frontiers in Artificial Intelligence

ORIGINAL RESEARCH
published: 18 March 2021
doi: 10.3389/frai.2021.624629

An Attempt to Boost Posterior Population Expansion Using Fast Machine Learning Algorithms

Przemysław Juda[1]* *and Philippe Renard*[1,2]

[1] Stochastic Hydrogeology and Geostatistics Group, Centre for Hydrogeology and Geothermics, University of Neuchâtel, Neuchâtel, Switzerland, [2] Department of Geosciences, University of Oslo, Oslo, Norway

In hydrogeology, inverse techniques have become indispensable to characterize subsurface parameters and their uncertainty. When modeling heterogeneous, geologically realistic discrete model spaces, such as categorical fields, Monte Carlo methods are needed to properly sample the solution space. Inversion algorithms use a forward operator, such as a numerical groundwater solver. The forward operator often represents the bottleneck for the high computational cost of the Monte Carlo sampling schemes. Even if efficient sampling methods (for example Posterior Population Expansion, PoPEx) have been developed, they need significant computing resources. It is therefore desirable to speed up such methods. As only a few models generated by the sampler have a significant likelihood, we propose to predict the significance of generated models by means of machine learning. Only models labeled as significant are passed to the forward solver, otherwise, they are rejected. This work compares the performance of AdaBoost, Random Forest, and convolutional neural network as classifiers integrated with the PoPEx framework. During initial iterations of the algorithm, the forward solver is always executed and subsurface models along with the likelihoods are stored. Then, the machine learning schemes are trained on the available data. We demonstrate the technique using a simulation of a tracer test in a fluvial aquifer. The geology is modeled by the multiple-point statistical approach, the field contains four geological facies, with associated permeability, porosity, and specific storage values. MODFLOW is used for groundwater flow and transport simulation. The solution of the inverse problem is used to estimate the 10 days protection zone around the pumping well. The estimated speed-ups with Random Forest and AdaBoost were higher than with the convolutional neural network. To validate the approach, computing times of inversion without and with machine learning schemes were computed and the error against the reference solution was calculated. For the same mean error, accelerated PoPEx achieved a speed-up rate of up to 2 with respect to the standard PoPEx.

Keywords: hydrogeology, inverse problem, posterior population expansion, binary classification, geostatistics, groundwater flow and transport, deep learning, ensemble learning

OPEN ACCESS

Edited by:
J. Jaime Gómez-Hernández, Universitat Politècnica de València, Spain

Reviewed by:
Vasily Demyanov, Heriot-Watt University, United Kingdom
Ruopu Li, Southern Illinois University Carbondale, United States

***Correspondence:**
Przemysław Juda
przemyslaw.juda@unine.ch

Specialty section:
This article was submitted to AI in Food, Agriculture and Water, a section of the journal Frontiers in Artificial Intelligence

Received: *31 October 2020*
Accepted: *22 February 2021*
Published: *18 March 2021*

Citation:
Juda P and Renard P (2021) An Attempt to Boost Posterior Population Expansion Using Fast Machine Learning Algorithms. Front. Artif. Intell. 4:624629. doi: 10.3389/frai.2021.624629

1. INTRODUCTION

Groundwater flow and contaminant transport in aquifers depend on subsurface parameters such as permeability, specific storage, or porosity. Due to the lack of their direct measurements and heterogeneity, solving inverse problem is essential to make reliable predictions of groundwater flow and crucial for water resources management.

The inverse problem consists in deducing the subsurface parameters given state variables measured in the field, such as hydraulic heads or tracer concentrations. The physical problem leading from parameters to state variables is called the forward problem; it often involves solving partial differential equations. While the forward problem has a unique solution, the inverse problem is usually ill-posed, with non-unique and unstable solution if framed in a deterministic manner. The inverse problem is especially difficult when dealing with highly heterogeneous parameter fields or categorical fields. Therefore, methods for solving the inverse problem in hydrogeology (and more broadly in geophysics) have been a topic of extensive research (Zhou et al., 2014; Linde et al., 2015). If defined in a probabilistic manner, the inverse problem is no longer ill-posed and the solution always exists (Tarantola, 2005). When formulated in a Bayesian framework, it needs defining prior knowledge which allows to properly account for subsurface heterogeneity. The associated difficulty lies in estimating the probability density over a non-linear space of parameters and requires using a Monte Carlo method for generating many parameter fields and forward model runs. Depending on the physical problem, the forward model can be computationally expensive (especially true for transient groundwater flow or contaminant transport models); the inversion would typically require long runs and using high-performance computing resources.

Machine learning, including deep learning, has gained momentum in water research (Shen et al., 2018), and can be used to improve the efficiency of the inverse methods (Marçais and de Dreuzy, 2017). There are two main areas of application of machine learning related to the inverse problem: first, using machine learning techniques for modeling the prior knowledge; second, emulating the forward problem.

Concerning the generation of samples from the prior distribution, generative adversarial networks (GAN) are interesting because of their ability to generate geologically realistic models. Despite significant training times, they are attractive for the fast generation of fields and because they offer a low dimensional parametrization, which allows an efficient exploration of model space (Laloy et al., 2017; Chan and Elsheikh, 2020). As a consequence, GANs have been successfully used in Markov Chain Monte Carlo algorithms for solving groundwater inversion problems (Laloy et al., 2017, 2018). Due to the possibility of computing gradients in GAN models space, deterministic inversion using GANs is also possible but can be hindered by the non-linearities of the problem (Laloy et al., 2019). Simpler models, such as Support Vector Machines (SVM) were also used to construct informative geological prior to constrain sampling for realistic reservoir models (Arnold et al., 2019).

Concerning the emulation of the forward problem, machine learning can also be used. For example, Tripathy and Bilionis (2018) used deep neural network to construct a surrogate for a stochastic partial differential equations and employed it in the context of high dimensional uncertainty quantification. Laloy and Jacques (2019) tested machine learning algorithms (including deep neural nets, Gaussian processes, polynomial chaos expansion) to emulate reactive transport model and found that deep neural networks perform reasonably well in the context of quantifying uncertainty. Dagasan et al. (2020) showed how GAN can emulate steady-state flow solver and used it in the posterior population expansion algorithm (Jäggli et al., 2018), showing that the results of the inversion were of similar quality as those obtained using the numerical flow solver.

All these approaches rely on machine learning techniques built specifically for a problem at hand and replacing state-of-the-art methods for either modeling the prior, or emulating the forward problem.

In this paper, we propose a slightly different approach based on a machine learning classifier, and not substituting the prior geostatistical model, nor the forward solver. We use the posterior population expansion method (PoPEx) to invert a categorical field and our aim is to accelerate the convergence of the algorithm. During the Monte Carlo exploration of the model space, models are generated along with the forward operator results. These data are then used to train a classifier, which predicts if models would have high or low likelihood and whether they would contribute much to the posterior distribution. Then, the classifier can decide if a forward operator should be called (when the parameter field is predicted to be favorable) or if a model can be discarded, saving computational time. This approach is generic, as it does not rely on a specific geostatistical prior model, neither on a forward solver type. It is similar in spirit to the approach of building a surrogate or emulated model (like Laloy et al., 2019; Dagasan et al., 2020), but instead of reproducing the forward response, it predicts if the response would match well with the observed data. A similar idea was proposed by Demyanov et al. (2010), where SVM classifier was applied to separate high likelihood models while stochastically sampling parameters space for reservoir predictions.

To test the efficiency of the proposed method, We consider an alluvial aquifer. Its geological heterogeneity is modeled using multiple-point statistics. The inverse problem consists in interpreting one tracer test. Once the geological heterogeneity and its remaining uncertainty is identified, the resulting distribution of geological fields is used to predict the 10-days capture zone. This problem is used since it is a frequent question in applied hydrogeology. One has to interpret tracer test data to then delineate protections zones around future drinking water production wells. Therefore, in addition to the analysis of the efficiency of the inverse method, we check also the quality of this prediction as compared to the reference.

