Seismic image focusing analysis using deep learning and residual migration

Joseph Jennings, Robert Clapp and Biondo Biondi

ABSTRACT

We present a workflow for training a deep convolutional neural network (CNN) that aids in the interpretation of geological features that are sub-optimally focused due to a complex overburden. We show that by training a CNN with prestack Stolt residual migration images, the network is able to segment poorly-focused channel features. While we trained the network on synthetic examples, we demonstrate the effectiveness of the network on a 3D test field dataset.

INTRODUCTION

Convolutional neural networks (CNNs) are the current state-of-the art for computer vision and image segmentation. These tools, along with other recent developments stemming from the current theory of statistical and deep learning, have completely transformed certain industries and are also having a large impact on the seismic exploration industry (**?**Pham et al., 2018). While CNNs have shown great success in segmenting geobodies such as faults, channel-like features and even salt bodies, much of this success has been on relatively well-focused seismic images. In practice, CNNs will of course need to be able to perform well on poorly focused seismic images.

In this report, we present a new way to train CNNs using 3D prestack Stolt residual migration images (Stolt, 1996). Prestack Stolt residual migration images contain all of the focusing information necessary for migration velocity analysis and provide the CNN with focusing information in order for it to segment the poorly focused seismic image. As prestack Stolt residual migration is inexpensive to compute, many features can be created quickly providing the images necessary for training. We show that with a network trained on 3D synthetic examples, we are able to segment channel-like features on a 3D Gulf of Mexico dataset.

This report is organized as follows: we first review the basic theory of deep learning and CNNs as well as the theory of prestack Stolt residual migration. Then we discuss the process of creating the synthetic training data and the training and design of our CNN. Finally we show segmentation results on both synthetic and a 3D field dataset.

THEORY

Deep learning

Deep learning falls under the suite of supervised statistical learning algorithms (Friedman et al., 2001). The basic idea of statistical learning is that given a number of data points X and their corresponding labels Y, can we find the mapping f between X and Y? To find this mapping requires three key components. The first of these is the specification of a loss function L that we desire to optimize. The optimization of this loss will hopefully result in a general and accurate mapping $\hat{f}$ where $\hat{f}$ is the estimate of the actual mapping f. The second step is an underlying model that we use in order to parameterize the mapping f. In the case of deep learning, this model takes the form of a deep neural network and is therefore non-linear and can be quite complicated. The last key to estimating $\hat{f}$ is to specify an algorithm for numerically optimizing the loss L. Typically a form of gradient descent is chosen that instead of calculating the total gradient, uses an instantaneous estimate of the gradient (known as stochastic gradient descent). In this work we have chosen to minimize the weighted binary cross-entropy loss and use a deep convolutional network that follows U-net architecture (Ronneberger et al., 2015). For the numerical optimization, we use a form of stochastic gradient descent known as ADAM (Kingma and Ba, 2014).

Prestack Stolt residual migration

To create the features that will be used for training the CNN, we use a prestack depth Stolt residual migration operator. Unlike conventional migration which maps seismic data (function of space and time) to a seismic image (function of space and depth), residual migration maps a seismic image to another seismic image without demigration. Due to this mapping from migrated image to migrated image, residual migration can be cast in terms of a ratio $\rho = v_0/v$ between the true velocity v and the migration velocity v_0 (Sava, 2003). In fact the mapping between the original image and the residually migrated image can be expressed with the following pre-stack dispersion relation

$$\begin{aligned} k_z =& \frac{1}{2}\sqrt{\frac{v_0^2}{v^2}\frac{\left[k_{z_0}^2 + k_{h_x}^2\right]\left[k_{z_0}^2 + k_{m_x}^2\right]}{k_{z_0}^2} - (k_{m_x} + k_{h_x})^2} \\ &+ \frac{1}{2}\sqrt{\frac{v_0^2}{v^2}\frac{\left[k_{z_0}^2 + k_{h_x}^2\right]\left[k_{z_0}^2 + k_{m_x}^2\right]}{k_{z_0}^2} - (k_{m_x} - k_{h_x})^2} \end{aligned} \tag{1}$$

where k_z is the residual migration image wavenumber, k_{z_0} is the wavenumber of the original image, k_{h_x} is the subsurface offset wavenumber and k_{m_x} is the midpoint wavenumber. We apply this migration operator (mapping) for many different values of ρ in order to obtain images that are locally focused for specific values of ρ.

