A scalable deep learning platform for identifying geologic features from seismic attributes

Lei Huang[1], Xishuang Dong[2], and T. Edward Clee[3]

Abstract

The modern requirement for analyzing and interpreting ever-larger volumes of seismic data to identify prospective hydrocarbon prospects within stringent time deadlines represents an ongoing challenge in petroleum exploration. To provide a computer-based aid in addressing this challenge, we have developed a "big data" platform to facilitate the work of geophysicists in interpreting and analyzing large volumes of seismic data with scalable performance. We have constructed this platform on a modern distributed-memory infrastructure, providing a customized seismic analytics software development toolkit, and a Web-based graphical workflow interface along with a remote 3D visualization capability. These support the management of seismic data volumes, attributes processing, seismic analytics model development, workflow execution, and 3D volume visualization on a scalable, distributed computing platform. Early experiences show that computationally demanding deep learning methods such as convolutional neural networks (CNN) provide improved results over traditional methods such as support vector machines (SVMs) and logistic regression for identifying geologic faults in 3D seismic volumes. Our experiments show encouraging accuracy in identifying faults by combining CNN and traditional machine learning models with a variety of seismic attributes, and the platform is able to deliver scalable performance.

Introduction

The modern requirement for analyzing and interpreting ever-larger volumes of seismic data to identify hydrocarbon prospects within stringent time deadlines represents an ongoing challenge in petroleum exploration. The current solutions in the petroleum industry are mostly based on commercial application systems that provide advanced interpretation and visualization capability; however, they are all desktop-based solutions. With the ever-increasing size of seismic data volumes and the number and variety of volume attributes, the industry demands a new platform that manages, accesses, queries, processes, analyzes, and visualizes them with scalable performance. Big data analytics platforms such as Apache Hadoop and Spark provide scalable performance on analyzing big data sets with high-level programming interfaces, which have thrived in many industrial domains, such as social networks, Internet of things, retail applications, security, advertising, and more. These big data analytics platforms bring a new powerful toolkit to data and domain scientists to extract insights and uncover hidden patterns from large volumes of data sets, which directly create additional revenue and/or reduce costs for these industries. Besides, on these platforms, machine learning techniques could be employed to construct identification models with high performance. The advances in machine learning techniques (Bishop, 2006; Abu-Mustafa et al., 2012), especially deep learning technologies, dramatically increase the accuracy of data analysis and prediction and provide new approaches to solving problems from different perspectives.

In this paper, we present our design of a scalable seismic analytics platform built upon Apache Spark and its functionality of managing, processing, analyzing, and visualizing seismic data. We also introduce the deep learning technology we employed in our experiments using the platform and the methodology for identifying geologic faults from seismic attributes.

Seismic analytics platform

We introduce the seismic analytics platform in three parts, namely, seismic software development kit (SDK), workflow, and visualization. These are combined with additional data analytics and infrastructure tools in a Spark environment with Hadoop distributed file system.

Seismic analytics software development toolkit. To provide a computer-based aid in addressing the petroleum interpretation challenge, we have developed the seismic data analytics SDK to facilitate the work of geophysicists and data scientists in analyzing and processing seismic volumes. As illustrated in Figure 1, the SDK is able to load seismic data from the Hadoop distributed file system (HDFS), partition, and aggregate data with or without overlapped areas according to a user's configuration in the Spark distributed computing system. It gets lines, traces, or samples from the distributed memory per user's request, transposes a volume in any dimensions, and saves the seismic volumes into HDFS. It also applies the user's functions to the entire seismic

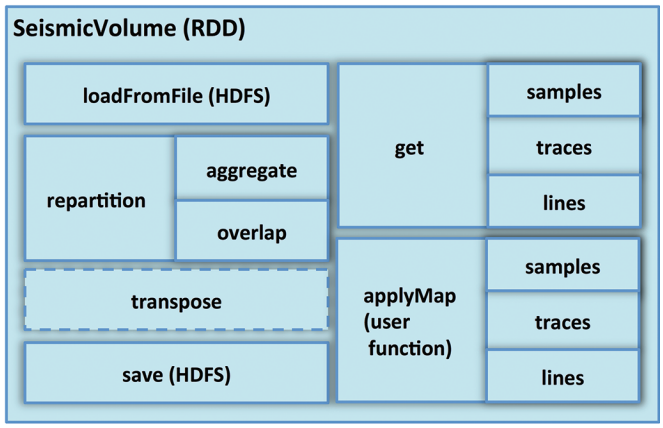

Figure 1. The seismic analytics SDK offers distributed operations on seismic volumes.

[1]Prairie View A&M University, Department of Computer Science.
[2]Prairie View A&M University, Department of Electrical and Computer Engineering.
[3]TEC Applications Analysis.

http://dx.doi.org/10.1190/tle36030249.1.