2. METHODS

In this section, we present a brief review of the posterior population expansion algorithm, and the machine learning schemes with their performance metrics used in this study. Finally, we explain how the schemes are used to accelerate PoPEx algorithm.

2.1. Inverse Problem and PoPEx Algorithm

PoPEx has been previously introduced and presented in detail in Jäggli et al. (2017) and Jäggli et al. (2018). It solves the inverse problem in a probabilistic manner. First, we explain how the problem is framed; then, we review the algorithm and its parameters; finally, we show how the solution can be used to generate a prediction.

2.1.1. Probabilistic Formulation

We will consider that $\mathbf{d}_{obs} \in \mathbb{R}^n$ is a data set obtained from an experiment with n defining the number of data points. The data are physical state variables, for example: hydraulic heads, contaminant concentrations, or flow rates. Let $\mathbf{g}: \mathcal{M} \rightarrow \mathbb{R}^n$ be the forward operator, mapping from the model space $\mathcal{M}$ to the data space $\mathbb{R}^n$. The model space defines the set of physical parameters, which fully describe the system (for example subsurface parameters) and allows to solve the forward problem (for example a groundwater flow problem). The forward operator generates the observable data given the model. In our case, the forward operator is a groundwater flow and transport model, and the model space is a set of all possible geological realizations of the subsurface, which map to permeability, porosity and specific storage.

The probabilistic solution of the inverse problem is given by (Tarantola, 2005):

$$\sigma(\mathbf{m}) = c\rho(\mathbf{m})L(\mathbf{m}), \quad \mathbf{m} \in \mathcal{M}$$

with σ the posterior probability density, ρ the prior probability density, L the likelihood, and c a normalization constant. The prior probability density is defined by expert knowledge about the parameter field. For example, it can constrain the type of geology which is considered. The likelihood evaluates the mismatch between the data and simulated values (output of the forward operator).

2.1.2. PoPEx Algorithm

In the following, we provide a rapid and brief overview of PoPEx. The posterior population expansion algorithm (Jäggli et al., 2018) is a modified adaptive importance sampler. It iteratively expands a sampled model set and learns the underlying probability distribution. It is designed for solving inverse problems when the space $\mathcal{M}$ contains categorical models. It requires a conditional geostatistical tool (e.g., a geostatistical algorithm which can generate a new model $\mathbf{m}$ from space $\mathcal{M}$ given conditioning points); and a forward solver which computes the likelihood given the model. At each iteration k it expands the sampled model set and uses all previously generated models and their likelihoods to define the conditioning set for the next model $\mathbf{m}_{k+1}$. The number of conditioning points at every iteration is uniformly drawn from $\{0, \ldots, n_c\}$, where n_c is the maximal number of conditioning points — a parameter specified by the user. Then, the new model is generated by the geostatistical tool honoring the imposed conditioning data and the forward solver computes its likelihood. The PoPEx algorithm stops after specified number of iterations N given by the user.

The choice of the conditioning data locations is guided by the Kullback-Leibler divergence (KLD) map computed between two discrete probability distributions P and Q on the probability space $\mathcal{X}$ and at each point in the domain:

$$D(P^k||Q) = \sum_{x \in \mathcal{X}} P^k(x) \log\left(\frac{P^k(x)}{Q(x)}\right). \tag{1}$$

$Q = \{q_1, \ldots, q_s\}$ corresponds to the prior probability maps for each category, with s the number of categories (e.g., geological facies). $P^k = \{p_1^k, \ldots, p_s^k\}$ correspond to the maps of facies probabilities but weighted by the normalized likelihoods estimated at iteration k: $\tilde{L}(m_j) = L(m_j)/(\sum_{r=1}^{k} L(m_r))$. The higher the KLD for a given point, the more likely it is chosen as a conditioning data location. Once conditioning point locations are determined, their values are sampled from the local P distribution.

2.1.3. Prediction

Typically, solutions of the inverse problem are used to generate some predictions. Let f be the function mapping from models to predictions, and μ the expected value of some quantity of interest (e.g., prediction of hydraulic head, capture zone of a well):

$$\mu = \int_{\mathcal{M}} \sigma(\mathbf{m}) f(\mathbf{m}) d\mathbf{m}. \tag{2}$$

PoPEx uses the set of generated models to approximate (2):

$$\mu = \sum_{i=0}^{N} \widehat{w}_i f(\mathbf{m}_i),$$

with $\widehat{w}_i$ a normalized *corrected* weight of the model i. The *corrected* weights are computed from the sampling weights w_i:

$$\widehat{w}_i = \frac{w_i^\alpha}{\sum_{i=1}^{N} w_i^\alpha},$$

with α the correcting factor. The weights w_i are given by:

$$w_i = L(\mathbf{m}_i)\frac{\rho(\mathbf{m}_i)}{\phi_i(\mathbf{m}_i)}, \tag{3}$$

where ρ is the prior distribution function and ϕ_i is the sampling distribution used at the iteration i. The weights need to be adjusted, as with large model spaces the distribution $W^N = \{w_1, \ldots, w_N\}$ can be dominated by few very large samples. The effective sample size n_e provides a measure of the skewness of the distribution:

$$n_e(W^N) = \frac{\sum_{i=1}^{N} w_i^2}{(\sum_{i=1}^{N} w_i)^2} \tag{4}$$

Correcting the weights consists of finding the factor $\alpha \in (0, 1]$ as close (or equal) to 1 such that the effective number of weights of the set $\{w_1, \ldots, w_N\}$ is at least l_0, with l_0 specified by the user.

2.2. Machine Learning Methods

Binary classification is a supervised learning task that has been extensively studied in the context of predictive data analytics (Kelleher et al., 2015), statistical learning (Hastie et al., 2009), and deep learning (Goodfellow et al., 2016). It consists in predicting the binary label (binary class, encoded as 0 or 1) given the input data (features). If X is the input (a feature vector, containing categorical or continuous variables), a binary classifier $\hat{f}$ maps X to the binary label $\hat{y}$: $\hat{f}(X) = \hat{y}$ with $\hat{y} \in \{0, 1\}$. The classifier must be fitted to known data with assigned labels, and then it can be used to generate predictions on previously unseen data. In this work, we considered three classifiers: AdaBoost, Random Forest, and Convolutional neural network (CNN). Here, we briefly introduce these methods and the cross-validation technique which was used to tune parameters of the classifiers and select the best algorithm. We refer to Hastie et al. (2009) for introductions about AdaBoost, Kelleher et al. (2015) about decision trees, Breiman (2001) about Random Forest, and Goodfellow et al. (2016) about CNN and deep learning in general.

2.2.1. Classifiers

AdaBoost is a boosting method (Freund and Schapire, 1997) which produces a sequence of weak classifiers. The weak classifiers are iteratively fitted to the data with weights modified at each step. More weight is given to previously missclassified samples (initially weights are equal). The final prediction of the AdaBoost algorithm is a majority vote of all classifiers. Typically, the weak classifiers used in AdaBoost are shallow decision trees. The error on the training sample is the average of the fraction of misclassified samples. The fitting is finished when perfect fit is achieved or when the total number of estimators has been reached.

Random Forests (Breiman, 2001), similarly to AdaBoost, rely on ensembles of decision trees. The decision trees are typically deeper than in AdaBoost and they are trained on data randomly sampled from the input. The sampling distribution from which the training data is drawn is the same for all trees in the forest. The main parameters of the method are the tree depth and the number of trees that form the forest. Breiman (2001) claimed that random forests compare favorably to AdaBoost, yielding similar errors and being more robust with respect to noise.

For gridded inputs, convolutional neural networks (LeCun et al., 1989) proved to be effective. Convolutional neural networks are a special type of neural nets (Goodfellow et al., 2016), composed of convolutional layers, combined with activation, pooling and dense layers. They are especially suitable for object recognition and are state-of-the art classifiers for working with images.

