DeepNet: An Ultrafast Neural Learning Code for Seismic Imaging*

J. Bar-hen, D. Reister, and V. Protopopescu

Center for Engineering Science Advanced Research
Computer Science and Mathematics Division
Oak Ridge National Laboratory
P. 0. Box 2008, Bldg. 6012
Oak Ridge, TN 3783 l-6367

Submitted to: International Joint Conference on Neural Networks, IEEE, July 10 – 16, 1999

*Research supported by the Engineering Research Program of the Office of Basic Energy Sciences, U. S. Department of Energy, under contract No. DE-AC05-96OR22464 with Lockheed Martin Energy Research Corporation.

DeepNet: An Ultrafast Neural Learning Code for Seismic Imaging

J. Barhen, David Reister, and V. Protopopescu

Center for Engineering Science Advanced Research
Oak Ridge National Laboratory
Oak Ridge, TN 37831-6355
barhenj @ornl.gov

Abstract

A feed-forward multilayer neural net is trained to learn the correspondence between seismic data and well logs. The introduction of a *virtual* input layer, connected to the nominal input layer through a special nonlinear transfer function, enables ultrafast (single iteration), near-optimal training of the net using numerical algebraic techniques. A unique computer code, named *DeepNet*, has been developed, that has achieved, in actual field demonstrations, results unattainable to date with industry standard tools.

1. Background and purpose

The ability to accurately predict the location of remaining oil in the neighborhood of existing production wells is of vital economic importance to the petroleum industry. For practical purposes, one typically targets volumes of fluid 10 meters thick and 200 meters in lateral extent at a distance of 200 meters from each well, requiring a resolution accuracy of 5% in terms of the distance from the observation well. Available oilfield information incorporates many datasets with different scales, uncertainties, sample volumes, and relevance. Well logs (e.g., porosity, gamma ray response, and resistivity) provide the most accurate possible sensor-based characterization of the geological formations encountered along the path of a well [1]. On the other hand, low-resolution seismic dam are generally used to conduct large-scale field assessments [2]. The *specific focus* of the research we report in this paper was to develop a methodology that would enable fast and accurate prediction of well pseudo logs from seismic data across an entire oil field.

2. Network Architecture

We consider a generalized multilayer feed-forward neural network. In general, the nodes (neurons) are organized in layers, namely: (i) ***input, (ii)*** (one or several) *hidden*, and (iii) *output*. Each layer l contains N_l nodes. In addition to these traditional layers, *we* introduce a *virtual input* layer between the input layer and the (first) hidden layer. The virtual input layer allows for a specific type of preprocessing of the input vectors. The associated computations between the actual and virtual input layers differ from the usual inter-layer neural network computations.

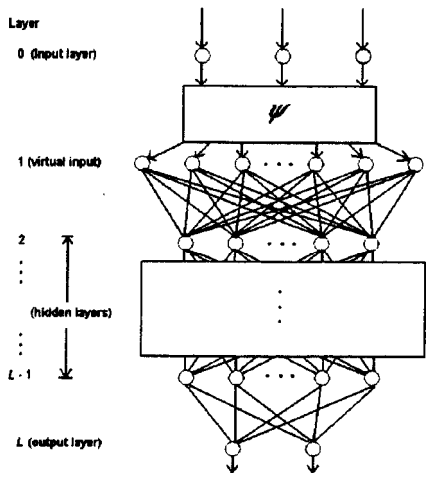

Figure 1: Neural net with virtual input layer

To simplify the description, and without loss of generality, we discuss a network architecture having N_0 = I input nodes, $N_1 = V$ virtual input nodes, one hidden layer with $N_2 = H$ nodes, and $N_3 = 0$ output nodes. The goal in the learning process is to determine the synaptic interconnection matrices $\{\mathbf{W}_{VH}\ ,\ \mathbf{W}_{HO}\}$. This will be achieved by minimizing (some norm of) the difference between the output values calculated by the net and the target outputs. In our approach, we decouple the nonlinearities of the transfer functions at each layer from the linear interlayer pattern propagation. This paradigm follows the seminal observations of Biegler-König and Barman [3], and Tam and Chow [4]. The essence of their approach was to minimize, solving a sequence of least squares problems, the learning error on each layer separately, rather than globally for the entire network. Since for most practical applications the number of training examples exceeds the dimensionality of the training patterns, approximate results are typically

