SPE-196087-MS

Deep Learning-Based Automatic Horizon Identification from Seismic Data

Harshit Gupta, Siddhant Pradhan, Rahul Gogia, Seshan Srirangarajan, Jyoti Phirani, and Sayan Ranu, Indian Institute of Technology Delhi

This paper was prepared for presentation at the SPE Annual Technical Conference and Exhibition held in Calgary, Alberta, Canada, 30 Sep - 2 October 2019.

Abstract

Horizons in a seismic image are geologically signficant surfaces that can be used for understanding geological structures and stratigraphy models. However, horizon tracking in seismic data is a time consuming and challenging task. Saving geologist's time from this seismic interpretation task is essential given the time constraints for the decision making in the oil & gas industry. We take advantage of the deep convolutional neural networks (CNN) to track the horizons directly from the seismic images. We propose a novel automatic seismic horizon tracking method that can reduce the time needed for interpretation, as well as increase the accuracy for the geologists. We show the performance comparison of the proposed CNN model for different training data set sizes and different methods of balancing the classes.

Introduction

Seismic interpretation plays an essential role in identifying geological horizons for oil reservoir exploration. Horizon tracking, facies classification, fault detection and, salt boundary identification from the 3D seismic data are difficult tasks due to the massive size of the data sets involved. Generally, the interpretation process consists of two main elements: software tools and human interpretation. Tools based on high-performance computing have helped to mitigate the processing time for seismic imaging [1–2]. Even for complicated cases or scenarios, the processing time for these tools is very low, but the need for manual inputs or processing using these tools still remains. Traditionally, horizon tracking was achieved by picking features such as peaks, troughs, or zero-crossing points across adjacent seismic traces. To reduce the manual effort, there is a need for a software tool which can incorporate geological expertise in order to analyze and interpret the acquired data. The colossal amount of vetted data that is available makes the deep learning methods suitable for the horizon identification task.

Deep learning has shown great assurance in computer vision [3] and we believe it can help in finding abstract features from images that can be used for tracking horizons. Convolutional neural networks (CNN) are known to learn non-linear functions [4] and underlying patterns [5], which can be hard to formulate and can be applied as part of standard attribute-based algorithms. We show that CNN can be used to identify the presence of the horizon in an image using feature-based pattern recognition. Deep learning allows us to build a learnable model that can help incorporate the expertise of a human interpreter and evolve as new

data is presented to the model. The computational costs involved in identifying horizons using the deep learning models are considerably less since most of the computational time or cost is incurred during model training, which is a one-time activity. Predictions can be made in a matter of seconds using a trained model.

One of the problems that we aim to address is to come up with a methodology for using the large amount of raw seismic data for training deep learning models. We propose a labeling strategy that is computationally efficient for preparing data with known labels. We formulate the horizon identification as a binary classification problem, where we train a patch-based CNN model to classify the image as 1 if it contains a horizon and 0 if it does not. After the model is trained, it computes the relevant features from the image patch and calculates the probability of the image containing a horizon. We carry out important experiments by varying the patch size, strides, and the number of training images.

In a seismic image, only a fraction of the pixels contain horizons, which induces a high-class imbalance for the binary classification problem. We present experimental results based on different strategies for handling class imbalance by under-sampling the majority class, oversampling the minority class, and using a penalty loss function during training. We propose different methods for efficiently labeling the seismic data, handling class-imbalance, and an automatic horizon tracking method.

Related Work

Horizon tracking has been one of the key focus areas in the field of seismic interpretation. Early work on semi-automatic horizon tracking was based on a human interpreter or geologist, providing an initial seed point [6] to start tracking the horizon in one seismic image. These methods can be categorized into three types [7]. In the first category, few horizon slices are manually picked, followed by interpolation for obtaining the horizon volume [8, 9]. The second category includes techniques that use local dips and least-squares method for automatically tracking the horizon [10]. The third category consists of a relative geological time data transformation method for automatic tracking [11]. However, all these methods need a starting point to track the whole horizon given by an interpreter. In addition, horizon tracking becomes inaccurate in the region around faults and at low depths due to low signal-to-noise ratio (SNR) of the seismic images.

Recent developments in deep learning have the ability to enable geologists to automate the seismic interpretation process. Researchers have used artificial neural networks and deep neural networks for fault detection [12], seismic facies identification, and salt body classification [13–15]. Our work with seismic data has shown a lot of promise for automating the horizon tracking task using deep learning.

