PSDM velocity model building using a Deep Convolutional Neural Network

GREG CAMERON

THRUST BELT IMAGING, GREG@TBI.CA

Abstract

Building PSDM velocity models in complex structure land environments is difficult. A machine-learning method using a convolutional neural network (CNN) incorporates both human and artificial intelligence to overcome these difficulties. The supervised learning process used a large representative dataset to train the CNN, learning the convolutional weights that best map the input seismic shot records to the target velocity model. With careful consideration of both training data and network architecture, the CNN can accurately predict velocity models on both synthetic and field data.

Introduction

Convolutional neural networks have been successfully used for many tasks, from computer vision to natural language processing. Seismic data, with its spatial and temporal dimensions, lends itself to applications using CNNs. In this paper, I propose a method using a CNN to predict PSDM velocity models in complex structure land environments (see Figure 1).

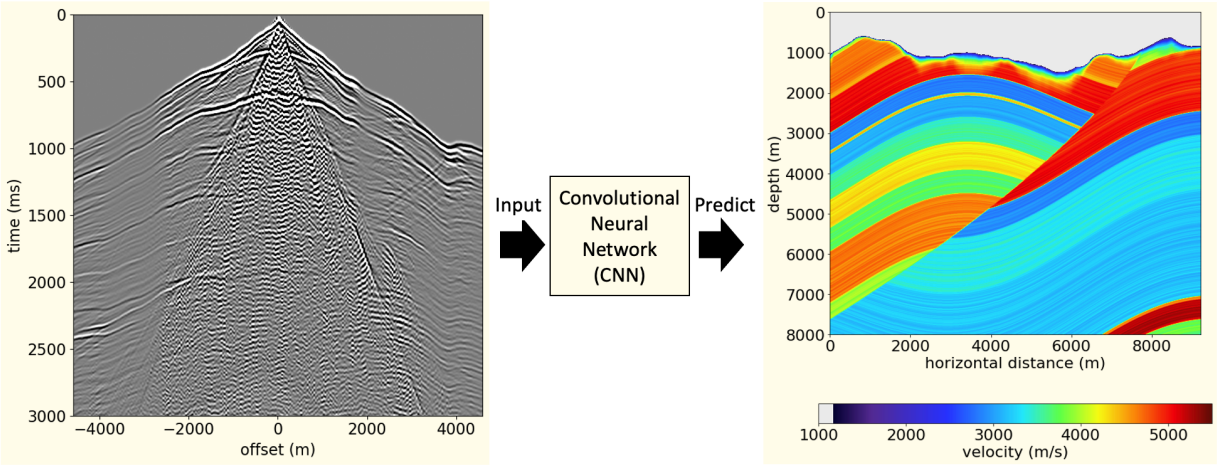

Figure 1. Mapping seismic shot records to velocity model using a deep convolutional neural network

It is challenging to build PSDM velocity models in complex structure land environments. We often have a lack of geological constraints, including limited well information and an incomplete understanding of the subsurface geometry. The seismic data acquired in these areas tends to

be sparse due to environmental and economic constraints. The data often contain high levels of noise and poor frequency bandwidth. This is due to the complex near-surface, which causes many distortions to the seismic data. Given these challenges, automated methods such as reflection tomography and full waveform inversion struggle in these environments. My objective was to see if a CNN-based method could achieve better results.

This method requires three steps as follows:

1. Build a large set of velocity models, incorporating geological structures one would expect to encounter in foothills environments.
2. Create synthetic seismic data over these velocity models using TTI acoustic finite difference modeling.
3. Train the CNN, predicting the velocity model from the synthetic seismic data. Once the neural network is trained, it can be used to predict velocity models on other synthetic and field data. The workflow is shown in Figure 2.

Training Data

This method relies on a large, representative dataset to train the CNN. Drawing from experience in complex-structure land environments, 3000 unique velocity models were created. These models were created automatically, each containing velocities, dips, and faults typical of foothills settings. I kept 20 models separate from the training process to be used to evaluate the CNN after the training was complete. Including the reciprocals of the remaining 2980 models resulted in 5860 training samples along with 100 validation samples.

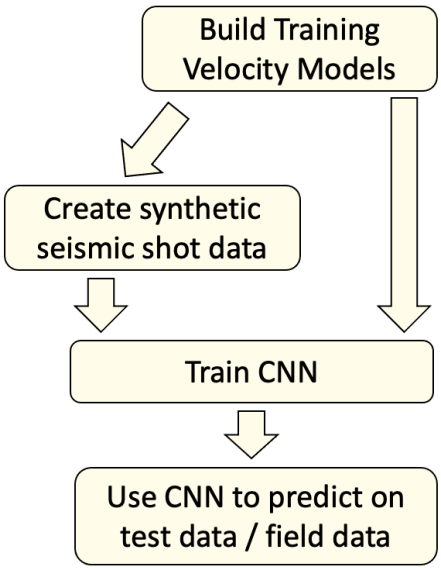

Figure 2. CNN velocity model prediction workflow

Each model contains many layers of variable thickness and velocity. These layers follow a dipping sinusoidal shape with variable dips, periods, and amplitudes. The models contain a number of listric faults with variable dips, curvatures, and orientations. Each model also contains uniquely variable topography and near-surface velocity to represent the weathering layer typically observed in these environments. Figure 3 shows one of the models.

