Using Generative Adversarial Networks as a Fast Forward Operator for Hydrogeological Inverse Problems

by Yasin Dagasan[1], Przemysław Juda[2], and Philippe Renard[2]

Abstract

Subsurface characterization using inverse techniques constitutes one of the fundamental elements of hydrogeological modeling applications. Available methods to solve inverse problems rely on a forward operator that predicts state variables for a given set of subsurface parameters. As the number of model parameters to be estimated increases, forward operations incur a significant computational demand. In this paper, we investigate the use of conditional generative adversarial networks (cGAN) as an emulator for the forward operator in the context of a hydrogeological inverse problem. We particularly investigate if the cGAN can be used to replace the forward operator used in the adaptive importance sampling method posterior population expansion (PoPEx) with reasonable accuracy and feasible computation requirement. The cGAN model trained on channelized geological structures has shown that the cGAN is able to reproduce the state variables corresponding to a certain parameter field. Hence, its integration in PoPEx yielded satisfactory results. In terms of the computational demand, the use of cGAN as a surrogate forward model reduces the required computational time up to 80% for the problem defined in the study. However, the training time required to create a model seems to be the major drawback of the method.

Introduction

Reconstruction of subsurface heterogeneities is a crucial step to make reliable predictions from groundwater flow modeling. Due to the lack of direct observations, inversion techniques are often required to identify the unknown parameters of the system using the observed state data. In hydrogeology, many approaches and solutions to the inverse problem have been proposed. Several review papers (Carrera 1988; de Marsily et al. 2000; Zhou et al. 2014; Linde et al. 2015) cover these methods in detail. An important aspect is that it is well known, since the work of Hadamard (1902), that inverse problems are generally ill-posed when they are framed in a deterministic manner: the solution may be nonunique, it may not exist, or it may be unstable. When the inverse problem is framed in a probabilistic manner (Tarantola and Valette 1982; Mosegaard and Tarantola 1995), these issues are not relevant any more. The aim is not to find a unique solution, but instead to obtain the probability distribution of the parameter values that is compatible with the observed state variables, the known physics, and the prior information about the parameters. In addition, the computational burden required to solve the inverse problem can be considered as a stumbling block. Most inverse modeling techniques are iterative and require a large number of forward model runs. Improvements to solve inverse problems often seek to address the above-mentioned issues.

Deep learning (DL) has gained momentum in recent years and could help to solve the inverse problem efficiently. Applications have shown the power of this technique in various fields (Collobert and Weston 2008; Hinton et al. 2012; Xu et al. 2014; He et al. 2016). It has lately been adopted in the geoscience community as well. As described by Marçais and De Dreuzy (2017), the main motivations to use DL techniques comprise mitigating computational costs, model reduction, category classification and uncertainty quantification. As such, several studies explored adopting DL in geosciences for low dimensional parametrisation (Laloy et al. 2017), inversion (Chan and Elsheikh 2017; Mosser et al. 2019), porous media reconstruction (Mosser et al. 2017, 2018a, 2018b) and generating realistic geological

[1]Corresponding author: Centre for Hydrogeology and Geothermics, University of Neuchâtel, Rue Emile-Argand 11, 2000 Neuchâtel, Switzerland; yasin.dagasan@graduate.curtin.edu.au

[2]Centre for Hydrogeology and Geothermics, University of Neuchâtel, Rue Emile-Argand 11, 2000, Neuchâtel, Switzerland

Article impact statement: In inverse problems, the most time-consuming step is to run the groundwater model. We show that neural networks can accelerate that step.

Received August 2019, accepted April 2020.
© 2020, National Ground Water Association.
doi: 10.1111/gwat.13005

models honoring physical measurements (Dupont et al. 2018). Particular implementations for hydrogeological problems include emulating computationnally expensive reactive transport models (Laloy and Jacques 2018), bidirectional mappings between the parameter space-model state space (Sun 2018) and surrogate modeling (Tripathy and Bilionis 2018; Zhu and Zabaras 2018; Mo et al. 2019a, 2019b).

The purpose of this paper is to investigate the feasibility of using a conditional generative adversarial network (cGAN) as a surrogate model for the forward operator in a hydrogeological probabilistic inversion problem. In distinction from previous surrogate modeling studies using DL (Tripathy and Bilionis 2018; Yang et al. 2018; Zhu and Zabaras 2018; Zhu et al. 2019; Mo et al. 2019a, 2019b), we assess the practical integration of a DL algorithm in the posterior population expansion (PoPEx) algorithm (Jäggli et al. 2018) to perform the inversion in a probabilistic manner. PoPEx is based on the principle of an adaptive importance sampling strategy (Bugallo et al. 2017) but designed specifically for the categorical inverse problem. PoPEx aims at identifying an ensemble of categorical fields representing, for example, the rock types—such as channels and lenses—within an aquifer such that the resulting models would reproduce the observations of state variables within an acceptable error range. In previous papers (Jäggli et al. 2017, 2018), it was shown that PoPEx was faster than other Markov Chain Monte Carlo methods to solve the inverse problem in the categorical case. However, as in most inversion algorithms, PoPEx needs a forward operator to compute the state values based on a given parameter field. This is the most time consuming part in the inversion procedure. In this paper, we evaluate the possibility to emulate and replace the forward operator with a cGAN. We propose a structure for the cGAN and evaluate the quality of the results when used in the inversion procedure. The tests are made using PoPEx, but the results are more general since the cGAN method could be used in many other inverse approaches.

The rest of the paper is organized as follows. Section "Methods" provides an overview of the inverse problem formulation, adaptive importance sampling algorithm and the cGAN methods. Section "Numerical Experiment" presents the problem setup used in the study and along with the cGAN architecture used. Details about the cGAN training, predictions and the use of the surrogate model in the inversion are given in Section "Results." The paper is concluded in Section "Discussion and Conclusion" with a discussion.

Methods

Inverse Problem Formulation

This section introduces the probabilistic inverse problem formulation following Tarantola (2005). The general aim is to identify the aquifer parameters from a set of state variable observations $\mathbf{d}^{obs}$ (e.g., hydraulic heads, contamination concentration, or discharge rates) and associated measurement errors. In a probabilistic framework, solving the inverse problem means identifying the posterior probability distribution of the model parameters. Using the terminology of Tarantola (2005), a *model* $\mathbf{m} = \{m_1, m_2, \ldots, m_n\}$ is a finite set of parameters which fully describes the physical system of interest. It can be, for example, a map of hydraulic conductivities.

In this paper, we consider the case of a categorical inverse problem: a model $\mathbf{m}$ represents a map in which each cell m_i can only take s possible different values corresponding for example to different rock types. This problem is difficult because most of the inverse methods, relying on partial derivatives or covariances, cannot handle discrete parameters properly (Linde et al. 2015). These models (or maps) can be generated using any geostatistical technique able to simulate categorical values. Each model is then a realization of a complex multidimensional and categorical prior probability distribution denoted $\rho(\mathbf{m})$.

Given a model $\mathbf{m}$, the flow response is computed using a forward operator $\mathbf{g}$ that solves the partial differential equations describing groundwater flow. This mapping allows to forecast the model responses $\mathbf{d} = \mathbf{g}(\mathbf{m})$ at the observation locations. The calculated values $\mathbf{d}$ are used to assess how well a given model $\mathbf{m}$ is able to reproduce the observed data $\mathbf{d}^{obs}$. This is quantified through the likelihood function $L(\mathbf{m})$ that is constructed by assuming a given distribution of errors.

The solution of the inverse problem is then expressed by characterizing the posterior probability $\sigma(\mathbf{m})$ as follows:

$$\sigma(\mathbf{m}) = c\rho(\mathbf{m})L(\mathbf{m}). \tag{1}$$

In the above equation, c denotes a normalization constant. Neither $\rho(\mathbf{m})$ nor $L(\mathbf{m})$ have a fully analytic expression and therefore most methods rely on Monte Carlo techniques (Mosegaard and Tarantola 1995).

Posterior Population Expansion

A detailed presentation of PoPEx is given in Jäggli et al. (2018). Here, we summarize the approach. The method is a modified adaptive importance sampler (Bugallo et al. 2017). The models $\mathbf{m}_i$ are generated from a proposal probability distribution and weighted according to their importance (see Section "Posterior Prediction"). The proposal distribution is adjusted during the iterations and tends toward the target posterior distribution $\sigma(\mathbf{m})$.