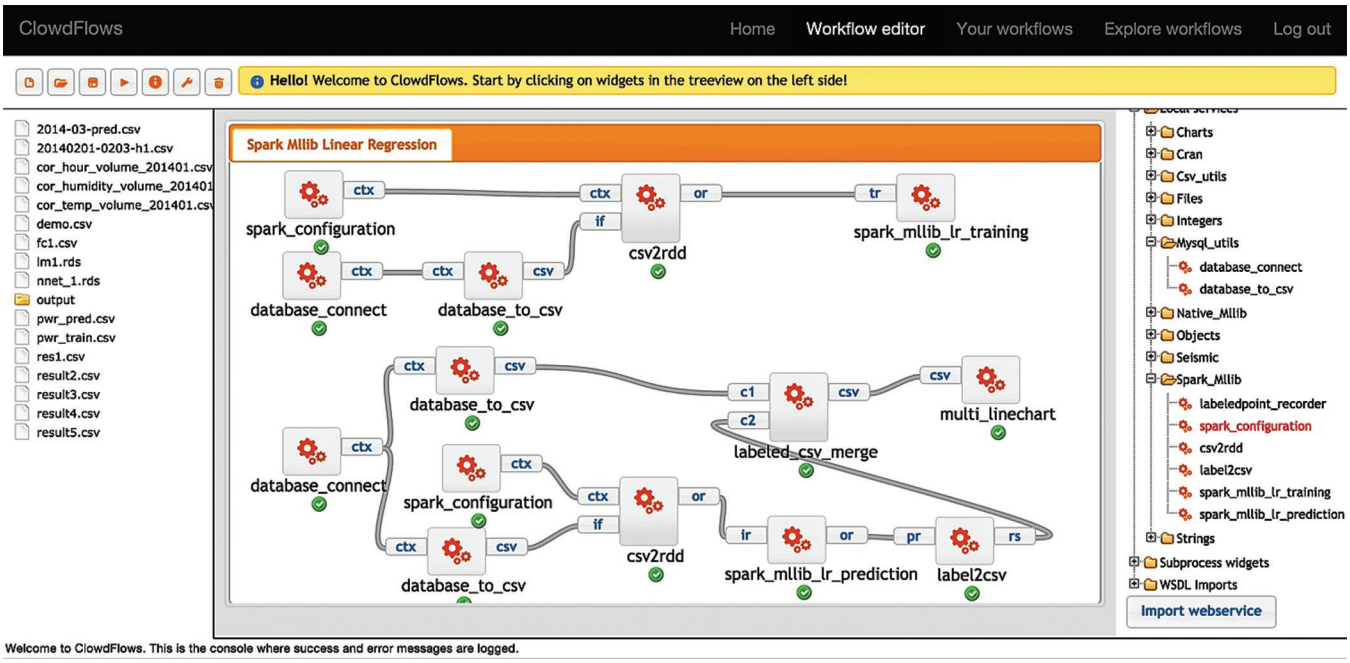

Figure 2. A seismic analytics workflow constructed in the Clowdflows platform.

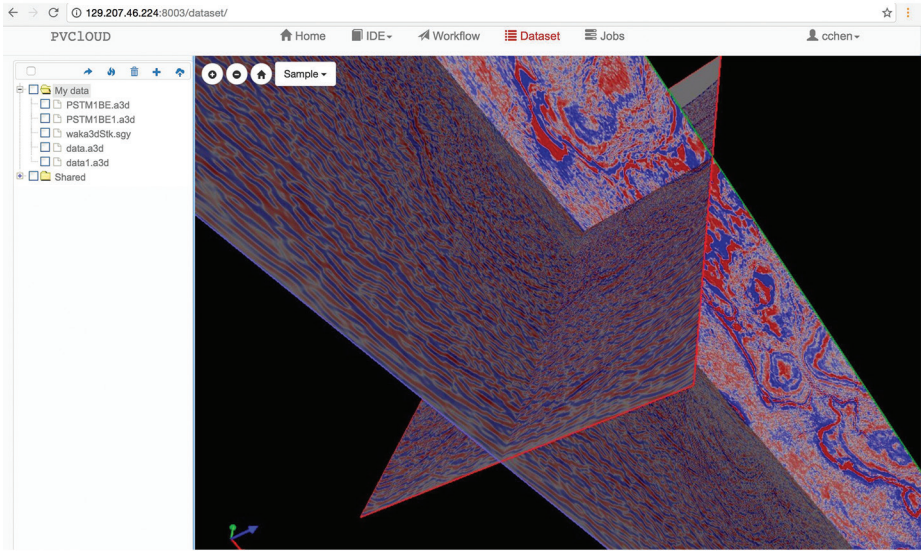

Figure 3. Web-based visualization of a 25 gigabyte seismic volume residing in memory on a Spark-based cluster.

volume in parallel, according to the user's configuration and parallel templates defined in our SDK.

Workflow. We have developed a web-based workflow interface based on the open-source ClowdFlows platform from ClowdFlows.org that constructs and executes data analytics workflows on Spark. Each work step in a workflow is a separate job submitted to the Spark platform and scheduled to run in the distributed system. Users can implement their own data analytics algorithms by using the seismic data analytics SDK and wrap it as a single work step. All work steps are connected via the inputs and outputs of each step, and most intermediate results are stored as Spark resilient distributed data sets (RDDs) that reside in memory without writing and reading to and from the file system, which increases performance significantly. Figure 2 shows a simple data analytics workflow for training a model to identify geologic faults.

Visualization. By collaboration with FEI Visualization Sciences Group, we have developed remote 3D visualization functionality in the platform to view seismic volumes using a variety of supported web browsers. We have implemented a data server that responds to the user's requests in viewing data in multiple directions. Based on the user's data requests, the data server gets seismic slices from the Spark RDDs using the SDK, and FEI's Open Inventor render server creates images that are pushed to the web browser for display. We are able to visualize large volumes of seismic data remotely in a web browser within a few seconds. Figure 3 is a screenshot from a demonstration of the 3D visualization capability on a web browser.