Creation of training data

As deep learning is a supervised learning algorithm, it requires a labeled dataset that is used for training the CNN. As at SEP we do not have any seismic volumes that have labeled geobodies such as channels, we created a training dataset from synthetic data. We used the model builder described in (Clapp, 2018) for building many 3D velocity models that contain river channels. Figure 1 shows one example of such a velocity model with channels. I then calculate the reflectivity from the synthetic velocity

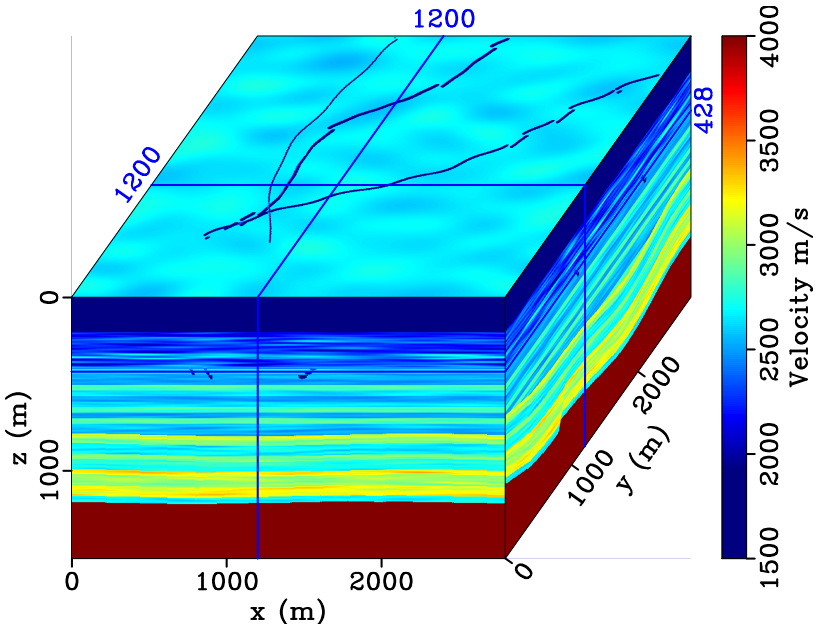


Figure 1: An example of a 3D synthetic velocity model created using the geologic model building software. Notice the channel system at approximately 300m depth shown in the top panel of the cube. [**ER**]

model and convolve with a non-stationary wavelet. This simulates a 3D poststack seismic image that has been imaged with the correct velocity. To create a prestack poorly focused seismic volume, two subsequent operations are required. First, we replicate the zero-offset cube over both angle (γ) and azimuth (ϕ) axes. This creates now a full 5D prestack image which with dimensions of depth (z), midpoint in the x direction (m_x), midpoint in the y direction (m_y), aperture angle (γ) and azimuth (ϕ). Figure 2 shows a slice of this 5D hypercube of data for a fixed m_y and ϕ.

The last step in order to create our synthetic training data is to apply a defocusing operator that can simulate velocity errors that cause defocusing when imaging seismic data. To achieve this defocusing, we first residually migrate the 5D prestack cube for a range of values of ρ. This results in diffractions that appear in the physical space (z, m_x, and m_y) and curvature in the angle and azimuth spaces. An example of this is shown in Figure 3 where we have applied prestack residual migration for a value of $\rho = 0.95$ and taken the same slice as shown in Figure 2. Note that the residually migrated images are a function of τ or zero-offset travel time. The reason for this is

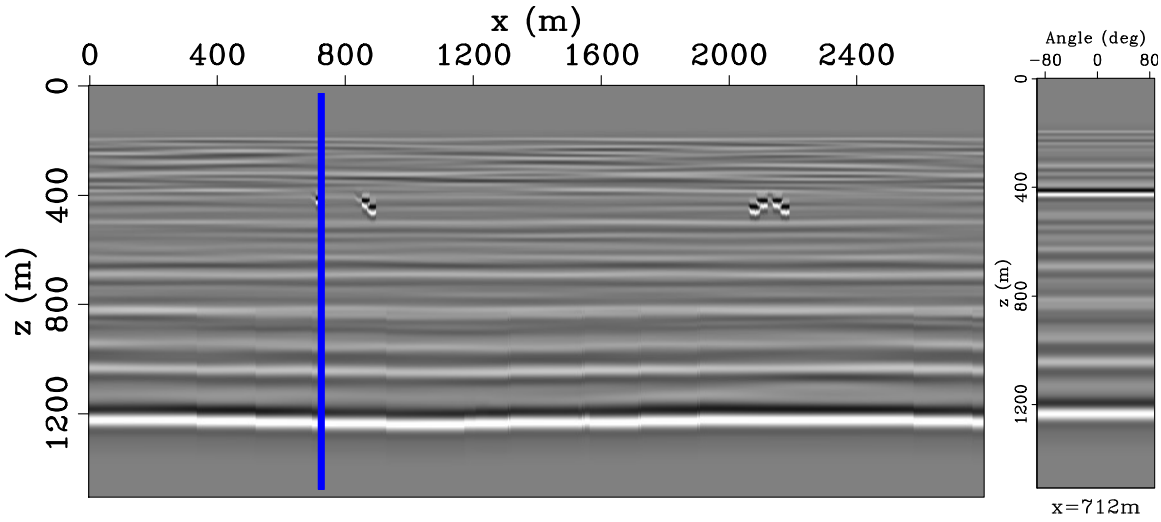


Figure 2: A slice of the 5D prestack image for a fixed m_y and ϕ. The blue line indicates the position at which the angle gather has been extracted along m_x. [**ER**]

because when performing residual depth migration, a vertical shift will be applied to the positions of the reflectors. Converting back to time will keep all reflectors in their original positions. We believe that this simplifies the training data as the network will not need to keep track of the shift in addition to the defocusing of the images.