In practice, the algorithm iteratively expands a set of models $\mathcal{M}^k = \{\mathbf{m}_1, \ldots, \mathbf{m}_k\}$. For each model, the mean-square error between the predicted $g_i(\mathbf{m}_j)$ and the reference values d_i^{obs} is computed

$$\mathrm{MSE}(\mathbf{m}_j) = \frac{1}{n_{obs}} \sum_{i}^{n_{obs}} [g_i(\mathbf{m}_j) - d_i^{obs}]^2. \tag{2}$$

n_{obs} represents the number of observation points and σ the standard deviation of the observation errors. The likelihood is expressed as a function of the mean square error

$$L(\mathbf{m}_j) = C\exp\left[-\frac{\mathrm{MSE}(\mathbf{m}_j)}{2\sigma^2}\right], \tag{3}$$

with the constant C being unknown. This is why, at each iteration, PoPEx uses (and update) a normalized likelihood:

$$\widetilde{L}(\mathbf{m}_j) = \frac{L(\mathbf{m}_j)}{\sum_{r=1}^{k} L(\mathbf{m}_r)}, \qquad j = 1, \ldots, k. \tag{4}$$

To generate a new model $\mathbf{m}_{k+1}$, PoPEx performs a geostatistical simulation with n_k conditioning data points. Three steps are involved.

First, the number of conditioning points n_k is drawn from a uniform distribution over $\{0, 1, \ldots, n_{\max}\}$, where $n_{\max}$ is the maximum number of conditioning points specified by the user.

In the second step, PoPEx identifies the locations and parameter values (rock types or facies) that correspond to high likelihood values. This is done by comparing two probability distribution maps. The first is denoted $Q = \{\mathbf{q}_1, \ldots, \mathbf{q}_s\}$, with s the number of facies and $\mathbf{q}_i$ the probability map for the facies i. It describes the prior information based only on the geological data and geostatistical model. The second, $P^k = \{\mathbf{p}_1^k, \ldots, \mathbf{p}_s^k\}$, is updated at each iteration k. It corresponds to the facies probability maps weighted by the normalized likelihood $\widetilde{L}(\mathbf{m}_j)$. To identify in a probabilistic manner the link between the flow data and the facies, PoPEx computes the Kullback-Leibler divergence (KLD):

$$D(P^k \| Q) = \sum_{i=1}^{s} p_i^k \log\left(\frac{p_i^k}{q_i}\right). \tag{5}$$

In every location, the KLD map shows how different the two distributions are. PoPEx will then place randomly the n_k conditioning data but with a higher probability in regions of high values of the KLD. At those locations, the facies values are then drawn randomly from the local P^k pdf.

The third and last step is the conditional geostatistical simulation. In this paper, PoPEx uses DeeSse (Straubhaar 2019), an efficient implementation of the Direct Sampling multiple-points statistic (MPS) method (Mariethoz et al. 2010; Mariethoz and Caers 2016).

Posterior Prediction

The parameters of a groundwater flow model are generally identified because the model is needed to make a prediction $f(\mathbf{m})$ of a quantity of interest (e.g., contaminant travel time, drawdown in a place of interest, maximum sustainable pumping rate, etc.). In a probabilistic framework, the aim is to estimate the expected value $\mu = \mathbb{E}_\sigma[f(\mathbf{m})]$ of that prediction. PoPEx, being an importance sampling technique, estimates μ using a weighted average:

$$\mu \approx \sum_{k=1}^{N} \widetilde{w}_k f(\mathbf{m}_k). \tag{6}$$

N is the number of models in the ensemble, $\widetilde{w}_k$ is the weight for model k. By assuming that the conditioning points are sufficiently distant to be considered independent (n_{max} must be small enough), Jäggli et al. (2018) show that it is possible to express the weights from the Q and P^k maps as follows:

$$w_k = L(\mathbf{m}_k) \prod_{j=1}^{n_k} \frac{q_{r(j)}(x_j)}{p_{r(j)}(x_j)}, \tag{7}$$

where $r(j)$ returns the category of the pixel at location $\mathbf{x_j}$. Since the weights can be extremely small (weight degeneracy problem), PoPEx raises them to an exponent α chosen such that the number of effective samples $n_e = \frac{\left(\sum_i^N w_i^\alpha\right)^2}{\sum_i^N (w_i^\alpha)^2}$ is at least higher than a value specified by the user. The weights are then normalized $\widetilde{w}_k = \frac{w_k^\alpha}{\sum_j^N w_j^\alpha}$. This is an approximation that converges to the exact result if the number of samples is sufficiently large.

Conditional Generative Adversarial Network

GANs were put forward by Goodfellow et al. (2014) and are able to learn an implicit description of the underlying probability distributions of the images in a training data set. The architecture of the GAN comprises two convolutional neural networks: a discriminator D and a generator G. In the original application, the generator is responsible for synthesizing realistic images $G(\mathbf{z})$ based on a latent variable z. The discriminator is responsible for distinguishing the generated images $G(\mathbf{z})$ from the images of the training set $x \sim p_{data}$. To be more specific, the discriminator computes the probability of a synthetic image $G(\mathbf{z})$ coming from the training data distribution $x \sim p_{data}$. While the generator strives for synthesizing more realistic images, the discriminator aims at better identifying the generated images as fake.

In this study, we adopted the pix2pix version of the cGAN (Isola et al. 2017), it can map an input image x into an output image y, $G: x \to y$. The objective function used for the cGAN that was utilized in this study is as follows:

$$\begin{aligned}\mathcal{L}_{cGAN}(G, D) = {} & \mathbb{E}_x \|1 - D(x, G(x))\|_2 \\ & + \mathbb{E}_{x,y} \|D(x, y)\|_2,\end{aligned} \tag{8}$$

where the aim is to minimize G and maximize D. x represents the input images and y denotes the corresponding output images. The discriminator is trained to label the real images as 1 and the synthesized images as 0. Hence, if the discriminator performs well, it will output 0 for the

the synthesized images and 1 for the real ones. Therefore, the overall loss function will have a high value (maximization objective of the discriminator). Similarly, if the generator is good, the discriminator will label the synthesized images $G(x)$ close to 1, which minimizes the loss function. The ultimate goal is to train a generator which produces very realistic images and the discriminator becomes unable to distinguish it from the real ones, producing values close to 0.5. In addition to the loss function defined above, the cGAN we used also includes an L1 type loss comparing the similarity of the generated image with the ground truth. This term ensures the low-frequency correctness of the generated images.

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y}|G(x) - y|. \tag{9}$$

The final objective function then becomes:

$$G^* = \arg\min_{G} \arg\max_{D} \mathcal{L}_{cGAN}(G, D) + \lambda\mathcal{L}_{L1}(G), \tag{10}$$

where λ assigns a weight for the L1 term (low frequency loss).

Numerical Experiment

The general aim of the numerical experiment presented in this paper is to investigate the feasibility of using the conditional GAN as a substitute for the flow simulation to accelerate the solution of the inverse problem using PoPEx. In this context, it is important to consider a synthetic reality case in which the underground is fully known. This allows testing thoroughly the quality of the results obtained with the proposed methodology. In a practical application, the map of the hydraulic conductivity field will never be known exhaustively and it would not be possible to test if the methodology properly identifies the position of the channel and the corresponding uncertainty.

The methodology is illustrated in Figure 1. It includes setting up an inverse problem which consists of identifying the position and uncertainty of highly conductive channels from a set of groundwater head measurements around a pumping well. Three inverse methods are applied. One is the most accurate but less efficient (importance sampler based on prior distribution), it provides the reference solution. Then, we apply the standard PoPEx algorithm that uses the flow simulation to evaluate the likelihood. Finally, we apply PoPEx using cGAN to replace the flow simulation. We then compare the accuracy of the results and numerical efficiency.

The detailed explanations for setting up this experiment are presented in the following subsections.

Problem Setting

The experiment was conducted on a two-dimensional heterogeneous aquifer comprising permeable channels in a less permeable matrix. The spatial structures of the aquifer were simulated using the multiple-point statistical (MPS) simulation tool DeeSse (Straubhaar 2019). The training image (TI) used for the simulations is shown in Figure 2 (Strebelle 2002). The DeeSse parameters are provided in Table 1. The size of the TI is 250×250 grid cells and each cell has a dimension of 1 m in each direction. The two different colors represent two different facies with different uniform hydraulic conductivity (K) values. As for the channels, the hydraulic conductivity value was set to 10^{-2} m/s whereas the matrix was two orders of magnitude less permeable 10^{-4} m/s. The simulation grid comprises 128 grid cells in the x and y directions, and 1 grid cell in the vertical direction. The thickness of the aquifer is constant and equal to 1 m. Constant head values were applied on the left and right boundaries as 1 m and 0 m, respectively, whereas the upper and lower boundaries had no flow boundary conditions. A pumping well was placed in the centre to extract 3 L/s.