We use CNN to extract features from the 2D images that are relevant for detecting the presence of a horizon. To locate the horizons, we present a patch-based approach where each image patch is classified using CNN based on the presence (or absence) of a horizon in that image patch. Since we detect the horizons based on the relationship between the features from the image, captured by the CNN model, our method provides a direct correlation between the seismic data and horizons. This allows us to track multiple horizons simultaneously.

Methodology

One of the significant challenges in training machine learning models on seismic data is the class imbalance. The F3 data set of dimensions 651 × 951 × 463 with five ground truth horizons has been considered in this work [16, 18]. The horizons are essentially pixel wide lines stretching from one end of the image to the other. After labeling, the ratio of the non-horizon to horizon class is around 100:1. Hence, binary semantic segmentation of the images becomes a difficult task, with heavy loss function penalties associated with the sparse class not leading to any substantial improvement in the results. Besides, the models required are very deep and require tremendous amounts of memory and time to converge to a solution. To make this problem more pliable, we assume that it is sufficient to localize the horizons with a margin of a few pixels

of their original location. This enables the use of shallow neural network models, which are more memory efficient and lead to faster convergence. To test our classifier, we apply the following process: first, we scale the amplitude values of the entire section to be between −1 and 1, then the histogram is clipped below a threshold to enhance visualization and to set the section as close as possible to the training conditions. Second, we extract one test image from the cube. Third, for each image, we extract patches. The workflow has the following steps: labeling the data, training the model, making predictions, and then performance evaluation or comparison. These steps are described in detail next.

Labelling the dataset

Our task is to track horizons across a seismic image drawn from the seismic cube. As shown in Fig. 1, the seismic cube is essentially an array of 2D images. Each image is split into a grid of tiles each of dimension n × n pixels. Each tile of the grid is a sub-image that is used as an input to the model for training. Only square tiles are used. To train the CNN classifier, we associate each tile with its class label. To determine if a tile contains a horizon or not, we could iterate through the list of horizon coordinates and do this for each tile in the image. However, this would be a very computationally time-consuming task. Thus, in order to prepare the labels for each tile, we create a cube of the same dimension as the data set and fill the horizon locations with 1s and the remaining locations with 0s, thus resulting in a binary cube. For each tile in the seismic cube, the presence of at least a 1 in the corresponding tile from the binary cube quickly indicates the presence of a horizon in the tile and thus increases the computational effectiveness of the data preparation step.

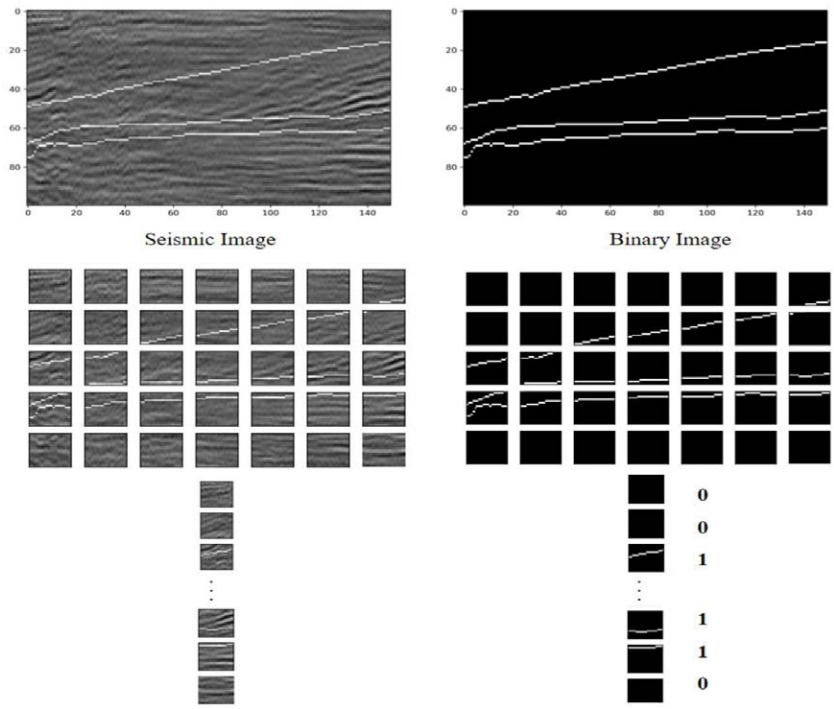

Figure 1—Labels for the seismic sub-images are generated by splitting the binary image slice corresponding to the seismic image slice into sub-images or tiles. We use the 'any entry is 1' to label the binary tile as positive class else the tile is labeled as negative class.