2.2.2. Cross-Validation

Cross-validation is a technique which allows to estimate the error (or a score) of the classifier on the unseen data. In K-fold cross-validation the whole training data is divided into K subsets and K iterations are performed. In each iteration $iter = 1, \ldots, K$, the $iter$ subset in a row is removed from the data to form the validation set. The rest of the data becomes the training set. The classifier is fitted to the training set and the error (or a score) is computed using the validation set: predictions are made on the validation set and they are compared to true values. Then, gathering the output of K iterations, one can compute the statistics of the error or the score function. Kohavi (1995) argued that the best choice is $K = 5$ or $K = 10$.

2.2.3. Performance Measures

We introduce here the commonly used performance measures for binary classifiers, which are based on the confusion matrix. Let us suppose that the classification results in TP true positives, TN true negatives, FP false positives and FN false negatives. Precision is defined as follows:

$$\text{precision} = \frac{\text{TP}}{(\text{TP} + \text{FP})}, \tag{5}$$

it describes how confident we can be that the predicted positive instance is correct. Recall is given by:

$$\text{recall} = \frac{\text{TP}}{(\text{TP} + \text{FN})}, \tag{6}$$

and it describes the ability of the model to find all positive instances. The harmonic mean of precision and recall is F score (or F_1 score) and it is generalized by F_β score (Rijsbergen, 1979):

$$F_\beta = \left(1 + \beta^2\right) \frac{\text{precision} \cdot \text{recall}}{\beta^2 \text{precision} + \text{recall}}, \tag{7}$$

where β parameter describes how more recall is important over precision. For $\beta = 1$, the F_β falls back on standard F score which is a harmonic mean of precision and recall.

2.3. Accelerating PoPEx

At each PoPEx iteration k, a new model $\mathbf{m}_k$ is generated. Instead of feeding it directly to the forward solver, a learning scheme can predict if the model's likelihood is significant enough to contribute to the solution of the inverse problem. If the model is not especially useful, it can be discarded; its likelihood is set to 0. Otherwise, if the learning scheme predicts that the model is good enough, the forward solver is called and the exact likelihood of the model is computed. In this way, the expensive forward solver is not called for every model. Then, PoPEx proceeds to draw another model and the procedure is repeated. The discarded models do not contribute to the solution; even if the learning scheme makes a mistake, it will not bias the results. Discarding useful models or marking low-likelihood models as good, would only slow down the convergence of PoPEx but would not introduce incorrect likelihood estimations.

2.3.1. Application of Learning Scheme

The learning scheme needs to be trained before being used with PoPEx. To this end, a sufficient dataset of pairs $(\mathbf{m}_i, L(\mathbf{m}_i))$ must be generated. To achieve this, we first run PoPEx in the normal

mode, for a specified number n_t of iterations, evaluating all models using the forward solver.

Then, the machine learning scheme is trained using the pairs $\{(X_i, y_i), i = 1, \ldots, n_t\}$ with X_i the input data and y_i the classification labels. X_i can be models $\mathbf{m}_i$ or some data obtained by transforming $\mathbf{m}_i$, for example collection indicator variables, or physical properties (porosity, conductivity) derived from $\mathbf{m}_i$. $y_i \in \{0, 1\}$ is a binary label describing if a model is useful (0 meaning an insignificant model with low likelihood and 1 – important model with high likelihood). We propose to sort all available likelihoods: $\{L(\mathbf{m}_i), i = 1, \ldots, n_t\}$, define a ratio $r \in (0, 0.5]$ and set $y_i = 1$ if $L(\mathbf{m}_i)$ is among the rn_t highest likelihoods, and $y_i = 0$ otherwise.

Once the learning scheme has been trained, the classifier is used to classify each new model $\mathbf{m}_j, j > n_t$. Let $\hat{y}_j$ be the prediction of the true label y_j. If $\hat{y}_j = 1$, the model $\mathbf{m}_j$ is fed to the forward solver and its true likelihood is evaluated. If $\hat{y}_j = 0$, we set $L(\mathbf{m}_j) = 0$ and the forward solver is not called. The true likelihood of the model remains unknown and the model does not contribute to the solution of the inverse problem, nor influences the sampling scheme of PoPEx.

This methodology is motivated by the fact that often only very few models have high likelihood (Jäggli et al., 2018). Correcting of the prediction weights was designed to deal with this problem. Therefore, it is justified to discard models with low likelihood, as they would not influence the PoPEx predictions anyway. Applying a perfect classifier in this way, should produce the same results as the original PoPEx algorithm.

A classifier yields false positives and false negatives. False positives (e.g., classifying low-likelihood models as useful) cause that an uninteresting model is being fed to the forward solver. A classifier producing a lot of false positives would have a weaker speed-up abilities. False negatives are more problematic, as it means that good models are discarded. A lot of false negatives would slow down the convergence of PoPEx, as the sampling scheme does not benefit from updating KLD maps and learns more slowly.

2.3.2. Speed-Up Score and Evaluation Metrics

Given the training set for the classifier, it is useful to estimate the possible speed-up when applying it in the PoPEx scheme. We suppose here that we want to achieve an inversion result equivalent (in the sense of prediction error) to running plain (standard) PoPEx for N iterations. Let us compare the total cost of expanding an ensemble of n_t PoPEx models to N models and the total cost of expanding to a bigger ensemble with the machine learning scheme with the equivalent number of significant models. More models are needed to account for the fact that due to the learning scheme, some good models are discarded. To simplify the calculation, we suppose that generated new models have the same proportion of good and bad models and we neglect the fact that using ML scheme influences PoPEx sampling and the quality of generated models.

Let c_m be the computational cost of generating a model $\mathbf{m}$, c_g the cost of running the forward problem solver and we set $c = c_m/c_g$. We assume that the forward solver is more expensive than running the geostatistical model, thus $c < 1$. The total computational cost of expanding the ensemble, e.g. adding $N_t = N - n_t$ models without ML is $N_t \cdot (c_m + c_g)$. Suppose that out of N_t models, ML scheme generates TP true positives ($\hat{y}_i = 1$ and $y_i = 1$), TN true negatives ($\hat{y}_i = 0$ and $y_i = 0$), FP false positives ($\hat{y}_i = 1$ and $y_i = 0$) and FN false negatives ($\hat{y}_i = 0$ and $y_i = 1$). More models need to be generated to account for the fact that some positives are not detected, so the total number of models N_t must be multiplied by (TP + FN)/TP. All true and false positives require the forward solver, thus the total cost with ML will be:

$$\frac{\mathrm{TP} + \mathrm{FN}}{\mathrm{TP}} \left(c_m N_t + c_g (\mathrm{TP} + \mathrm{FP}) \right).$$

We propose to use the ratio of the total computational cost without ML divided by the cost when using ML scheme as a speed-up estimator. Let us call it *s-score* ($\mathrm{s_{score}}$):

$$\mathrm{s_{score}} = \frac{N_t \cdot (c_m + c_g)}{\left[c_m N_t + c_g (\mathrm{TP} + \mathrm{FP}) \right] \cdot (\mathrm{TP} + \mathrm{FN}) / \mathrm{TP}}$$

Using the definitions for precision and recall (5, 6) and the ratio of "good" models $\mathrm{TP} + \mathrm{FN} = rN_t$, we obtain:

$$\mathrm{s_{score}} = \frac{N_t (c_m + c_g)}{\frac{1}{\mathrm{recall}} c_m N_t + c_g \frac{1}{\mathrm{precision}} r N_t} = \frac{1 + c}{\frac{c}{\mathrm{recall}} + \frac{r}{\mathrm{precision}}}. \quad (8)$$

When generating a model is significantly cheaper than running the forward solver, $c \to 0$ and we obtain a convenient approximation:

$$\mathrm{s_{score}} \approx \mathrm{precision}/r, \quad (9)$$

useful to evaluate rapidly if a learning scheme is potentially a good candidate for accelerating PoPEx sampling. If c values are greater than 1, the machine learning scheme is not interesting to apply, even if the recall is high. Smaller r would yield potentially large *s-score* if the precision is high. The *s-score* can also be expressed in terms of F_β score (7):

$$\mathrm{s_{score}} = \frac{1 + c}{r + c} F_\beta,$$

where $\beta^2 = c/r$. In this case β is lower than 1 and close to 0, which attributes more importance to precision than recall.