Now, after residual migration, we have a 6-dimensional prestack image with the additional dimension of ρ. All of the focusing information is contained within this hypercube. To apply a defocusing operator to this cube, we apply a filter that shifts the data along the newly formed ρ axis. For example, this will shift image points that were located at $\rho = 1.02$ now to $\rho = 1$. While this could be done deterministically, when training the network, we desire to have as many different defocusing scenarios as possible. We therefore do this randomly by creating a smooth cube of random numbers that describes this shift along the ρ axis. Figure 4 shows a slice of this cube for a fixed m_y.

Using this random field, we then construct a filter that will apply the shifts along the ρ axis. The result of applying this shift to our prestack image is shown in Figure 5. This figure shows the image for $\rho = 1$. Note that the focusing of the image now changes as a function of space. This is evident by the small diffractions seen at the locations of the channels and also by the fact that many of the reflectors exhibit curvature along the aperture angle axis indicating an incorrect migration velocity (improper focusing of seismic waves). Figure 6 shows another slice of the same cube but now for a fixed z, γ and ϕ.

Applying this same operation of residual migration and defocusing for many different velocity models, we are able to create a training dataset that consists of residual migration images as features and the position of the channels as label. Figure 7 shows one training example.

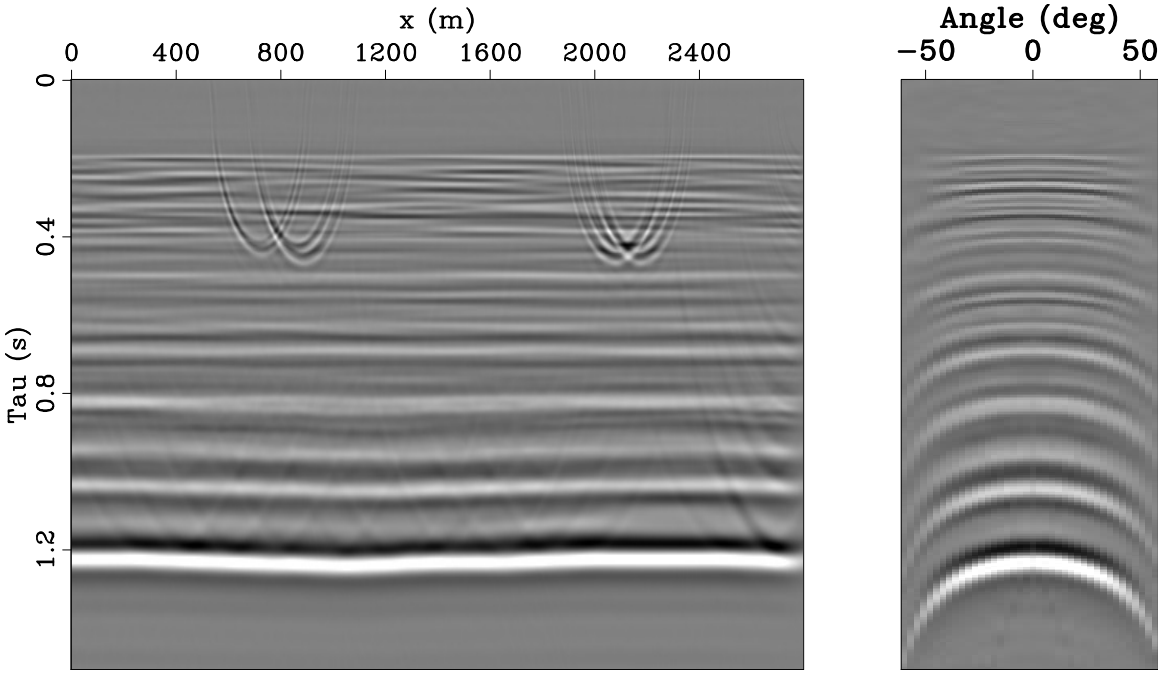


Figure 3: Prestack residual migration applied to the prestack image shown in Figure 2. This residual migration was calculated with $\rho = 0.95$ [**ER**]