Deep learning methodology

In recent decades, the development of deep learning technology has significantly improved the performance of forecasting in many fields (Donahue et al., 2015). It is developed from the artificial neural networks drawn heavily on the knowledge of the human brain, statistics, and applied mathematics. Deep learning, as a branch of machine learning algorithms, attempts to model high-level data representation by using multiple layers of neurons and multiple nonlinear transformations (LeCun et al., 2015). Recently, with the increment of layers in the deep learning models, it has become the most popular and powerful when it is built on big data. Besides, due to more powerful computers and larger

data sets, training deeper networks becomes faster and easier. The deeper the model's layer of network is, the stronger the model's ability of representing raw data becomes, resulting in better model performance.

Additionally, deep learning allows the computer to build complex concepts out of simpler concepts by constructing deeper neural networks. A variety of deep learning architectures have been developed, such as deep neural networks, convolutional deep neural networks, deep belief networks, and recurrent neural networks with deep feature layers. These architectures assume that these layers of features corresponding to levels of abstracted representation of inputs can enhance the forecasting effectiveness. These deep learning algorithms with different architectures have been employed to address challenges in computer vision, automatic speech recognition, natural language processing, audio recognition, and bioinformatics, producing state-of-the-art results.

Implementing applications based on deep learning depends significantly on developments of software libraries designed for this purpose. Among many others, TensorFlow and Caffe are two most typical and practical of such deep learning packages. TensorFlow is an open-source software library provided by Google. It is used for numerical computation using data flow graphs, in which nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. Our software environment also includes Caffe, a deep learning framework developed mainly by the Berkeley Vision and Learning Center (BVLC).

Convolutional neural networks (CNN) are a type of feed-forward artificial neural network in which the network structure is formed among artificial neurons inspired by the organization of human neurons (LeCun et al., 2015), which has been applied to solve problems such as image and video recognition (Krizhevsky et al., 2012; Karpathy et al., 2014), recommender systems (van den Oord et al., 2013), and natural language processing (Mikolov et al., 2013). The CNN architecture is formed by a stack of distinct hidden layers that transform the input data into an output volume, as shown in Figure 4. Each convolution layer consists of a set of learnable filters, which have a small receptive field but which extend through the depth of the input volume. During the forward feedback of parameters, each filter is convolved across the input volume by computing the dot product between the filter and the input of former layers. In addition, it is common to periodically insert a pooling layer in between successive convolution layers. The function of the pooling layers is to progressively reduce the spatial size of the representation for decreasing the amount of parameters and computation with pooling options. At the end of the structure of the CNN model, in the fully connected layer, artificial neurons are fully connected to all outputs in the previous layer. CNN's powerful ability of representing data enables it to extract valuable information from data as a new data representation allowing the construction of high-performance models.

In this paper, we report experimental results for applying the CNN to represent seismic data in identifying geologic faults in seismic data volumes with traditional machine learning models. We choose the Google TensorFlow application as the deep learning library and integrate it as a work step in our seismic analytics workflow to train a model of faults identification. The model is later applied to represent seismic data for identifying faults in a new seismic volume.

Use case: Geologic faults identification

Our initial use case is the identification of geologic faults by applying deep learning technology on the seismic data analytics platform. The objective is illustrated in Figure 5, which shows the time-migrated stack data (Figure 5a) and faulting (Figure 5b) estimated by image processing techniques (Hale, 2013). The data are from a West Cameron Gulf of Mexico data set made public by Geco-Prakla (Schlumberger-Geoquest). We seek to employ suitable machine learning models for fault identification using known fault locations in a suite of attributes derived from a seismic data volume. We then apply the model to other data sets to quickly identify similar fault structures.

The workflow contains the following four steps:

1) calculate seismic attributes;
2) extract features;
3) train CNN models; and
4) predict geologic faults by applying these CNN models.

We describe these four steps in the following sections respectively.

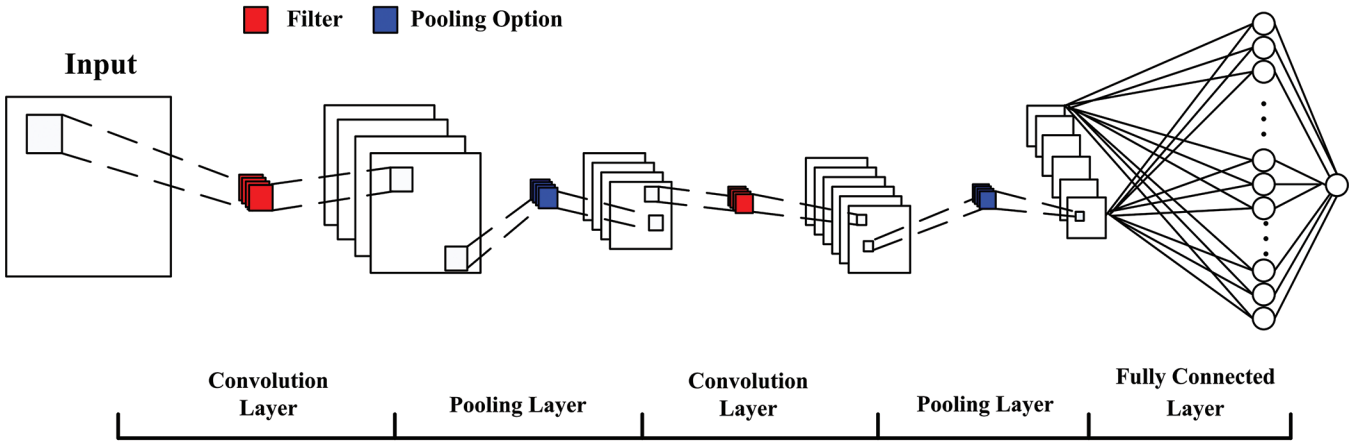

Figure 4. Convolutional neural network architecture.