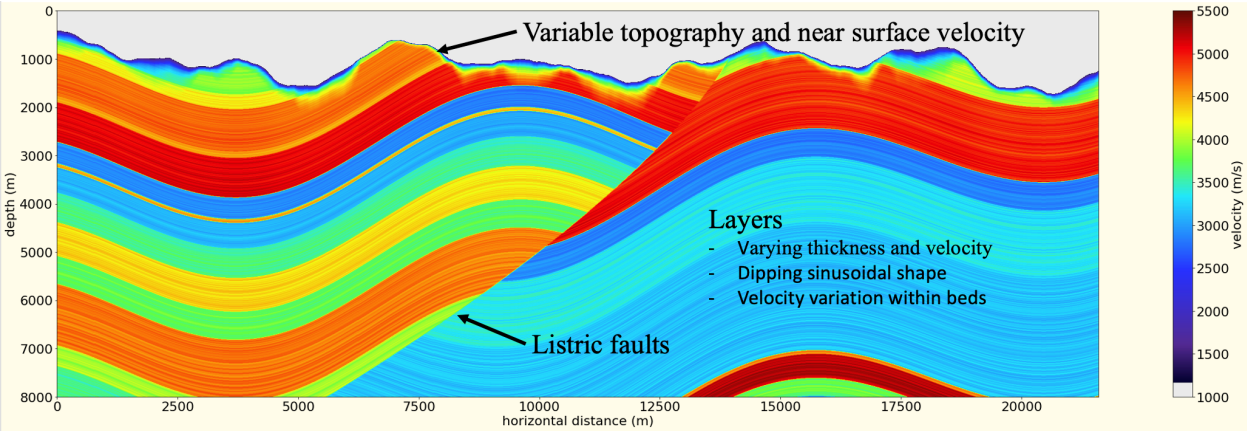

Figure 3. One of 3000 velocity models created automatically using realistic subsurface parameters

As they are created with random variables, the individual models may not look geologically plausible. However, collectively they are representative of the characteristics of subsurface features we expect to see in the field. This provides the CNN a rich training dataset to learn from, allowing it to predict accurate models from new synthetic data it has not seen before. If the training data captures the variety of data seen in the field, the CNN will also be able to predict velocities on field data. Figure 4 shows the first 30 training models.

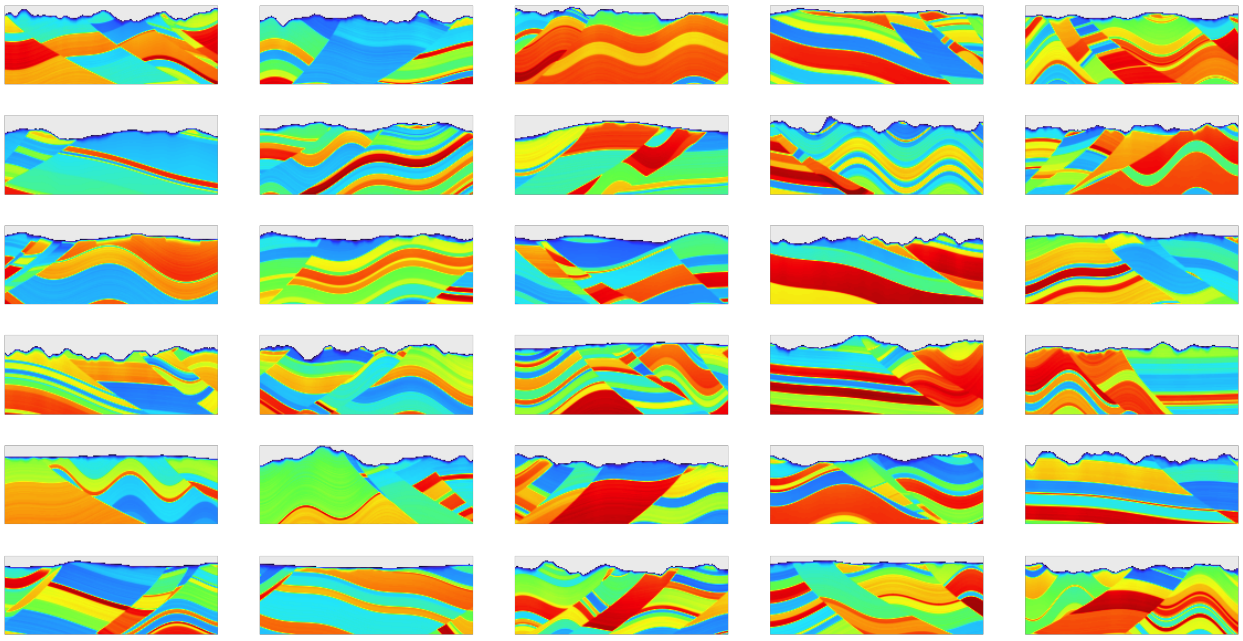

Figure 4. 30 velocity models created automatically using realistic subsurface parameters