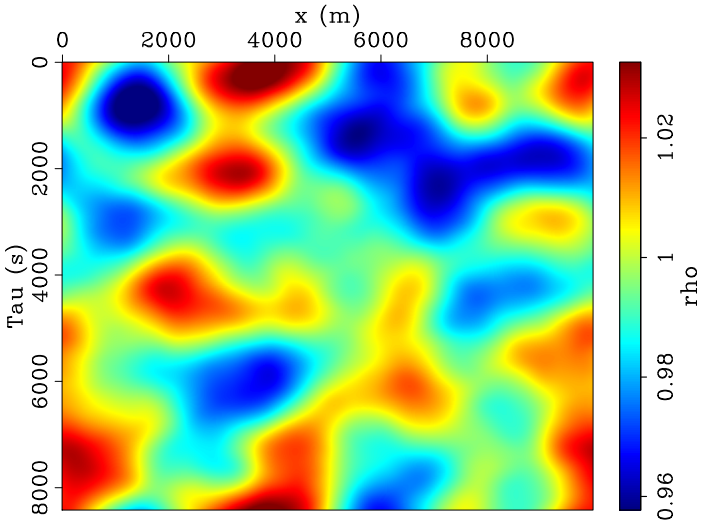


Figure 4: An example of a randomly generated map that describes where in space and for which value ρ the image will be focused. [**ER**]

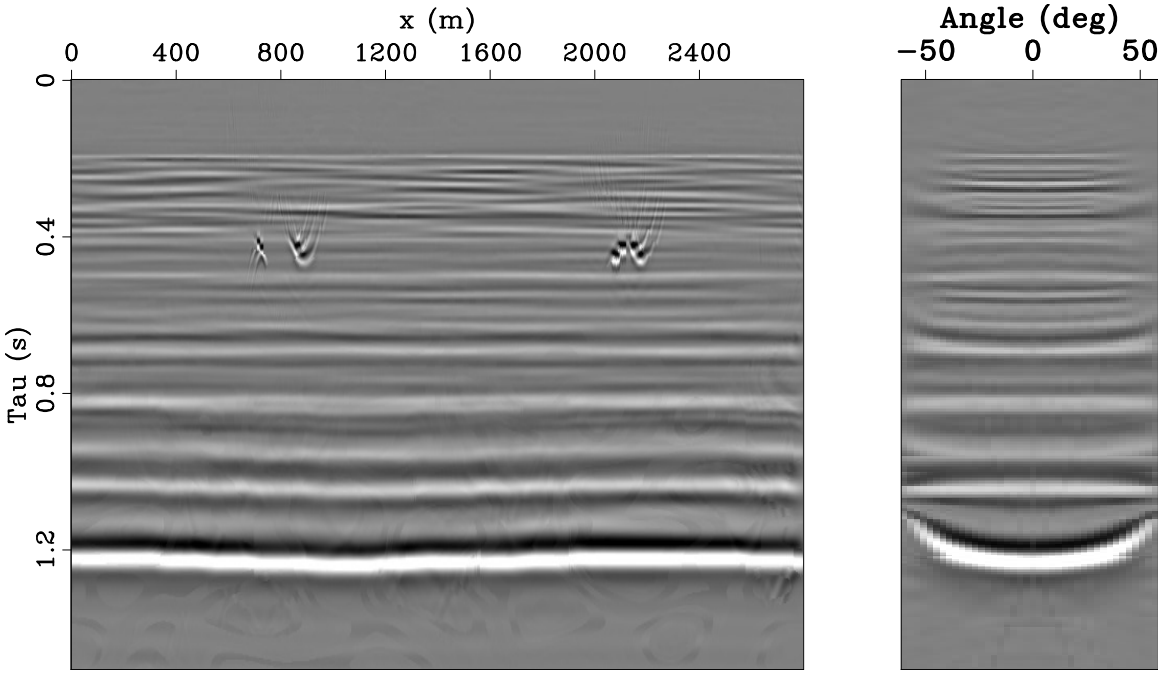


Figure 5: The results of a defocusing operation applied to the image shown in Figure 3. This residual prestack image is for $\rho = 1$. Note that as expected for a real seismic image, the image in they physical space is not completely focused and the angle gathers are not all flat. This represents velocity error in the migration velocity model. [**ER**]