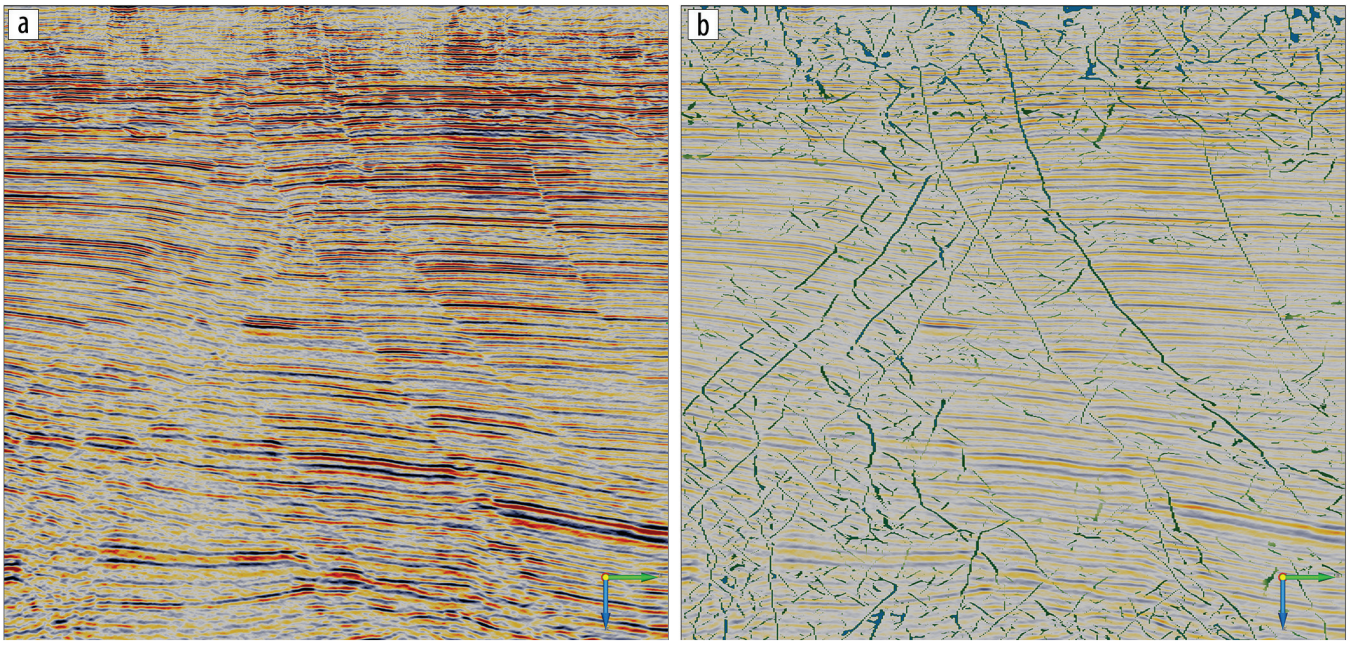

Figure 5. Center inline from the West Cameron data showing clear evidence of faulting: (a) time-migrated stack data and (b) with superimposed fault likelihood attribute.

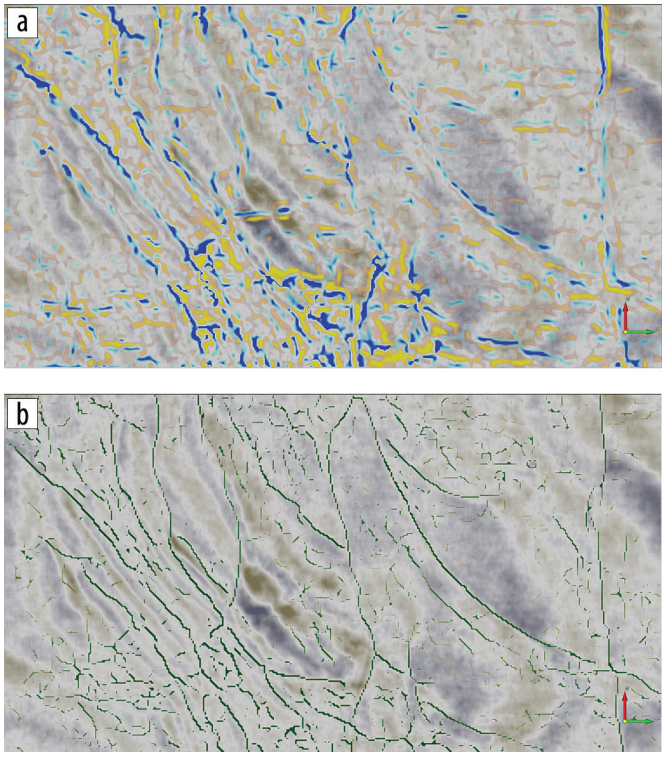

Figure 6. Typical seismic attributes used as inputs for machine learning, displayed for a time slice through the middle of the West Cameron data. Large values of (a) maximum curvature (yellow and bright blue), and (b) fault likelihood (green) are overlaid on seismic amplitude.