Tiles from the data images are matched with the corresponding tiles from the binary images. If any pixel value in the binary tile is 1, we attach a 1 label to the corresponding data tile indicating that it belongs to the horizon class. For our model, the training data is the array of n × n sub-images, and the training label is the corresponding 0/1 array, created as described above. In the best-case scenario, this method improves the non-horizon to horizon class ratio from around 100:1 to around 100:n.

Training

We use two different publicly available datasets for training and testing, namely F3 Netherlands Offshore and Penobscot data set [18]. We adopt supervised learning to train the model where the model attempts to learn the underlying patterns given the input sub-images and the corresponding target labels indicating the presence or absence of horizons.

To address the class imbalance, data duplication of the horizon class tiles or random sampling from the non-horizon class tiles is carried out until the ratio of horizon to non-horizon tiles passed to the model for training is 1:1. The images are selected in three different ways to train the model. The first method is the sequential selection. The idea behind choosing the first few images is to emulate a real-world scenario where a geologist will label horizons in the first x% of the images, and the model will predict the horizons in the remaining images. However, this is a case in which it is difficult for the model to produce good results because the pattern and formation of rocks change as we move in one direction, and the model is being effectively trained locally only in one section of the cube.

In the second method, we shuffle images along one direction of the cube, either inline or crossline, and train on the first x% of the images. This methodology helps to counteract the problem mentioned above by allowing the model to be trained on geological features found across the cube. Nevertheless, it can prove to be a challenge for a geologist to label a horizon several hundred images away, since its exact location may change.

In the third method, we select the images from the cube which are fixed number of steps apart. The idea is to label a few images which are important to train the model. Sequential images may contain redundant or repeated information, whereas random shuffling may not give proper insight into what the model has seen. In ordered selection, we control information that is being passed to train the model and our experiments show that it is the best method to choose images for training the model.

We applied and tested several different CNN architectures and hyper-parameters, along with different input patch sizes. Beginning with the common LeNet [17], we added complexity since deeper networks are better at differentiating classes. Another aspect of deeper networks is that small convolution masks of size such as 3×3 pixels are expected to give better results. However, adding too many layers to a network can be detrimental since the training time increases drastically and the model will also be prone to overfitting. Additionally, the training sessions used few common parameters such as a learning rate of 0.001, momentum of 0.9, and input batch size of 32 patches.

In all our experiments, we ran our model for a batch size of 256 tiles. We also capped the epochs at 100. We employed early stopping in case the accuracy metric during training did not improve over 5 consecutive epochs. In our experiments, the model tends to converge before 100 epochs. The experiments were run on NVIDIA K40 (12GB, 2880 CUDA cores) GPU.

Network Architecture

The proposed CNN model architecture is shown in Fig. 2. The model consists of 3 convolutional layers with kernel size (3, 3), and 32 filters each. The stride in either direction is 1 and the activation function employed for each layer is ReLU [20]. The first three layers are responsible for capturing the pattern information from the sub-images. Between two consecutive convolutional layers, there is a batch normalization layer. Batch normalization layer normalizes the output to zero mean and unit variance which speeds up the training and performs dropout independently. This is followed by a MaxPool layer of kernel size (2, 2), and stride (2, 2). Output of the MaxPool layer is flattened and fed to a fully connected layer of length 128 with a ReLU activation function [20], and finally terminating with another fully connected layer of length 2, with a SoftMax activation function. Between the two fully connected layers, there is a dropout layer with a ratio of 0.5. Dropout is an effective way of preventing model overfitting without having to explicitly include a regularizer [21]. A stochastic optimization algorithm is used to update weights while training the network [19].

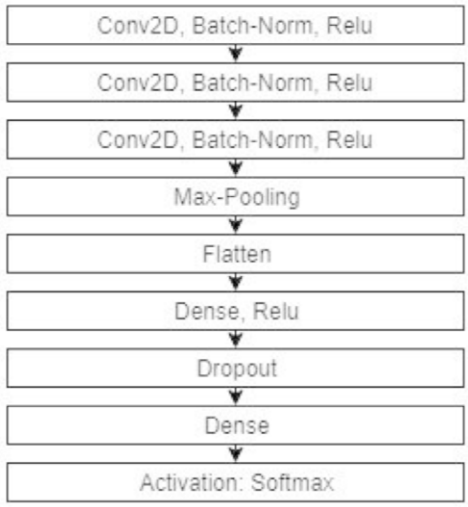

Figure 2—Network architecture: First three blocks consist of 2D convolutional layers followed by batch normalization and ReLU activation function. Max pooling layer down-samples the input representation and dropout layer minimizes overfitting.

Prediction

After the model has been trained, the model is employed to track horizons on all images excluding the ones that were used for training. The model produces a probability indicating the likelihood of the image containing a horizon. Since the classification is binary, the predicted class is the one with a probability greater than 0.5.

