Ω

SPE 49026

Society of Petroleum Engineers

Stochastic Reservoir Simulation Using Neural Networks Trained on Outcrop Data

Jef Caers, Stanford University and André G. Journel, SPE, Stanford University

This paper was prepared for presentation at the 1998 SPE Annual Technical Conference and Exhibition held in New Orleans, USA., 27-30 September 1998.

Abstract

Extensive outcrop data or photographs of present day depositions or even simple drawings from expert geologists contain precious structural information about spatial continuity that is beyond the present tools of geostatistics essentially limited to two-point statistics (histograms and covariances). A neural net can be learned to collect multiple point statistics from various training images, these statistics are then used to generate stochastic models conditioned to actual data. In petroleum applications, the methodology developed can be a substitute for objects based-algorithms when facies geometry and reservoir continuity are too complex to be modelled by simple object such as channels. The performance of the neural net approach is illustrated using training images of increasing complexity, and attempts at explaining that performance is provided in each case. The neural net builds local probability distributions for the facies types (categorical case) or for petrophysical properties (continuous case). These local probabilities include multiple point statistics learned from training images and are sampled with a Metropolis-Hastings sampler which also ensures reproduction of statistics coming from the actual subsurface data such as locally varying facies proportions.

Introduction

Stochastic simulation is a tool that has been driven by the need for more "representative" *images* than the smoothed *maps* produced by regression techniques such as kriging. Although locally accurate in a minimum error variance sense, kriging maps are poor representations of reality with various artefacts. Kriging produces conditional bias in the sense that through smoothing small values are overestimated and large values are underestimated. Moreover, this smoothing is not uniform since it is minimal near the data locations and increases as estimation is performed further from the data locations. Smoothed maps should not be used where spatial patterns of values is important such as in the assessment of travel times or production rates in flow simulations. Reproduction of the spatial connectivity of extremes is critical in any study involving the determination of risk of a rare event happening.

Most current methods of simulation rely on the modeling of two-point statistics such as covariances and variograms and the subsequent reproduction of these second moments. Yet, many phenomena are more complex in the sense that their spatial patterns cannot be captured by two-point statistics only. In particular, identification of two-point statistics only is not sufficient to reproduce patterns of connectivity of extreme values over long ranges. Strings of extreme values, curvilinear or wedge-shaped clusters of similar values are common occurrences in earth sciences, and their importance in water or oil flow simulations cannot be overstated.

The most common way of performing stochastic simulation is through some form of Gaussian simulation (Deutsch and Journel, 1997, p. 139). This class of simulation algorithms is based on the congenial properties of multi-Gaussian models, which import some severe restrictions not always fully appreciated:

- The marginal distribution is normal which is rather thin-tailed. This is the least restrictive aspect of the multi-Gaussian assumption since we can always "scale" the simulated Gaussian field by employing an appropriate long-tailed marginal back-transformation (program trans in GSLIB, Deutsch and Journel, 1997, p. 132).

- The multivariate and thus all conditional distributions are normal which leads to extremes which in the asymptotic limit, i.e. as the random function is taken to higher threshold levels, become independent. The location of extremes of a stationary isotropic Gaussian field is equivalent to a spatial Poisson process.

- Both the shape and the variance of conditional distributions are homoscedastic, i.e., they are independent of the conditioning data values.

It is sometimes naively thought that a mere univariate back-transformation of a normal distribution into a non-normal one, renders the original Gaussian stochastic simulation a non-Gaussian one, see most major publications on stochastic simulation, except for Goovaerts (1997, p. 266). Actually, a univariate transformation merely scales the simulated field back to the target histogram but does not correct for the last two drawbacks cited above.

Improvement in connecting extremes could be obtained by using an indicator simulation method (Deutsch and Journel, 1997, p. 149). In this case, specific transition probabilities of extremes can be imposed but the method is still restricted to two locations only and therefore to two-point statistics. Also, the modeling of multiple indicator covariances can be difficult due to lack of data, particularly at extreme thresholds.

The quest for "better" models or methods is motivated by the fact that in many circumstances additional structural information on the spatial distribution is actually available. This information need not be hard data. It can densely sampled training images of a deemed similar field, e.g. outcrop photographs, secondary data exhibiting similar spatial variation. Training images can be generated using physical dynamical models or they may be hand-drawn by an expert geologist. Every bit of information gives us more *intelligence* about the actual spatial distribution, particularly that of extreme values. Also, by drawing, producing or making training images, one explicitely states one's prior belief about what the spatial model looks like, without hiding it behind some mathematical formulation such as a multi-Gaussian distribution. Using training images the whole concept of stochastic imaging is turned around: one first selects a realistic image and subsequently tries to construct a model that captures the essence of that image. Thus, instead of imaging an analytical model, one models an image deemed representative.

A framework for direct incorporation of additional multipoint information has been introduced with the concept of multi-point statistics and extended normal equations (Journel and Alabert, 1989, Guardiano and Srivastava, 1992 and Srivastava, 1992). Information is now pooled at more than two locations at the same time, not being restricted to pairs of data such as in the variogram. The problem is one of inference of the multipoint (more than two) covariances involved and ensuring their consistency. Such inference requires extensive training images and important RAM to store the multiple point covariances obtained by scanning.

In literature one can find various other methods for incorporating multi-point statistics

- Incorporating multi-point statistics using simulated annealing: the technique tolerates some amount of inconsistency (non-positive definite situation), however as the number of classes becomes high and the order of the multi-point statistics raises, CPU and RAM quickly become a problem.
- Iterative methods for conditional simulation (Srivastava, 1992; Wang, 1996). CPU is less of a problem but RAM is, since all conditional distributions (which are multipoint statistics) scanned from the training image must be stored. There is no explicit modeling of these multi-point statistics, so no data-expansion is made. Hence, the training image may have to be very large or there are many of them to allow for enough DEV's to be recognized.
- Markov Random Fields (Besag, 1974; Tjelmeland, 1996). In this case one explicitely states a model for the multivariate distribution of the values at all spatial locations, typically a non-Gaussian model. The method is internally consistent and not too CPU intensive. Yet, the model contains hundreds of parameters which are difficult to interpret and hence difficult to fit using actual data or a training image. Therefore one typically restricts the model calibration to a few parameters, the other parameters being set to arbitrary values.

Methodology outline

This paper aims at demonstrating that neural networks applied to stochastic simulation can overcome many of the previously mentioned problems. A neural network is in essence a multi-parameter non-linear regression model that defines a general non-linear mapping from any space into any other space both of arbitrary dimension. In our case, the neural network is trained to determine conditional distributions from a training image, then it proceeds with simulation using these distributions. Conditional distributions relate the value at any location to neighbouring data values within a template centred at that location, see Figure 1. The neighbourhood template can be anisotropic (its anisotropy could be associated to variogram ranges) and there could be different templates of different sizes if a multiple grid approach is used (Goodman and Sokal, 1989; Tran, 1994). The limitation to a finite neighbourhood is similar to the screening effect assumption; it is commonly used in kriging. Within such data neighbourhoods, the neural network is learned to determine values y at a any given location given all neighbouring data values $\mathbf{x}$.

More precisely, the neural network is trained to model the local conditional probability density function (cpdf)

$$f(y|\mathbf{x})\,\mathrm{d}y = \Pr\{y < Y < y + \mathrm{d}y|\mathbf{x}\}$$

or its integral, the conditional cumulative distribution function (ccdf). The neural network maps the multi-point input (the training image) into a one dimensional output, the previous conditional distribution. In the training phase, we input the conditioning data y and $\mathbf{x}$, that is the training image, and the net determines the model $f(y|\mathbf{x})$. In the generalization phase we input $\mathbf{x}$ and draw values of y from these cpdf's, see Figure 2. The network serves as a non-linear connection between input and output.

In the training phase the network is learned by scanning a training image using a certain template, retrieving multi-point information in the form of conditional distributions. This approach is somewhat different from the classical training phase of a neural network, where the network output is compared with true or target output values, the learning then proceeds by back-propagating the mean square difference between network output and target output through adjusting the network parameters.

The output in our case are probability distributions which are not available beforehand. We will show that an error function can still be defined to fine tune the training.

In the generalization phase, one visits each node i of the N nodes of the simulation grid (except the edges) and draws a simulated value y_i conditioned to the data $\mathbf{x}_i$ found within the prescribed template. The term generalization refers to the fact that, in this phase, the neural network is applied to assess distributions conditioned to point information different from those found in the training data. Generalization thus entails data expansion.