Seismic attributes for fault identification. Many authors have described advanced techniques and methods for using seismic volume attributes to interpret faults, such as Hale (2013) and Machado et al. (2016) and many others cited in these two papers. It is not our purpose here to present innovative attribute constructions to obtain the very best possible fault identifications, but rather we seek to demonstrate the use of our scalable workflow to apply machine learning techniques in the solution of an analytics problem of interest to the exploration community. For this purpose, we choose a suite of attributes computed from a 3D poststack time-migrated volume, designed to provide a variety of independent representations of the imaged region that may help to guide the machine learning process. We choose attributes based on the original amplitude data, the structural slopes of the amplitude data, and properties of the semblance-based fault likelihood measure as described by Hale (2013), as well as curvature-based attributes described by Roberts (2001). We select for use in our analysis a subset including nine of these attributes: log-scaled stack trace values, sine of local dip, eigenvalue-based planarity, maximum curvature, dip curvature, curvature shape index, semblance-based fault likelihood, sine of fault likelihood dip angle, and cosine of fault likelihood azimuth. Two of these attributes are illustrated in Figure 6 for the West Cameron data set.

These entire attribute data sets are computed from input seismic volumes using our seismic data analytics SDK. The SDK loads seismic volumes with the SEGY file format, and creates a resilient distributed data set (RDD) that is a global data abstract on Apache Spark to enable in-memory computing. The computation of these attributes follows the predefined parallel templates in the SDK, which are either one-to-one or m-to-n with optional overlapping in data distribution. We implemented the kernel functions for these attributes and chose the right templates with overlapping parameters. The SDK applies the kernel function to each entire seismic volume in parallel on the Spark platform to compute these attributes in scalable fashion. By using the SDK, a geophysicist does not need to handle the details of data distribution, parallelism, and data communication. Instead, he or she can focus on the kernel function that computes the attribute for a single trace, line, or subvolume, and the SDK is able to apply the user kernel function to the entire volume without programming effort by the user.

Feature extraction. The fault-detection model was trained using our suite of nine attributes computed from a synthetic volume of more than 2.2 million sample points constructed using a simple seismic volume generation program provided with the public domain IPF (image processing for faults) software provided by Hale (2014). This training data set shown in Figure 7 has five faults generated in two episodes of faulting. The faults illustrated (and the complementary zero-value nonfault data points) form the label data set for training the machine learning fault-detection model.

To conduct the model training in the machine-learning workflow, we need to extract meaningful features from these seismic attributes. We extract a 7 × 7 × 7 subvolume for each voxel in these seismic volumes to create a vector with 343 features. These features represent the seismic attributes in a localized area, and we rely on CNN to learn the common characteristics for faults. Since we are using synthetic data in our training, we are able to generate the label volume that represents the geologic faults in the seismic volume. The label volume is combined with the feature vector to indicate if the center point of the 7 × 7 × 7 subvolume is in a geologic fault or not.

Model training. Figure 8 shows the workflow of constructing a geologic fault-identification model using the training features extracted from these seismic attributes. The first step is to configure a CNN model and then employ it to represent seismic data as fault-identification results for these nine training feature data sets extracted from the seismic attribute volumes. In our experiments, we construct CNN models to build the submodels using five layers: (1) a convolution layer with 32 filters; (2) a 2 × 2 max-pooling layer; (3) a convolution layer with 64 filters; (4) a 2 × 2 max-pooling layer; and (5) a fully connected layer with 1024 outputs. We use the Adam Optimizer parameter estimation algorithm provided in the TensorFlow system.

In the image-processing domain, we usually employ multichannel CNN to train models. It is similar to our case because seismic data contain many attributes that can be treated as image channels if seismic data are viewed as images. However, training on that data consumes excessive computational resources and is a lengthy process. Therefore, for reducing computation pressure and time for building models, we design to train models on different attribute data simultaneously, which can be completed in a parallel method. We also adopt logistic regression and support vector machine as classifiers to construct the final fault-identification model.

This training is a lengthy process, and prior to acquisition of our own GPU-enhanced cluster, we have taken advantage of the opportunity to run on a GPU-based cluster at the University of Houston. After the submodel training processing is completed, the next step is to recognize faults and merge all recognition results as a new data representation of training data. In the final step, we train a traditional classifier based on support vector machines (SVM) and logistic regression on the new data representation and construct the final geologic fault-identification model.

After these models are trained completely based on a traditional classifier, we apply them to identify geologic faults on a new seismic synthetic volume. This testing workflow, illustrated in Figure 9, is very similar to that of training the identification model. The same suite of seismic attributes is computed for the testing seismic volume, and then the feature vectors are extracted from these attributes, as in the training process. Then we use the CNN submodels obtained in the training procedure to recognize faults on these subsets and merge the recognition results as a new data representation of seismic testing volume. Finally, we apply the SVM/logistic regression fault-identification model obtained in the training step to identify the final recognized faults for the seismic testing volume on its new data representation.

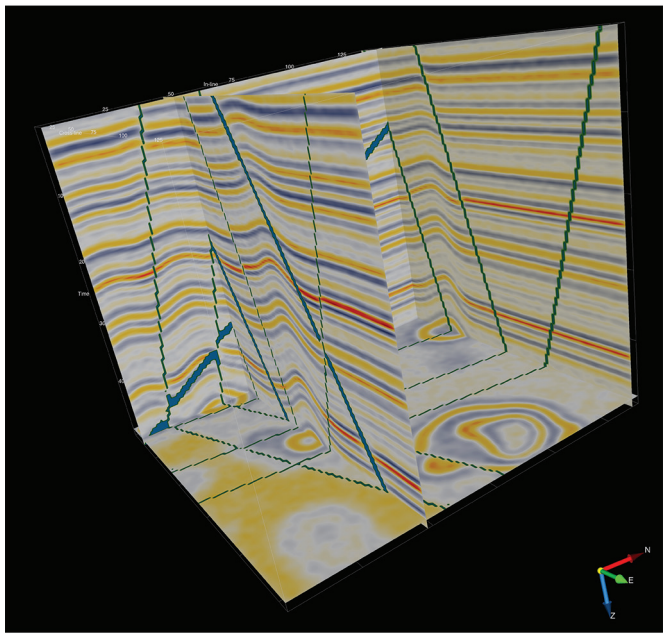

Figure 7. A synthetic seismic volume with five faults used to train the fault-detection model.