A reference parameter field was generated using an arbitrary seed and was considered to be the true field. Thirteen measurement locations were determined to extract the hydraulic head values for performing likelihood estimations for the inverse problem. A random Gaussian noise with $\sigma = 0.05\,\text{m}$ was added to the extracted hydraulic head values. The reference parameter field along with its corresponding water heads can be seen in Figure 3.

The cGAN was then used to emulate the forward operator that was used to compute the likelihood of a given realization for the parameter field. For training the adversarial network, 600 MPS realizations were performed and the flow simulations for each of the geologies were created using flopy interface of the Modflow software (Bakker et al. 2016). Some examples of the parameter fields and corresponding flow responses can be seen in Figure 4.

cGAN Architecture and Implementation

The generator utilized the U-Net architecture with skip connections (Reed et al. 2016). It comprised an encoder (downsampling) and a decoder (upsampling) parts. Given n total number of layers, the skip connections concatenates the i^{th} layers with those of $n-1$. This structure allows the passing of low-level information between the input and output across the net.

The convolutions performed in both the generator and the discriminator used a 4×4 spatial filter with a stride of 2. The spatial filters are matrices comprising weights and are used to detect/extract features through multiplication operations. The stride corresponds to the step that is used when sliding the filter. The resulting convolved images become the feature maps. Following this step, the feature maps are normalized and the activation function is applied to introduce nonlinearity. In this study, the batch normalization was performed with a momentum value of 0.8. As for the activation function, the the Rectifier Linear Unit (ReLU) operation (Nair and Hinton 2010) for the encoder was carried out using Leaky ReLU (Maas et al. 2013) with a slope value of 0.2. The decoder did not have a slope value for the leaky ReLU operations.

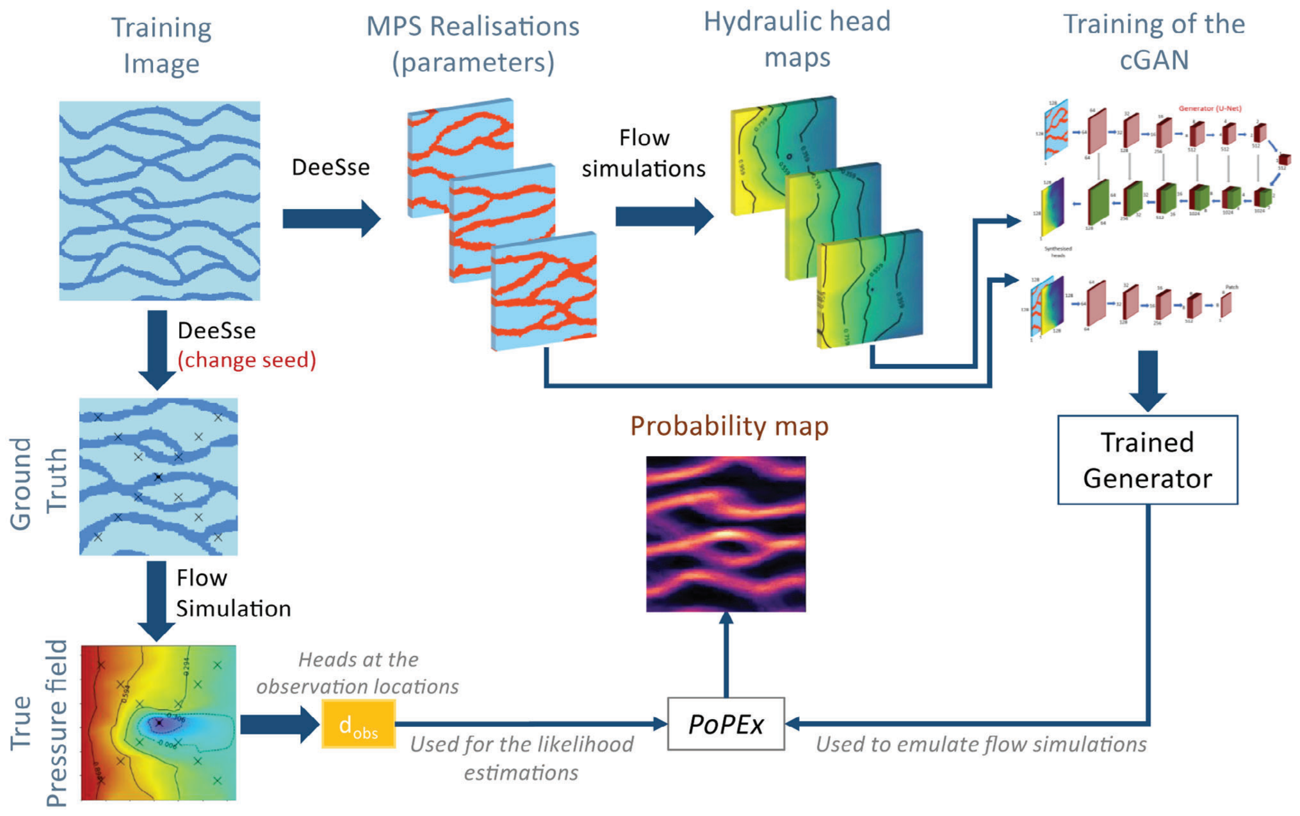

Figure 1. General overview of the methodology. A channelized training image was used to create a reference true-unknown field and its corresponding flow response. Same training image was then used to create realizations using a different seed value. The created realizations and their corresponding responses were used to train the cGAN model. The trained model was then incorporated in PoPEx to perform the inversion.

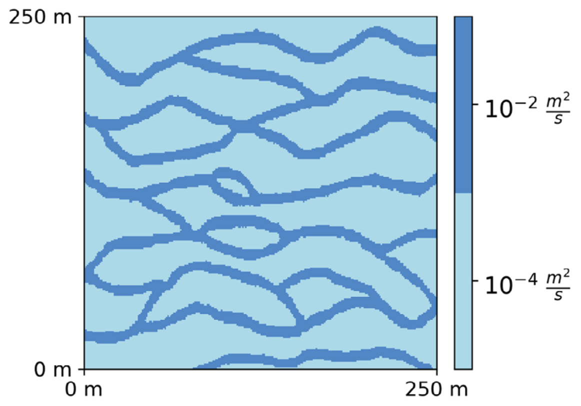

Figure 2. The TI used to simulate the aquifer parameters.

The output layer of the decoder was applied a hyperbolic tangent activation function.

The discriminator used was a Markovian discriminator which computed the texture/style loss. The convolutions were performed using a 4×4 spatial filter with a stride of 2. The convolution process returned an $N \times N$ patch to compute the goodness of the generated images. The target for the real images was an 8×8 patch with values of one at each pixel. Whereas, the target for the generated images was an 8×8 patch with values of zero at each pixel. The parameters of the discriminator were

Table 1
DeeSse Parameters Used for the Model Generation

Parameter	Value
Search radius	40
Number of neighboring nodes	30
Distance threshold	0.02
Maximal scan fraction	0.25
Number of postprocessing paths	1

optimized to minimize the difference between the target and the predicted patches. The architecture of the G and D can be seen in Figure 5.

The implementation of the methodology was performed using the Keras DL library in Python. The sizes of the input and output images (hydraulic conductivity fields and pressure heads, respectively) were 128×128. 500 image pairs were used for training and 100 images were used for testing. Prior to the training, min-max normalization was applied for the training dataset the and values of the images were scaled between 0 and 1. The adam optimization solver was used with a learning rate of 0.0002 and a forgetting rate of 0.5. The number of epochs was 200 and 50 images were used in the minibatches. The λ coefficient for the low frequency loss was set as

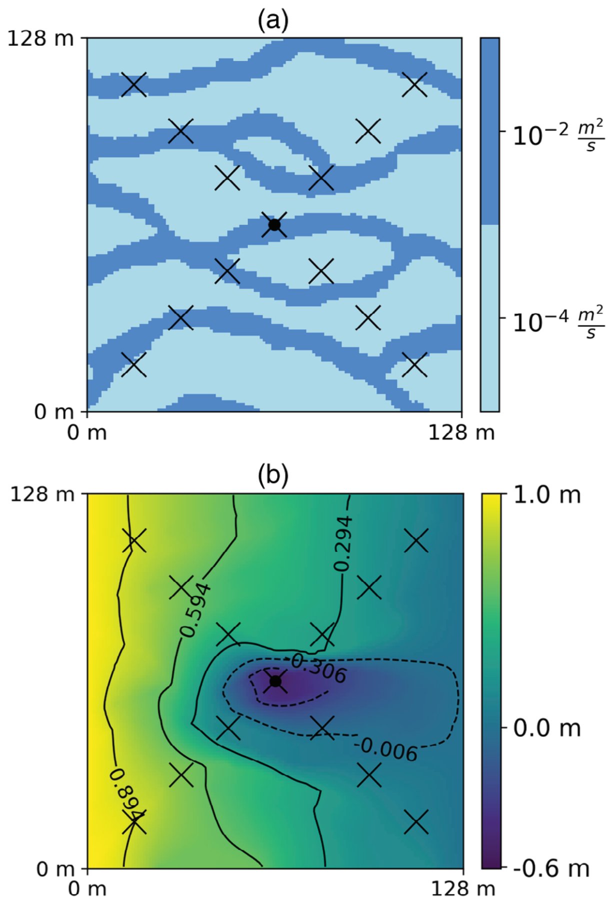

Figure 3. (a) Reference parameter field that was considered as ground truth and (b) its corresponding flow response.