The generalization phase consists of generating stochastic simulations (images). The problem is that, initially, we do not have all the neighbouring values of y; typically only a few conditioning data are available. The idea is to initialize the whole field with random values (or according to some prior marginal distribution) and then update each of these values using the network generated cpdf's $f(y|\mathbf{x})$. A random path is defined that visits each node or location, and one keeps cycling through the **whole** field until some convergence is obtained. We will define later criteria for such convergence. This procedure is tantamount to setting up a Markov Chain (Metropolis-Hastings sampler) sampling a stationary distribution. All original conditioning data values remain frozen. The general outline of our method is as follows:

1. Training (=Modeling the image)
   (a) Define a neighbourhood template with which the training image is scanned
   (b) Train the neural net with the scanned data, extracting only "essential" features of the image. The results are a model for the cpdf $f(y|\mathbf{x})$ known for all values of y and $\mathbf{x}$.
2. Generalization (=Simulating an image)
   (a) Define a random path visiting all non-frozen nodes
   (b) Initialize the simulation by freezing the original data values at their locations and filling-in the remainder nodes with values drawn from the global target histogram (prior cdf)
   (c) Start loop : visit all nodes, at each node do
      - Draw a new value for that node (either at random or according to the prior cdf)
      - Retrieve from the neural net the cpdf of the old and new value
      - Change old value into new value according to some probabilistic criterion (e.g. Metropolis criterion).
   (d) Stop loop when convergence is reached, if not goto (c)
   (e) If another image is required start all over from (a)

First, we will develop the general methodology for stochastic simulation using local conditional distributions, and a Metropolis-Hastings sampler. Next we elaborate on how the neural network models these conditional distributions.

Conditional simulation with Markov Chains

Sequential simulation Sequential conditional simulation methods rely on the conditional distribution of the value at each node to be simulated given the original conditioning data and previously simulated values. Points on a regular grid are visited along some random path and their values are drawn. In the case of Gaussian sequential simulation, the parameters of the normal conditional distributions are identified to the kriging means and variances. The dimension of the kriging system rapidly becomes unwieldy if all previously simulated values are retained. An approximation is obtained by retaining only those conditioning values that are most "consequential". In most cases only the "closest" data are retained, where closest is defined with respect to some variogram distance measure. Retaining the closest data amounts to assume that the RF features some Markov behaviour, i.e. the closest data screen the information provided by further away data.

In the case of Direct Sequential Simulation, only the means and variances of the conditional distributions (not necessarily Gaussian) are identified to the kriging results (Xu and Journel, 1994).

Metropolis sampling Suppose now that, through some algorithm, we have obtained the probability distribution of the attribute value at any specific location to be simulated conditional to L neighbouring data values; that distribution being generally non-Gaussian. We also assume a Markov property, i.e. the conditioning is restricted to the L neighbours. We could then implement a sequential simulation method with such conditional distributions.

A powerful yet simple method for enforcing such conditional distributions is the Metropolis-sampler. Denote the set of N pixel locations in the simulation grid by $S = \{1 \ldots N\}$. The simulated values over that grid is then the array $\mathbf{y} = \{y_1 \ldots y_N\}$. Denote the neighbourhood data of a grid node i by a vector $\mathbf{x}_i$, i.e. the values closest to y_i in some sense. Denote by $f(\mathbf{y})$ the joint multivariate density of all N pixel values.

Conditional simulation with a Metropolis sampler calls for iterative visits of each grid node (Figure 3). The initial grid is filled with random values. At each successive visit of a grid location i with present value y_i, a new value y_i^* is randomly drawn from the set of possible values and the original y_i is replaced by y_i^* with probability

$$\min\left\{1, \frac{f(\mathbf{y}^*)}{f(\mathbf{y})}\right\}$$

where $f(\mathbf{y}^*)$ is the joint distribution of the new pixel grid, where $\mathbf{y}^*$ differs from $\mathbf{y}$ only by the value y_i^* at the current grid location i. Using the conditional probability paradigm this can be rewritten as

$$\min\left\{1, \frac{f(y_i^*|\text{ all } y_j, j \neq i)}{f(y_i|\text{ all } y_j, j \neq i)}\right\}$$

If only conditioning on local neighbouring values is considered, the transition probability from $\mathbf{y}$ to $\mathbf{y}^*$ reduces further to

$$P(\mathbf{y} \to \mathbf{y}^*) = \min\left\{1, \frac{f(y_i^*|\mathbf{x}_i)}{f(y_i|\mathbf{x}_i)}\right\} \qquad (1)$$

The local conditional distributions $f(y_i^*|\mathbf{x}_i)$ are required, they were determined by the neural net in the training phase. The original conditioning data always remain frozen at their locations.

The Metropolis sampler thus amounts to a Markov chain with known transition probability matrix $P(\mathbf{y} \to \mathbf{y}^*)$ defined by the expression (1). P is zero whenever $\mathbf{y}$ differs *by* more than one pixel from $\mathbf{y}^*$. The stationary distribution of the Markov chain is the joint distribution $f(\mathbf{y})$.

Honouring global statistics with the Metropolis-Hastings sampler Many geostatistical simulation methods honour global statistics such as the histogram and a global variogram. The Metropolis-Hastings sampler, a more general form of the Metropolis sampler leaves the flexibility to explicitely honour a global histogram. The general form of the Metropolis-Hastings probability is written as, see Hastings, 1970; Ripley, 1987, p. 115:

$$P(\mathbf{y} \to \mathbf{y}^*) = \min\left\{1, \frac{f(y_i^*|\mathbf{x}_i)t(y_i^* \to y_i)}{f(y_i|\mathbf{x}_i)t(y_i \to y_i^*)}\right\}$$

where we have now introduced a new but completely arbitrary transition probability t. t describes a transition of one pixel value into another pixel value that is independent of the neighbouring values in $\mathbf{x}_i$. We find back the original Metropolis criterion (1) when t is symmetric, i.e. $t(y_i^* \to y_i) = t(y_i \to y_i^*)$. To honour a specific histogram we will define an objective function that measures the difference between the target histogram and the histogram at the current iteration in the Markov chain. If the global histogram is described by C quantiles q_c then the objective function becomes

$$O(\mathbf{y}) = \sum_{c=1}^{C}(q_c - q_c^{(s)}(\mathbf{y}))^2$$

where $q_c^{(s)}$ are quantiles of the global histogram at the current simulation step, depending on the current set of pixels $\mathbf{y}$. The objective function is then used to define the ratio of transition probabilities as follows

$$\frac{t(y_i^* \to y_i)}{t(y_i \to y_i^*)} = \exp(-(O(\mathbf{y}^*) - O(\mathbf{y}))/k_B)$$

where k_B is a positive constant defined such that the previous ratio is of the same order of magnitude than $f(y_i^*|\mathbf{x}_i)/f(y_i|\mathbf{x}_i)$. The Metropolis-Hastings sampling criterion then becomes

$$P(\mathbf{y} \to \mathbf{y}^*) = \min\left\{1, \frac{f(y_i^*|\mathbf{x}_i)}{f(y_i|\mathbf{x}_i)}\exp(-(O(\mathbf{y}^*) - O(\mathbf{y})/k_B)\right\} \qquad (2)$$

Note that the objective function $O(\mathbf{y})$ is easily updatable after each iteration, since only one grid node value changes, hence the honouring of a target histogram can be obtained at virtually no extra cost. Experience has shown that the easiest histogram to honour is the uniform histogram. It is therefore advisable to make a uniform score transform of the training image.

Multiple grid approach Many images have large scale features that cannot be represented with a template of limited size. Large scale trends may be present or some structures such as channels may run over the entire image. In such case we can define multiple grids, see Figure 4: simulation starts on the coarser grid allowing to capture the large scale structure, then proceeds onto the finer grids to depict the smaller scale details. The values simulated on the coarse grid are frozen as conditioning data in the subsequent finer grids. In our approach, we need a set of cpdf's for each nested grid, which means that we have to train different neural nets for different grids, with scanned data using templates of different sizes, see Figure 4. It is advisable to use multiple grids when the number of simulation nodes exceeds vastly the original sample size (number of conditioning data).

A network architecture for building local conditional distributions

A one layer network The neural network is used to model all local conditional distributions needed in the generalization phase of the conditional simulation method. Markov chain algorithms ask for the conditional distribution $f(y_i|\mathbf{x}_i)$ at the visited grid location i given any current realization $\mathbf{x}_i$ of the neighbouring data. The neural network utilizes a general non-Gaussian cpdf model, which is "fitted" or "trained" on the training set. There is no close form mathematical representation for the corresponding multivariate distribution model, the net delivers the table of all required conditional distributions, once their parameters are established. Before describing the parameter fitting algorithm, we must define the network architecture. We first consider a network with one hidden layer (Bishop, 1995).

Consider again gridded values which take continuous outcomes, the y's in Figure 2. We assume a Markov screening to hold in the 2- or 3-dimensional space. *Target value* is typical neural network language for the value y_i for which the conditional distribution $f(y_i|\mathbf{x}_i)$ has to be derived.