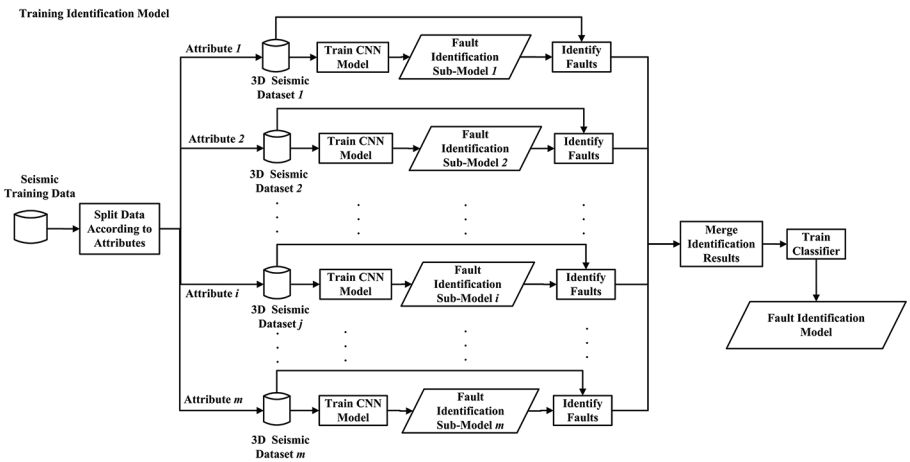

Figure 8. Training the fault-identification model on a suite of seismic attributes data volumes.

Geologic faults identification results. We applied the trained model for prediction of faults in another synthetic data set constructed to contain three faults. Since both training and testing (prediction) volumes are synthetic data, we are able to evaluate the fault-identification model accurately. Figure 10 shows the testing model both with the known (Figure 10a) and the predicted faults (Figure 10b).

We next discuss the statistical measures used to evaluate the success of the machine learning process.

Evaluation Metrics

The machine learning statistical evaluation metrics we chose are: precision, recall, and their combination $F_{measure}$, which are recognized as adequate methods to evaluate results for classification experiments (Bai et al., 2017).

These metrics focus on evaluating prediction performance of the positive samples, although they also capture information about the rates and kinds of errors made (Powers, 2011). For the application of geologic fault identification, the positive samples indicate faults and their locations in a seismic volume. The prediction possibilities are shown in Table 1.

The precision can be defined as the ratio between the number of true positives and the summation of true positives and false positives, while recall is defined as the ratio between true positives and the summation of true positives and false negatives. Mathematically, precision P and recall R are defined in terms of the tabulated prediction counts (TP, TN, FP, FN) as

$$P = \frac{TP}{TP + FP} \tag{1}$$

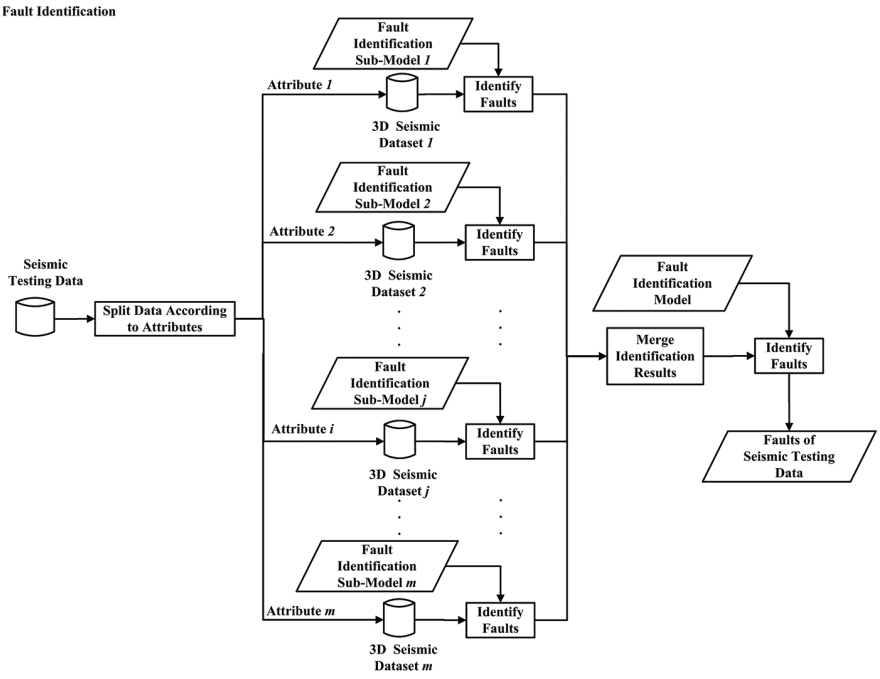

Figure 9. Identifying faults on another seismic testing data set, using the same suite of attributes as for the training step.