To track horizon(s) on a seismic slice, we split the slice into multiple square tiles as shown in Fig. 3. Each tile is passed through the CNN model and predicted as either belonging to the positive class (i.e., containing a horizon which is marked in green) or negative class (i.e. not containing a horizon and marked in red). These predicted tiles are then stitched together to create the final tracked seismic image. For each prediction, we estimate the quality of the classification by using standard metrics such as accuracy, F-score, and visual evaluation on the test data set. The scores can either be calculated for the classes on the total number of tiles of that particular class across all the test images, or can be calculated for the classes in each image, and then be averaged across all the images. We evaluate our model by using the latter methodology to better understand the classification results of our CNN model. For visualization, a random image is stitched back together using the constituent sub-images.

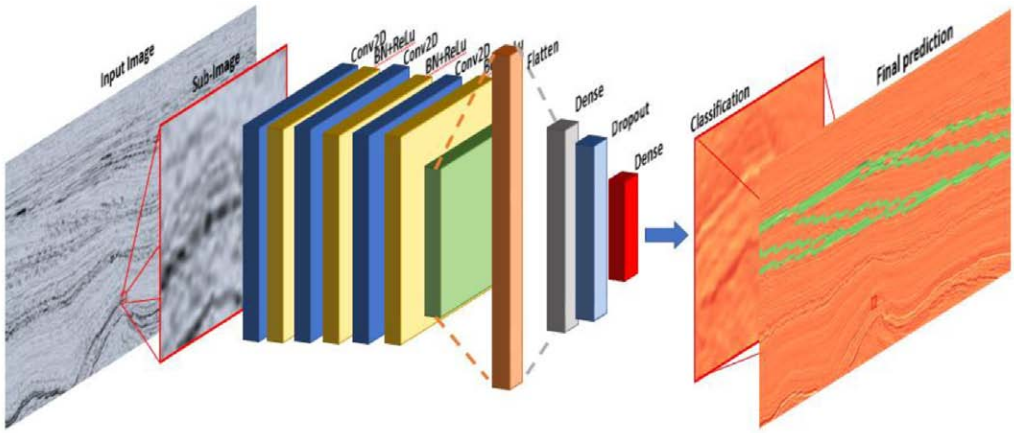

Figure 3—Test image is split into sub-images. For each square sub-image, the model classifies it as positive (green) or negative (red) class. Finally, all the predicted sub-images are stitched together resulting in the final tracked horizons in the original image.

Evaluation Metric

Precision (P) is the fraction of predicted horizon sub-images that actually belong to the horizon class. Recall (R) is the fraction of actual (ground truth) horizon class sub-images that were correctly predicted. The harmonic mean of precision and recall is called F-score.

We define true positives (TP) as those samples or sub-images which belong to the positive class (or horizon class) as per the ground truth labels for the data set and predicted as belonging to the positive class by the model. False positives (FP) are the sub-images which belong to the negative (or non-horizon) class but were predicted as belonging to the positive class by the model. False negatives (FN) are those sub-images which were from the positive (or horizon) class but were predicted as belonging to the negative (or non-horizon) class.

Mathematically, precision (P), recall (R) and F-score (F) can be expressed as:

$$P = \frac{TP}{TP+FP} \qquad R = \frac{TP}{TP+FN} \qquad F = \frac{2PR}{P+R}$$

In order to better understand these evaluation metrics, consider the example shown in Table 1. Assume that the test data set contains a total of n = 165 samples for which the model needs to make a class prediction. As per the ground truth class labels for this data set, 105 samples belong to the positive class and 60 samples belong to the negative class. After making a prediction, 100 of the positive class samples were predicted correctly, hence TP = 100. On the other hand, 10 of the negative class samples were incorrectly predicted to belong to the positive class, hence FP = 10. Also, 5 positive class samples were predicted to belong to the negative class hence FN = 5, and 50 negative class samples were correctly predicted to belong to the negative class.

Table 1—Example data set with the corresponding evaluation metrics.

***n* = 165**	**Predicted class: NEGATIVE**	**Predicted class: POSITIVE**	
Actual class: NEGATIVE	*TN* = 50	*FP* = 10	60
Actual class: POSITIVE	*FN* = 5	*TP* = 100	105
	55	110	

Results

In our experiments, we first trained our model on one set of parameter values (data set, sub-image size, number of training images, sampling technique, data arrangement). We fine-tuned the hyperparameters using these set of parameters, and then carried out further study on the other parameters.