3. TEST CASE

We consider a 2D synthetic reference field representing a fluvial aquifer. The data used for the inversion is the synthetic tracer breakthrough curve recorded at the pumping well. The reference probabilistic solution for the inverse problem is obtained after 40,000 PoPEx iterations, it will be used to assess the solutions generated using the learning schemes.

3.1. Synthetic Model of a Fluvial Aquifer

The 2D geology is modeled by multiple-point statistics (MPS), which allows to account for subsurface heterogeneity and represent categorical fields (Mariethoz and Caers, 2015). The

MPS algorithm requires a training image (TI), an example of the field which is used as a spatial pattern database. Here, we consider a TI composed of 4 geological facies, representing a fluvial aquifer (**Figure 1**), and first published by Jäggli et al. (2018). The TI has a dimension of 1,000 pixels by 800 pixels with cell size of 5 m by 5 m. To obtain the synthetic reference field (**Figure 2A**), we performed a Direct Sampling (Mariethoz et al., 2010) simulation on a regular grid representing 500 m by 500 m area (100 × 100 pixels). The DeeSse code with multi-resolution capabilities (Straubhaar et al., 2020) was used with the following parameters: 2 pyramid levels (with a pyramid for each indicator variable) with reduction factor 2 at each level and each direction (x, y); search neighborhood radius: 40 in each direction, number of neighboring nodes: 60, distance threshold: 0.01, maximal scan fraction: 0.04. The reference realization was generated with seed value of 201,913. Moreover, points at $x = 374.5$ m, $y = 249.5$ m and $x = 124.5$ m, $y = 249.5$ m are considered to have known facies of category 4, in other words conditioning data is imposed in these two locations with value 4.

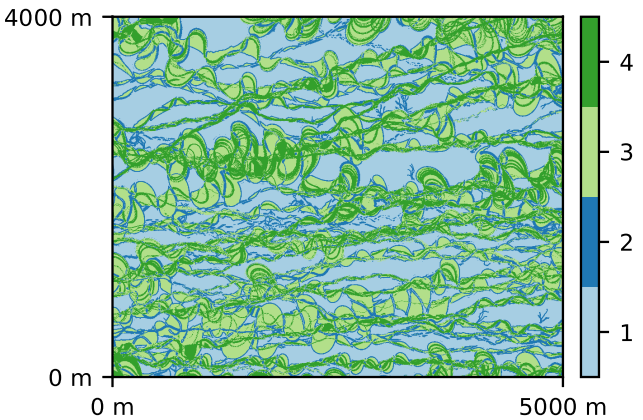

FIGURE 1 | Training image used for generating the synthetic reality and in the inversion process.

3.2. Groundwater Flow and Transport

The aquifer is a confined fluvial aquifer with thickness of 10 m and modeled as a single layer. Each geological facies has a uniquely defined physical properties (hydraulic conductivity, porosity and specific storage) and can be interpreted as different rock type: silt, fine sand, coarse sand, and gravel (**Table 1**). The maps of conductivity, porosity and specific storage on a 100 × 100 regular grid obtained by means of DeeSse simulations are refined by factor of 5 in each direction, resulting in a 500 × 500 m model with cell size 1 m by 1 m. The value of a parameter at each location is equal to its value in the parent cell.

A pumping well is placed at $x = 374.5$ m, $y = 249.5$ m and it pumps water with constant rate of 0.07 m^3/s. The left (west) boundary is at constant head 0.5 m, the right boundary (east) at constant head 0 m and the hydraulic heads at top (north) and bottom (south) boundaries are linearly interpolated between 0.5 m and 0 m. The groundwater flow and transport model was implemented using the flopy python package (Bakker et al., 2016). The steady-state solution of the flow problem is used as initial condition for a transient groundwater flow and transport simulation of a tracer test. The injection well is placed at $x = 124.5$ m, $y = 249.5$ m. The tracer is injected at a constant concentration of 1 kg/m^3 with a constant injection rate of 1 m^3/h, so that a total of 1 m^3 water is injected during 1 h.

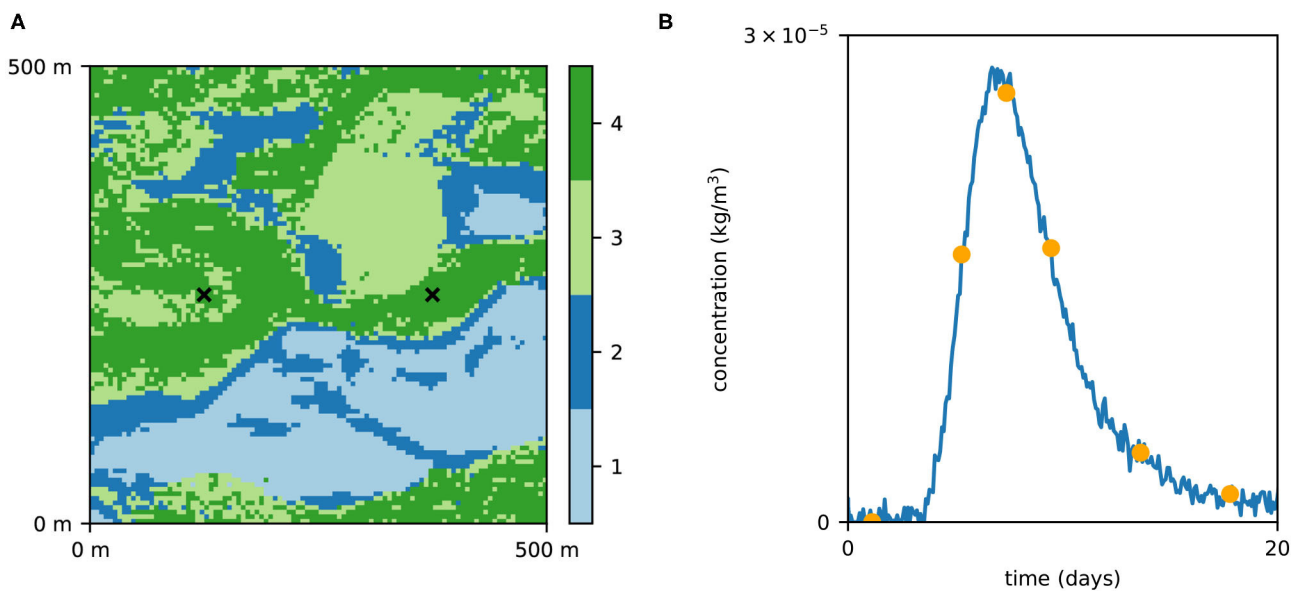

FIGURE 2 | The reference field **(A)** used as the synthetic reality (considered unknown) and the corresponding concentration curve **(B)**. The orange dots correspond to data which were used for likelihood calculation.

TABLE 1 | Physical properties associated with different geological facies.

Facies ordinal	1	2	3	4
Facies	Silt	Fine sand	Coarse sand	Gravel
Conductivity (m/s)	10^{-5}	10^{-4}	10^{-3}	10^{-1}
Porosity	0.40	0.35	0.30	0.25
Specific storage (m^{-1})	10^{-3}	5×10^{-4}	10^{-4}	10^{-5}

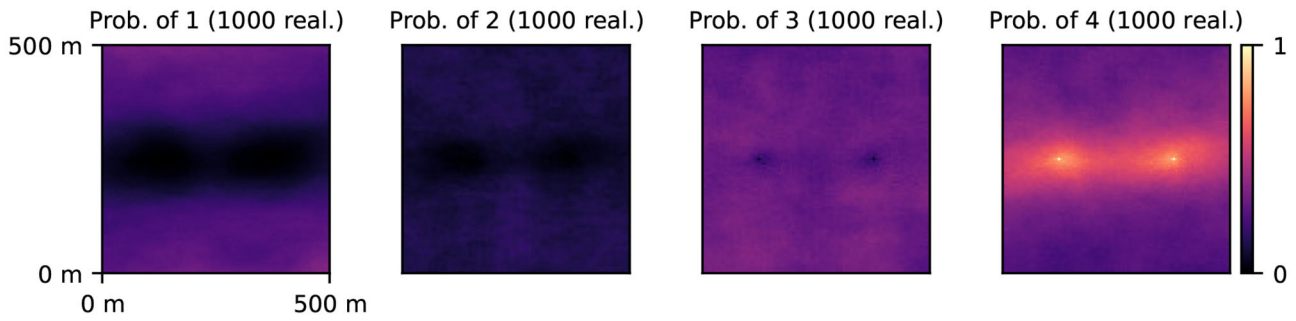

FIGURE 3 | Prior probability maps associated with different geological facies (labeled 1, 2, 3, 4).