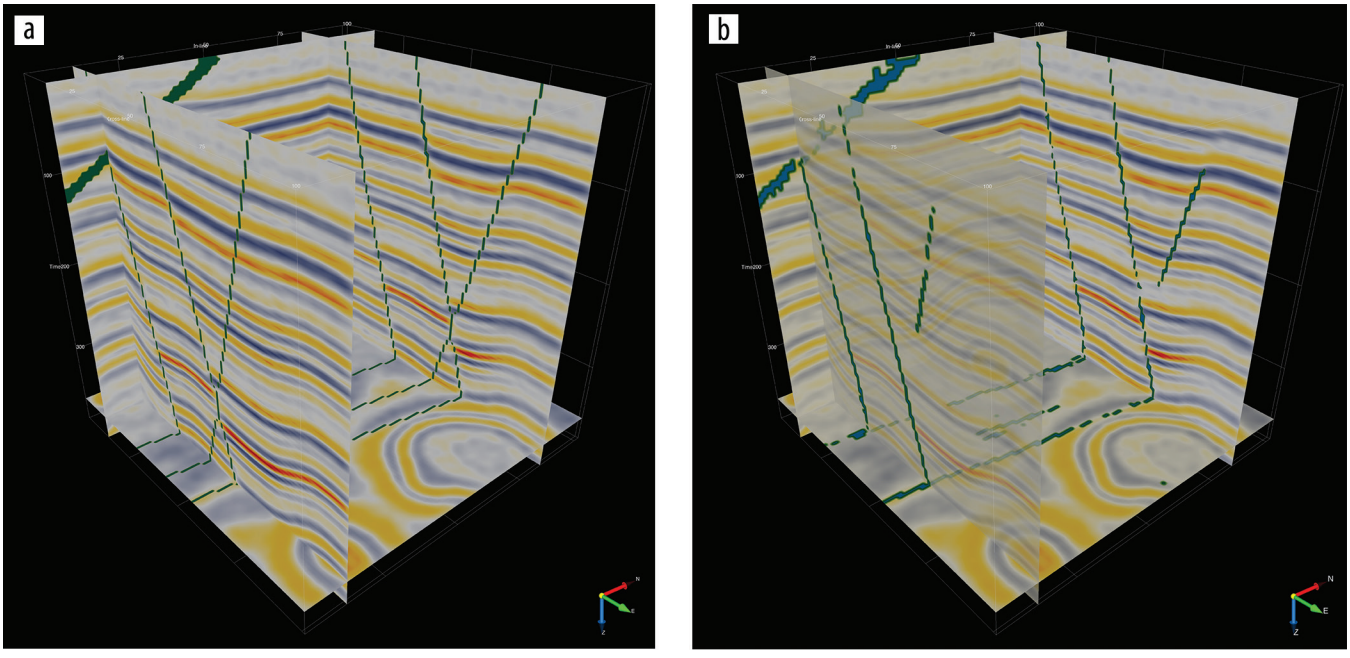

Figure 10. A synthetic seismic volume with three faults showing (a) labeled faults and (b) faults predicted by the CNN model trained from the five-fault synthetic data set. The left-hand inline in the prediction display has some transparency applied, revealing fault predictions on the *z*-slice at bottom.

Table 1. Fault prediction success possibilities.

	Target is "Fault"	Target is "Not Fault"
Prediction says "Fault"	True positive (TP)	False positive (FP)
Prediction says "Not Fault"	False negative (FN)	True negative (TN)

Table 2. Fault prediction performance measures for the seismic attribute suite.

Attribute Name	Precision	Recall	$F_{measure}$	Accuracy
Log-scaled stack	0.6756	0.6324	0.6533	0.8027
Local dip	0.7057	0.4857	0.5754	0.7199
Planarity	0.7163	0.7343	0.7252	0.8540
Maximum curvature	0.5203	0.6212	0.5663	0.7857
Dip curvature	0.5122	0.6381	0.5683	0.7907
Shape index	0.4244	0.6179	0.5032	0.7746
Fault likelihood	0.8399	0.8342	0.8370	0.9120
Fault dip	0.8178	0.8325	0.8251	0.9068
Fault azimuth	0.3500	0.7501	0.4773	0.7938

Table 3. Performance measures for the consolidated prediction step.

Classifier	Precision	Recall	$F_{measure}$	Accuracy
CNN + logistic regression	0.5849	0.7603	0.6612	0.9827
CNN + support vector machine	0.5774	0.7851	0.6654	0.9847

and

$$R = \frac{TP}{TP + FN}. \quad (2)$$

Furthermore, $F_{measure}$ is defined as

$$F_{measure} = \frac{2PR}{P + R}, \quad (3)$$

and we also express the prediction performance using accuracy A defined as

$$A = \frac{TP}{TP + FP + FN + TN}. \quad (4)$$

Each of these measures ranges from 0 to 1, with 1 representing best performance.

In our experiments, we first construct CNN models on each of the different attributes and evaluate the fault-identification results. The results are shown in Table 2, where we see that different attributes can exhibit different identification performance.

The results tell us that we need to carefully select useful attributes so as to construct identification models with high performance. On the other hand, the method for combining these attributes should be elaborated if we hope to obtain a high-performance model on the combined attributes.

Finally, we complete the prediction process by using logistic regression and SVM to build the final geologic fault-identification models. The experimental results in Table 3 show that these two models have similar performance because the difference between their $F_{measure}$ values is less than 0.01. Each classifier has its own advantages: logistic regression has higher precision; SVM has higher recall. We conclude that the selection of classifier for constructing fault-identification models would not significantly affect the performance of predictions.