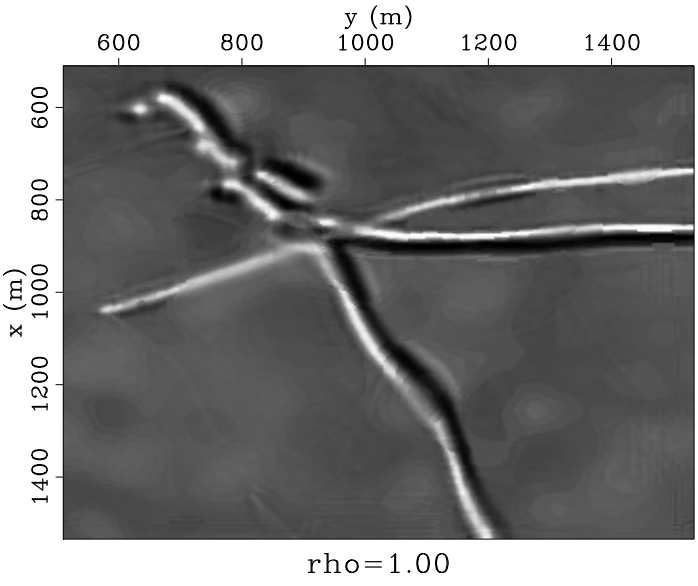


Figure 6: Another slice of the defocused prestack residual migration cube for $\rho = 1$. Notice regions of improper focusing (e.g., at approximately m_x=1200 m and m_y=1000 m the image is not well focused.) [**ER**]

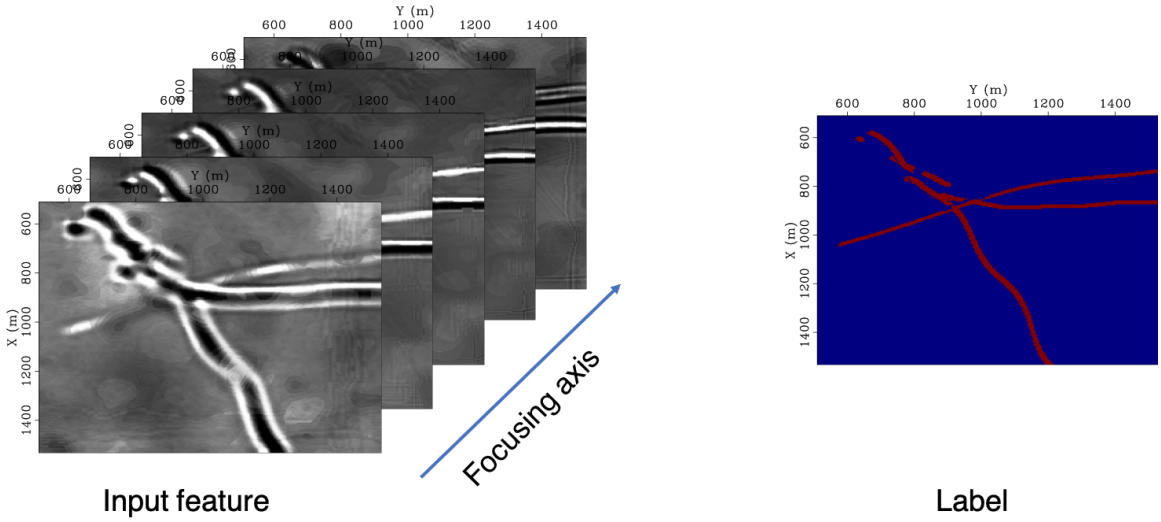


Figure 7: One training data example used to train the CNN. The stack of images on the left show residual migration images of a poorly focused seismic image (the focusing axis is ρ). On the right is the label which takes on a value of one at the position of the channel and zero everywhere else. [**ER**]

Training the CNN

While ideally we would like to pass in the entire 6D hypercube to the CNN for training, this would be very computationally challenging in terms of training. Instead, we pass only slices of the cube for a fixed z, γ and ϕ (what is shown in Figure 6). We further slice each of these slices along both m_x and m_y to create four images from each depth slice. This results in 7296 training images for training the CNN. Further more, we split the training images into 80% for training and 20% for validation.

For the design of the network, we decided to go with a U-net-like (Ronneberger et al., 2015) architecture but without the skip layers. Figure 8 shows a schematic of the network. Each input image was 128×128 samples with ten input channels for ten different residual migration images.

Using a binary-cross entropy loss function, we trained the network for 40 epochs using the ADAM (Kingma and Ba, 2014) optimizer. We discuss the results of the training on both synthetic and field data results in the next section.

RESULTS

To validate the results, we first tested the network on synthetic data on which it had not been trained. We then applied the network to a residually-migrated cube provided to SEP by Chevron (Lomask, 2007).