The input to the CNN is seismic shot records. For each velocity model a set of shot records was created using proprietary TTI acoustic modeling software written using the open-source Devito library. Thomsen's anisotropic parameters (Thomsen, 1986), epsilon and delta, were held

constant at 0.12 and 0.03 respectively and the dips were defined parallel to bedding. Each model used the same sparse acquisition geometry, with shot spacing of 64m and receiver spacing of 36m. Offsets were limited to 4300m. This geometry mimics the geometry of the field data that was used to evaluate the method. Each of the 3000 models has 144 source locations, giving a total of 432,000 shots.

Figure 5 shows the test model with a wavefield snapshot overlaid along with a shot acquired in the middle of the line. The shot record exhibits high complexity in its reflectivity, due to the complex topography and near surface velocity. This gives the shot record a realistic appearance. Also, note the high-amplitude near-offset noise that was caused by numerical dispersion. The wavefield propagation parameters were not fine enough for the very low near-surface velocity. Creating these shots with the parameters required to minimize this noise resulted in a ~10× increase in runtime. As a tradeoff, it was decided to allow the distortion to remain in the dataset. It could be considered to represent noise observed on field data, such as ground roll or s-wave arrivals.

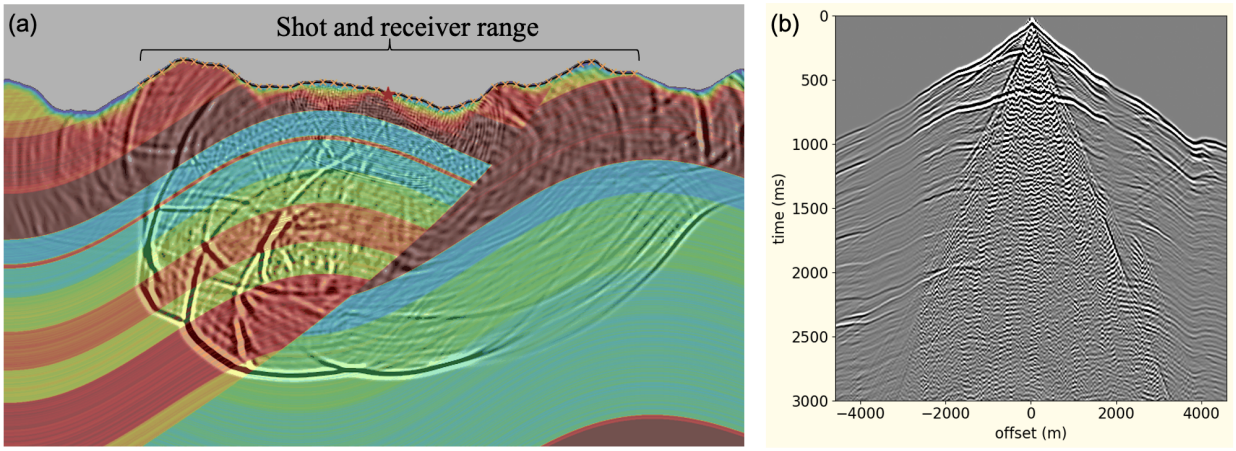

Figure 5. Wavefront propagation through synthetic velocity model (a) and associated shot record (b)

While the shot record has some of the complexity we expect to see in the field, it lacks the wavelet distortion, dispersion effects, and ambient noise we typically observe on field seismic data. Noise was added to our shot records to mimic these effects in two steps. First, each trace was convolved with a unique operator that modelled wavelet distortion and dispersion. Then a variable amount of ambient noise was added to each trace. Figure 6 shows the effects of the added noise on the test shot. The added noise is not related to the model, so the CNN did not learn anything about the velocities from the noise. However, as noise is present in our field data, the CNN will learn to predict velocities in the presence of noise. The CNN was trained on both the clean and noisy data to evaluate its effect on both synthetic and field data. Now that the training data is defined, we can turn our attention to the neural network.