The total simulation time is 1 h plus 20 days, discretized in 36 time-steps during initial 1 h and 240 time-steps during the rest of the simulation time. During this period, the flow is modeled in transient regime to account for the perturbation due to the injection. The transport simulation is done with the following parameters: diffusion coefficient of 1×10^{-9} m^2/s; longitudinal dispersivity of 4 m and transversal dispersivity of 0.4 m. The concentration curve at the pumping well was recorded and random Gaussian noise with mean 0 and standard deviation $\sigma_L = 0.05 \times 10^{-5}$ kg/m^3 was added to emulate measurement error (**Figure 2B**).

3.3. Reference Solution of the Inverse Problem

In this scenario, the prior distribution of the geological fields is modeled using the MPS technique with the same parameters as those used to generate the reference realization. Two conditioning points are imposed: the facies 4 (permeable, gravel) at the pumping well and injection well. A prior ensemble of 1,000 realizations was generated and facies probability maps (**Figure 3**) were used as input for the PoPEx algorithm.

The likelihood for this problem should be written as proportional to $\exp\left(-1/2\sigma_L \sum_{i=1}^{n}(\mathbf{g}_i(\mathbf{m}) - \mathbf{d}_i^{obs})^2\right)$ with σ_L the standard deviation of measurement error, $\mathbf{g}(\mathbf{m})$ the simulated values of concentration of the model $\mathbf{m}$, and $\mathbf{d}^{obs}$ the vector of measurements. However, when using this complete formula, the likelihood values are zero for most realizations (considering the fact that numerical representation of floating point numbers has a finite number of digits) and PoPEx converges very slowly (Jäggli et al., 2018). Therefore, to avoid this numerical issue, and following Jäggli et al. (2018), we considered a reduced data set including only six time steps to estimate the likelihood. The "sampled" likelihood formula was: $\exp\left(-1/2\sigma_L \sum_{i\in I}(\mathbf{g}_i(\mathbf{m}) - \mathbf{d}_i^{obs})^2\right)$ with $I = \{50, 100, 125, 150, 200, 250\}$.

PoPEx was run for 40,000 iterations with $l_0 = 100$ and $n_c = 10$. The 10 days groundwater capture zone (van Leeuwen et al., 1998) was predicted with the forward particle tracking module of flopy. The particle tracking was performed on the steady-state solution of the groundwater flow using the original grid composed of 100 × 100 pixels with the cell size of 5 m × 5 m. For each pixel in the domain, a probability of being in the 10 days zone was computed a priori (**Figure 4A**) and a posteriori (**Figure 4B**). **Figure 5** shows the prior and posterior concentration probabilities at the pumping well.

3.4. Application Scenario

While it is possible to apply a classifier directly on a set of models $\mathbf{m}$, in the context of groundwater transport it is favorable to transform the input prior to feeding it to the ML scheme. Below we describe how ML inputs are formed and specify the algorithm parameters.

3.4.1. Transforming Model Into ML Input

The groundwater velocity vector (v_x, v_y) controls advection and hydrodynamic dispersion

$$v_x = -\frac{K}{n_e}\frac{\partial h}{\partial x}, \quad v_y = -\frac{K}{n_e}\frac{\partial h}{\partial y} \tag{10}$$

with K the hydraulic conductivity, n_e the effective porosity and h the hydraulic head. The hydraulic heads are computed using a steady-state MODFLOW (flopy) simulation. Values of conductivity and porosity are defined according to **Table 1**. Two-point numerical derivative is then used to calculate the hydraulic gradient.

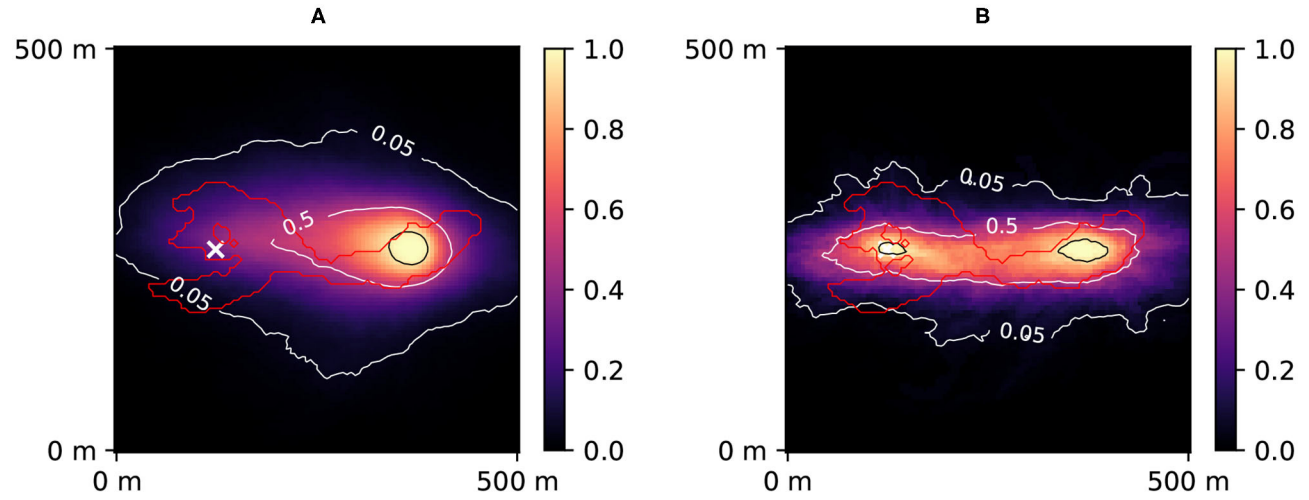

FIGURE 4 | Prior **(A)** and posterior **(B)** probability maps of 10 days zone. The red contour corresponds to the reference 10 days zone (simulated using the synthetic reality; considered unknown). The white cross represents the injection well. The black contour corresponds to 95 % confidence. The posterior was obtained after 40,000 PoPEx iterations.