obtained. Our proposed algorithm, on the other hand, implements a sequence of alternating direction singular value decompositions (SVD), on a network architecture having a monotonically decreasing number of nodes in successive layers. We modify the traditional neural network architecture by introducing a *virtual input* layer connected to the input nodes through a nonlinear transfer function. The virtual layer acts as a preprocessor of the input vectors, and replaces a highly over-determined linear system with a uniquely solvable one.

3. Network Training

The learning algorithm begins at the output layer. A full iteration consists of a backward pattern propagation to the virtual layer, followed by a forward sweep back to the output layer. Our notation is as follows. A bar indicates a quantity calculated by the network (as opposed to a "target" or initialized quantity); a superscript f denotes a forward calculation. Each node implements a nonlinear transfer function $\varphi : \Re \rightarrow (0,1)$, typically a sigmoid. Since φ is bijective, the inverse φ^{-1} is well defined. The K seismic signatures used for training are stored as rows of matrices; the number of columns of each matrix equals the number of nodes of the corresponding processing layer. For convenience, the matrix dimensions are explicitly indicated as subscripts. The following two equations relate the presynaptic inputs T to the postsynaptic outputs S at the output layer.

$$\mathbf{T}_{KO} = \varphi^{-1}(\mathbf{S}_{KO}) \tag{1}$$

$$\overline{\mathbf{T}}_{KO} = \overline{\mathbf{S}}_{KH}\, \mathbf{W}_{HO} + \mathbf{b}_K\, \mathbf{v}_O. \tag{2}$$

In Eq. (2) $\mathbf{b}_K$ is a column vector of constant biases, and $\mathbf{v}_O$ is a row vector of bias weights.

The first phase of the learning algorithm minimizes $\|\mathbf{T}_{KO} - \overline{\mathbf{T}}_{KO}\|$ by solving the system

$$(\mathbf{T}_{KO} - \mathbf{b}_K\, \mathbf{v}_O) = \overline{\mathbf{S}}_{KH}\, \mathbf{W}_{HO} \tag{3}$$

for $\overline{\mathbf{S}}_{KH}$. Here, $\mathbf{T}_{KO}$ is known, and $\mathbf{W}_{HO}$ and $\mathbf{v}_O$ are assumed randomly initialized. We compute $\overline{\mathbf{S}}_{KH}$ using an SVD of $\mathbf{W}_{HO}$ from the right. When $\mathbf{W}_{HO}$ is a nonsingular square matrix, this step reduces to inverting $\mathbf{W}_{HO}$.

Typically we need to modify the calculated solution $\overline{\mathbf{S}}_{KH}$, since in the forward calculations the corresponding entries are output values of the sigmoid function φ, with range (0,1). Some of the calculated values of $\overline{\mathbf{S}}_{KH}$ may be outside the range of φ. Thus, we need to find an invertible renormalization matrix Γ with the property that

$$[\tilde{\mathbf{S}}_{KH} \mid \mathbf{b}_K] = [\overline{\mathbf{S}}_{KH} \mid \mathbf{b}_K]\, \Gamma. \tag{4}$$

$\tilde{\mathbf{S}}_{KH}$ has values in the range of φ, while preserving the final column $\mathbf{b}_K$.

We use $\tilde{\mathbf{S}}_{KH}$ to determine $\tilde{\mathbf{T}}_{KH} = \varphi^{-1}(\mathbf{S}_{KH})$. If more than one hidden layer is used, one minimizes $\|\tilde{\mathbf{T}}_{KN_\ell} - \overline{\mathbf{T}}_{KN_\ell}\|$ as above for each hidden layer ℓ backwards, until the first hidden layer (counting from the input side) is reached.