100. The training was performed using two Nvidia Tesla K40c GPUs.

Results

cGAN Application

To visually check the performance of the trained network, two randomly chosen parameter fields from the test dataset were considered, as can be seen in Figure 6. The results in Figure 6 give a quick insight on the performance of the cGAN implementation. The results indicate that the generated images are rather similar to their originals and appear to be slightly noisy, as compared to their originals.

Since the trained model is aiming to replace the forward operator within an inversion process in this study, quantitative assessment of the performances were also carried out. Within PoPEx, the likelihood estimation is performed based on the error between the observed and the simulated heads. Therefore, MSE and the coefficient of determination (R^2) values of the generated pressure fields against the actual head maps were used to assess the cGAN performance. About 100 test images that were not included in the training set were used to perform the predictions using the trained cGAN model. The predicted head maps were then used to compute the associated coefficient of determination scores (R^2). The mean of the R^2 score was 0.97. This also indicates that the trained cGAN model appears to perform well.

As with many DL algorithms, it takes a fair amount of time to train a model. For 200 epochs, a minibatch size of 50, and 500 TIs in total, the training times were about 5.5 h for the 128×128 cells problem, and 17.3 h for the 256×256 cells problem. We have further investigated if there would be a potential avenue to reduce the computation time at a cost of reasonable accuracy loss. Our investigation consisted of changing the number of epochs and TIs used and observing the errors in terms of MSE.

The MSE obtained for different number of epochs and TIs used during training are shown in Figure 7. The MSE observed with 200 epochs and 500 training dataset size was found as 0.009. Whereas, mean and standard deviations of the proxy errors were 0.018 and 0.028, respectively. These values indicate that the errors are small when compared to the assumed data error used in the likelihood.

The scatterplots of the actual flow simulations against the predicted ones for different epochs and the training sizes are shown in Figures 8 and 9. Since the response of the flow simulations are different depending on whether the well intersects a channel or not, we provide both cases. The results show that as the number of epochs increases from 100 to 300, the error decreases. However, the MSE starts to significantly increase from 0.009 to 0.10 after epoch number 300. This behavior could be due to two reasons. First, the discriminator could become relatively more powerful than the generator with an increasing number of epochs. Hence, the generator faces difficulty in fooling the discriminator. One solution would be to modify the learning rates of the discriminator. Secondly, the discriminator might have undergone an overfitting issue. A possible solution to avoid this could be adding Gaussian noise to the input images. Nevertheless, since we have already obtained sufficient accuracies with epoch numbers less than 300, we have not further investigated this.

In the light of the above information, to lower the CPU cost, the model can also be trained using a reduced number of epochs such as 100 at a cost of reasonable accuracy loss. As for the number of TIs, the error decreases with the increasing number of TIs, as expected. Similarly, the training time can be reduced using a reduced training size of 300 to 400 by sacrificing reasonable accuracy.

A test was performed to compare the processing power demand of the cGAN model with the Modflow engine. In order to have a fair comparison between the two, we have performed a single CPU estimation of 500 head fields and repeated it 10 times. In addition, we have created a similar setup with 256×256 grid resolutions. Similarly, we have trained the cGAN with 500 image pairs and performed the predictions. This allowed us to see the effect of the number of parameters in

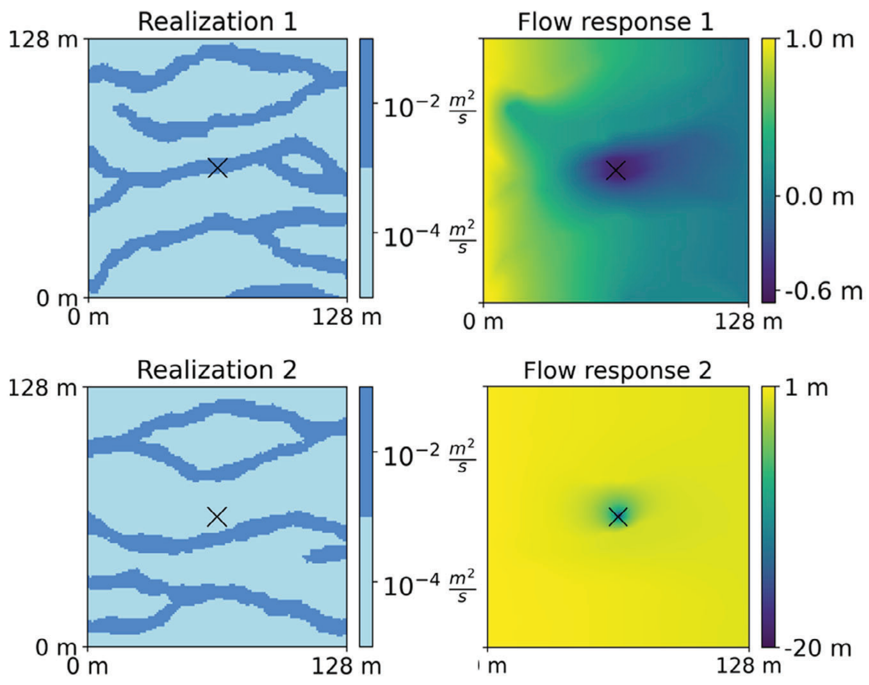

Figure 4. Some examples from the dataset used to train the cGAN model. Realization 1 and its corresponding flow response demonstrate the case in which the well intersects highly permeable channels. Whereas the realization 2 and its corresponding flow response represent the cases in which the well intersects the low permeability matrix.