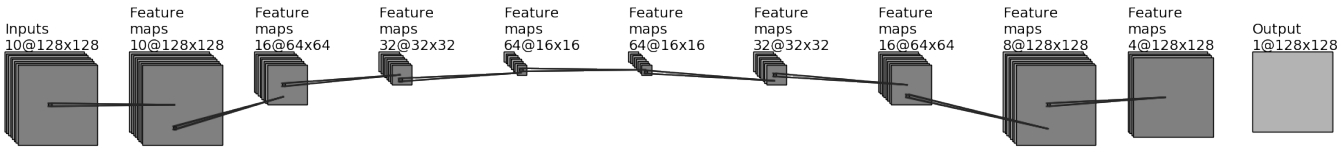


Figure 8: Convolutional neural network architecture used for segmenting the poorly focused seismic images. A filter kernel size of 5×5 was used throughout the network. Relu activitation functions and strided convolutions were used to move from one layer to the next. [**ER**]

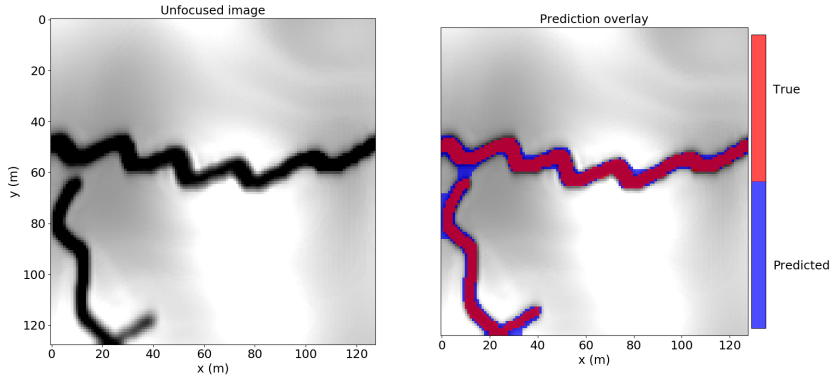
Synthetic data results

Figure 9 shows the segmentation results from the CNN for channels coming from three different velocity models. The images in the left column are one image from the residual migration images that all serve as input to the CNN. The images in the right column show the true locations of the channel feature (red pixels) and the locations classified by the network (blue pixels). Visually, the classification from the network is in good agreement with the true locations of the channels. The small discrepancies that occur when different channels come close is due to how we chose to threshold the predictions. If we had chosen a higher threshold, we would not observe this discrepancy. What we are observing here is essentially an inverse crime of our network. The validation data is very similar to the training data and therefore, it is very easy for the network to segment these geobodies.

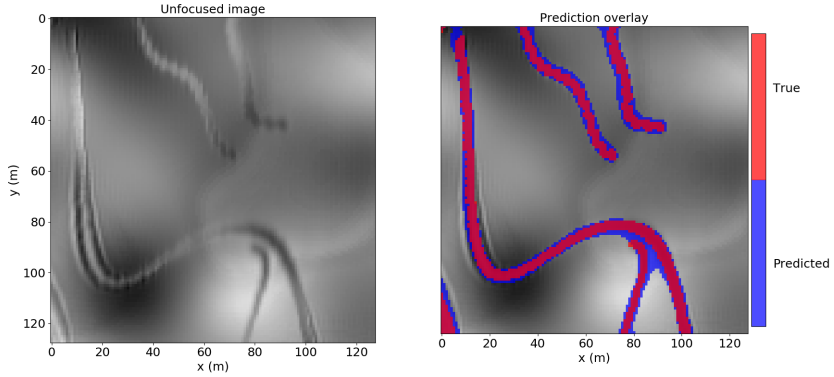
Field data results

To better test the performance of our network, we first residually migrated a 3D poststack seismic volume and extracted one time slice from that volume. Figure 10 shows the time slice for $\rho = 0.85$ and for $\rho = 1$.

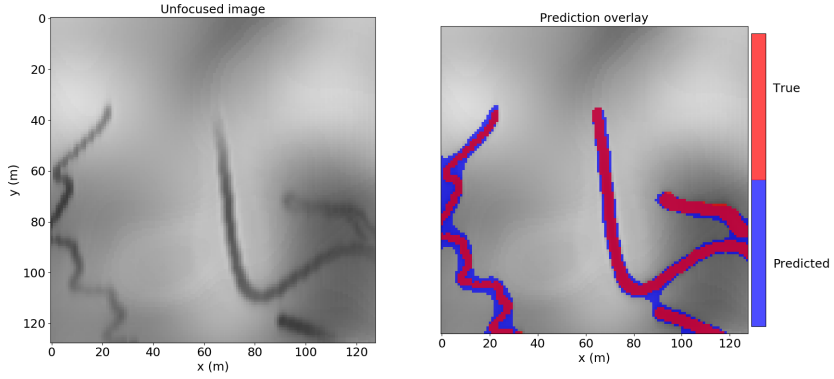
We then split the residually migrated time slices into 128×128 images which are then input to the CNN. The results of the CNN segmentation are shown in Figures 11 and 12. The leftmost of column of these figures shows the residual migration images for $\rho = 0.85$, the middle column shows $\rho = 1$ and the rightmost column shows the results of the segmentation of the network. Compared to the segmentation results on the validation data, the segmentation is much noisier and visually, does not agree as well with the channel features. However, it does successfully classify many pixels that appear to belong to channel features.