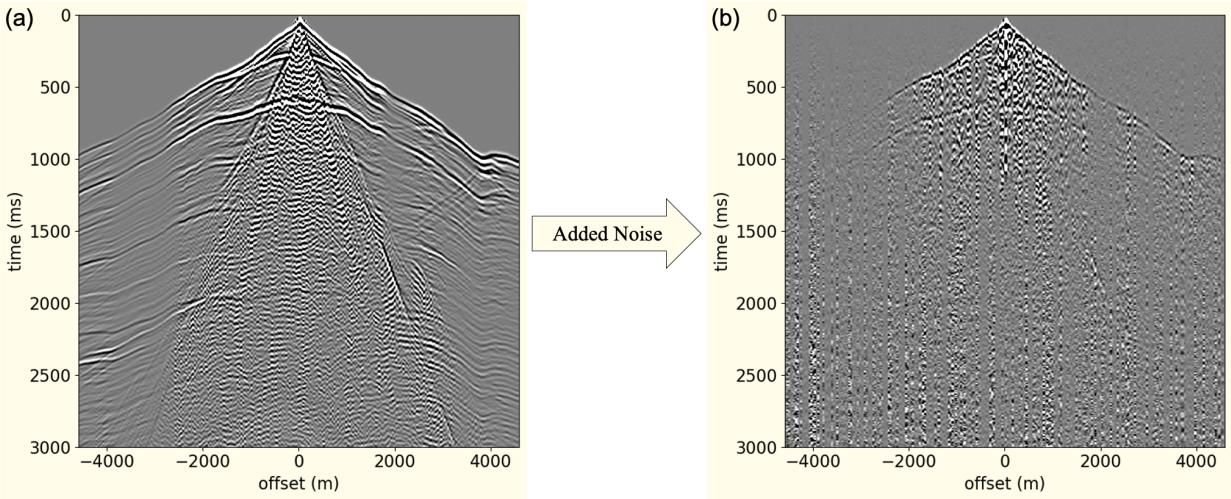

Figure 6. Synthetic shot record without (a) and with (b) noise added

CNN Architecture

Convolutional neural networks vary in size and shape but all have the same objective: to learn the convolutional weights (features) that best map an input to a target. In this case, the input is a set of seismic shot records and the target is the velocity model.

The CNN designed for this experiment follows a U-net architecture, as shown in figure 7. The shot records are input on the left of the network and pass through several types of layers before predicting the target velocity model. The convolutional layers each map the input to output through a set of 3×3 convolutions. The max pooling layers reduce the dimensionality of the data by selecting the maximum value from each 2×2 samples. The transpose convolutional layers are similar to the convolutional layers but also expand the data by a factor of 2×2. This architecture allows the network to learn larger scale features as it progresses deeper in the network before expanding back to the resolution of the velocity model. There are also skip connections, which allow finer resolution features to skip a portion of the network. This provides a higher resolution output and generally improves the training.

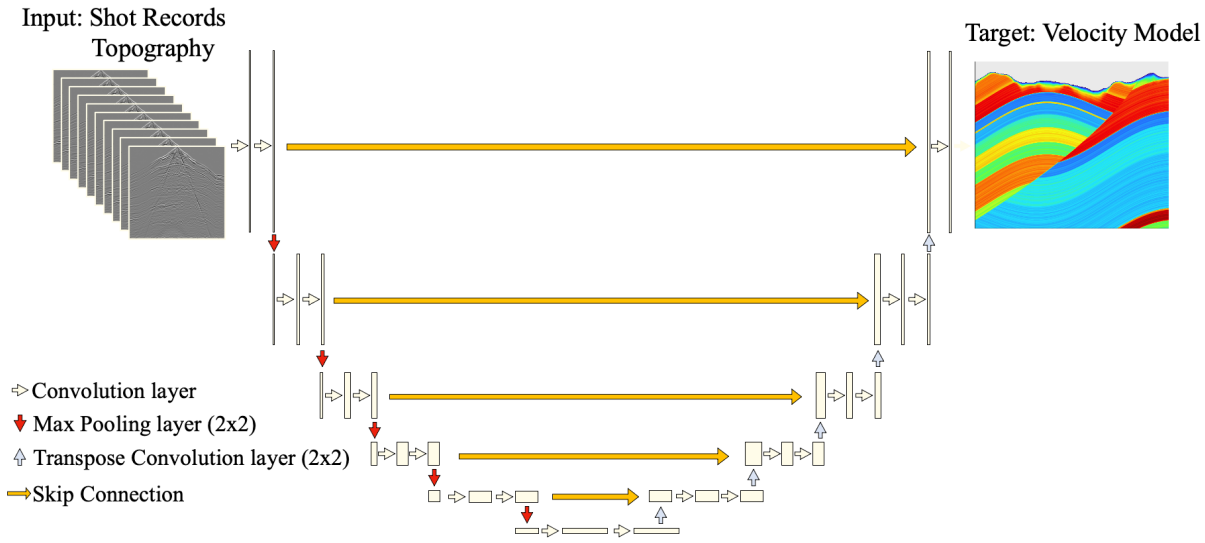

Figure 7. CNN U-net architecture

A RELU activation function was used for each of the convolutional layers to introduce nonlinearity into the network. The network was trained with back propagation using stochastic gradient descent. The training minimizes the mean-squared error (MSE) between the predicted and target velocity models.

Several different CNN structures were evaluated and it was observed that that modifying the data structure improved the performance of the network. The initial architecture was based on a CNN described by Yang and Ma (2019). They propose each input shot being treated as its own channel, resulting in the number of input channels equal to the number of shots. This is analogous to how RGB images are typically handled in a CNN, with each colour being a separate channel. An example showing the first few layers of this architecture is shown in figure 8a. In this simple example, there are input 3 shots and 8 convolutions per layer. Once the two convolution layers are complete, the data are passed to the next convolutional layer through a max pooling layer and also passed across the network using a skip connection.