A neural network consists of neural nodes which contain mathematical operations and branches which direct the results of these operations to other neural nodes[1]. Most common operations consist of summation, identity functions (input=output) and non-linear tranforms using some sigmoidal functions. The neural nodes are arranged in layers which are connected. The simplest neural network is a network that connects the input to a middle layer and then to the output layer. The middle layer is called the *hidden layer*. A single hidden layer feed forward neural network consists therefore of three layers. If the hidden layer has K internal nodes the ccdf is modeled by mapping the

[1] to avoid confusion with nodes of the simulation grid we will use the terms *neural nodes* and *grid nodes* to differentiate wherever necessary

input $y|\mathbf{x}$ into the output, see Figure 5:

$$F(y|\mathbf{x}) = \sum_{k=1}^{K} o_k \, T\left(\sum_{l=1}^{L} \widetilde{w}_{k,l} x_l + v_k y\right) \qquad (3)$$

where o_k are K output weights and $\widetilde{w}_{k,j}$ are $K \times L$ input weights linking the hidden layer, v_k are weights between the input target value y and the hidden layer. K is the number of neural nodes in the hidden layer and L is the number of data values x_l in the template $\mathbf{x}$. T is a non-linear transfer function, typically a sigmoid with some known mathematical expression. In our case, there is a further restriction on the function T in that it should be a licit cdf. Since a linear combination of cdf's also is a cdf, if $o_k \geq 0$, $\sum_{k=1}^{K} o_k = 1$, the resulting network output (3) is a legitimate cdf.

For convenience, without changing its architecture, we can rewrite the network as

$$F(y|\mathbf{x}) = \sum_{k=1}^{K} o_k \, T\left(v_k\left(y - \sum_{l=1}^{L} w_{k,l} x_l\right)\right), \qquad (4)$$

with $\widetilde{w}_{k,j} = -v_k w_{k,j}$. The network architecture is drawn in Figure 5.

From expression (4), it is clear that the ccdf appears as a linear combination of "kernel" cdf's. One can calculate the mean of this ccdf, i.e. the conditional mean as

$$\begin{aligned} \mathrm{E}[Y|\mathbf{x}] &= \int_{-\infty}^{\infty} y \, \mathrm{d}F(y|\mathbf{x}) \\ &= \sum_{k=1}^{K} o_k \int_{-\infty}^{\infty} y \, \mathrm{d}T\left(v_k\left(y - \sum_{l=1}^{L} w_{k,l} x_l\right)\right) \end{aligned}$$

if $T(y)$ is a cdf with zero mean and unit variance then

$$\begin{aligned} \int_{-\infty}^{\infty} y \, \mathrm{d}T\left(v_k\left(y - \sum_{l=1}^{L} w_{k,l} x_l\right)\right) &= \sum_{l=1}^{L} w_{k,l} x_l \\ &= m_k(\mathbf{x}) \end{aligned}$$

and the conditional mean becomes

$$\mathrm{E}[Y|\mathbf{x}] = \sum_{k=1}^{K} o_k m_k(\mathbf{x}) = m(\mathbf{x}) \qquad (5)$$

Similarly, the conditional variance is derived as:

$$\begin{aligned} \mathrm{Var}[Y|\mathbf{x}] &= \int_{-\infty}^{\infty} (y - m(\mathbf{x}))^2 \, \mathrm{d}F(y|\mathbf{x}) \\ &= \sum_{k=1}^{K} o_k \left(\frac{1}{v_k^2} + (m_k(\mathbf{x}) - m(\mathbf{x}))^2\right) \qquad (6) \end{aligned}$$

which is a classical result from variance analysis. Thus the one hidden layer network architecture results in a ccdf which is a mixture of K cdf kernels of the same type (i.e. a cdf T with zero mean and unit variance) but centred at different values $m_k(\mathbf{x})$. For this one layer network the parameter vector is: $\underline{\theta} = \{o_k, v_k, w_{k,l},\ k = 1, \ldots K,\ l = 1, \ldots, L\}$. The conditional variance (6) is heteroscedastic in that it is dependent of the data values $\mathbf{x}$. The conditional mean (5) is a linear function of the data $\mathbf{x}$ in that each of the K kernel means $m_k(\mathbf{x})$ is linear in $\mathbf{x}$. This linear relation of $m(\mathbf{x})$ with $\mathbf{x}$ may be too restrictive to model a general image, hence a second hidden layer should be added to obtain a more general non-linear approximator to the conditional distribution function. Correspondingly, the computational effort will increase considerably and, as will be shown, the learning process becomes more difficult.

A two layer network The fact that the conditional mean in Eq. (5) is linearly dependent on the conditioning data values is a restriction to the flexible approximation capabilities of the neural network. To remove this restriction, we will add a second hidden layer of neural nodes (Bishop, 1995). A first hidden layer containing K_1 nodes is inserted between the input and the original hidden layer with K_2, see Figure 6. A preliminary non-linear transform S of the data $\mathbf{x}$ is applied prior to applying the T transform. In neural net jargon, the S transform constitutes the first layer and the cdf-type transform is then called the second hidden layer, see Figure 6. Thus, the first hidden layer containing K_1 neural nodes is inserted between the input and the original hidden layer (now called second layer) with K_2 neural nodes. This results in the following conditional distribution:

$$F(y|\mathbf{x}) = \sum_{k_2=1}^{K_2} o_{k_2} \, T(v_{k_2}(y - \sum_{k_1=1}^{K_1} w_{k_2,k_1} \, S \sum_{l=1}^{L} u_{k_1,l} x_l))) \qquad (7)$$

where $u_{k_1,l}$ are weights associated to the first layer and S is any non-decreasing function. Based on the same notation for the one layer neural network, we can simplify Eq. (7).

$$F(y|\mathbf{x}) = \sum_{k_2=1}^{K_2} o_{k_2} \, T\left(v_{k_2}\left(y - m_{k_2}(\mathbf{x})\right)\right)$$

where we replace the linear regression $m_k(\mathbf{x})$ of expression (5) by a combination of K_1 **non-linear** functions of type S

$$m_{k_2}(\mathbf{x}) = \sum_{k_1=1}^{K_1} w_{k_2,k_1} \, S\left(\sum_{l=1}^{L} u_{k_1,l} x_l\right) \qquad (8)$$

The following comments can be made about this two layer network

- For $F(y|\mathbf{x})$ to be a permissible ccdf it suffices that $\sum_{k_2=1}^{K_2} o_{k_2} = 1$, $o_{k_2} \geq 0$ and T is a permissible cdf.
- S is valued within the probability range of the cdf T, which in this case is the whole real line $[-\infty, +\infty]$.
- The two layer network appears as a mixture of K_2 kernel cdf's of type T, each with zero mean and unit variance, and centred on

$$m_{k_2}(\mathbf{x}) = m_{k_2}(\mathbf{x}, \mathbf{w}, \mathbf{u}) = \sum_{k_1=1}^{K_1} w_{k_2,k_1} \, S\left(\sum_{l=1}^{L} u_{k_1,l} x_l\right)$$

Note that $m_{k_2}(\mathbf{x})$ is non-linear in $\mathbf{x}$. Finally:

$$\begin{aligned}
\mathrm{E}[Y|\mathbf{x}] &= \sum_{k_2=1}^{K_2} o_{k_2} m_{k_2}(\mathbf{x}, \mathbf{w}, \mathbf{u}) = m(\mathbf{x}) \\
\mathrm{Var}[Y|\mathbf{x}] &= \sum_{k_2=1}^{K_2} o_{k_2} \left(\frac{1}{v_{k_2}^2} + (m_{k_2}(\mathbf{x}, \mathbf{w}, \mathbf{u}) - m(\mathbf{x}))^2 \right)
\end{aligned}$$

It appears that the sole function of the added first layer of K_1 nodes, is to make the regression $\mathrm{E}[Y|\mathbf{x}]$ non-linear.

Understanding the network parameters It is essential for the implementation of a fast learning neural net to have a better insight into the approximation capabilities of the network and the meaning of each network parameter. We proceed as follows. From the cumulative conditional distribution in expression 7, the conditional density is calculated as the derivative

$$f(y|\mathbf{x}) = \sum_{k_2=1}^{K_2} o_{k_2}(v_{k_2} T'(v_{k_2}(y - m_{k_2}(\mathbf{x})))) \tag{9}$$

where T' is the derivative of T with respect to its argument. If one introduces the function h as

$$h(y - m_{k_2}(\mathbf{x}), v_{k_2}) = v_{k_2} T'(v_{k_2}(y - m_{k_2}(\mathbf{x}))$$

then

$$f(y|\mathbf{x}) = \sum_{k_2=1}^{K_2} o_{k_2}(h(y - m_{k_2}(\mathbf{x}), v_{k_2})) \tag{10}$$

The function h is a bell-shaped pdf if T is a sigmoidal cdf. For example, if T is Gaussian cdf, h is a Gaussian pdf.

The parameter v_{k_2} is a shape parameter controlling the width of the bell. Expression (10) can be interpreted as a decomposition of the output density into various "kernel densities" or "nodal densities" h. The neural network outputting the conditional density is drawn in Figure 7.