When the presynaptic input matrix $\tilde{\mathbf{T}}_{KH}$ for the (first) hidden layer has been calculated, we alter the process and begin a forward sweep, in which the weights are updated. We observe that $\tilde{\mathbf{T}}_{KH}$ and the postsynaptic output $\overline{\mathbf{S}}_{KV}$ of the virtual input layer are related by

$$\tilde{\mathbf{T}}_{KH} = \overline{\mathbf{S}}_{KV}\, \mathbf{W}_{VH} \tag{5}$$

where $\tilde{\mathbf{T}}_{KH}$ is known from the backward sweep and $\overline{\mathbf{S}}_{KV}$ is calculated from $\mathbf{S}_{KI}$. Thus, we do not need to compute $\tilde{\mathbf{S}}_{KV}$. Rather, we carry out an SVD of $\overline{\mathbf{S}}_{KV}$ from the left to determine the interconnection weight matrix $\overline{\mathbf{W}}^f_{VH}$. Using Eq. (5), we derive a forward estimate of $\mathbf{T}_{KH}$

$$\overline{\mathbf{T}}^f_{KH} = \overline{\mathbf{S}}_{KV}\, \overline{\mathbf{W}}^f_{VH}. \tag{6}$$

The forward estimate of $\mathbf{S}_{KH}$ is

$$\overline{\mathbf{S}}^f_{KH} = \varphi(\overline{\mathbf{T}}^f_{KH}). \tag{7}$$

The augmented weight matrix (obtained by adding a row vector v to W) between the hidden layer and the output layer is updated using the inverse of the renormalization matrix:

$$\hat{\mathbf{W}}^f_{HO} = \Gamma^{-1}\, \hat{\mathbf{W}}_{HO}. \tag{8}$$

Finally, a forward estimate of $\mathbf{T}_{KO}$ is obtained:

$$\overline{\mathbf{T}}^f_{KO} = \hat{\mathbf{S}}^f_{KH}\, \hat{\mathbf{W}}^f_{HO}, \tag{9}$$

which yields

$$\overline{\mathbf{S}}^f_{KO} = \varphi(\overline{\mathbf{T}}^f_{KO}). \tag{10}$$

The error is measured as some norm (e.g. L_2) of the difference between the provided target output pattern matrix and the corresponding forward estimate:

$$E = \|\mathbf{S}_{KO} - \overline{\mathbf{S}}^f_{KO}\| \tag{11}$$

If additional hidden layers are used, each hidden layer matrix $\overline{\mathbf{S}}_{KN_\ell}$ is renormalized by postmultiplying it by an appropriate matrix Γ_ℓ as in Eq. (4) during the backward

sweep, and the augmented weight matrix $\hat{\mathbf{W}}_{N_\ell N_{\ell+1}}$ is updated by premultiplying it by Γ_ℓ^{-1}, as in Eq. (8).

4. Renormalization Matrix

The matrix Γ scales the backward computation at each hidden layer (e.g., $\overline{\mathbf{S}}_{KH}$) so that its elements are in the range of φ. A simple method for scaling is the transformation $s_{ij} \rightarrow a \cdot s_{ij} + \beta$, given appropriate constants α, β. The augmented matrix

$$\hat{\mathbf{S}}_{KH} = [\overline{\mathbf{S}}_{KH} \mid \mathbf{b}_K]$$

includes a fmal column $\mathbf{b}_K$, the elements of which are multiplied by the bias weights in $\hat{\mathbf{W}}_{HO}$. The scaled matrix

$$[\tilde{\mathbf{S}}_{KH} \mid \mathbf{b}_K] = [\overline{\mathbf{S}}_{KH} \mid \mathbf{b}_K]\ \Gamma$$

should preserve the last column. For $1 \le i \le K$, the transformations

$$(s_{ij} \rightarrow \alpha \cdot s_{ij} + \beta, 1 \le j \le H;\ b_i \rightarrow b_i) \qquad (12)$$

can be implemented, for positive values of α, as a nonsingular and easily invertible matrix Γ [5]. Note that