The modification made to this architecture is shown in figure 8b. The entire set of shot records is treated as a single channel, adding a third dimension to the input data. As with the original network there are 8 convolutions per layer, this time resulting in 8 filter maps for each shot. As before, the data are passed to the next convolutional layer through a max pooling layer. However, when using a skip connection to pass the data across the network we first need to reduce the data to 2 dimensions per filter map. This is achieved using a global max pooling layer, which assigns the maximum value from all shots for each sample to create a single set of filter maps.

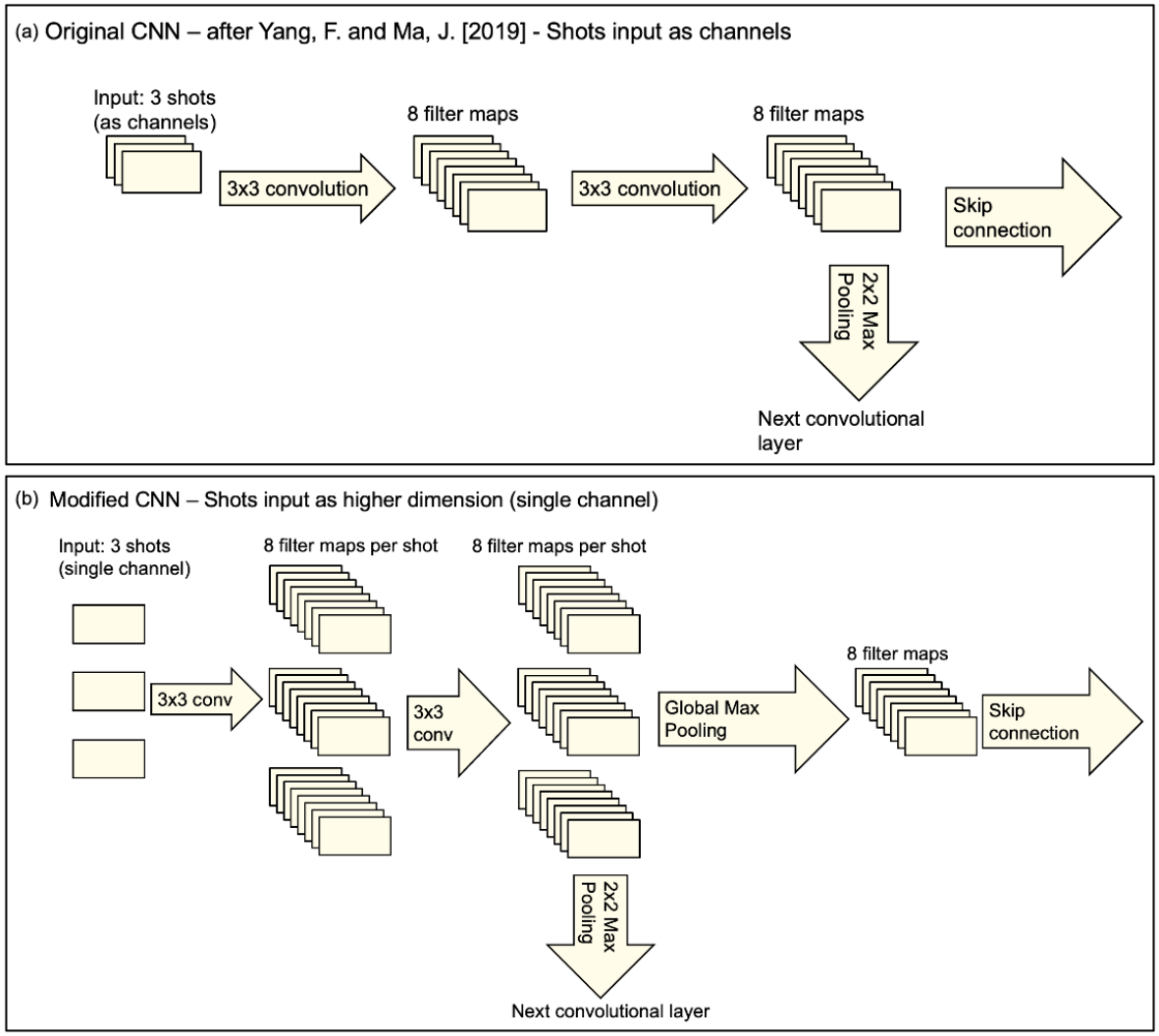

Figure 8. Original (a) and updated (b) U-net architecture

The above modification to the CNN resulted in much better velocity model predictions. There are several reasons for this. In the original CNN, the shots are combined into a single set of filter maps after the first convolutional layer. This results in more difficult feature extraction in subsequent convolutions. In the modified CNN, the shots are not mixed until they are passed to the right side of the network, using the global max-pooling layer. Also, with the original architecture, the first layer learns independent features for each shot. In the modified CNN, the features are shared between the shots. This results in more general features being learned, which stabilizes the training process. The global max-pooling layer extracts the most important features. This should improve the training, particularly for complex and noisy data. Finally, the modification allows the CNN to handle a variable number of shots. The number of weights

trained in a CNN is dependent on the number of input channels. By removing this dependency, the CNN can be evaluated on a different number of shots than it was trained on.