The function h is seen as the conditional density of the target y given the conditioning data $\mathbf{x}$ *and* node k_2 in the neural network

$$h(y - m_{k_2}(\mathbf{x}, \mathbf{w}, \mathbf{u}), v_{k_2}) = h(y|\mathbf{x}, \mathbf{k}_2) \tag{11}$$

Since the set $\{o_{k_2}\}$ is a valid probability mass, the nodal densities $h(y|\mathbf{x}, \mathbf{k}_2)$ are compounded with the weights o_{k_2} into the final cpdf:

$$f(y|\mathbf{x}) = \sum_{k_2=1}^{K_2} o_{k_2} h(y|\mathbf{x}, \mathbf{k}_2) \tag{12}$$

$\mathbf{k}_2$ are all the network parameters that define the node k_2, it thus includes all the parameters of $\underline{\theta}$ except the mixing parameters o_{k_2}. In statistical jargon, expression (12) is termed a Gaussian mixture. The parameters o_{k_2} thus appear as prior probabilities or relative contributions of each of the K_2 nodes of the second hidden layer. The parameters v_{k_2} are scale parameters controlling the width of the radial basis functions h and the parameters $\mathbf{w}$, $\mathbf{u}$ define the non-linear connection between the means of these radial basis functions and the input data $\mathbf{x}$.

Determining the parameters of the neural net

To determine the values of the various parameters $\underline{\theta} = \{o_{k_2}, \ v_{k_2}, \ \mathbf{w}, \ \mathbf{u}, k_2 = 1, \ldots, K_2\}$ the network must be learned from training data. The training data can be obtained by scanning a training image for all available combinations $\{y, \mathbf{x}\}$. Edge effects are ignored. The classical method of training neural nets, where some mean square deviation of present versus desired output is iteratively mininimized, cannot be used since the output here is a conditional probability which is not known. However, we can use the principle of maximum likelihood to obtain the optimal parameters $\underline{\theta}$. In the maximum likelihood paradigm, one uses the joint density of all grid node values of the training image given the network parameters as the conditional likelihood for the training image given the model. This joint density depends on the network parameters and can be maximized, yielding a maximum likelihood estimate for $\underline{\theta}$.

Application of the likelihood principle can intuitively be justified as follows and is symbolically depicted in Figure 8. We assume that our training image is one sample of an ensemble of images that is generated by a theoretical model defined by the set of parameters $\underline{\theta}$. This assumption need not be true, yet it is the basis of many methods of statistical inference. Such a model can be a Metropolis-Hastings sampler using a neural network to model the conditional distribution. The question now arises about where the training image should fit into the distribution of images defined by the model, see Figure 8. It is argued that the training image should be situated somewhere in the middle of that distribution since among all images it is the most likely to occur, hence the maximum likelihood principle. Note that in the likelihood principle we do not use the actual conditioning data, we do use though the training image.

Suppose, we have a training image consisting of the set of pixel values $\mathbf{y}^{(o)} = \{y_i^{(o)}, i = 1, \ldots, N\}$. Maximizing the likelihood of drawing the specific training image $\mathbf{y}^{(o)}$ requires knowledge of the joint density $f(\mathbf{y}|\underline{\theta})$ for any set $\mathbf{y}$ of all N pixel values given any choice for the parameters $\underline{\theta}$. Although this joint density can in theory be calculated from knowledge of the local cpdf's $f(y_i|\mathbf{x}_i, \underline{\theta})$, that calculation is not practical (see Caers, 1998). Thus we need to compromise and replace the joint density by, for example, the product of all N known local cpdf's $f(y_i|\mathbf{x}_i, \underline{\theta})$. More precisely, the set of parameters $\widehat{\underline{\theta}}$ retained is that which maximizes the pseudo-likelihood defined as (see Besag, 1975; Gidas, 1993)

$$\ell(\underline{\theta}, \mathbf{D}) = \prod_{i=1}^{N} f(y_i^{(o)}|\underline{\theta}, \mathbf{x}_i^{(o)}) \tag{13}$$

Expression (13) is called the "likeliness" of the training image $\mathbf{D}$ constituted by the N observed pixel values $\{y_i^{(o)}, i = 1, \ldots, N\}$. It can be shown that in many cases, maximizing the likeliness yields a consistent estimate of the network parameters. Actually, instead of maximizing the likeliness, we will minimize minus the log-likeliness, defined as the error function:

$$Er(\underline{\theta}, \mathbf{D}) = -\log(\ell(\underline{\theta}, \mathbf{D})) = -\sum_{i=1}^{N} \log(f(y_i^{(o)} | \underline{\theta}, \mathbf{x}_i^{(o)})), \quad (14)$$

where Er stands for error function.

Many methods could be considered to minimize this error-function, such as steepest descent or conjugated gradient methods. However, these usually take a lot of iterations, rendering the training procedure slow and tedious. Also there is a risk to end up in a local minimum. In Caers and Journel (1998), an iterative minimization of error function (14) is presented based on the EM-algorithm (Expectation-Maximization). It is important to note that the numbers K_1 and K_2 are fixed a priori, they are not determined through our optimization procedure.

Obtaining the essential features of an image

The neural network architecture is defined by the numbers (K_1, K_2) of internal nodes, the input, output and how all these are linked. Training on an image is performed through maximization of the error function (14) through iterative EM-steps. One cannot however simply fit the data extracted from an image using the EM-algorithm, since the fitting error will gradually decrease and the neural network may end up fitting the training data exactly beyond their essential and exportable spatial features. Indeed, by using enough nodes (K_1, K_2 large) one could fit exactly all training patterns. A way to overcome this problem is to stop the training early, also denoted in neural language as *early stopping*. However, when to stop is not clear.

Consider a second image, or if not available, we divide up our training image such that an independent set can be obtained. The idea is to evaluate the same likelihood error function on the second cross-validation set using the parameters fitted on the first training set. We define then a second error function which is calculated at each step of the EM-algorithm

$$Er_{cv}(\underline{\theta}_{tr}, \mathbf{D}^{cv}) = -\log(\ell_{cv}(\underline{\theta}_{tr}, \mathbf{D}^{cv})) = -\sum_{i=1}^{N} \log(f(y_i^{cv} | \underline{\theta}_{tr}, \mathbf{x}_i^{cv})) \quad (15)$$

where y_i^{cv} is the cross-validation data. Note that the cross-validation error is calculated exclusively with the cross-validation data $\mathbf{D}^{cv}$ but using the training parameter values $\underline{\theta}_{tr}$. The cross-validation data set is *not* used to determine the parameters $\underline{\theta}_{tr}$ in the minimization, yet the cross-validation error (15) it is updated at each step of the EM-algorithm. The cross validation error in (15) will tend to have a minimum as the EM-algorithm proceeds, see Figure 9. This can be understood by viewing the training with maximum likelihood as a result of two opposing tendencies. On one hand, the goodness of fit of the training patterns by the neural net will increase as the likelihood is maximized, on the other hand the amount of learning will decrease.

The cross-validation error will tend to decrease at first as the network learns the essential features of the first training image but will increase as the network starts overfitting the first training data modeling features which are not shared by the second cross-validation image. We will show examples of training and cross-validation eror functions in the examples.

Example training images

A range of example training images (constructed images and real outcrop) is shown in order to assess the performance of the stochastic simulation method proposed.

Sand dunes. A first image (see Figure 10) selected is a digitized section of aeolian sandstone images with black and white pixels. The black pixels correspond to the fine sediments with good horizontal connectivity. The white pixels correspond to coarser higher permeability sediments. The most important feature of this image is the curvilinear shape of the fine sediments which would act as vertical flow barriers. Such curved shapes are difficult to reproduce using traditional two-point simulation algorithms. The image is scanned with the template shown in Figure 1 (left). Due to the small size of the image, early stopping (after 60 iterations of the EM-algorithm) was preferred to cross-validation. The neural network trained contains 20 nodes in the first and 10 nodes in the second hidden layer. Figure 10 shows three conditional simulations: the connectivity of black pixels with their curvilinear feature is reasonably reproduced.

Walker Lake data set. The Walker lake data set is a set of 78000 (260 × 300) topographic measurements (altitudes in feet) of a mountain range near Walker Lake. Figure 11 gives the exhaustive data set which features the mountain ridges and flat salt lakes in the valleys. A normal score transform of the data was taken. To obtain a training and a cross-validation set we split the image in two halfs (bottom and top). The aim of this study is to produce unconditional simulations on small 130 × 150 grid and conditional simulations on a larger 260 × 300 grid. To be able to reproduce the long range structure along the NS-direction we used the multiple grid approach. For conditional simulation, three successive grids were retained of size 65 × 75, 130 × 150 and 260 × 300. For each grid a neural network was trained with 20 nodes in the first and 10 nodes in the second hidden layer. For each of these grids we used the same template, see Figure 1 (left). 390 regularly gridded sample locations (0.5 % of the total amount of pixels) were selected on the reference Walker Lake data set to provide local conditioning data. Figure 11 shows two resulting conditional simulations. The large scale feature of mountain versus lake area is reproduced, whereas smaller details within the mountain are not. Two unconditional simulations on a 130 × 150 grid are also shown in Figure 11. The unconditional simulations reproduce the transition between mountain and lake area characteristic of the exhaustive data set.