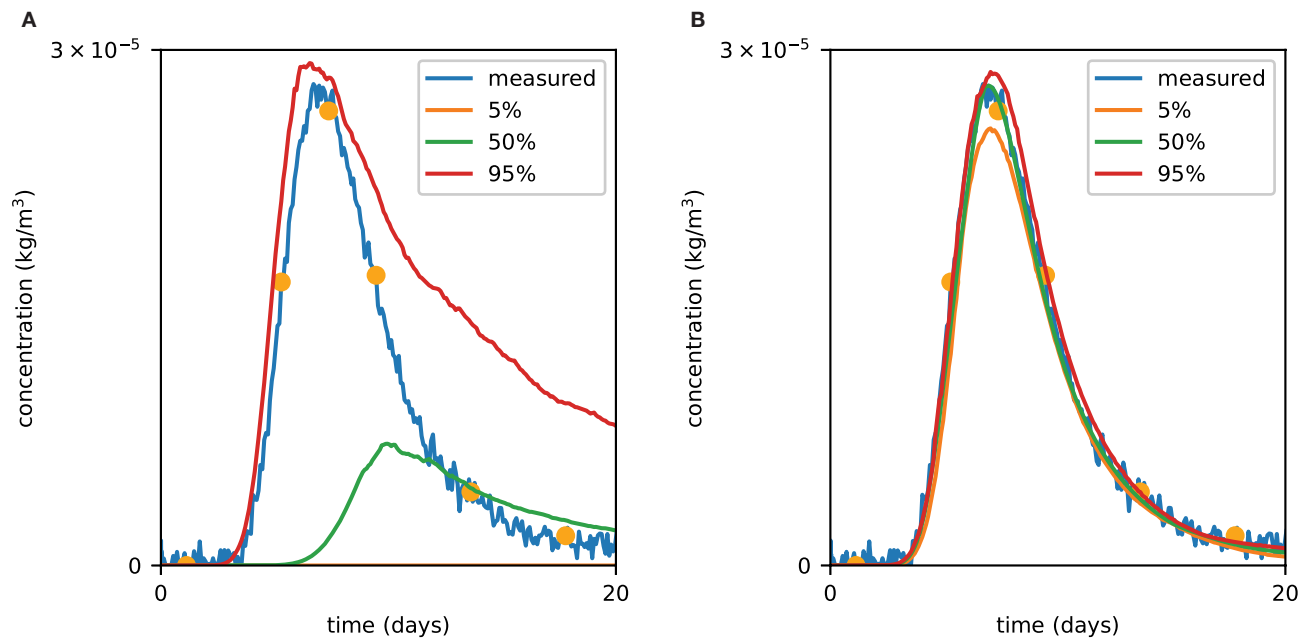

FIGURE 5 | Prior **(A)** and posterior **(B)** tracer concentration curves. The orange dots correspond to data which were used for likelihood calculation. The posterior was obtained after 40,000 PoPEx iterations.

Given the model $\mathbf{m}_i$ which is 100 × 100 pixels, X_i is formed by stacking v_x and v_y (each of size 98 × 98 pixels) and normalizing them to the interval [0,1], using one global minimum and maximum (the maximal/minimal value found in all vectors X_i, $i = 1, \ldots, n_t$). A 98 × 98 image with two channels is formed. The total number of features in the input is 98 × 98 × 2. For AdaBoost and Random Forest the input is flattened, while CNN benefits from the spatial arrangement of the features. The classification labels are obtained by labeling ratio r of best likelihoods as good (1), and the rest not significant (0), as described in section 2.3.1.

3.4.2. Initial PoPEx Ensemble

We generated a PoPEx solution using only 500 iterations (**Figure 6**). The initial ensemble is needed for training the machine learning scheme. The generated set of models with their likelihoods will be used to tune parameters of the machine learning schemes, and to select the best classifier for two different

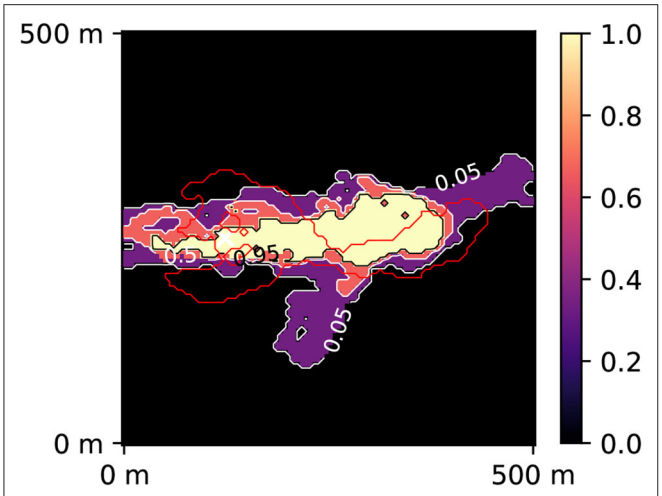

FIGURE 6 | Posterior probability map of 10 days zone obtained after 500 PoPEx iterations. The red contour corresponds to the reference 10 days zone (simulated using the synthetic reality; considered unknown). The white cross represents the injection well.

ratios $r = 0.4$ and $r = 0.2$. To this end, we will apply five-fold cross-validation. For each ratio, parameters of all methods will be re-tuned and the corresponding best classifiers will be used for validating the ML-accelerated PoPEx.

3.5. Test Procedure

To test the methodology, we will compare the results obtained by the standard PoPEx algorithm after $N = 2{,}000$ iterations with the results obtained by ML-accelerated PoPEx algorithm after $N = 4{,}000$ iterations, taking into account computing time and error with respect to the reference solution. We will consider two cases: ratio $r = 0.2$ and $r = 0.4$. This test procedure will be repeated 4 times. Each time with a different initial PoPEx ensemble (neither being the ensemble used for ML parameter tuning).

To evaluate the results, the computing time and discrepancy with the reference solution (40,000 PoPEx iterations) will be computed. The discrepancy will be measured on the prediction: 10-days zone probability maps compared by means of then Jensen-Shannon divergence:

$$J(\widehat{\mu}||\mu^{ex}) = \frac{1}{2}\left(D(\widehat{\mu}||\bar{\mu}) + D(\mu^{ex}||\bar{\mu})\right) \tag{11}$$

with $\widehat{\mu}$ the estimated probability map, μ^{ex} the exact probability map, $\bar{\mu} = (\widehat{\mu} + \mu^{ex})/2$, and $D(\cdot||\cdot)$ Kullback-Leibler divergence (1).

4. RESULTS

In the first test, we perform a comparison of the performances of three machine learning algorithms: AdaBoost, Random Forest, and CNN, and different ratios of models labeled as good (significant) models. This test corresponds to the practical application of the methodology to accelerate PoPEx: the choice of the ML method would be based on limited number of realizations without knowing the true solution. In the second test, we verify how the best performing methods according to the first step are able to accelerate the inversion.

Each inversion task was performed on a computing node composed of 2 Intel(R) Xeon(R) CPU D-1541 @ 2.10GHz CPUs running 64 threads in total. The total RAM of a node is 256 GB. PoPEx was run with 63 processes (workers) and one master process. Running inversion with $N = 500$ iterations required about 24 h to finish. The approximate average time required for completing one geostatistical simulation by the computing node is $c_m = 6.3$ s and for the forward solver: $c_g = 1.7 \times 10^2$ s; $c = c_m/c_g$ is approximately equal to 0.037.

4.1. Hyperparameter Tuning

The hyperparameter tuning was performed with the initial ensemble containing 500 realizations. Depending on the ratio, different number of models were marked as good: 100 models for $r = 0.2$, and 200 models for $r = 0.4$. The validation scores from five-fold cross-validation on these datasets were used to compare the models. The cases of two ratios are treated separately.

4.1.1. Random Forest and AdaBoost

The Random Forest and AdaBoost were implemented using scikit-learn library (Pedregosa et al., 2011). The mean five-fold cross-validation *s-score* on the validation sets was used as performance metric. The Random Forest was tested with the following range of base estimators: 10 to 10,000. The number of samples (measured as fraction of all samples in the training dataset) were varied and scores reported for the ratio $r = 0.4$ (**Figure 7A**) and $r = 0.2$ (**Figure 7C**). The AdaBoost classifier was tested with the learning rates in the range 0.0001 to 0.1 and number of estimators varied between 10 to 1,000. The scores for the ratios $r = 0.4$ and $r = 0.2$ are shown in **Figures 7B,D**, respectively. In the case $r = 0.4$, Random Forest slightly outperformed AdaBoost. The following parameters were selected: number of classifiers 1,000 and fraction of samples 0.2. In the case $r = 0.2$ AdaBoost outperformed Random Forest. The corresponding optimal parameters are: number of estimators 500 and learning rate 0.01.