(a)



(b)



(c)

Figure 9: Results of neural network segmentation on the validation data. The images in the left column are poorly focused seismic images from different synthetically generated velocity models. The images in the right column are the same image but with overalid the true position of the channel (red pixels) and the predicted positions by the CNN (blue pixels). Note that the true and predicted channel positions are in good agreement. [**CR**]

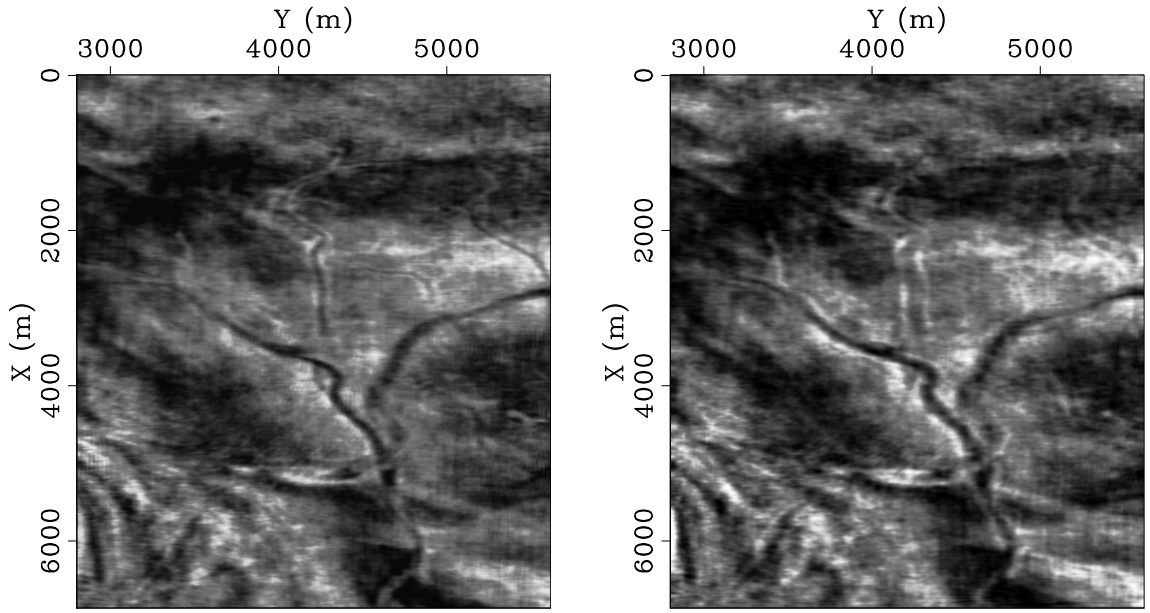
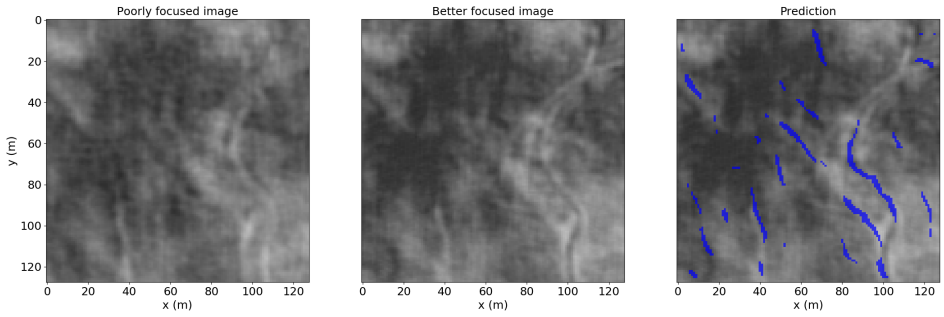


Figure 10: Slices of a residually migrated cube . Panel 10a shows the time slice extracted from the $\rho = 1$ cube. Panel 10b shows the time slice extracted from the $\rho = 0.85$ cube. Notice that the channel features are much broader in Figure 10b than appear in Figure 10a (m_x=2000 m, m_y=4000 m). [**ER**]