The disadvantages of the modified CNN relate mainly to efficiency. For a given input there is more data passing through the network. This results in fewer features being learned (given limited GPU memory) and longer training time. In this example, the network learned half of the features and the training took twice as long. Given the uplift in results, this is a reasonable trade-off.

4 CNNs were trained, using the clean and noisy training data for each of the original and modified CNN architectures. Each CNN was trained over 30 epochs (passes through the training data). The velocity was predicted over the area of source and receiver coverage to 3000m below surface as training any deeper reduced the accuracy of the results.

Results – synthetic data

Figure 9 compares the true model to the predicted model from each of the 4 CNNs. Figure 9a is the target model. Figure 9b and 9c show the predicted models using the clean shot records as inputs to the original and updated CNN. Figure 9d and 9e show the predicted models using the noisy shot records as input. The best prediction is using the clean shot records as input to the updated CNN. All 4 predictions show reasonable estimates of the first few hundred meters below the surface. The original CNN shows a poor prediction below the strong velocity inversion, particularly when using the noisy input. The updated CNN shows good predictions of the major velocity boundaries over most of the model, with the clean-shot-record version producing a slightly better result. It is interesting that 9c and 9e are so similar given the amount of noise added for the training of 9e. This suggests that the CNN is still able to extract useful features despite the noise. Also, note that none of the CNNs predicted the high velocity in the footwall of the thrust fault more than 1 km deep. Looking more closely at the synthetic data, the limited acquisition and strong velocity inversion in the shallow model likely prevent the illumination of the fault plane reflector.

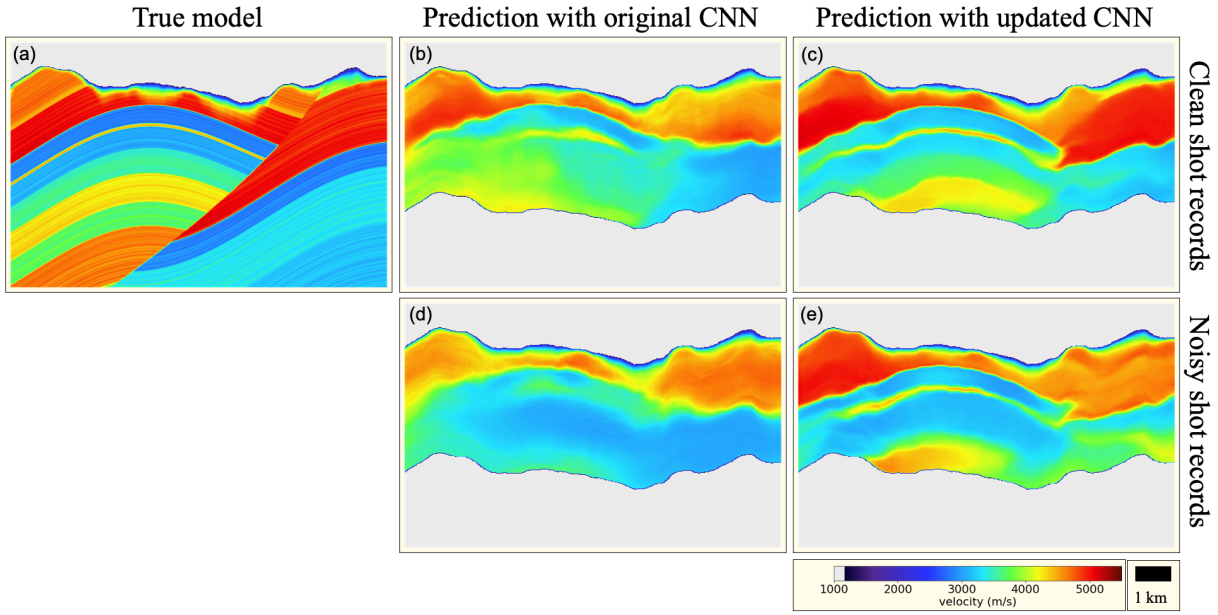

Figure 9. Comparison of true model vs predictions with different CNNs

Results – field data

Once satisfied with the predictions on synthetic data, the CNN was tested on field data. I evaluated the 4 CNNs on a 2D line from the foothills of the Llanos basin in Colombia. Figure 10 shows a selection of shot records from west to east over this line. The data quality is variable, with higher noise content to the west. The best velocity prediction was obtained using