Discussion and further work

Having succeeded in making preliminary predictions of faults in synthetic seismic volumes, we note that the dip direction seems to have some influence on the quality of prediction. Our next steps will be to bring dip invariance into the learning process by using a training data set that contains a wider variety of dip azimuths. We will further investigate which seismic attributes contribute better to the final prediction to select the right set of attributes in our model training. Moreover, we will conduct pixel-based model training and prediction to reduce the computation complexity and increase the accuracy.

We also recognize the need to demonstrate machine learning applied to field-recorded data sets. We are investigating whether models trained on synthetic data can be used to predict fault locations in field-recorded data and are also developing procedures to use expert-selected instances of faults and nonfaulted zones in field-recorded data as labeled inputs for training the machine learning model.

Beyond our initial use case of fault detection, we will investigate whether we can use similar techniques to delineate other feature types such as stream channels or salt flanks. One beneficial feature of the deep learning platform is that the network will train the learning filters themselves and decrease the feature dimensions by pooling operations, which, compared to traditional machine learning, dramatically reduces the number of learning parameters to lower the computation complexity. We will be able to spend more effort on research in various seismic attributes that are sensitive to the occurrence of such structural features in seismic volumes.

In addressing scalable performance, we note that the major computational tasks in this work are the feature extraction and the training phase of deep learning. The computation of attributes, feature extraction, and predictions has been implemented on our scalable seismic analytics platform to significantly reduce the execution time. The CNN training phase is currently undertaken on Nvidia GPUs

that reduced the time from multiple days on CPUs to just hours. As part of our ongoing research, we are working on applying distributed deep learning model training in a Spark-based system equipped with GPUs on multiple nodes to further speed up the training phase.

Conclusions

The work presented here has shown a number of useful results. We have implemented a cloud-based seismic data analytics platform that can manage seismic volumes, calculate seismic attributes, conduct feature extraction and selection, and apply machine learning including deep learning models to infer geologic features to facilitate the seismic interpretation process. The platform is equipped with a seismic data analytics SDK with deep learning package, a web-based data management tool and workflow interface to interact with the computing platform, and a remote visualization capability to view large seismic volume data sets via a web browser. Our initial case study of geologic faults identification has demonstrated an example for usage of the seismic analytics platform on which we can build future capabilities of interest to the exploration community. We continue to work on a variety of enhancements to address scalability performance issues, especially for larger volumes of field-recorded data.

Acknowledgments

This work has been supported by a number of grants from the U.S. National Science Foundation (IIP-1518140, IIP-1543214, ACI-1229744, ACI-1531814). We also appreciate the contribution of Dave Hale, who counseled us in the implementation of his public-domain software, and of Schlumberger-Geoquest for their release of the West Cameron data set recorded by Geco-Prakla. We acknowledge dGB Earth Sciences for our use of the OpendTect package for preparing the seismic data displays, and the FEI Visualization Sciences Group for their collaboration on the remote visualization of distributed volumes. Finally, we would like to thank the Prairie View A&M University students and research associates who have provided technical support for this work, especially Yuzhong Yan, Chao Chen, and Darshita Mistry. TLE

Corresponding author: lhuang@pvamu.edu

References

Abu-Mostafa, Y. S., M. Magdon-Ismail, and H. Lin, 2012, Learning from data – A short course: AMLBook.com.

Bai, X., B. Shi, C. Zhang, X. Cai, and L. Qi, 2017, Text/non-text image classification in the wild with convolutional neural networks: Pattern Recognition, Elsevier, available online 11 December 2016, ISSN 0031–3203, http://dx.doi.org/10.1016/j.patcog.2016.12.005.

Bishop, C. M., 2006, Pattern recognition and machine learning: Springer-Verlag New York Inc.

Donahue, J., L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, T. Darrell, and K. Saenko, 2015, Long-term recurrent convolutional networks for visual recognition and description: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2015, 1473–1482, http://dx.doi.org/10.1109/CVPR.2015.7298878.

Hale, D., 2013, Methods to compute fault images, extract fault surfaces, and estimate fault throws from 3D seismic images: Geophysics, **78**, no. 2, O33–O43, http://dx.doi.org/10.1190/geo2012-0331.1.

Hale, D., 2014, Seismic image processing for geologic faults, https://github.com/dhale/ipf, accessed 10 November 2016.

Karpathy, A., G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, 2014, Large-scale video classification with convolutional neural networks: 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1725–1733, http://dx.doi.org/10.1109/cvpr.2014.223.

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, and G. Hinton, 2015, Deep learning: Nature, **521**, no. 7553, 436–444, http://dx.doi.org/10.1038/nature14539.

Machado, G., A. Alali, B. Hutchinson, O. Olorunsola, and K. J. Marfurt, 2016, Display and enhancement of volumetric fault images: Interpretation, **4**, no. 1, SB51–SB61, http://dx.doi.org/10.1190/INT-2015-0104.1.

Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, 2013, Distributed representations of words and phrases and their compositionality; Advances in Neural Information Processing Systems: Curran Associates, Inc., 3111–3119.

Powers, D. M. W., 2011, Evaluation: from precision, recall and F-measure to ROC, informedness, markedness, and correlation: Journal of Machine Learning Technologies, **2**, no. 1, 37–63.

Roberts, A., 2001, Curvature attributes and their application to 3D interpreted horizons: First Break, **19**, no. 2, 85–100, http://dx.doi.org/10.1046/j.0263-5046.2001.00142.x.

van den Oord, A., S. Dieleman, and B. Schrauwen, 2013, Deep content-based music recommendation, Advances in Neural Information Processing Systems: Curran Associates, Inc., 2643–2651.