Geological modeling using Recursive Convolutional Neural Networks

Sebastián Ávalos†, Julián M. Ortiz‡*

†The Robert M. Buchan Department of Mining, Queen’s University, Canada

Abstract

Resource models are constrained by the extent of geological units that often depends on lithology, alteration and mineralization. Three dimensional geological units must be built from scarce information and limited understanding about the geological setting. In this work, a new technique for Multiple-Point Geostatistics simulation based on a Recursive Convolutional Neural Network approach (RCNN) is presented. The method requires conditioning data and a training image. The training image is used to learn geological patterns and simulation is carried out by using the previous conditioning data.

A lithological modeling process is carried out in a synthetical 3D surface-based model lithology to demonstrate the method. Spatial metrics are used to measure performance and characterize the RCNN properties. Also, strengths and weaknesses of the methodology are discussed by briefly reviewing the theoretical perspective and looking into some of its practical aspects.

Keywords: Geostatistics, Multiple-point statistics, Training Image, Categorical Variable, Deep Learning, Convolutional Neural Networks

1. Introduction

Multiple-point statistics (MPS) simulation methods provide a framework for spatial modeling of variables in geoscience problems. The reproduction of complex patterns and spatial uncertainty assessment are their main goals. They can be applied for continuous or categorical variables and the methods can be extended to the multivariate case, suiting applications in diverse fields (oil and gas, hydrological reservoirs, ore deposits).

MPS simulation has seen an explosive development since Strebelle’s seminal paper in 2002 (Strebelle, 2002). Many different approaches have been proposed (Arpat and Caers, 2007; Mariethoz G, 2010; Parra and Ortiz, 2011) and some have reached industrial applications (Mariethoz G, 2010). The methods associated to MPS are intimately related to image and texture analysis (Daly, 2004; Parra and Ortiz, 2011; Zahner et al., 2015). Excellent reviews of multiple-point simulation methods are available (Tahmasebi, 2018) and further theoretical and practical details can be found (Mariethoz and Caers, 2014).

This paper is motivated by the possibilities offered by deep learning methods, particularly Convolutional Neural Networks (CNNs) (LeCun et al., 1998) which have shown extraordinary performances in image analysis, including classification, reconstruction, segmentation, labeling and more (Dosovitskiy et al., 2015; Eilertsen et al., 2017; Gatys et al., 2016; He et al., 2016). The main aim is to create a bridge between CNNs and MPS. This is done by proposing a Recursive Convolutional Neural Network (RCNN) approach that enables the architecture to learn features of an underlying phenomenon from a training image and then perform simulations on different domains reproducing the phenomenon complexity, conditioned by hard data at sample locations.

2. Convolutional Neural Networks

CNNs belong to a set of deep learning architectures with great abilities on image analysis. Their features extraction properties, from raw images, have made them suitable for a variety of image analysis applications.
Connecting neural networks with MPS is not new (Caers and Journel, 1998) however deep learning techniques have only recently been connected to MPS (Laloy et al., 2018).

CNNs have the main feature of sharing inner parameters across the network, which leads to architectural properties of scale, shift and distortion invariance, making them a powerful tool for image feature extraction. Those properties mean that regardless where and how a specific raw feature appears in the image, a suitable and well-trained CNN is able to capture that feature. Once features are extracted, images can be classified, segmented or even reconstructed.

CNNs are composed by a feature extraction block and a classification block (Fig. 1). The first block receives a grid-like topology input and extracts representative features in a hierarchical manner. The second block receives the top hierarchical feature and delivers a final matrix of prediction.

Once the inner process of a CNN is understood, the inner process of a CNN is understood, a three dimensional image grid-like topology is used to depict the inner process of a CNN. The building blocks of the feature extraction section are: input image $X$, filters $W_m$, bias vectors $b_m$ and hidden layers $H_m$. Then, $H_m$ results of convolving $W_m$ over $H_{m-1}$, adding a bias vector $b_m$ and then passing the temporary result through (1) a Batch-Normalization function, (2) a non-linear activation function and (3) a pooling function as:

$$H_m = \begin{cases} 
    X & \text{if } m = 0 \\
    \text{pool}(g(BN(W_mH_{m-1} + b_m))) & \text{if } 0 < m \leq M 
\end{cases} \quad (1)$$

A convolution is understood as the process of sliding the filter over the input while performing the sum of an element-wise multiplication between the filter values and the corresponding section of the input. Activation functions, $g(\cdot)$, a set of non-linear transformations, are required to suitably extract features. They are applied element-wise over each feature map in the hidden layers. Some activation functions are Sigmoid, TanH and ReLU. Each activation function has an active zone, that is, an interval where the derivative of the function is not zero.