Fig. 4 shows the results on a seismic image taken from the F3 data set. Red colored blocks represent the horizon class sub-images and blue colored blocks represent the non-horizon class sub-images. A close similarity between the ground truth horizons and the predicted horizons is observed.

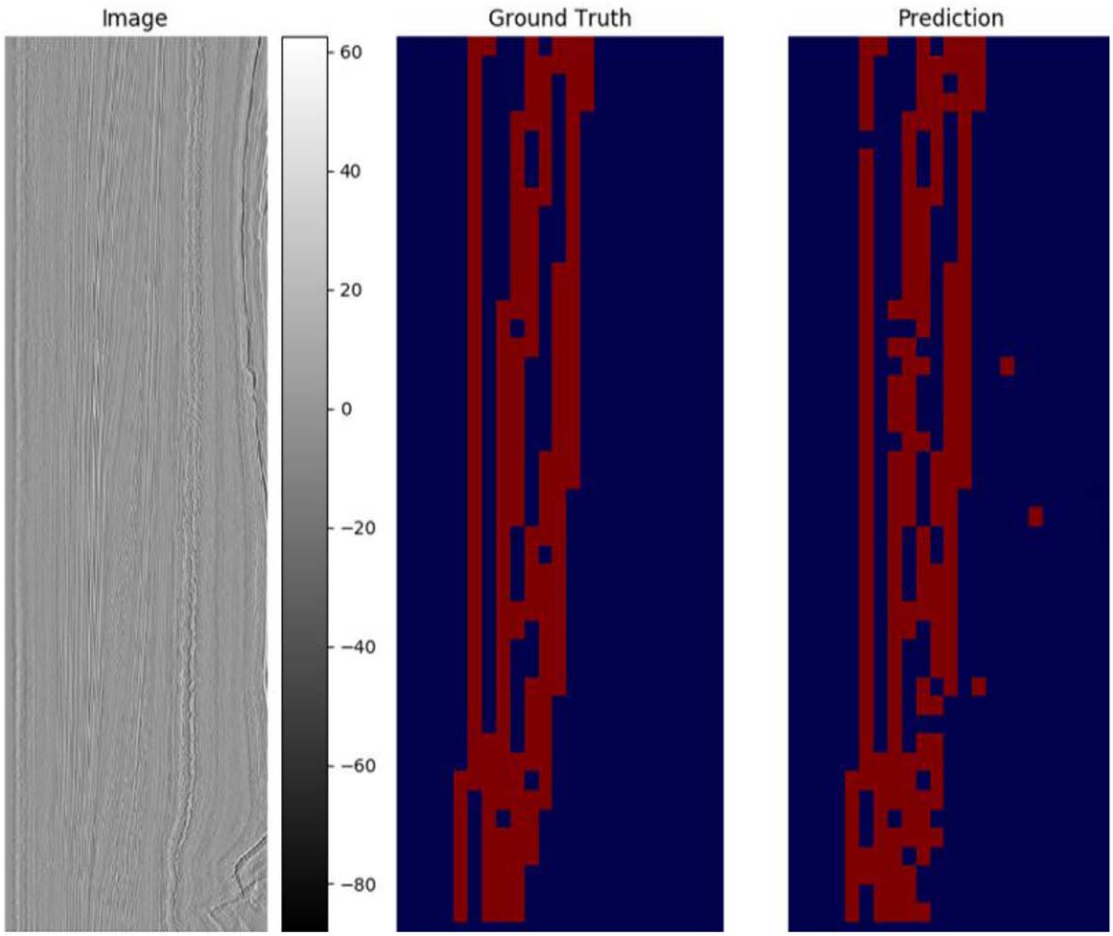

Figure 4—Results on a sample seismic image from the F3 data set.

Table 2 shows the classification results for horizon identification on the F3 dataset. The results are obtained by training on 10 % of the total images (8 % training, 2 % validation), on shuffled data by over-sampling the positive class. We observe that the F-score of the predictions steadily improves with an increase in the sub-image size. Increasing the sub-image size also aids the model by mitigating the class imbalance expressed in terms of the initial imbalance ratio. We define the initial imbalance ratio as the ratio of the number of unique negative class sub-images to the number of unique positive class sub-images. However, this comes at the cost of the prediction being coarser and not being able to localize the horizon within a small band around it, which is one of the primary objectives of the trained model.

Table 2—Precision, recall, and F-score values for different sub-image sizes.