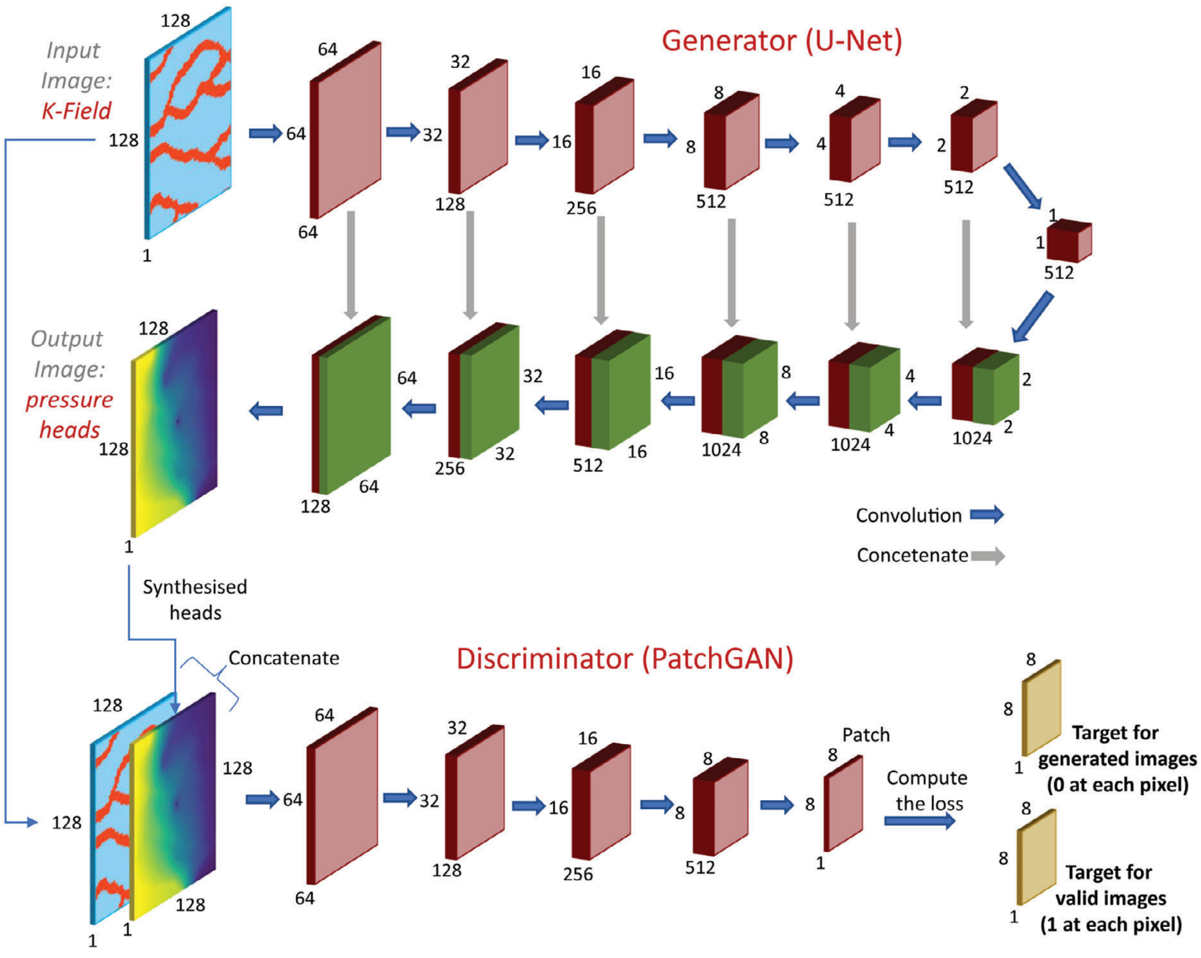

Figure 5. General overview of the architecture for the generator and discriminator. The discriminator here evaluates the validity of the generated image using the output patch generated.

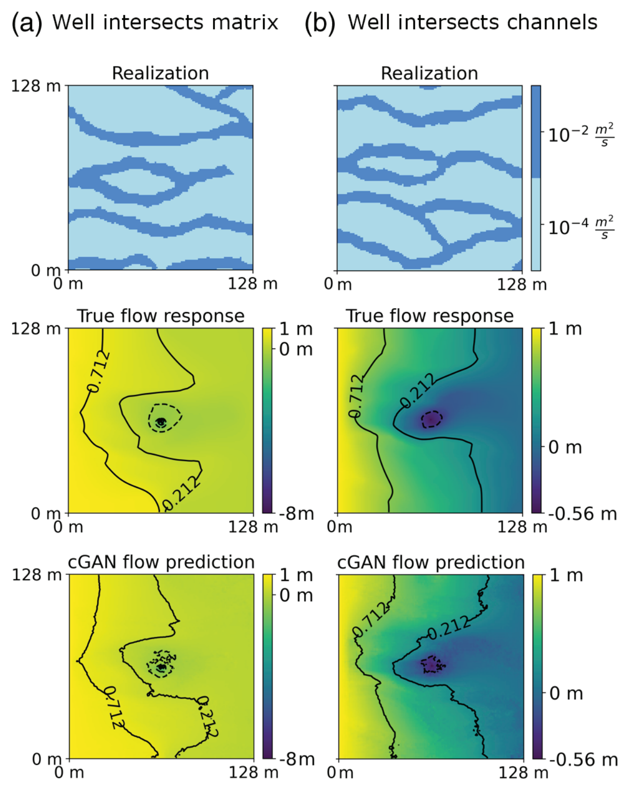

Figure 6. Validation images to visually check the prediction performance of the trained adversarial network: (a) flow simulation and the cGAN prediction for the given realization (well location intersects the matrix) and (b) flow simulation and the cGAN prediction for the given realization (well location intersects the channel).

the conductivity field on both the cGAN and Modflow simulations. The CPU used was i7-8750H and the GPU used for the predictions was GeForce GTX 1050. The results in Figure 10 demonstrate that cGAN required less computational power as compared to Modflow in both resolutions. Considering the tests done on single CPUs, we have observed 51% and 30% reduction in the computation times for the 128×128 and 256×256 resolutions, respectively. The same test done on the GPU, revealed a significant improvement. Use of GPU demonstrated an 83% computational reduction for the 128×128 resolution and 79% reduction for the 256×256 resolutions.

There is 105 s time difference between Modflow and GPU processing times to compute 500 flow simulations.

PoPEx Results

To test the performance of the GAN-flow emulator, we used it as the forward operator in the PoPEx procedure. We then compared the PoPEx solution using the cGAN forward emulator with two alternatives: the reference sampler and the PoPEx solution with Modflow forward simulator. Figure 11 shows the posterior probability maps of the channel for the three set-ups. The prior probability map for the channel is uniform over the entire domain with the value of 0.72.

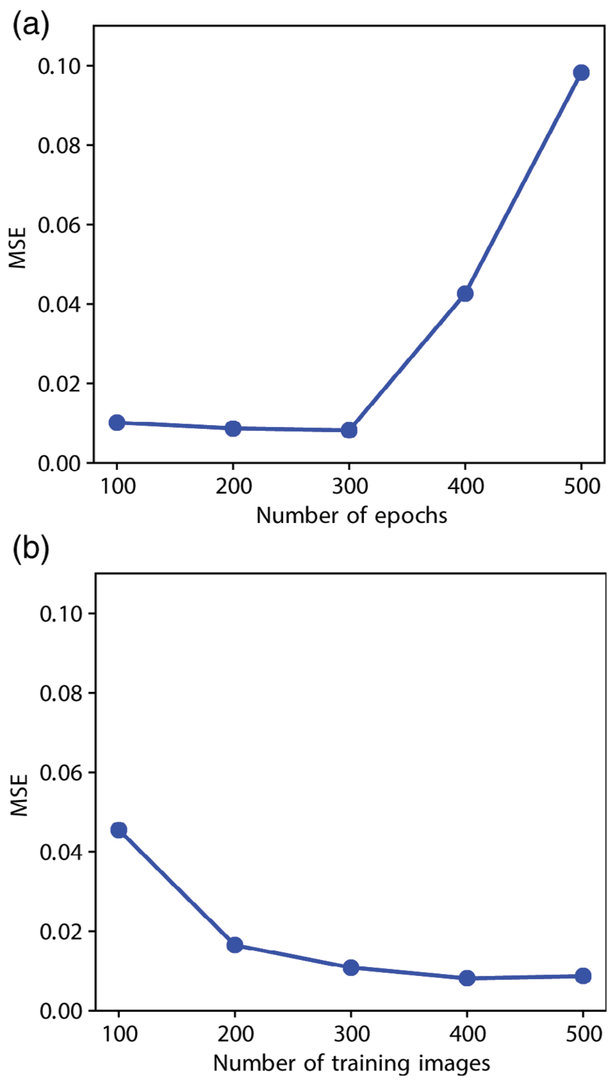

Figure 7. Mean squared error between the actual and the cGAN predicted head maps based on (a) different number of epochs and (b) different training dataset size.

To compute the reference solution, we applied the importance sampling method using the prior distribution as the proposal distribution (Cary and Chapman 1988). In that case, the probability map is obtained using Equation 6 with the weights being simply the normalized likelihood values (Jäggli et al. 2018). This method is not the most efficient but since we applied it for a large number of models (300,000) and because the problem is of relatively small size, we ensure high accuracy of the result.

Model generation in all the cases was done by the DeeSse algorithm with the same parameter set defined in Table 1, which was also used for training of the GAN-flow emulator. The PoPEx solutions (both with GAN-flow and with Modflow) are based on 10,000 iterations, the minimal number of models used for prediction is $l_0 = 100$ and maximal number of conditioning points guiding simulations is 20.