The Batch Normalization function (BN) (Ioffe and Szegedy, 2015) normalizes $(W_mH_{m-1} + b_m)$ before passing it through the activation function by subtracting the mean and dividing by their respective standard deviation over each feature map. This process has three main properties: (1) it avoids vanishing gradient problems during training, by adjusting the input values to the active zone; (2) it accelerates the training process; and (3) it serves as a regularization method.

The pooling function, $\text{pool}(\cdot)$, seeks to reduce the size of the representative hidden layer by taking small regions of each feature map. Several functions exist including the widely used max pooling, and others such as average pooling, min pooling and L2-norm pooling. They are usually applied to go from a moving window of $(2, 2, 2)$ to a single value $(1, 1, 1)$. Max pooling keeps only the maximum value among the nodes in
the small region. This pooling function significantly reduces the number of learning parameters, improving statistical efficiency and reducing the memory storage consumption (Goodfellow et al., 2016).

The classification section is composed by a set of feedforward networks (Hornik et al., 1989) whose building blocks are: weight matrices \( W_{FC_f} \), bias vectors \( b_f \) and hidden fully connected layers \( FC_f \). Then, \( FC_f \) results of a matrix multiplication between \( W_{FC_f} \) and \( FC_{f-1} \), adding a bias vector \( b_f \) and then passing the temporary result only by (1) a Batch-Normalization function and (2) an activation function as:

\[
FC_f = \begin{cases} V \text{ec}(H_f) & \text{if } f = 0 \\ g(BN(W_{FC_f} FC_{f-1} + b_f)) & \text{if } 0 < f \leq F \end{cases}
\]  

(2)

The first feedforward network \( FC_0 \) corresponds to a vector-representation \( V \text{ec}(H_M) \) of the last hidden layer in the feature extraction block. The BN and activation functions act exactly as presented before.

The final layer \( FC_F \) must have the same dimensions as the number of categories when categorical distributions are required. Let \( K \) be the number of categories, \( n_{FC}^{(F)} = K \), and \( s_k \in \{s_1, ..., s_K\} \) the non-normalized conditional probability of each class in \( FC_F \). By passing \( FC_F \) through a softmax function:

\[
\hat{p}(k) = \frac{e^{s_k}}{\sum_{c=1}^{s_K} e^{s_c}}
\]  

(3)

the expected conditional probabilities for each class are obtained. Then, the final predicted state \( \hat{s}_k \) is inferred by taking the position \( k \) which contains the highest probability, that is, using the argmax function.

Inner parameters \( \Theta = \{W, b\} \) are initialized as random values from a normal gaussian truncated function. Optimum \( \Theta \) values are obtained as result of training the CNN by applying the Adam optimizer (Kingma and Ba, 2014) algorithm. The loss functions to minimize during training (Eq. (4)) for categorical variables, considering the predicted probability of each class (Eq. (3)) and the real probability \( p(k) \), is the negative log-likelihood of the Cross Entropy (CE):

\[
CE : L(\Theta) = -\sum_{k=1}^{K} \left( \log \hat{p}(k) \right) \cdot p(k)
\]  

(4)

3. The Recursive Convolutional Neural Network approach

Let \( SG \) and \( IP \) be the search grid and inner pattern, whose dimensions are odd positive integers to ensure the existence of a collocated center (Fig. 2. Left). In MPS terms, the SG is the neighbourhood (template) that contains the data event \( d_n \) (conditioning data).

The main idea is that a first CNN1 is trained to predict a conditional cumulative distribution function (ccdf) of each location (node) inside an IP by receiving, as input, a data event \( (D) \) from the TI and assigning them to all \( d_n \) embedded in a SG (Fig. 2. Left). Then a second CNN2 is trained to predict the same ccdf but now knowing the same input and the IP predicted by the previous CNN1. The third CNN3 is trained using same input and the two previously predicted IP by CNN2 and CNN1. The process can be done again (Fig. 2. Right) giving the recursiveness property to the RCNN approach. The idea behind the recursiveness is the improvement on results quality by taking into account the previously simulated information.

The RCNN training algorithm starts by creating \( N+1 \) simulation grids (domains), namely \( D^0, D^1, ..., D^N \), whose dimensions are equal to the TI and then extracting a percentage of hard data (\( perDC \)) randomly from the TI and assigning them to all \( D^i \). The simulation process, explained later, is carried. Once all \( D^i \) are simulated, every pair-list database (DB') of input-output as \( X' \leftrightarrow IP^{Real} \) is created in order to train the respective CNNi. This is done by extracting \( \forall i \) SG\( \left(D^i\right) \) in order to create \( X' \) and the respective collocated IP\( ^{Real} \).