Channels. Fluvial reservoirs are characterized by sinuous sand-filled channels within a background of mudstone. A realistic modeling of the curvilinear aspect of such channels is critical for reliable connectivity assessment and for flow simulation. The modeling of such channels usually occurs within well-defined stratigraphic horizons. An object-based method (fluvsim, Deutsch and Wang, 1996) was used to generate a training and cross-validation image, see Figure 12. The corresponding petrophysical properties were generated as a random texture within both the mudstone and the channel. The training and cross-validation images clearly exhibit long-range structure due to the channeling. To reproduce such long-range structures, we must simulate on multiple grids. Three grids were selected of sizes 25 × 25, 50 × 50 and 100 × 100. For each of these grids we used the same template, see Figure 1 (left). For each grid, a neural network is learned from the scanned data using the corresponding template. In each case we opted for a net with 30 nodes in the first and 15 nodes in the second hidden layer. Figure 12 shows two unconditional simulations and two conditional simulations, each with a different amount of conditioning data, in order to assess the influence of conditioning data on the location of the channels. When using conditioning data, the meandering and thickness of the channels is better reproduced. The conditioning data locations were taken randomly from the training image.

Conclusions

In this paper we propose a stochastic simulation method based on sequential modeling of local conditional distributions (cpdf) using neural networks. The conditional distribution of a set of unknown variables given a set of known variables is at the root of many statistical prediction algorithms and the methology developed here carries over to other estimation problems, not necessarily spatial ones. The key aspect of the methodology proposed consists of integrating into the cpdf's, right from the start, essential textures read from training images. This approach is markedly different from most geostatistical method where the texture is embedded into the random function model (such as Multi-Gaussian or multiple indicator models) which borrows from reality only 2-point statistics. It is also different from the simulated annealing approach where an objective function is defined to impose reproduction of various multi-point statistics scanned from the training image. Simulated annealing does not provided a method to *decide* which multi-point statistics are essential and thus should be retained in the objective function. The use of a cross-validation data set allowed the neural network approach to avoid overfitting the multi-point patterns scanned from the training image. In case of simulated annealing, if too many multi-point statistics borrowed from the training image are imposed, the training image would be reproduced exactly, which is never the ultimate goal. The neural network approach allows for data expansion, a concept that is of critical importance in any imaging technique within the earth sciences.

The proposed method does not involve any of the classical steps of a geostatistical study such as kriging, modeling of variograms or making univariate transforms, nor does it involve any kind of consistency condition and correction. At the same time this simplification entails its major disadvantage: we do not know what textures are being borrowed from the training images. From the examples it appears that the neural network approach has the capability of reproducing curvilinear features, fractures or repetitive patterns.

Many practical issues still need to be addressed: how large should the template(s) be ? How many nodes should each hidden layer in the net contain ? When should we stop the visit of the whole image in the Metropolis-Hastings sampler? How can we speed up the learning phase ? Many of the answers lie in a better understanding of the inner working of the neural network and how the two hidden layer interact with each other during learning and generalization phases.

Nomenclature

y	=	value at any node
y_i	=	value at node i, Image has N nodes.
$\mathbf{y}$	=	values at all nodes = one image
$\mathbf{x}$	=	set of neighbourhood data in template centrec
x_j	=	one specific neighbourhood datum in any tem
$\mathbf{x}_i$	=	set of neighbourhood data in template centrec
$\underline{\theta}$	=	set of parameters
f	=	density (conditional, marginal or joint)
F	=	cumulative distribution (conditional, margina
K_2	=	number of neural nodes in second hidden lay
K_1	=	number of neural nodes in first hidden layer
T, S	=	non-linear functions
$\mathbf{u}$	=	set of parameters feeding in the first hidden l
$\mathbf{w}$	=	set of parameters feeding in the second layer
v_k	=	scale parameters of the radial basis function
o_k	=	weight into the output layer
ℓ	=	likeliness
Er	=	error function

Acknowledgements

The authors would like to acknowledge the Stanford Center for Reservoir Forecasting, the BAEF (Belgian American Educational Foundation) and the NATO for support of this research.

References

1. BESAG, J., 1974. Spatial interaction and the statistical interaction of lattice systems (with discussion). *J. Roy. Stat. Soc. B*, 36, 192–236.

2. BESAG, J., 1975. Statistical interaction of non-lattice data. *The Statistician*, 24, 179–195.

3. BESAG, J., GREEN, P., HIGDON, D. AND MENGERSEN, K., 1995. Bayesian computation and stochastic systems (with discussion). *Statistical Science*, 10, 3–66.

4. BISHOP C.M, 1995. Neural networks for pattern recognition. Oxford University Press.

5. DEMPSTER, A.P., LAIRD, N.M. AND RUBIN, D.B., 1977. Maximum likelihood from incomplete data via the EM-algorithm (with discussion). *J. Roy. Stat. Soc.*, B, 39, 1–38.

6. DEUTSCH, C., 1992. Annealing techniques applied to reservoir modeling and the integration of geological and engineering (well test) data. Ph.D. thesis, Department of Petroleum Engineering, Stanford University, Stanford.

7. DEUTSCH, C., AND WANG, L., 1996. Hierarchical object-based stochastic modeling of fluvial reservoirs. *Math. Geol.*, 28, 857–880.

8. DEUTSCH, C., AND JOURNEL, A.G., 1997. GSLIB: Geostatistical Software Library. Oxford University Press.

9. CAERS, J., 1998. Stochastic simulation using neural networks. 11th Annual report, Stanford Center for Reservoir Forecasting, Stanford, USA, 66p.

10. CAERS, J., AND JOURNEL, A.G., 1998. A neural network approach to stochastic simulation. GOCAD ENSMSP meeting, Nancy, France, June 4-5.

11. GUARDIANO, F. AND SRIVASTAVA, M., 1992. Borrowing complex geometries from training images: the extended normal equations algorithm. *SCRF Report*, 1992.

12. IGELNIK, B. AND PAO, Y.H., 1995. Stochastic choice of basis functions in adaptive functional approximation and the Functional Link net. *IEEE Trans. on Neural Networks*, 6, 1320-1329.

13. GOODMAN J. AND SOKAL A.D., 1989. Multigrid Monte Carlo method. Conceptual foundations. *Physical Revue D, Particles and Fields*, 40, 2035-2071.

14. GOOVAERTS, P., 1997. Geostatistics for natural resources evaluation. Oxford University Press.

15. HUSMEIER, D. AND TAYLOR, J.G., 1997. Neural networks for predicting conditional probability densities: improved training scheme combining EM and RVFL. King's College, Department of Mathematics, London. Pre-print KCL-MTH-97-47.

16. JORDAN, M.I. AND JACOBS, R.A., 1994. Hierachical mixtures of experts and the EM-algorithm. *Neural Computation*, 6, 181–214.

17. JOURNEL, A.G. AND ALABERT, F., 1989. Non-Gaussian data expansion in the earth sciences. *Terra Nova*, 1, 123–134.

18. HASTINGS, W.K., 1970. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57, 97–109.

19. PRESS, W.H., TEUKOLSKY, S.A., VETTERLING, W.T. AND FLANNERY, B.P., 1996. Numerical recipes in Fortran 90, volume 2 of Fortran numerical recipes. Cambridge University Press.

20. RIPLEY, B.D., 1987. Stochastic simulation. Wiley, New York.

21. RIPLEY, B.D., 1996. Pattern recognition and neural networks. Cambridge University Press.

22. SRIVASTAVA, M., 1992. Iterative methods for spatial simulation. *SCRF Report*, 1992.

23. TJELMELAND, H., 1996. Stochastic models in reservoir characterization and markov random fields for compact objects. Ph.D. Thesis, Department of Mathematical Sciences, Norwegian University of Science and Technology, Trondheim.

24. TRAN, T., 1994. Improving variogram reproduction on dense simulation grids. *Computers and Geosciences*, 20, 1161–1168.