Sub-image Size (n x n)	Unique Positive Sub-images	Unique Negative Sub-images	Initial Imbalance Ratio	Final Positive Sub-Images	Final Negative Sub-Images	Total Number of Sub-Images	Precision	Recall	F-score
8 × 8	32828	404362	12.31	404362	404362	808724	0.37	0.65	0.46
12 × 12	20882	174248	8.34	174248	174248	348496	0.68	0.71	0.69
16 × 16	14575	92805	6.36	92805	92805	185610	0.83	0.82	0.81
20 × 20	11117	59148	5.32	59148	59148	118296	0.88	0.89	0.88
24 × 24	8862	39303	4.43	39303	39303	78606	0.89	0.92	0.90
28 × 28	7317	27003	3.69	27003	27003	54006	0.92	0.92	0.92
32 × 32	6051	20339	3.36	20339	20339	40678	0.92	0.94	0.93

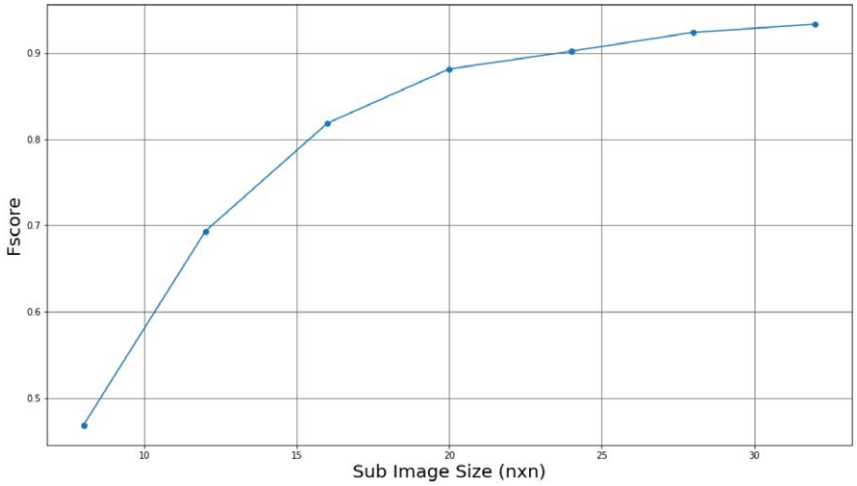

Figure 5—F-score as a function of the sub-image size.

In Fig. 4, we plot the F-score as a function of the sub-image size. The value on the x-axis corresponds to one dimension of the square sub-image or tile. The experiments were done using 10 % of the total data set for training and validation. The horizon class is over-sampled, and the data cube is shuffled. We observe that until a sub-image size of 20 × 20 pixels, the F-score improves significantly and beyond this sub-image size the F-score value appears to saturate. Thus, for further experiments and the results shown next, we use a sub-image size of 20 × 20 pixels.

In practice, random shuffling can result in an ordering of images such that some of the images can be closer to each other than the expected average distance between sampled images. This leads to sub optimal amount of information passed to the model. To tackle this problem, we deterministically shuffle the images, such that the training images have a fixed average distance between them. This ensures that information given to the model is uniform across the cube. However, that deterministically shuffling does not guarantee a set which gives the optimal solution. It simply helps us reach closer to it.

In Table 3 we present the benchmark metrics of our CNN model using different methodologies and experiments carried out using the F3 data set. In each experiment, 10 % of the total data has been used for training and validation. Our model achieves the best F-score of 0.914 by over-sampling the positive class sub-images and choosing training images which are a fixed number of steps apart over the entire data set.

Table 3—Precision, recall, and F-score values under different parameter settings.

Sampling	Data Arrangement	Precision	Recall	F-score
Normal	Ordered	0.647	0.265	0.370
Normal	Random Shuffle	0.903	0.817	0.851
Normal	Deterministic Shuffle	0.942	0.840	0.882
Weighted	Ordered	0.643	0.358	0.454
Weighted	Random Shuffle	0.912	0.822	0.857
Weighted	Deterministic Shuffle	0.948	0.866	0.901
Under-Sampling	Ordered	0.474	0.601	0.524
Under-Sampling	Random Shuffle	0.571	0.935	0.704
Under-Sampling	Deterministic Shuffle	0.451	0.977	0.609
Over-Sampling	Ordered	0.670	0.408	0.499
Over-Sampling	Random Shuffle	0.882	0.893	0.882
Over-Sampling	Deterministic Shuffle	0.917	0.918	0.914

Based on our experiments with different data arrangement methods, it is noted that the deterministic shuffling of the training images yields the best possible results. This is expected as it ensures that images from across the entire data cube are uniformly chosen for model training. The only exception is the case when under-sampling of the negative class sub-images was employed. In an ideal scenario, fully random and choosing sub-images with a fixed step should lead to similar results. However, in practice it depends on the nature or state of the data across the cube.