Using the CE of Eq. (4), the loss function of each CNNi is:

\[
L_i(\Theta^i) = -\sum_{a=1}^{IP_a} \sum_{b=1}^{IP_b} \sum_{c=1}^{IP_c} \sum_{k=1}^{K} \left( \log \hat{p}(k | RS_{a,b,c;}(X',\Theta^i)) \right) \cdot p(k | IP^{Real}_{a,b,c;}(X'))
\]  

(5)
Eq. (5) represents the sum of all CE in IP between the predicted conditional probability and the real probability $p(k)$ from IP$^{\text{Real}}$. Any location $(a, b, c)$, whose category in IP$^{\text{Real}}$ is $k$, has a real probability vector of $[0, 1, 0]$ with 1 at the $k$-position, so Eq. (5) is highly simplified when calculated.

Each CNN$^i$ receives a mini-batch of $m$ samples from DB$^i$ and estimates the gradient with respect to the loss function, $\nabla_{\Theta} L_i(\Theta^i) \approx \nabla_{\Theta} \frac{1}{m} \sum_{k=1}^{m} L_i(\Theta^i; X_k^i)$, and performs a parameter update $\Theta^i_t \leftarrow \Theta^i_t + \Delta \Theta^i$ by inferring the updated direction $\Delta \Theta^i$ with respect to the gradient $\nabla_{\Theta} L_i(\Theta^i)$ by using the Adam Optimizer (Kingma and Ba, 2014). After all CNNs have been trained by all mini-batches, the first epoch is completed. The entire process is carried out again as many times as the number of epochs previously defined, or until the entire network shows signs of convergence.

The RCNN simulation process begins by migrating the conditioning data to the closest node at each $D^i$. First $D^1$ is fully simulated, then $D^2$ until $D^N$ is completely informed. The sequence of nodes to be simulated at each $D$ is given by the same random path, previously defined. Following that path and at unknown locations over $D^i$, the collocated SG associated to $D^{i-1}, D^{i-2}, ..., D^0$ are extracted, concatenated and fed into the CNN$^i$ to obtain the IP$^i$. Then, instead of freezing all IP$^i$ values in $D^i$, only a random percentage of them are selected and frozen at unknown locations. Particularly, the unknown center is always simulated. The percentage of random values used across this paper is 50%.

4. Experiments

A three dimensional binary lithological surface-based model is synthetically built with an extension of $100 \times 100 \times 50$ pixels. Categories are indexed as $(s_0 : 0)$ unknown category, $(s_1 : 1)$ category 1 and $(s_2 : 2)$ category 2. The field is split in 4 sectors of $50 \times 50 \times 50$. The first one is used as training image to train the RCNN and the other three as ground truths, named $S1$, $S2$ and $S3$, as shown in Fig. 3.

Fig. 3: Training image and three ground truth. All of them with sizes $50 \times 50 \times 50$. 
At each sector \( S1, S2 \) and \( S3 \) two sets of vertical drill-holes samples are randomly selected and used as conditioning data, one with 5\% and the other with 2\%. For instance, Fig. 4 shows the selected conditioning data over \( S3 \).

![Randomly conditioning data (drill-holes) at S3 with 2 % (left) and 5% (right).](image)

The RCNN architecture consists of four identical nested CNNs with four hidden layers \((M = 4)\), three fully connected layers \((F = 3)\) and two categories \((K = 2)\). Two metrics are used to measure the quality of reproducing the spatial complexity of the underlying phenomena:

- **Visualizations** One random realization, the \( E \)-type over a hundred realizations and the respective local variance map.
- **Variograms** Omni-horizontal and vertical experimental variograms over the TI, GT and each realization are compared, providing a measure of the two-points statistics.

5. Results

After training the RCNNs and performing 100 realizations at each of the three sectors with both sets of conditioning data, the following results are achieved.

- **Visualizations** From a visual point of view, the quality of lithological structure reproduction is similar between sectors.
  
  Fig. 5 shows visual results obtained over \( S2 \). Each realization honours the presence of structures with noise, connecting the conditioning data. Areas with clear absence of structure are also honoured. The \( E \)-type and VarMap help visualizing the predicted categories. Shapes and continuities are reasonably captured with 5\% of hard data but they become more diffuse with 2\% of conditioning data, specially in areas larger than the search grid where there are not conditioning data.