25. WANG, L., 1995. Modeling complex reservoir geometries with multi-point statistics. *SCRF Report*, 1995.

26. XU, W. AND JOURNEL, A.G., 1994. DSSIM: A general sequential simulation algorithm. *SCRF report*, 1994.

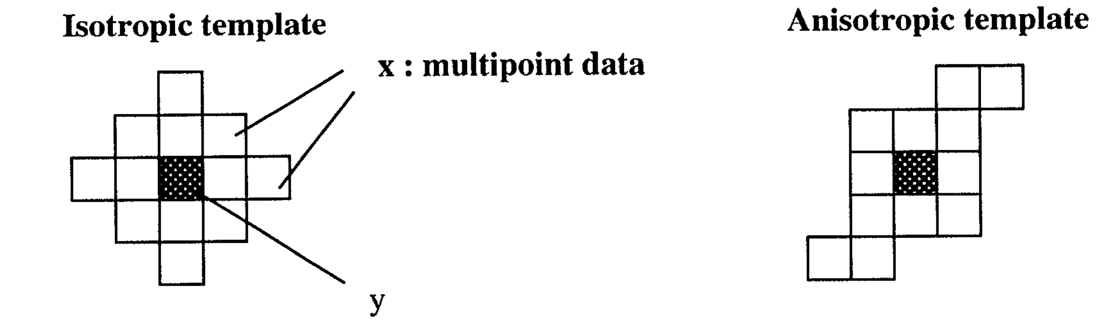
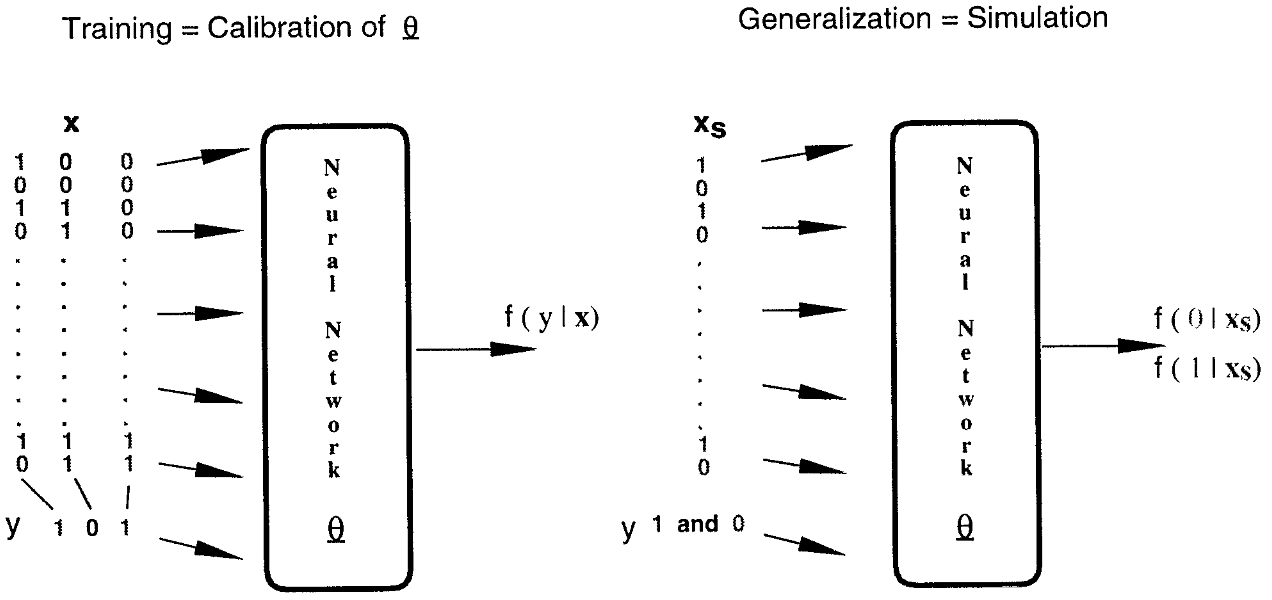


Figure 1: *Defining a neighbourhood or scanning template*



y : value of visited node (could be continuous)

x : value of neighbors of visited node

$\underline{\theta}$: network parameters

f : conditional distribution

Figure 2: *General network configuration for stochastic simulation for a binary case.*

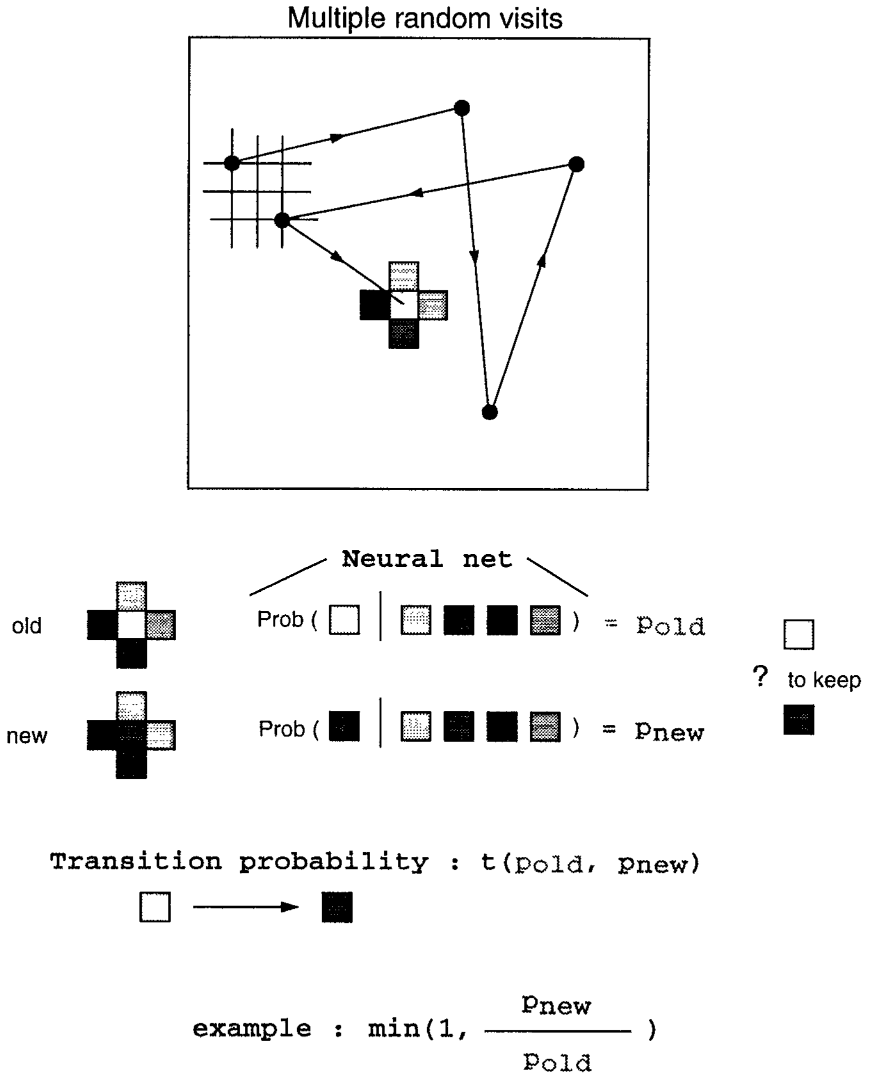

Multiple random visits
Neural net
old
new
Prob (□ | ▨ ■ ■ ▨) = Pold
Prob (■ | ▨ ■ ■ ▨) = Pnew
? to keep
Transition probability : t(Pold, Pnew)
example : min(1, Pnew / Pold)

Figure 3: *Metropolis sampling : the field is initially random. Each node is visited and updated according to the ratio p_{new}/p_{old}.*