Of the different sampling strategies that were employed to address the class imbalance, it is found that over-sampling the positive class sub-images leads to the best results. This is expected as over-sampling the positive class sub-images effectively increases the amount of data used for training, whereas under-sampling the negative class sub-images results in an effective reduction in the amount of training data. The only exception to this is the ordered case where under-sampling the negative class sub-images gives the best results. We do not observe any clear pattern as far as the precision and recall values are concerned, but the tradeoffs discussed above allow us to achieve the best F-score in almost every case. In Fig. 6 and Fig. 7, we show the relationship between F-score and the number of training images. In Fig. 6, we consider the case when the positive class sub-images are over-sampled, and the F-score values are plotted for different data arrangement techniques. In Fig. 7, we consider the deterministic shuffling of the data cube and plot the F-score values for the different data sampling strategies.

Figure 6—F-score as a function of the training data set size for different data arrangement techniques when the positive class sub-images are over-sampled.

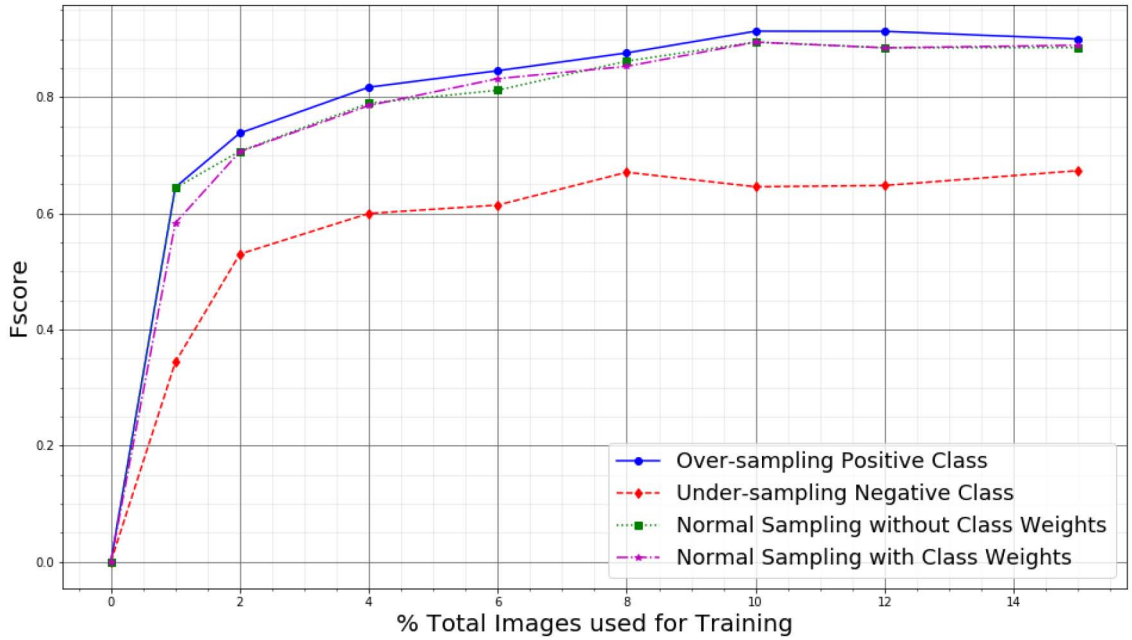

Figure 7—F-score as a function of the training data set size for different data sampling strategies when using deterministic shuffling of the training data.

From Fig. 6, we confirm our earlier expectation regarding the effect of the different data arrangement methods on the classification performance. Keeping the data cube intact would make it easier for the geologist to label the data, however it leads to poor F-score values. On the other hand, shuffling the data cube would make the geologist's job more difficult, but helps in significantly improving the F-score and thus the classification performance.

Another observation that can be made from Fig. 6 is the relatively smooth nature of the plots in the case of shuffling the data cube as opposed to the "rocky" or "jolty" nature of the plot when keeping the data cube intact. When the data cube is kept intact, the training occurs locally in the volume. As the training data set size increases, the model learns more about the various geological and sediment patterns in the data. This results in the model weights changing abruptly and so does the nature of prediction or the classification performance. This is in contrast to the cases when the data cube is shuffled, and the model learns about features from across the data cube, leading to a gradual updating of the model weights with increasing training data set size.

From Fig. 7, it is observed that under-sampling the negative class sub-images leads to significantly poorer results than the other three data sampling techniques. This is most likely due to the fact that the positive class sub-images in the data set are already very few in number and as a result of this the sampled negative class sub-images will also be very few. This leads to a general reduction in the total amount of training data that is input to the model for a fixed number of training images, leading to poor F-score values.