a. The range $(0,1)$ of φ corresponds to domain values $(-\infty, \infty)$. We chose to restrict to the subrange [0.1, 0.91, which corresponds to the finite subdomain $[-\ln(9), \ln(9)]$.
b. We use $a \le 1$ to contract the values $\{s_{ij}\}$ to fit in an interval of length ≤ 0.8. If the range of $\{s_{ij}\}$ is less *than 0.8,* it is not necessary to expand the range, and we let $a = 1$.
c. The parameter β represents a shift of the values to tit within [0.1, 0.9]. We have selected a and β so that the average value μ for the set $\{s_{ij}\}$ is mapped to 0.5, the center of the interval [0.1, 0.9], and the value s_{ij} farthest from μ is mapped to either 0.1 or 0.9.

5. Predicted error of learning results

The learning algorithm introduces minimum norm approximations for two (or more) linear systems, since we use SVD computations. It is well known that piecewise optimization does not in general produce globally optimal results. However, we can predict near-optimal results for the learning algorithm if the following three conditions are met:

(1) The training set does not contain duplicate input vectors.
(2) The number H of hidden-layer nodes is greater than the number 0 of output nodes. (If more than one hidden layer is used, the number of hidden-layer nodes decreases in successive hidden layers.)
(3) The mapping from the input matrix $\mathbf{S}_{KI}$ to the virtual-layer matrix $\overline{\mathbf{S}}_{KV}$ produces a nonsingular square matrix (with $V = K$).

If condition (1) is not met, then the training set contains either duplicate copies of the same (input-output) pair, or contradictory patterns in which the same input is matched with more than one output pattern. Preprocessing of the training patterns can be introduced to handle such situations.

Condition (2) implies that me linear system

$$(\mathbf{T}_{KO} - \mathbf{b}_K \mathbf{v}_O) = \overline{\mathbf{S}}_{KH}\, \mathbf{W}_{HO}$$

is underdetermined (provided that the rows of $\mathbf{W}_{HO}$ are linearly independent, as is usually the case) and hence the minimum norm solution for $\overline{\mathbf{S}}_{KH}$ should be exact (one of infinitely many).

Condition (3) implies that $\overline{\mathbf{S}}_{KV}$ is invertible, and the singular value decomposition gives the inverse. Thus the minimum norm solution for W_{VH} should also be exact.

Since the inverse of the transfer function φ is known explicitly, the preceding analysis predicts that, in theory, the learning algorithm should result in error $E = 0$ (or as close to zero as can be expected from repeated finite-precision calculations with matrices).

6. Virtual Input Layer

The processing between the input and virtual layers is specified in a special way. For a given set of training vectors, we assume that there exists a particular nonlinear transfer function ψ that maps row vectors from the input pattern matrix $\mathbf{S}_{KI}$ to row vectors of the postsynaptic matrix $\mathbf{S}_{KV}$ output by the virtual layer. The mapping is direct, and does not involve the separate calculation of the row vectors of T. The function ψ is not altered during the learning process. We have

$$\mathbf{S}_{KV} = \psi(\mathbf{S}_{KI}). \qquad (13)$$

Condition (2) above illustrates the critical role of the virtual input layer. It is common for the number K of training examples to exceed the number of input nodes I. If we attempt to use the training process without the virtual layer, the linear system $\mathbf{T}_{KH} = \mathbf{S}_{KI}\, \mathbf{W}_{IH}$ is overdetermined for $\mathbf{W}_{IH}$,, and the resulting "best-fit" solution may be a poor approximation.

For example, for the standard XOR function with $I = 2$ binary inputs and $K = 4$ *training* examples, the training algorithm applied without the virtual input layer would

give $\bar{S}^{f}_{KO}$ = [0.5 0.5 0.5 0.5]^T as an approximation of the target output vector S_{KO} = [0 1 1 0]^T !