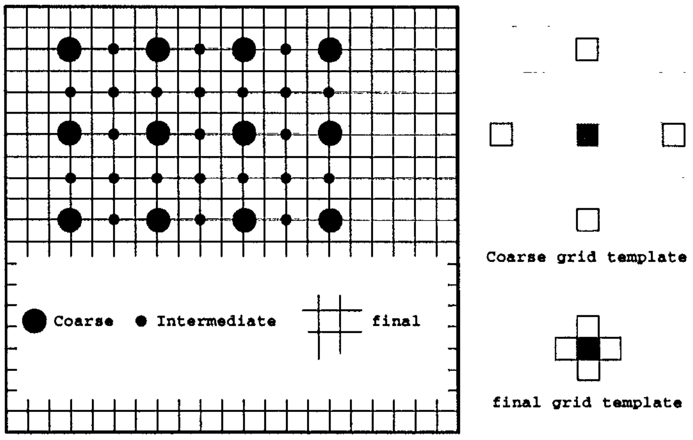


Figure 4: *Using multiple grids in stochastic simulation.*

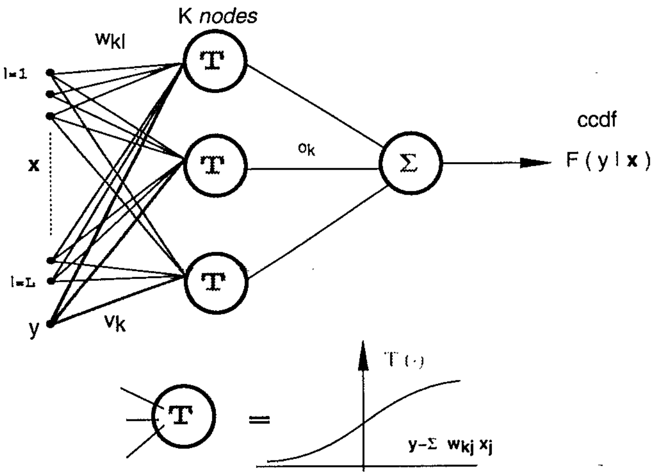


Figure 5: *Single hidden layer network.*

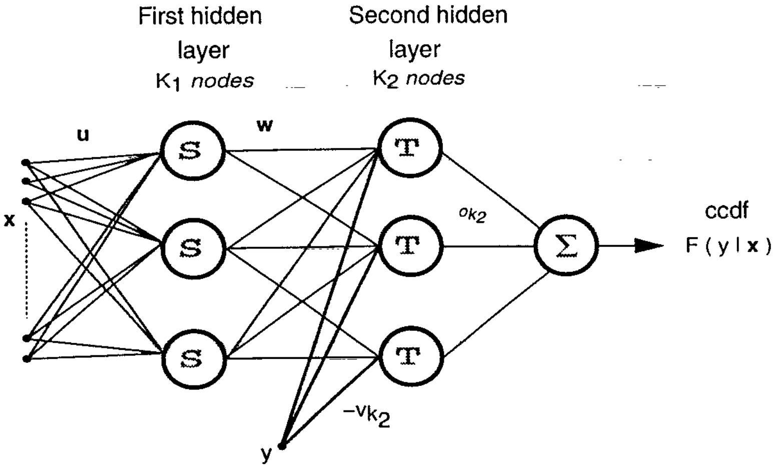


Figure 6: *Adding a second layer.*

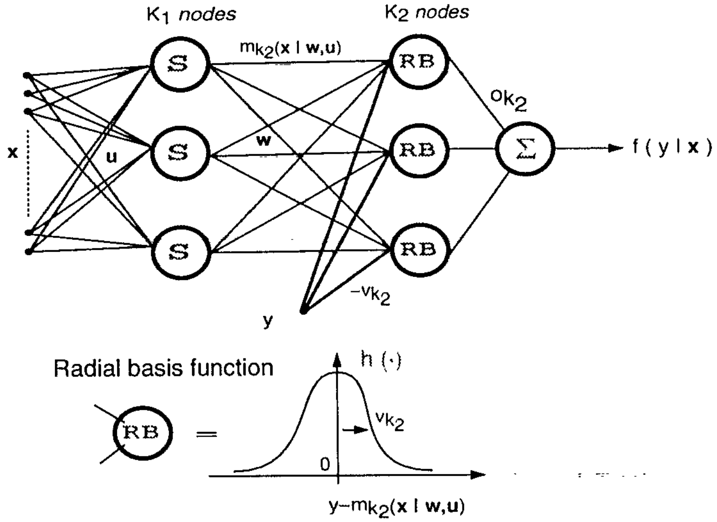


Figure 7: *Neural network that outputs a density instead of a cdf.*

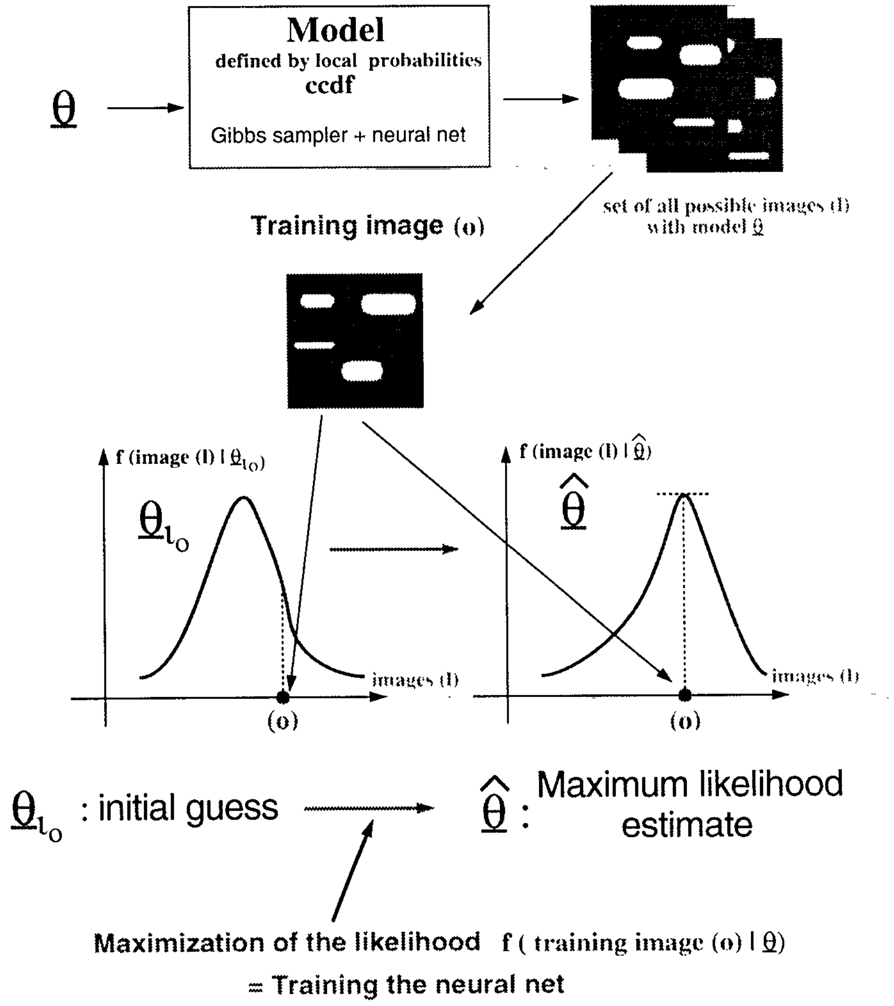

θ
Model
defined by local probabilities
ccdf
Gibbs sampler + neural net
set of all possible images (I)
with model θ
Training image (o)
f (image (I) | θ_{l0})
θ_{l0}
images (I)
(o)
f (image (I) | θ̂)
θ̂
images (I)
(o)
θ_{l0} : initial guess
θ̂ : Maximum likelihood estimate
Maximization of the likelihood f (training image (o) | θ)
= Training the neural net

