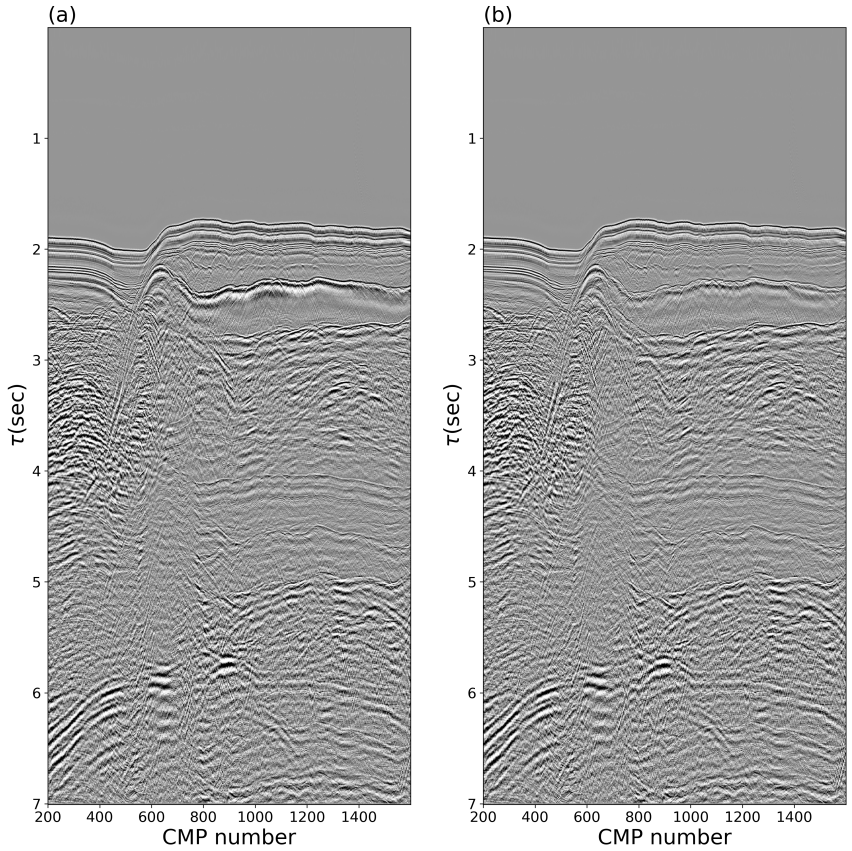


Figure 4: (a) Stack section obtained with manual analysis velocity model. (b) Stack section obtained via the velocity model predicted by the CNN.

model failed to predict the velocity field correctly, transfer learning with a small portion of the target data was used to improve predictability.

Acknowledgements

The authors are grateful to the sponsors of Signal Analysis and Imaging Group (SAIG) at the University of Alberta for their continued support.

References

Abadi, M., P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., 2016, Tensorflow: a system for large-scale machine learning.: OSDI, 265–283.

He, K., X. Zhang, S. Ren, and J. Sun, 2016, Deep residual learning for image recognition: Proceedings of the IEEE conference on computer vision and pattern recognition, 770–778.

Krizhevsky, A., I. Sutskever, and G. E. Hinton, 2012, Imagenet classification with deep convolutional neural networks: Advances in neural information processing systems, 1097–1105.

LeCun, Y., Y. Bengio, et al., 1995, Convolutional networks for images, speech, and time series: The handbook of brain theory and neural networks, **3361**, 1995.

LeCun, Y., Y. Bengio, and G. Hinton, 2015, Deep learning: nature, **521**, 436.

LeCun, Y., B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, 1989, Backpropagation applied to handwritten zip code recognition: Neural computation, **1**, 541–551.

Neidell, N. S., and M. T. Taner, 1971, Semblance and other coherency measures for multichannel data: Geophysics, **36**, 482–497.

Pan, S. J., Q. Yang, et al., 2010, A survey on transfer learning: IEEE Transactions on knowledge and data engineering, **22**, 1345–1359.

Simard, P. Y., D. Steinkraus, and J. C. Platt, 2003, Best practices for convolutional neural networks applied to visual document analysis: null, IEEE, 958.

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, 2014, Dropout: a simple way to prevent neural networks from overfitting: The Journal of Machine Learning Research, **15**, 1929–1958.

Taner, M. T., and F. Koehler, 1969, Velocity spectraâĂŤdigital computer derivation applications of velocity functions: Geophysics, **34**, 859–881.

Yosinski, J., J. Clune, Y. Bengio, and H. Lipson, 2014, How transferable are features in deep neural networks?: Advances in neural information processing systems, 3320–3328.