4.1.2. CNN Architecture Selection

The CNN was implemented using Keras library and the initial architecture (**Table 2**) is an adapted version of the AlexNet (Krizhevsky et al., 2012) to match the input size in our study. We considered the following variants of the initial architecture: included different blocks (1,2,3,4,5) and *kernel factors* (1,2,4,8). Five architecture variants were considered: first, including only block 5; second, including blocks 4,5; third, including blocks 3, 4, 5; etc. As block 4 and 5 are dense layers, the architecture with one and two blocks correspond to feed-forward neural nets. The remaining architectures are CNNs with two dense layers before the output layer. We also introduced a second architectural parameter, *kernel factor*. It was used to divide the kernel size of convolutional layers and number of nodes in the dense layers. A higher number decreases the number of parameters of the neural net. In other words, the *kernel factor* defines a number by which the last column in the **Table 2** is divided. In this way, 20 variants

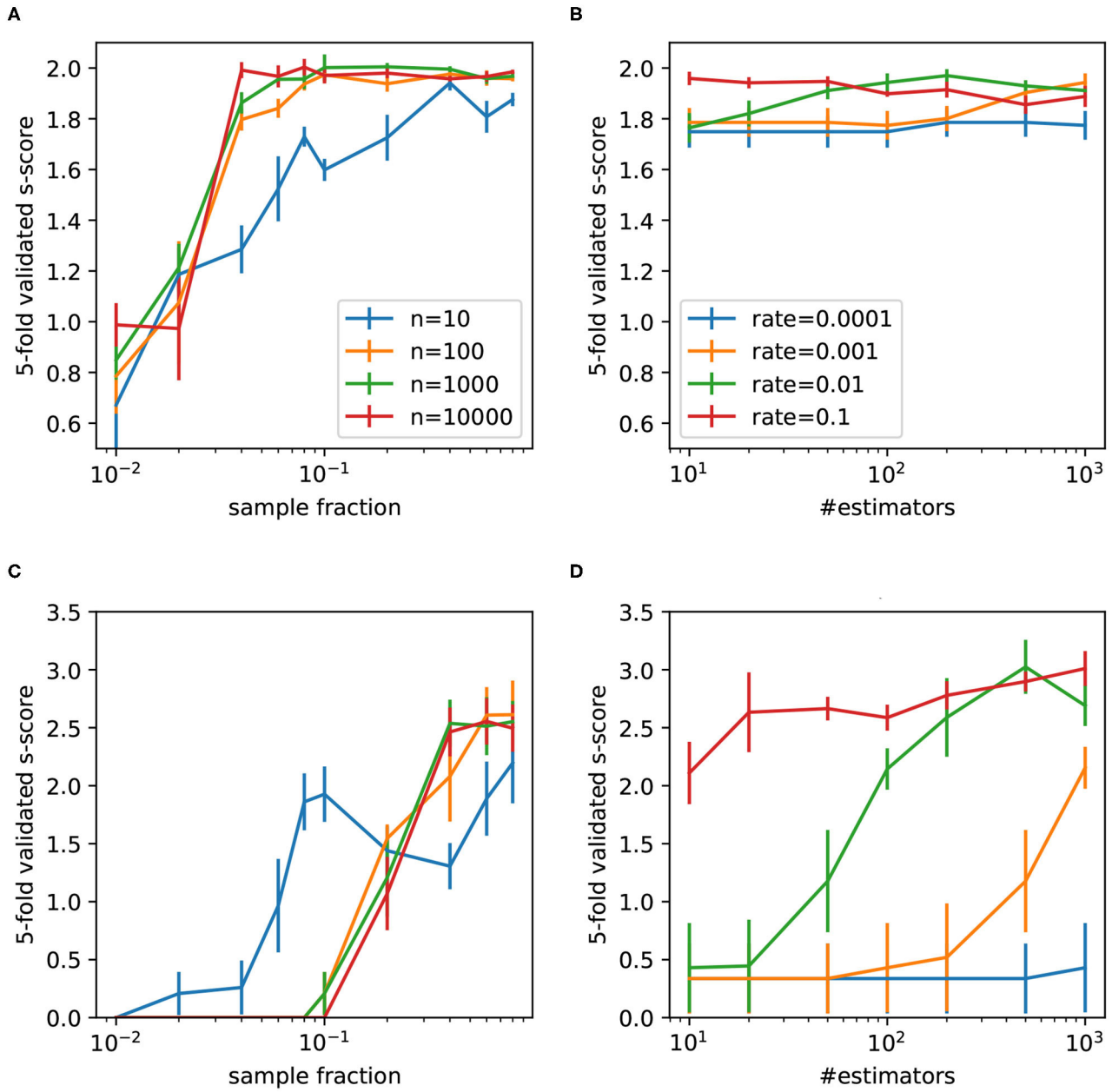

FIGURE 7 | Five-fold cross-validated *s-scores* obtained by Random Forest **(A,C)** and AdaBoost **(B,D)** for two ratios 0.4 (upper row) and 0.2 (lower row). Mean values are reported with the standard error of the mean.

of CNN architecture were formed. For each variant, a five-fold cross-validated learning curve was obtained and the epoch corresponding to the minimal validation loss was noted. The batch size was that of the full training set, and Adam optimizer was used with binary-crossentropy loss and constant learning rate 0.001. The loss is weighted according to class weights: r for the class 0 (not significant models) and $1 - r$ for the significant ("good") models. The weights (as in the case of Random Forest) are used to correct the fact that the training set is not class-balanced; it forces the algorithm to pay more attention to the underrepresented class. Then, the five-fold cross-validated *s-score* after the optimal epoch was retained. This procedure was applied for ratio $r = 0.4$ and $r = 0.2$ (**Figures 8A,B**) respectively. Simpler architectures perform generally better than those containing all the blocks and simple feed-forward nets outperform the more complex architectures.

4.2. Choosing ML Algorithm

The five-fold cross-validated scores of tuned AdaBoost, Random Forest, and CNN algorithms are reported in **Table 3**. The

TABLE 2 | Layers of the convolutional neural network used for binary classification.

Block	Layer type, activation	Kernel/pool size	Output shape	#kernels or #nodes
Block 1	Conv. 2D, ReLU	5 × 5, stride 2 × 2	(47, 47, 24)	24
	Max-pool	3 × 3, stride 2 × 2	(23, 23, 24)	
Block 2	Conv 2D, ReLU	5 × 5, stride 2 × 2	(19, 19, 64)	64
	Max-pool	3 × 3, stride 2 × 2	(9, 9, 64)	
Block 3	Conv 2D, ReLU	5 × 5, stride 2 × 2	(7, 7, 96)	96
	Conv 2D, ReLU	5 × 5, stride 2 × 2	(5, 5, 96)	96
	Conv 2D, ReLU	5 × 5, stride 2 × 2	(3, 3, 64)	64
	Max-pool	3 × 3	(1, 1, 64)	
Block 4	Flatten			
	Dense, ReLU		1,024	1,024
Block 5	Flatten			
	Dense, ReLU		1,024	1,024
Output	Dense, sigmoid		1	1

following mean scores and standard errors of the mean are compared: *s-score* (according to the formula 8), precision, recall, and approximate *s-score* (precision/r). The *s-score* value, as explained in the methodology, can be interpreted as an optimistic estimator for the real speed-up value.

Table 3 shows that the *s-score* and the precision obtained with Random Forest and AdaBoost are higher than those obtained with CNN for both values of r. The recall values are similar for all classifiers for $r = 0.4$, and the recall for $r = 0.2$ is better for CNN than for AdaBoost and Random Forest. It means that AdaBoost and Random Forest were able to achieve a high proportion of good models among all labeled as good, but missed some good models. Consequently, these results confirm that the approximate *s-score* overestimated the actual ones.

4.3. Test of ML-Accelerated PoPEx

Random Forest with 1,000 estimators and samples fraction 0.4 was chosen for validating ML-accelerated PoPEx for ratio of 0.4, and AdaBoost with 500 estimators and learning rate 0.001 for validating ML-accelerated PoPEx for ratio of 0.2. In the validation experiment, PoPEx in the standard mode was run for $N = 2{,}000$ iterations and two ML-accelerated PoPEx versions were run for $N = 4{,}000$ iterations: RF-accelerated PoPEx with $r = 0.4$ and AdaBoost-accelerated PoPEx with $r = 0.2$.

For each of these three PoPEx modes, 4 independent runs were performed and averages used to compute the mean errors with respect to the reference solution. The total running times, and the number of models fed to the forward solver were recorded. The accelerated modes of PoPEx were trained once after $n_t = 500$ iterations and from that iteration, the ML schemes were used.