Figure 8: *Likelihood principle in spatial statistics.*

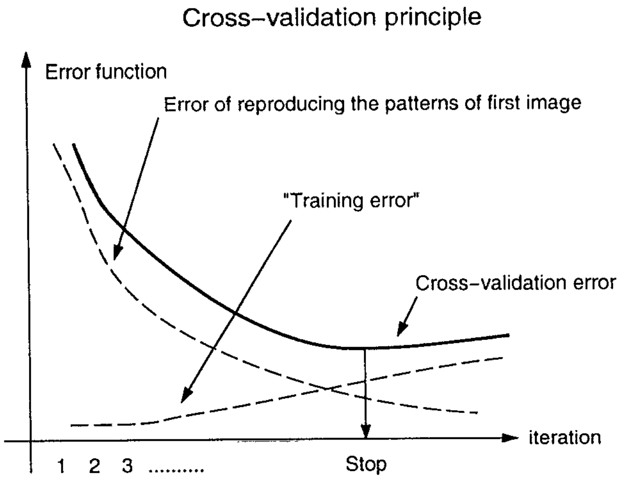


Figure 9: *Training error and cross-validation error.*

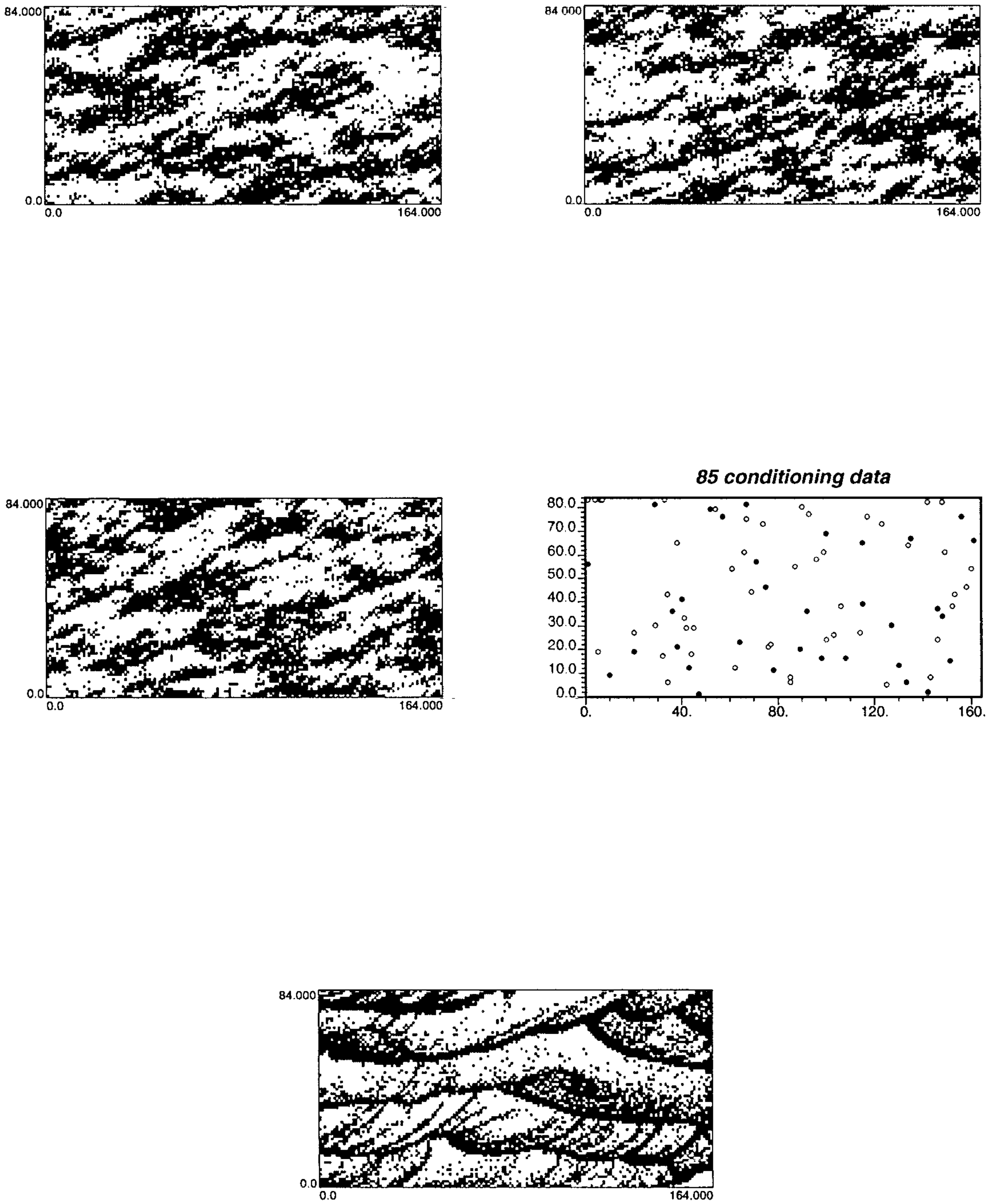


Figure 10: *Dunes: results.*

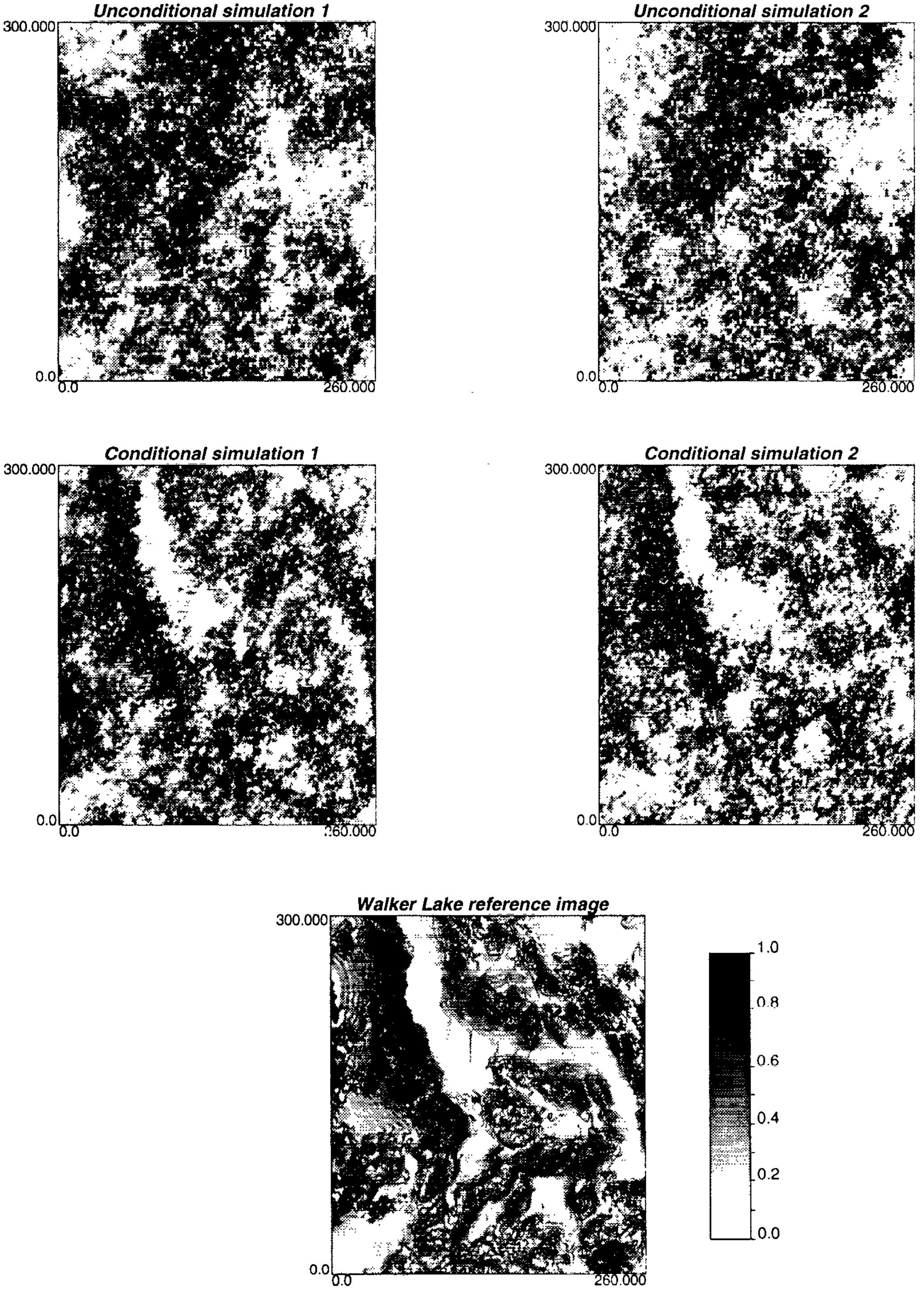


Figure 11: *Walker Lake: simulation results*

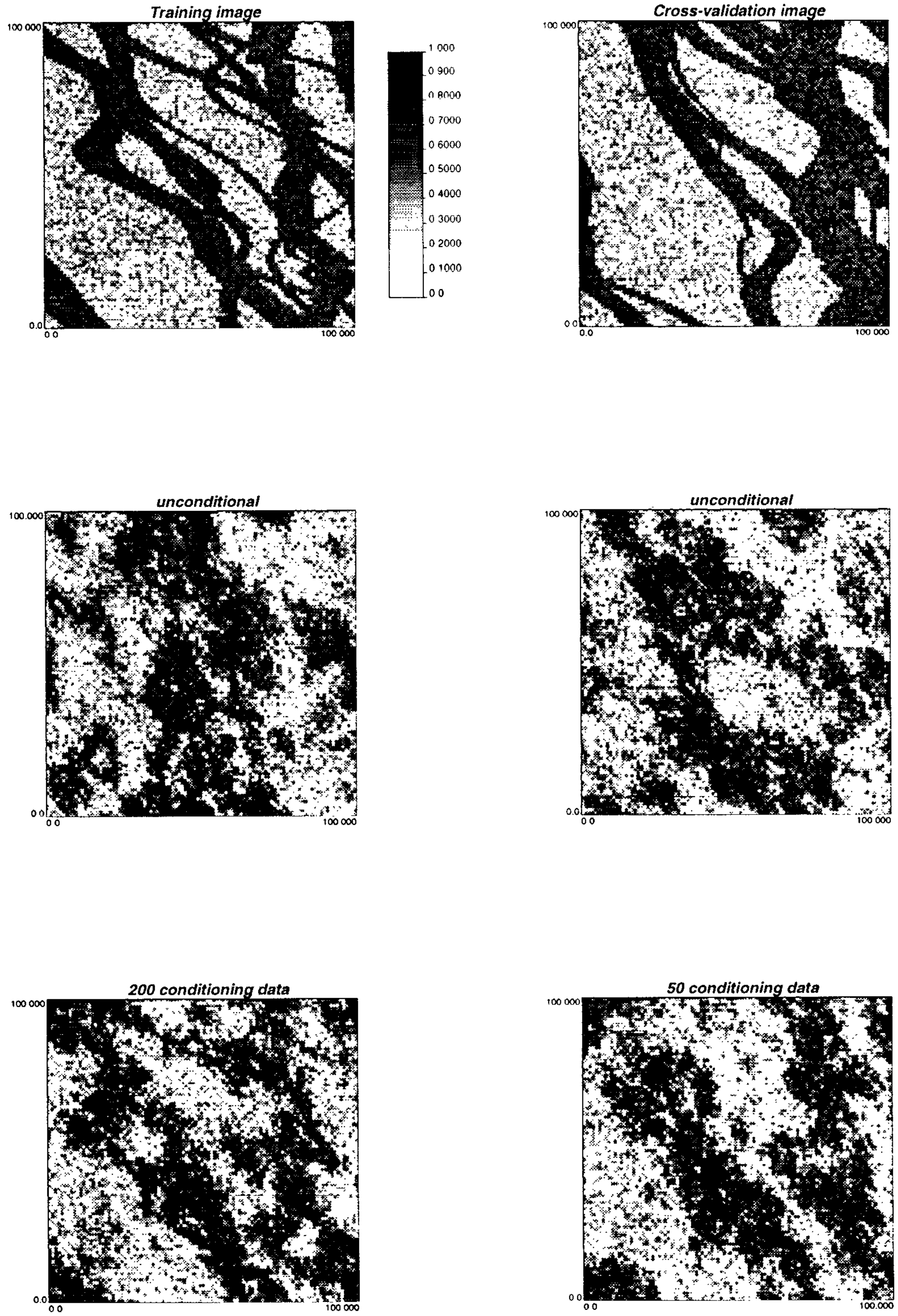


Figure 12: *Channel simulation results.*