The results given in Figure 11 show that both PoPEx solutions correctly reproduced the channelized structure of

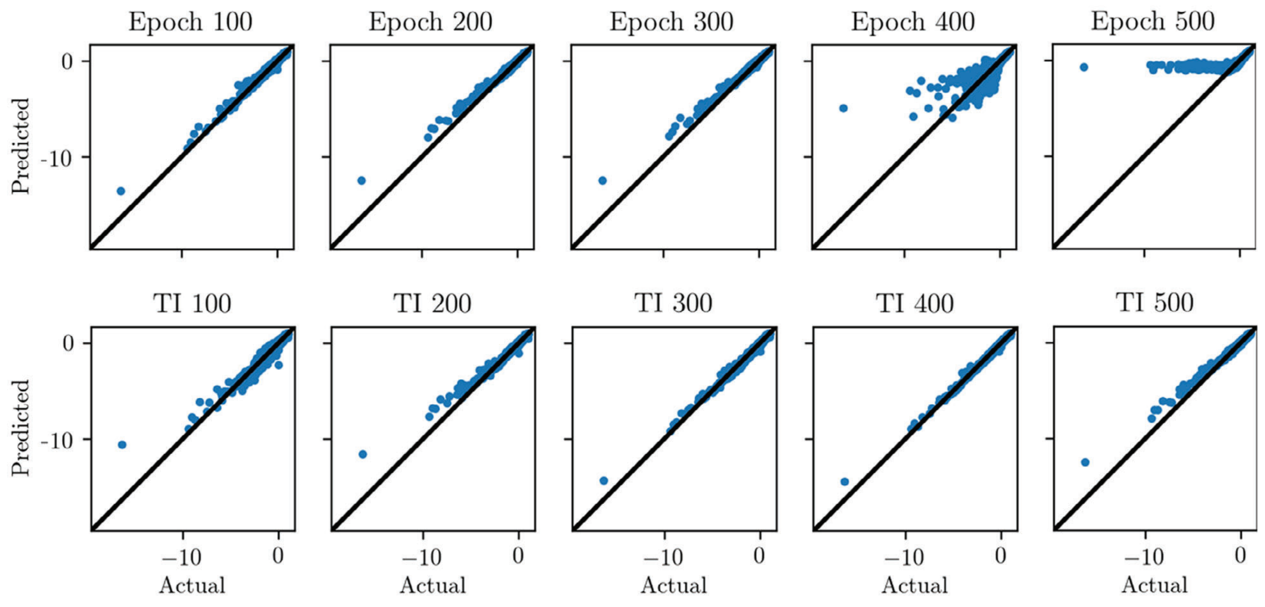

Figure 8. Scatterplot of the actual head values against the cGAN predictions for different epochs and training set sizes. Plots belong to the flow simulations in which the pumping well intersects the matrix.

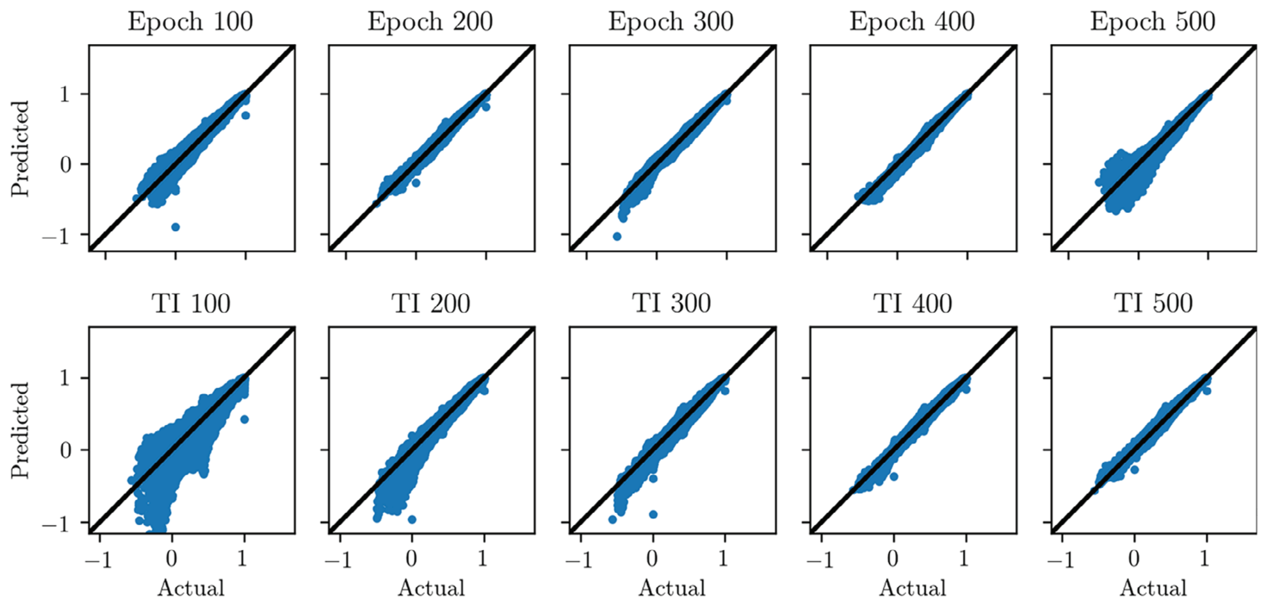

Figure 9. Scatterplot of the actual head values against the cGAN predictions for different epochs and training set sizes. Plots belong to the flow simulations in which the pumping well intersects the channels.

the probability map. The quality of the PoPEx with GAN-flow solution is visually comparable to that of PoPEx with Modflow, only slight underestimation of the uncertainty of having a channel (stronger contrasts), is visible on the right side of the image.

To quantify the error of the PoPEx solution with respect to the prior sampling solution, we used the Jensen-Shannon divergence. Considering the posterior solution $\widehat{\mu}$, the Jensen-Shannon divergence with respect to the reference prior sampling solution μ^{ex} is given by:

$$J(\widehat{\mu}\|\mu^{ex}) = \frac{1}{2}[D(\widehat{\mu}\|\,\overline{\mu}) + D(\mu^{ex}\|\,\overline{\mu})], \qquad (11)$$

where $D(\cdot\,\|\cdot)$ is the Kullback-Leibler divergence as defined in Equation 5 and $\overline{\mu} = \frac{1}{2}(\widehat{\mu} + \mu^{ex})$. The errors for the PoPEx solutions $\widehat{\mu}$ using GAN-flow and Modflow as a function of the number of iterations is plotted in Figure 12.

For the solution obtained after 10,000 iterations, we obtained the following values of the averaged Jensen-Shannon divergence: 0.007 for Modflow and 0.021 for GAN-flow. The maps of errors given by Jensen-Shannon divergence for both flow simulators are plotted in Figure 13. The convergence plot (Figure 12) demonstrates that GAN-flow error goes down systematically with the Modflow error. Even if the GAN-flow error is slightly

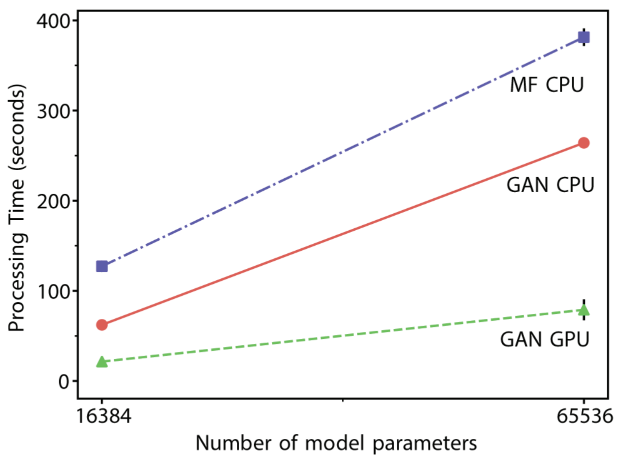

Figure 10. Average processing time required for running the forward operator on 500 flow problems as a function of the number of parameters in the input parameter field.

bigger than the Modflow error, both convergence rates compare well with the theoretical rate of $k_{it}^{-1/2}$ (where k_{it} is the number of iterations) predicted by the Central Limit Theorem (Durrett 2019).

Discussion and Conclusion

We have implemented cGAN to emulate the forward operator used in a hydrogeological inverse problem. The model was trained using a dataset comprising (1) 500 hydraulic conductivity fields that were created using MPS simulations and (2) corresponding flow simulations obtained using Modflow. Having trained the cGAN, it was then used to emulate the forward operator of the PoPEx algorithm to compute the likelihoods. Prediction performances were assessed using a separate test dataset containing 100 parameter fields and corresponding flow simulations.

The results show that the cGAN is capable of mapping from a given hydraulic conductivity field to the corresponding flow simulations. Although the generated pressure maps are slightly noisy as compared to their referenced ones, there exists a high correlation between the predicted and actual head values. These errors were sufficiently small to have little impact on the results of the probabilistic inversion. The probability maps obtained using the cGAN were very close to the ones obtained using the exact forward solution. As compared to the PoPEx solution, the cGAN produced slightly less contrast at some locations. Nevertheless, this experiment demonstrated the capability of the cGAN to be used in the inversion successfully with less computational demand than the Modflow engine.