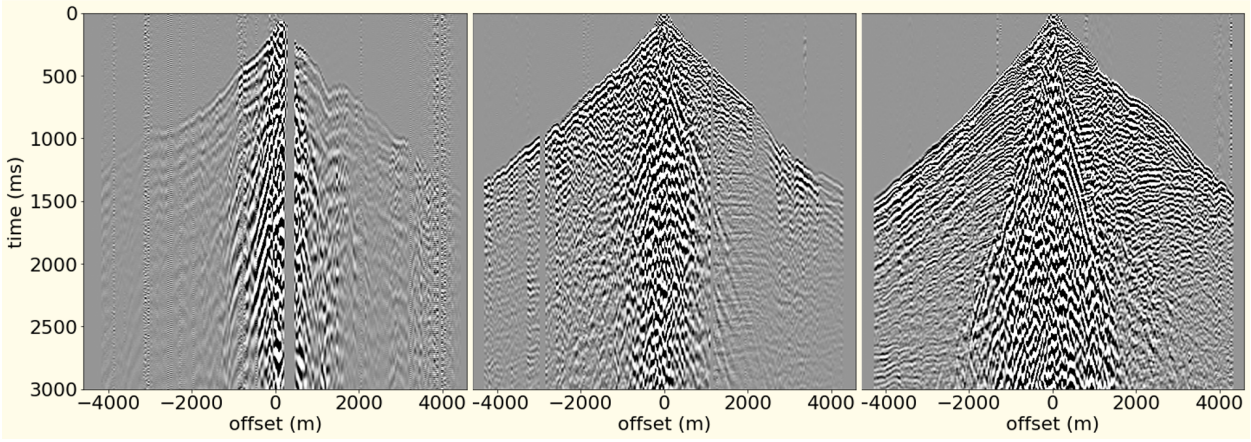

Figure 10. Shot records from Colombian 2D line

the CNN with the noisy input and modified architecture. This is shown in Figure 11a. We have processed this line through PSDM several times and the predicted model agrees with our understanding of the subsurface geometry. The location of the major faults, anticlines, and synclines are shown on the model. Additionally, the shallow velocities are similar to those generated by first arrival tomography, shown in Figure 11b.

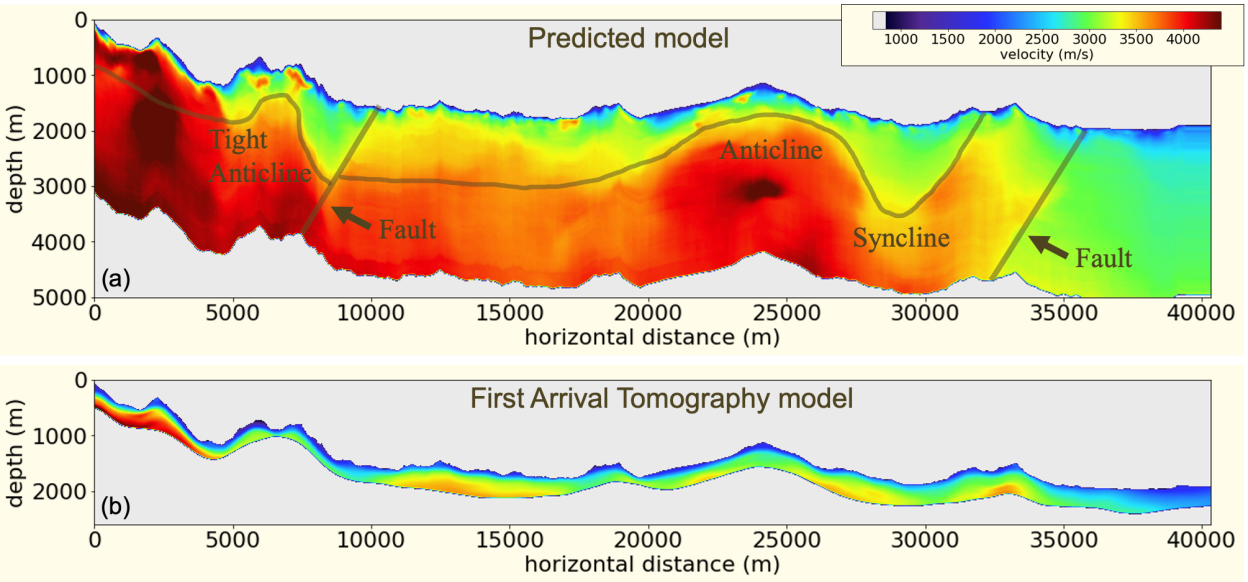

Figure 11. Predicted velocity model (a) and first arrival tomography model (b)

Both the location of the structural features and the similarity to the first-arrival tomography model suggest this is a reasonable prediction of subsurface velocities. Figure 12 shows the predicted velocities using the 4 trained CNNs. Both the addition of noise to the training data and the modification to the network architecture have improved our prediction on the field data.