In the following, we present a mapping ψ which always produces a nonsingular matrix $\bar{S}_{KV}$. We first consider the special case of input vectors of dimension 1. Let D denote an upper bound for the maximum distance between two distinct training input values. For any input v, let $\psi(v)$ be the K-dimensional vector whose jth component is

$$\psi_j(v) = 1 - \frac{|v - s(j)|}{D} \qquad (14)$$

where $s(j)$ is the jth training input value, j= 1,2,...K. Note that the jth component of $\psi(s(j))$ is 1.

Let $\bar{S}_{KV}$ denote the virtual-layer matrix whose jth row is $\psi(s(j))$. We may assume without loss of generality that the training input values are ordered from smallest to largest. If not, we can interchange rows of the matrix, two at a time, and interchange the corresponding two columns. The net effect on the determinant is multiplication by $(-1)^2 = 1$. Let $d(i)$ denote the distance between $s(i)$ and $s(i+1)$. We have proven [3] the following theorem:

$$\det(\bar{S}_{KV}) = 2^{K-2}\left(\prod_{i=1}^{K-1}\frac{d(i)}{D}\right)\left(2 - \sum_{i=1}^{K-1}\frac{d(i)}{D}\right) > 0. \qquad (15)$$

The proof does not require that D be an upper bound for the maximum distance between distinct training input values. Such a value simply guarantees that the determinant is positive; in practice, we only need the matrix to be nonsingular, and a different value D may be used, as long as

$$\left(2 - \sum_{i=1}^{K-1}\frac{d(i)}{D}\right) \neq 0.$$

If the input vectors have dimension $I > 1$, the one-dimensional procedure above is applied one component at a time. For each component m = 1, 2, . . . I, construct a $K \times K$ matrix $\bar{S}^{(m)}_{KV}$ in which

$$\bar{S}^{(m)}_{KV}(i,j) = 1 - \frac{|s(i)_m - s(j)_m|}{D_m} \qquad (16)$$

with $i, j = 1, \ldots K$, and $m = 1, \ldots I$. Here $s(i)_m$ denotes the mth component of the itb $training$ vector $s(i)$ and D_m is the maximum of $|\, s(i),,, - s(j)_m \,|$ over all i, j = 1, ... K.

Lets $_{KV}$ be the block diagonal matrix whose m-th block is given by Eq. (16). The determinant of the full matrix is

$$\det(\bar{S}_{KV}) = \prod_{m=1}^{I}\left(\frac{1}{2}\prod_{j=1}^{K-1}\frac{2d_m(j)}{D_m}\right) > 0 \qquad (17)$$

The block implementation of $\bar{S}_{KV}$ for $I > 1$ above guarantees that the matrix is nonsingular, but increases the size of me matrix. As an alternative, the mapping ψ used in the one-dimensional case can be generalized to higher dimensions by defining the (i,j)th component to be

$$[\bar{S}_{KV}]_{i,j} = 1 - \frac{\|s(i) - s(j)\|}{D}, \; i, j = 1, \ldots K, \qquad (18)$$

where D is the maximum distance between input vectors in the training set. For some training sets, the matrix $\bar{S}_{KV}$ obtained by using the transformation (18) may be singular. However, the SVD method does not require an invertible matrix. If the linear system is overdetermined, the SVD will generate a minimum norm approximation.

Currently, we are considering a number of different virtual-layer transformations for use with this general approach to neural network training. A different choice of virtual layer transformation may provide better representation of the data, yielding better learning results and/or better ability of the trained neural network to generalize.

7. Results

To test the proposed methodology, the Pompano field, located in the Gulf of Mexico, was selected. Pompano is in deep water and has a significant potential for compartmentalized oil. The line scale heterogeneity caused by the channel depositional environment is well below the resolution of 3D seismic data. The information available to us included 3D seismic data, well logs, core samples, oil location and production profiles.

Five seismic variables were provided: the reflected seismic signal, acoustic impedance (AI), and three components of the Hilbert transform of the reflected seismic signal (amplitude, frequency, and phase). Each of the five datasets had 80 megabytes of data with a spatial resolution of 4 km in x and 7 km in y. An x-t plot of the reflected seismic signal is displayed in Fig. 2. For the case of normal incidence, the amplitude of the reflected signal depends on the change in acoustic impedance at the interface between two materials, where AI is the product of density and the speed of sound in the material.