It is also observed that using normal sampling of the data - with and without class weights - leads to similar results for different number of images used for training the models. Over-sampling the positive class sub-images leads to slightly better results than when using normal sampling. This can be attributed to the extra, albeit non-unique and previously seen, training data being input to the model for a given training data set size. The results presented here primarily refer to the experiments carried out with the F3 data set; however similar patterns have been observed with the Penobscot data set as well.

Conclusions

We have presented a method for end-to-end automatic horizon tracking in seismic images based on a deep convolutional neural network. The CNN model learns relevant information and features from training data,

and results in high classification accuracy. We have presented results from the proposed model on the Netherlands F3 and Penobscot data sets. With a limited amount of manually labelled training samples, the model successfully tracks horizons on all the other seismic slices, in both inline and crossline directions. The horizon tracking results were compared with ground truth and quantified using precision, recall, and F-score values, which are conventional metrics for evaluating the classifier performance. With the capability of emulating human expertise and improving the model through training as new labeled images become available, CNN has the potential for automating the key task of horizon tracking in seismic images.

Acknowledgements

This work was supported by the Schlumberger India Technology Centre Pvt. Ltd. (research grant no. FT/05/255/2018). The authors thank IIT Delhi HPC facility for the computational resources. The F3 and Penobscot data sets used in this work are provided by dGB Earth Sciences. We would also like to acknowledge the use of OpendTect software in our work.

References

[1] Ortigosa, F., et al (2008). Benchmarking 3D RTM on HPC platforms. *SEG Technical Program Expanded Abstracts*, pp. 2879-2883.

[2] Rastogi, R. (2011). High-Performance Computing in Seismic Data Processing: Promises and Challenges, *HPC Advisory Council Switzerland Workshop*.

[3] LeCun Y. et al (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, **1**(4), 541-551.

[4] Choromanska, A., LeCun, Y., and Arous, G. B. (2015). Open problem: The landscape of the loss surfaces of multilayer networks. *In 28th Annual Conference on Learning Theory* (pp. 1756-1760).

[5] Zhao, T., Jayaram, V., Roy, A., & Marfurt, K. J. (2015). *A comparison of classification techniques for seismic facies recognition. Interpretation*, **3**(4), pp. SAE29-SAE58.

[6] Howard, R. E. (1991). *U.S. Patent No. 5,056,066. Washington, DC: U.S. Patent and Trademark Office*.

[7] Wu, X., and Hale, D. (2015). *Horizon volumes with interpreted constraints*. *Geophysics*, **80**(2), pp. IM21-IM33.

[8] Zeng, H., Backus, M. M., Barrow, K. T., and Tyler, N. (1998). Stratal slicing, Part I: Realistic 3-D seismic model. *Geophysics*, **63**(2), pp. 502–513.

[9] Zeng, H., Henry, S. C., and Riola, J. P. (1998). Stratal slicing, Part II: Real 3-D seismic data. *Geophysics*, **63**(2), pp. 514–522.

[10] Lomask, J. (2006). Seismic volumetric flattening and segmentation (Doctoral dissertation). *Stanford University, USA*.

[11] Stark, T. J. (2005). Unwrapping instantaneous phase to generate a relative geologic time volume. *SEG Technical Program Expanded Abstracts*, pp. 1707–1710.

[12] Xiong, W. et al (2018). Seismic fault detection with convolutional neural network. *Geophysics*, **83**(5), pp. O97-O103.

[13] Maniar, H., Ryali, S., Kulkarni, M. S., and Abubakar, A. (2018). Machine learning methods in geoscience. *SEG Technical Program Expanded Abstracts*, pp. 4638-4642.

[14] Chen, J., and Y. Zeng, 2018, *Application of machine learning in rock facies classification with physics-motivated feature augmentation. arXiv e-prints*, pp. arXiv:1808.09856.

[15] Waldeland, A. U., Jensen, A. C., Gelius, L.-J., and Solberg, A. H. S. (2018). Convolutional neural networks for automated seismic interpretation. *The Leading Edge*, **37**(7), pp. 529-537.

[16] *dGB Earth Sciences - Software*. Retrieved from https://www.dgbes.com/index.php/software [last accessed 07-June-2019].

[17] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, **86**(11), pp. 2278-2324.

[18] *TerraNubis - Open Seismic Repository information*. Retrieved from https://terranubis.com/osr [last accessed 07-June-2019].

[19] Kingma, D. P., and Ba, J. L. (2015). Adam: A method for stochastic optimization. *In International Conference on Learning Representations*.

[20] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pp. 807-814.

[21] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, **15**(Jun), pp. 1929–1958.