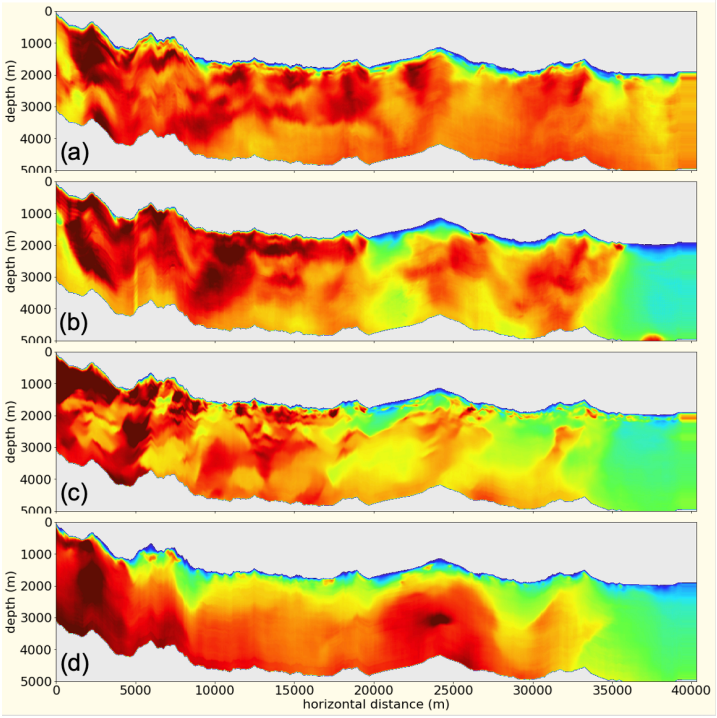

Figure 12. Predicted velocity model using 4 different CNNs: (a) original CNN, clean training data, (b) original CNN, noisy training data, (c) modified CNN, clean training data, (d) modified CNN, noisy training data

Conclusions and future work

This deep learning method for velocity model prediction consists of three primary components. The first is a set of many representative training models. I used our experience in processing complex structure data to produce training models with the ranges of velocities, dips, and faults one would expect in the field. The second component is accurate synthetic seismic data. Using proprietary TTI finite-difference forward-modeling code, synthetic shot records were created from each of our training models. We

added noise that is typical of data we observe in the field. The final component, a tuned neural network, was created using a modified U-net architecture to optimize the results.

The results on synthetic test data were very promising, with the CNN accurately predicting velocities to several kilometres below the surface. The results were encouraging on the Colombian field data example. More work is needed to verify these results by migrating with these velocities.

Since completing this work, I have modified the workflow, using image data rather than raw shot records as input to the CNN. This provides several advantages. It allows for an iterable process, where the predicted velocity model can be used to migrate the data for input to the next iteration. Both input and target are in depth, allowing the required convolutions to be more local. I am also evaluating ways to quantify the uncertainty in the velocity model predictions.

Convolutional Neural Networks offer an alternative to traditional seismic data processing methods. In addition to velocity prediction, CNNs are being developed for interpolation and noise attenuation of seismic data. These methods all require a good understanding of geophysics to build representative training data as well as neural networks to optimize the CNN.

Acknowledgements

I would like to thank the following people and companies for their help on this project:

- Rob Vestrum for his many insights into velocity model building that formed the basis for this work
- Tim MacArthur for his work in building the forward modelling software used for this project
- Thrust Belt Imaging for the time and encouragement to work on this project
- EcoPetrol for the permission to show the Colombian data example

References

Faust, L.Y. [1951] Seismic velocity as a function of depth and geologic time, *Geophysics,* **16,** 192-206.

Kingma, D. and Ba, J. [2014] Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*

Krizhevsky, A., Sutskever, I. and Hinton, G. [2012] ImageNet Classification with Deep Convolutional Neural Networks, *Neural Information Processing Systems,* **25**, 10.1145/3065386.

Louboutin, M., Lange, M., Luporini, F., Kukreja, N., Witte, P. A., Herrmann, F. J., Velesko, P., & Gorman, G. J. [2019]. Devito (v3.1.0): An embedded domain-specific language for finite differences and geophysical exploration. Geoscientific Model Development, **12(3)**, 1165–1187. https://doi.org/10.5194/gmd-12-1165-2019

Luporini, F., Louboutin, M., Lange, M., Kukreja, N., Witte, P., Hückelheim, J., Yount, C., Kelly, P. H. J., Herrmann, F. J., & Gorman, G. J. [2020]. Architecture and Performance of Devito, a System for Automated Stencil Computation. ACM Transactions on Mathematical Software, **46(1)**, 1–28. https://doi.org/10.1145/3374916

Ronneberger, O., Fischer, P. and Brox, T. [2015] U-Net: Convolutional networks for biomedical image segmentation, *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 234– 241.

Thomsen, L., 1986, Weak elastic anisotropy: Geophysics, **51**, 1954–1966.

Yang, F. and Ma, J. [2019] Deep-learning inversion: A next-generation seismic velocity model building method, *Geophysics*, **84,** 583-599

About the Author

Greg Cameron *received his BScE in Geological Engineering from Queen's University in 2001 and his Masters in Data Science and Analytics from the University of Calgary in 2022. He spent 11 years working for WesternGeco, advancing to the role of Area Geophysicist with a focus on land seismic data processing. Since 2012 he has worked at Thrust Belt Imaging as processing manager and geophysical advisor. His current research interest is in building machine learning models to assist in the imaging of complex-structure land seismic data.*