The log data is sampled at regular intervals along the well. In Pompano, most wells are not vertical (of the 17 wells studied here, only three are vertical). The *DeepLook* consortium of petroleum companies provided us with the rate of deviation for each well. We calculated the (x,y,z) coordinates for each data sample in the log data. The seismic data has coordinates of (x,y,t), where t is the two way travel time. To convert from t to z, we used the *DeepLook* smooth estimates of the average velocity [v = (2 z)/t]. Such estimates are less detailed than the seismic data.

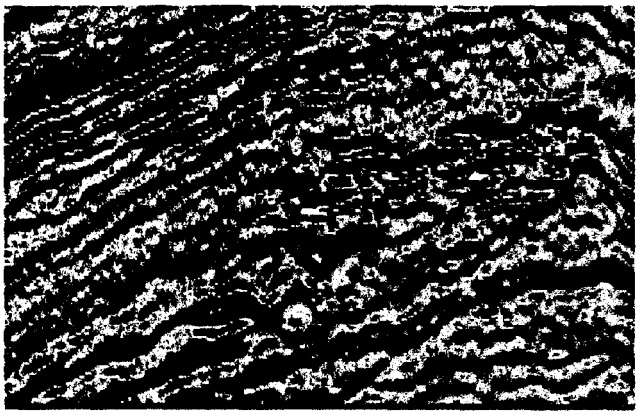

Figure 2: An x-t cross section of a reflected seismic signal. Lighter colors indicate positive data.

The x-y grid spacing is 26.67 m and the vertical spacing is 4 ms in two-way travel time. Using the velocity values for the seismic data, the vertical resolution ranges **from** 3.66m to 4.09 m. The spacing of the data values for the well logs is either 0.61 m or 0.76 m for most wells. Thus, the vertical resolution for the well logs is about a factor of five greater **than** the resolution for the seismic data

We have produced integrated **datasets** for each of the 17 wells in a common format. The integrated **datasets** contain 17 variables: the coordinates and two way travel times, gamma ray, resistivity, the time for sound to travel a fixed distance, density, porosity, calculated acoustic impedance, acoustic impedance, reflected seismic signal, and the three components of the **Hilbert** transform (frequency, phase, amplitude).

Whenever we lacked measured values for a log variable, we substituted average values **from** the other wells. For most wells, we had values of True Vertical Depth Sub Sea (TVDSS), which is the depth below sea level. If the log file for a well did not have the TVDSS, we calculated it using the measured distance down the well and the rate of deviation. The deviation of the wells and the associated approximation in the time to depth conversion may contribute to a lower correlation factor between the estimated and real log measurements. The same is true when substituting average log values in lieu of real log data.

The learning methodology described in this paper was programmed into a new computer *code* named ***DeepNet.*** The code is written in FORTRAN-90 and is implemented in Microsoft FORTRAN **PowerStation** 4.0 running under Windows NT 4.0. Results to date are very encouraging, both in terms of me exceptional speed of the learning process, and the quality of prediction obtained with test data. For instance, the **typical** training time using a **dataset** of several hundred seismic signatures is of the order of seconds on a Dell Workstation 610 configured with 2 Pentium II Xeon processors operating at 400 MHz. The training and prediction are illustrated in Figs. 3 and 4. A separate net was trained for each log response.

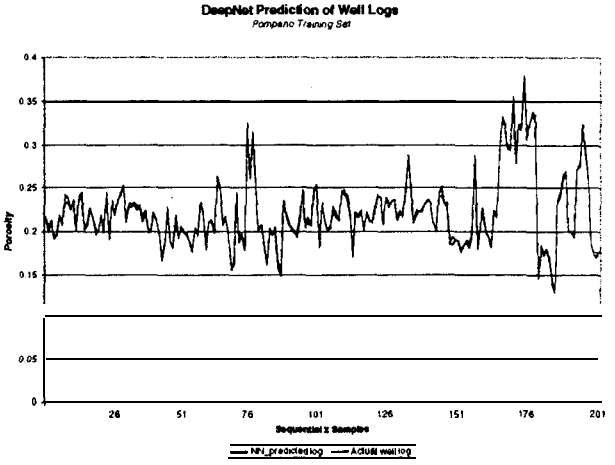

Figure 3: Almost perfect ***DeepNet*** prediction of porosity log using training set data.