The use of cGAN is of course not limited to PoPEx. Any inverse method that uses repeatedly the forward model to evaluate the misfit could benefit from this technique. The gain will be the highest for inverse methods that make intensive use of the forward model such as Markov chain Monte Carlo methods (Mosegaard

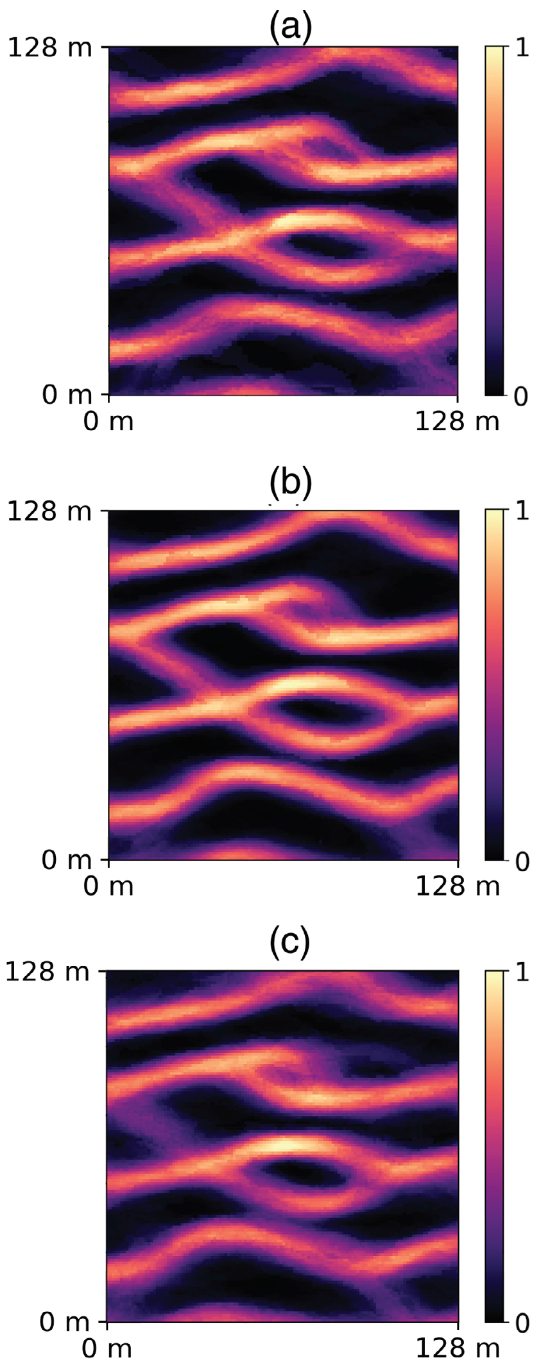

Figure 11. Comparison of posterior facies probability maps for channel, obtained using: (a) the reference technique (300,000 iterations), (b) PoPEx with Modflow forward (10,000 iterations), and (c) PoPEx with GAN-flow forward (10,000 iterations).

and Tarantola 1995), Particle or Ensemble Kalman Filters (Evensen 2018).

However, there are several drawbacks. First, as in most DL algorithms, training a model requires the creation of a training dataset. For complex geologies with numerous grid cells, this could take significant time. For such cases, a smaller sized dataset containing a smaller number of training instances can be still be used to perform reasonable predictions at the cost

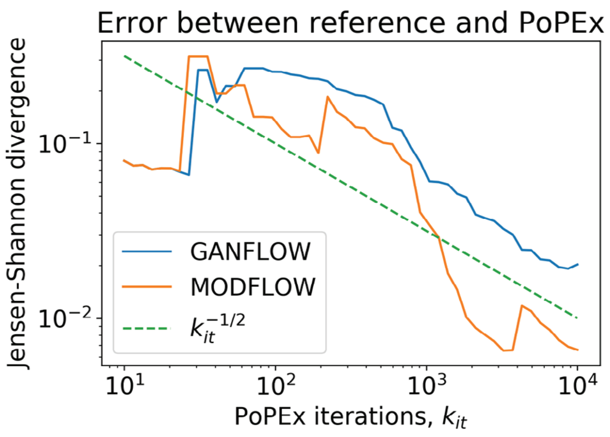

Figure 12. Comparison of the error of the inverse problem solution with respect to the prior sampling solution as a function of the number of PoPEx iterations k_{it} for the two different forward simulators. Error is defined by average Jensen-Shannon divergence.

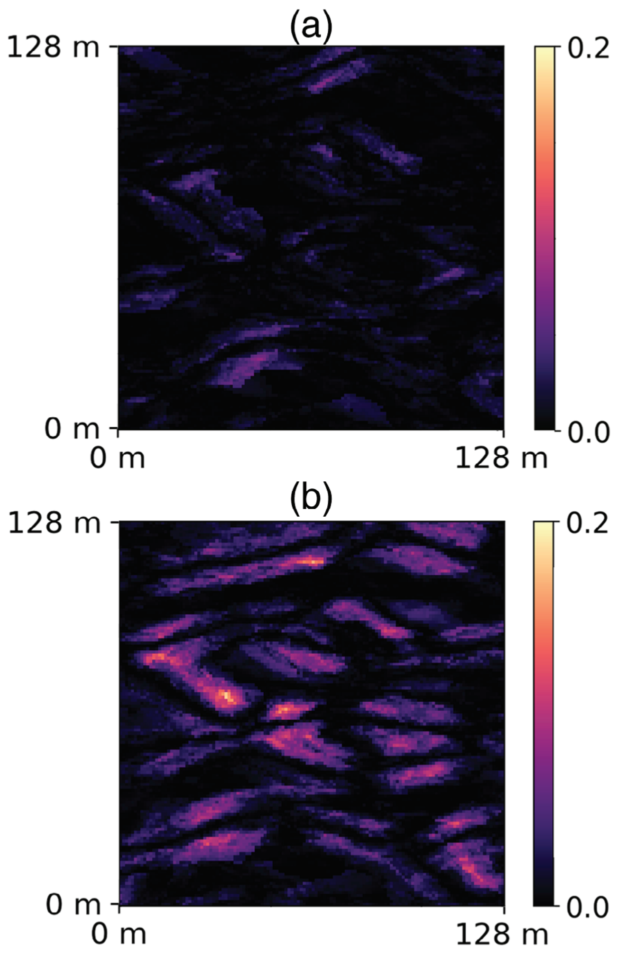

Figure 13. Maps of pixel-wise errors of PoPEx solution obtained after 10,000 iterations: (a) using Modflow and (b) using GAN-flow.

of some accuracy loss. Second, training of GANs can require significant time. To compensate for the training times, in the example that was treated in this paper, we require roughly 100,000 iterations of PoPEx. In other words, the use of cGAN becomes advantageous only after around a hundred thousand iterations. Considering that PoPEx can provide an accurate solution already after 10,000 iterations, we observe that the required number of iterations to compensate the training is quite large. This may hinder its use in practical applications. One solution could be to find some ways to simplify the generator network architecture to have fewer layers and to ensure faster training. This will probably reduce the quality of the approximation (Mallat 2016). However, for the specific application discussed in this paper, we do not necessarily need to predict the complete hydraulic head field. A simpler surrogate model could be sufficient to predict the likelihood with enough accuracy and be much faster. Another solution could be the use of transfer learning (Pan and Yang 2009). A pretrained model on a similar problem can be used to retrieve the initial parameters of the network architecture. The subsequent training would then involve fine-tuning of the parameters in a shorter period of training time for the specific application (Ng et al. 2015). These techniques have been shown to be very efficient.

To conclude, this paper shows the strength of the cGAN method in general but also highlights some practical limitations that are not frequently discussed in the literature concerning the application of these methods for groundwater modeling. It emphasizes the need to conduct further research in this field.

Acknowledgments

The authors would like to thank Julien Straubhaar and Christoph Jäggli for their help and suggestions concerning this work. We also thank the editor Henk M. Haitjema, two anonymous reviewers and Jean Marçais for their valuable comments and suggestions. We also appreciate Boris Galvan help in providing an access to the GPU cluster. Yasin Dagasan is grateful for the funding by the Swiss Government Excellence Scholarship.

Authors' Note

The authors do not have any conflicts of interest or financial disclosures to report.

References

Bakker, M., V. Post, C.D. Langevin, J.D. Hughes, J.T. White, J.J. Starn, and M.N. Fienen. 2016. Scripting modflow model development using python and flopy. *Groundwater* 54, no. 5: 733–739.