  Fig. 6 shows the \( E \)-types of each sector using 5\% of conditioning data and the associated Ground Truths. It shows the ability of the RCNN to predict the presence and absence of lithology at each sector honouring and using the information of the available hard data. The binary proportions are not imposed in the loss function and remain a challenge in the training of the RCNN.

- **Variograms** From a variographic perspective (Fig. 7), the RCNN is able to reproduce the two-point statistics illustrated in the shapes of realization-variograms compared with the training image and ground truths. As the distance increases, variograms from realizations fluctuate as the GT does rather than the fluctuations of TI-variogram. Also, reasonable variance are reached by the realization-variograms: \((0.17, 0.19)\) with 2\% and 5\% of conditioning data respectively, compared with the variance of 0.18 found in the TI/GT.
Fig. 5: Visualization of S2. (Left) one random realization, (center) $E$-type and (right) variance map, over 100 realizations. (Top) using 5% and (bottom) using 2% of conditioning data.

Fig. 6: Visualization of $E$-types (top) and Ground Truths (bottom) at S1 (left), S2 (center) and S3 (right) using 5% of conditioning data.
6. Conclusions

One of the hardest issues to accomplish by MPS techniques is their implementation and effectiveness in three dimensions. During the present work, the methodology of a 3D implementation of the proposed RCNN technique was presented, applying it into a lithological case where good results were achieved and measured, showing the benefits of bringing Deep Learning techniques in the MPS framework.

By means of visualizations and variograms the ability of RCNN to learn the spatial arrangement of lithological structures, by training, and the reproduction of it, by simulation, was displayed, measured and interpreted. Shapes and connectivities were captured using 5% and 2% of hard data.

Acknowledgments. The authors acknowledge the funding provided by the Natural Sciences and Engineering Council of Canada (NSERC), funding reference number RGPIN-2017-04200 and RGPAS-2017-507956, and the Computer and Geoscience Research Scholarship 2018. Special thanks to Honggeun Jo and Prof. Michael J. Pyrcz from the Texas Center for Geostatistics for providing the training image.

References

Arpat, G. B., Caers, J., 2007. Conditional simulation with patterns. Mathematical Geology 39 (2), 177–203.
Caers, J., Journel, A. G., 1998. Stochastic reservoir simulation using neural networks trained on outcrop data. In: SPE Annual Technical Conference and Exhibition. Society of Petroleum Engineers.
Daly, C., 2004. Higher Order Models using Entropy, Markov Random Fields and Sequential Simulation. In: Leuangthong, O., Deutsch, C. (Eds.), Geostatistics Banff 2004 (Quantitative Geology and Geostatistics, vol. 14). pp. 215–224.
Dosovitskiy, A., Tobias Springenberg, J., Brox, T., 2015. Learning to generate chairs with convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1538–1546.
Eilertsen, G., Kronander, J., Denes, G., Mantiluk, R. K., Unger, J., 2017. HDR image reconstruction from a single exposure using deep CNNs. ACM Transactions on Graphics (TOG) 36 (6), 178.
 Gatys, L. A., Ecker, A. S., Bethge, M., 2016. Image style transfer using convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2414–2423.
Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep learning. Vol. 1. MIT press Cambridge.
He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778.
Hornik, K., Stinchcombe, M., White, H., 1989. Multilayer feedforward networks are universal approximators. Neural networks 2 (5), 359–366.
Ioffe, S., Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167.
Kingma, D. P., Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
Lakay, E., Héraut, R., Jacques, D., Linde, N., 2018. Training-Image Based Geostatistical Inversion Using a Spatial Generative Adversarial Neural Network. Water Resources Research 54 (1), 381–406.
LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86 (11), 2278–2324.
Mariethoz, G., Caers, J., 2014. Multiple-point geostatistics: stochastic modeling with training images. John Wiley & Sons.
Mariethoz G, Renard P, S. J., 2010. The direct sampling method to perform multiple point geostatistical simulations. Water Resources Research 46 (11), w11536.
Parra, A., Ortiz, J. M., 2011. Adapting a texture synthesis algorithm for conditional multiple point geostatistical simulation. Stochastic Environmental Research and Risk Assessment 25 (8), 1101–1111.
Strebelle, S., 2002. Conditional simulation of complex geological structures using multiple point statistics. Mathematical geology 34 (1), 1–21.
Tahmasebi, P., 2018. Multiple Point Statistics: A Review. Handbook of Mathematical Geosciences, 613.
Zahner, T., Lochbühler, T., Mariethoz, G., Linde, N., 2015. Image synthesis with graph cuts: a fast model proposal mechanism in probabilistic inversion. Geophysical Journal International 204 (2), 1179–1190.