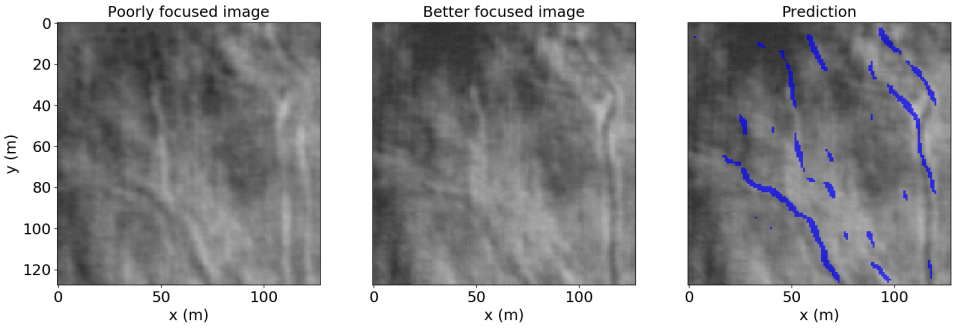
DISCUSSION AND CONCLUSION

While the field data results are quite noisy and not ideal, they are encouraging for a first go and given that the training data are not very representative of the noise we observe in the field data. There are many areas to explore for improvement. The first is how to create more realistic training data. While the channel shapes do appear realistic, the background seismic amplitudes in the synthetic cubes are far from what we observe in the field data. Also, more needs to be done with the channel shapes. For example, in Figure 12a, note that an obvious channel has been completely ignored by the CNN. This is because none of the channels present in the training data were as wide as that channel feature in terms of pixels. A broader range of channels sizes must be created to make more realistic data.

Additionally, we plan to explore much beyond the simple segmentation problem. While implicitly, the CNN is learning image focusing in order to segment the channel features, we would like to attempt to create a CNN that can either tell us how well an image is focused, or perhaps even improve the focusing of an image.

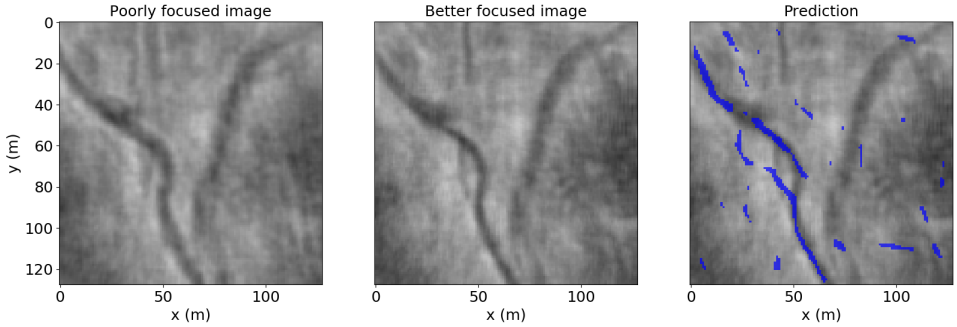


(a)

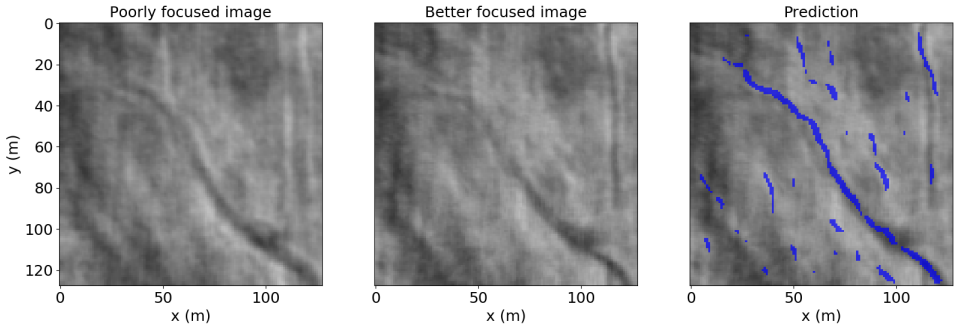


(b)

Figure 11: Segmentation results from the neural network applied to the residually migrated slices shown in Figure 10. Left column shows a poorly focused image $\rho = 0.85$, right column shows a better focused image $\rho = 1.0$ and right column shows the neural network segementation overlain on the image. [**CR**]



(a)



(b)

Figure 12: Same as Figure 11 but for different portions of the images shown in Figure 10. [**CR**]

ACKNOWLEDGEMENTS

We thank Total SA for their financial support of this project and Fantine Huot for discussions and advice.

REFERENCES

Clapp, R. G., 2018, Synthetic model building for training neural networks in a Jupyter notebook.

Friedman, J., T. Hastie, and R. Tibshirani, 2001, The elements of statistical learning: Springer series in statistics New York, NY, USA:, **1**.

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

Lomask, J., 2007, Seismic volumetric flattening and segmentation: PhD thesis, Stanford University.

Pham, N., S. Fomel, and D. Dunlap, 2018, Automatic channel detection using deep learning, *in* SEG Technical Program Expanded Abstracts 2018: Society of Exploration Geophysicists, 2026–2030.

Ronneberger, O., P. Fischer, and T. Brox, 2015, U-net: Convolutional networks for biomedical image segmentation: International Conference on Medical image computing and computer-assisted intervention, Springer, 234–241.

Sava, P. C., 2003, Prestack residual migration in the frequency domain: Geophysics, **68**, 634–640.

Stolt, R. H., 1996, A prestack residual time migration operator: Geophysics, **61**, 605–607.