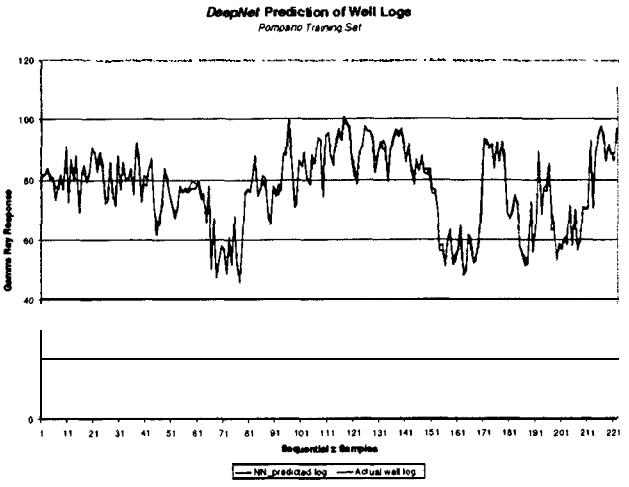

Figure 4: Almost perfect *DeepNet* prediction of gamma ray log using training set data.

It was important to assess the quality of predictions that can be obtained with ***DeepNet.*** The network is initially trained using a small subset of the available data: typically, we have used the seismic-to-log correspondence for one, two, or *three* wells. ***DeepNet*** *was then used* to generate pseudo logs at other wells in the Pompano tield. For comparison purposes the same pseudo logs were generated using two state-of-the-art neural network algorithms (i.e., the nearest neighbor and **Nadaraya-**Watson paradigm **[6]**). The ***DeepNet*** results are illustrated in Figures 5 and 6. The latter results are given in Figures 7 and 8. Pompano well B-10 was used for the prediction test.

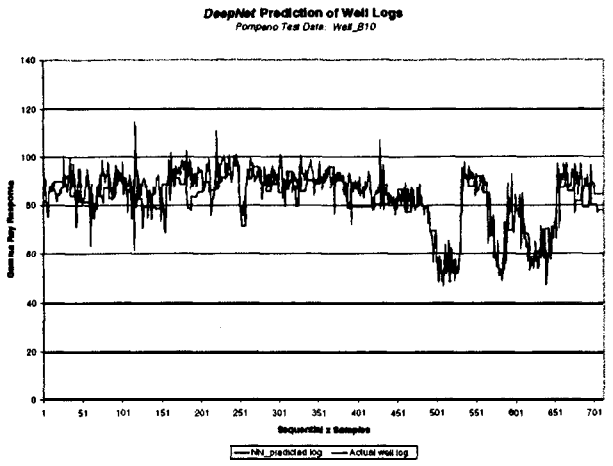

Figure 5: *DeepNet* prediction of gamma ray logs for test dataset.

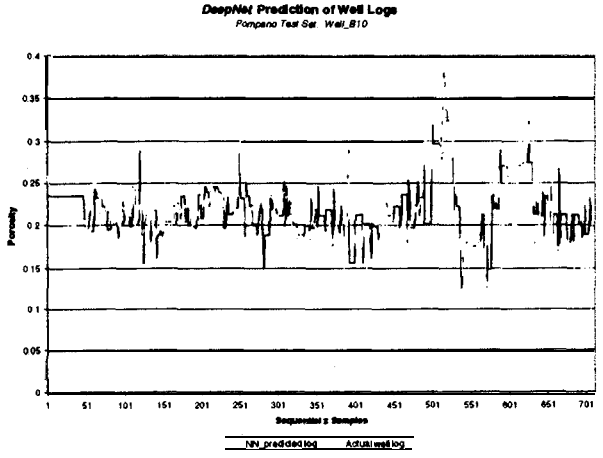

Figure 6: *DeepNet* prediction of porosity for test dataset.