Bugallo, M.F., V. Elvira, L. Martino, D. Luengo, J. Miguez, and P.M. Djuric. 2017. Adaptive importance sampling: The past, the present, and the future. *IEEE Signal Processing Magazine* 34, no. 4: 60–79. https://doi.org/10.1109/MSP.2017.2699226

Carrera, J. 1988. State of the art of the inverse problem applied to the flow and solute transport equations. In *Groundwater flow and quality modelling. NATO ASI Series (Series C: Mathematical and Physical Sciences)*, Vol. 224, ed. E. Custodio, A. Gurgui, and J. Ferreira, 549–583. Dordrecht: Springer.

Cary, P., and C. Chapman. 1988. Automatic 1-D waveform inversion of marine seismic refraction data. *Geophysical Journal International* 93, no. 3: 527–546.

Chan, S. and A. H. Elsheikh. Parametrization and generation of geological models with generative adversarial networks. *arXiv preprint arXiv:1708.01810*, 2017.

Collobert, R., and J. Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, 160–167. New York: ACM.

de Marsily, G., J. Delhomme, A. Coudrain-Ribstein, and A. Lavenue. 2000. Four decades of inverse problems in hydrogeology. In *Theory, Modeling, and Field Investigation in Hydrogeology: A Special Volume in Honor of Shlomo P. Neuman's 60th Birthday*, Vol. 348, ed. D. Zhang, and C. Winter, 1–17. Boulder, Colorado: The Geological Society of America.

Dupont, E., T. Zhang, P. Tilke, L. Liang, and W. Bailey. Generating realistic geology conditioned on physical measurements with generative adversarial networks. *arXiv preprint arXiv:1802.03065*, 2018.

Durrett, R. 2019. *Probability: theory and examples*, Vol. 49. Cambridge, UK: Cambridge University Press.

Evensen, G. 2018. Analysis of iterative ensemble smoothers for solving inverse problems. *Computational Geosciences* 22, no. 3: 885–908. https://doi.org/10.1007/s10596-018-9731-y

Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, Montreal, Canada: Neural information processing systems foundation, 2672–2680.

Hadamard, J. 1902. *Sur les problèmes aux dérivées partielles et leur signification physique*, 49–52. Princeton, New Jersey: *Princeton University Bulletin*.

He, K., X. Zhang, S. Ren, and J. Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778. New York: *IEEE*.

Hinton, G., L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, B. Kingsbury, et al. 2012. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine* 29: 82–97.

Isola, P., J.-Y. Zhu, T. Zhou, and A.A. Efros. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1125–1134. New York: *IEEE*.

Jäggli, C., J.A. Straubhaar, and P. Renard. 2018. Parallelized adaptive importance sampling for solving inverse problems. *Frontiers in Earth Science* 6: 203.

Jäggli, C., J. Straubhaar, and P. Renard. 2017. Posterior population expansion for solving inverse problems. *Water Resources Research* 53, no. 4: 2902–2916. https://doi.org/10.1002/2016WR019550

Laloy, E., R. Hérault, J. Lee, D. Jacques, and N. Linde. 2017. Inversion using a new low-dimensional representation of complex binary geological media based on a deep neural network. *Advances in Water Resources* 110: 387–405.

Laloy, E. and D. Jacques. Emulation of CPU-demanding reactive transport models: comparison of gaussian processes, polynomial chaos expansion and deep neural networks. *arXiv preprint arXiv:1809.07305*, 2018.

Linde, N., P. Renard, T. Mukerji, and J. Caers. 2015. Geological realism in hydrogeological and geophysical inverse modeling: A review. *Advances in Water Resources* 86, no. A: 86–101.

Maas, A.L., A.Y. Hannun, and A.Y. Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the ICML*, Georgia, USA: PMLR, Atlanta, Vol. 30, 3.

Mallat, S.. Understanding deep convolutional networks. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, Royal Society, 374 (2065): 20150203, 2016. https://doi.org/10.1098/rsta.2015.0203.

Marçais, J., and J.-R. De Dreuzy. 2017. Prospective interest of deep learning for hydrological inference. *Groundwater* 55, no. 5: 688–692.

Mariethoz, G., and J. Caers. 2016. *Multiple-Point Geostatistics: Stochastic Modeling with Training Images*. Chichester, UK: John Wiley & Sons, Ltd. ISBN 978-1-118-66295-3 978-1-118-66275-5.

Mariethoz, G., P. Renard, and J. Straubhaar. 2010. The direct sampling method to perform multiple-point geostatistical simulations. *Water Resources Research* 46: W11536.

Mo, S., N. Zabaras, X. Shi, and J. Wu. 2019a. Deep autoregressive neural networks for high-dimensional inverse problems in groundwater contaminant source identification. *Water Resources Research* 55, no. 5: 3856–3881.

Mo, S., Y. Zhu, N. Zabaras, X. Shi, and J. Wu. 2019b. Deep convolutional encoder-decoder networks for uncertainty quantification of dynamic multiphase flow in heterogeneous media. *Water Resources Research* 55, no. 1: 703–728.

Mosegaard, K., and A. Tarantola. 1995. Monte Carlo sampling of solutions to inverse problems. *Journal of Geophysical Research: Solid Earth* 100, no. B7: 12431–12447.

Mosser, L., O. Dubrule, and M. J. Blunt. Deepflow: History matching in the space of deep generative models. *arXiv preprint arXiv:1905.05749*, 2019.

Mosser, L., O. Dubrule, and M. J. Blunt. Conditioning of three-dimensional generative adversarial networks for pore and reservoir-scale models. *arXiv preprint arXiv:1802.05622*, 2018a.

Mosser, L., O. Dubrule, and M.J. Blunt. 2018b. Stochastic reconstruction of an oolitic limestone by generative adversarial networks. *Transport in Porous Media* 125, no. 1: 81–103.

Mosser, L., O. Dubrule, and M.J. Blunt. 2017. Reconstruction of three-dimensional porous media using generative adversarial neural networks. *Physical Review E* 96, no. 4: 043309.

Nair, V., and G.E. Hinton. 2010. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, Madison, WI, United States: Omnipress, 807–814.

Ng, H.-W., V.D. Nguyen, V. Vonikakis, and S. Winkler. 2015. Deep learning for emotion recognition on small datasets using transfer learning. In Proceedings of the *2015* ACM on international conference on multimodal interaction, 443–449. New York: ACM.

Pan, S.J., and Q. Yang. 2009. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering* 22, no. 10: 1345–1359.

Reed, S.E., Z. Akata, S. Mohan, S. Tenka, B. Schiele, and H. Lee. 2016. Learning what and where to draw. *Advances in Neural Information Processing Systems* 29: 217–225.

Straubhaar, J.. DeeSse user's guide. Technical report, Centre for Hydrogeology and Geothermics (CHYN), University of Neuchâtel, Switzerland, 2019.

Strebelle, S. 2002. Conditional simulation of complex geological structures using multiple-point statistics. *Mathematical Geology* 34, no. 1: 1–21.

Sun, A.Y. 2018. Discovering state-parameter mappings in subsurface models using generative adversarial networks. *Geophysical Research Letters* 45, no. 20: 11–137.

Tarantola, A. 2005. *Inverse problem theory and methods for model parameter estimation*. Philadelphia, Pennsylvania: Society for Industrial and Applied Mathematics.

Tarantola, A., and B. Valette. 1982. Inverse problems = quest for information. *Journal of Geophysics* 50: 159–170.

Tripathy, R.K., and I. Bilionis. 2018. Deep uq: Learning deep neural network surrogate models for high dimensional uncertainty quantification. *Journal of Computational Physics* 375: 565–588.

Xu, Y., T. Mo, Q. Feng, P. Zhong, M. Lai, I. Eric, and C. Chang. Deep learning of feature representation with multiple instance learning for medical image analysis. In *2014 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 1626–1630. New York: IEEE, 2014.

Yang, L., D. Zhang, and G. E. Karniadakis. Physics-informed generative adversarial networks for stochastic differential equations. *arXiv preprint arXiv:1811.02033*, 2018.

Zhou, H., J.J. Gómez-Hernández, and L. Li. 2014. Inverse methods in hydrogeology: Evolution and recent trends. *Advances in Water Resources* 63: 22–37.

Zhu, Y., N. Zabaras, P.-S. Koutsourelakis, and P. Perdikaris. 2019. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics* 394: 56–81.

Zhu, Y., and N. Zabaras. 2018. Bayesian deep convolutional encoder-decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics* 366: 415–447.