8. Conclusions

As evidenced by the tests, our method exhibits an excellent prediction performance. *The DeepNet* methodology will enable the oil exploration and production industry to gain an unprecedented insight into fluid types and distributions in reservoirs of interest.

Acknowledgements

This research was performed at the Center for Engineering Science Advanced Research, Computer Science and Mathematics Division, Oak Ridge National Laboratory. Funding was provided by the *DeepLook* Consortium under Agreement Number ERD-97-1506 with Lockheed Martin Energy Research Corporation, and by the Engineering Research Program of the DOE Office of Basic Energy Sciences under contract DE-AC05-96OR22464 with Lockheed Martin Energy Research Corporation.

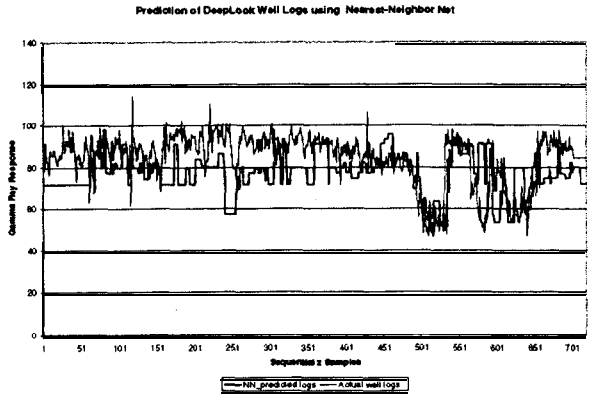

Figure 7: Prediction of gamma ray log using nearest neighbor net for test well B-10.

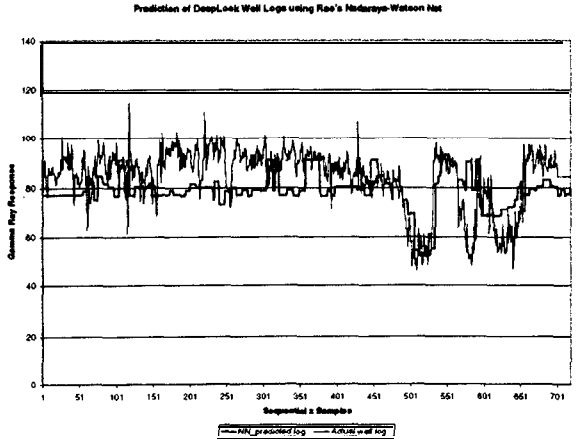

Figure 8: Prediction of gamma ray log using Nadaraya-Watson algorithm for test well B-10.

References

1. P.M. Wong *et al*, "An Improved Technique in Porosity Prediction: A Neural Network Approach", *IEEE Trans. Geosc. Rem Sens.*, **33(4)**, 971-980 (1995).

2. J.Schuelke, J. Quirein, and J. Pita. "Prediction of Reservoir Architecture and Porosity Distribution using Multiple Seismic Attributes", *Procs., 30th Offshore Technology* Con-, pp. 83-88 (1998).

3. F. Biegler-König and F. Bärman, "A Learning Algorithm for Multilayered Neural Networks based on Linear Least Squares Problems", *Neural Networks.* **6**, 127- 131 (1993).

4. Y. F. Tam and T. W. S. Chow, "Accelerated Training Algorithm for Feedforward Neural Networks based on Least Squares Method", *Neural Processing Letters*, **2(4)**, 20-25 (1995).

5. J. Barhen, R. Cogswell, and V. Protopopescu, "Single Iteration Training Algorithm for Multilayer Feed Forward Neural Networks", *Neural Proc. Lett.*, (submitted, 1999).

6. Rao, N.S.V., et al., "Learning Algorithms for Feedforward Networks Based on Finite Samples," *IEEE Trans. on Neural Networks*, **7(4)**, *926-960* (1996).