Example results (1 out of 4) are shown in **Figure 9**, where predictions of 10-days zones after all iterations (2,000 for the standard mode, 4,000 for ML-accelerated) are reported with Jensen-Shannon error maps. The reference solution obtained after 40,000 iterations (**Figure 4**) was used in the Jensen-Shannon formula (11). In this example, the exact solution took 94 h 45 min to compute, the RF-accelerated with $r = 0.4$ took 88 h 12 min, and the AdaBoost-accelerated with $r = 0.2$ only 47 h 40 min. The lowest mean JS error achieved RF-accelerated mode with $r = 0.4$: 0.010, the exact mode: 0.015 and AdaBoost-accelerated mode with $r = 0.2$: 0.014.

Figure 10 shows the convergence of the different PoPEx modes with respect to the number of iterations. As expected, ML schemes slow down the convergence due to incorrect model classifications. Nevertheless, the convergence rate of the RF-accelerated mode with $r = 0.4$ is close to the standard mode. We can also see that this accelerated mode achieves the lowest values of mean error. Not all the generated models in ML-accelerated modes are sent to the forward solver, therefore the time required to achieve a specified number of iterations vary. This is why it is more representative of computing times to plot convergence with respect to the total number of models fed to the forward solver (**Figure 11A**) or to the total computing time (**Figure 11B**). These plots show that both ML-accelerated methods need computing fewer forward models to achieve the same error as the standard PoPEx version (**Figure 11**). These graphs show also that the convergence rate is similar for both values of r. If converted to total computing time, however, the RF-accelerated mode with $r = 0.2$ presents similar convergence rate as the standard mode, because many models need to be generated by the geostatistical method but are rejected. RF-accelerated mode with $r = 0.4$ still requires systematically less computing time than the standard mode.

Figure 12 presents speed-up for different mean J-S errors. The speed-up was calculated as follows. It is the ratio of the computing time of the standard PoPEx algorithm by the computing time of ML-accelerated PoPEx for obtaining the same mean J-S errors. The simulation time was counted from iteration 500 to the iteration by which a specified mean J-S error was achieved. ML-accelerated modes achieve peak speed-ups greater than 2x for large errors, that is at the beginning of inversion procedure. The speed-ups for both ratios are similar and the *s-score* estimates well the real speed-up rate for $r = 0.4$. For $r = 0.2$ the *s-score* overestimated the speed-up.

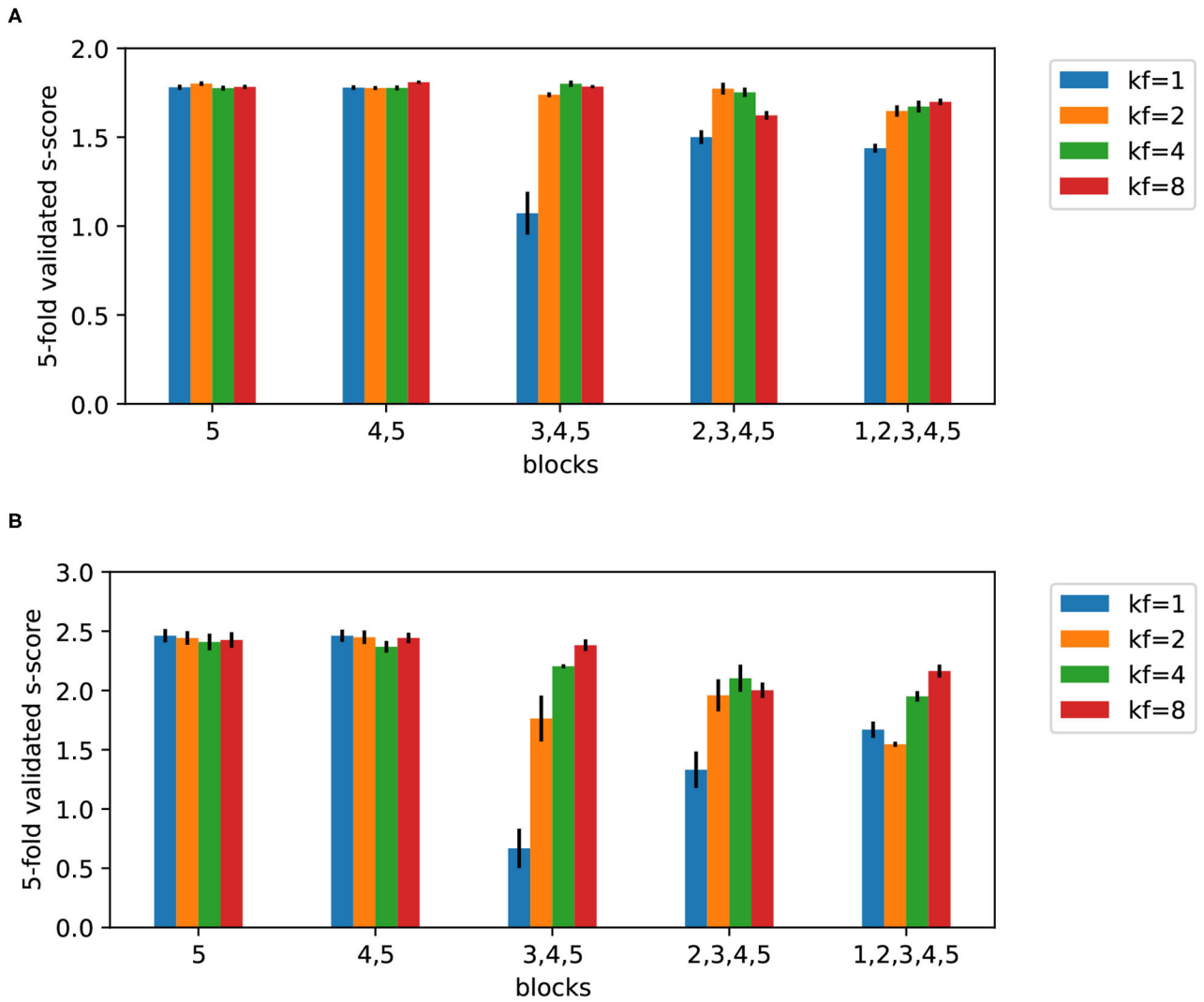

FIGURE 8 | Five-fold cross-validated *s-scores* on test sets obtained by CNN for two ratios 0.4 **(A)** and 0.2 **(B)**, and different kernel factors (kf). Mean values are reported with the standard error of the mean.

TABLE 3 | Five-fold cross-validated performance of classifiers on the initial ensemble containing 500 realizations.

Classifier	r	*s-score*	Precision	Appr. *s-score*	Recall
AdaBoost	0.4	1.98 ± 0.03	0.84 ± 0.02	2.10 ± 0.03	0.78 ± 0.05
Random Forest	0.4	2.01 ± 0.02	0.86 ± 0.02	2.15 ± 0.03	0.77 ± 0.03
CNN	0.4	1.82 ± 0.01	0.76 ± 0.01	1.91 ± 0.03	0.80 ± 0.03
AdaBoost	0.2	3.1 ± 0.3	0.77 ± 0.06	3.9 ± 0.3	0.48 ± 0.06
Random Forest	0.2	2.6 ± 0.2	0.95 ± 0.05	4.8 ± 0.3	0.21 ± 0.04
CNN	0.2	2.48 ± 0.06	0.56 ± 0.03	2.8 ± 0.2	0.63 ± 0.05

5. DISCUSSION AND CONCLUSION

In this article, we introduced a generic framework for accelerating the posterior population expansion algorithm (PoPEx) using machine learning techniques, and tested it on a groundwater transport problem. Our approach is generic and does not interfere with the geostatistical method used for modeling the prior and can be adapted to any type of forward problem. We found that in our set up, both Random Forest and AdaBoost performed well for classifying if generated models are of significance. While AdaBoost and Random Forest performed similarly for the two ratios, we

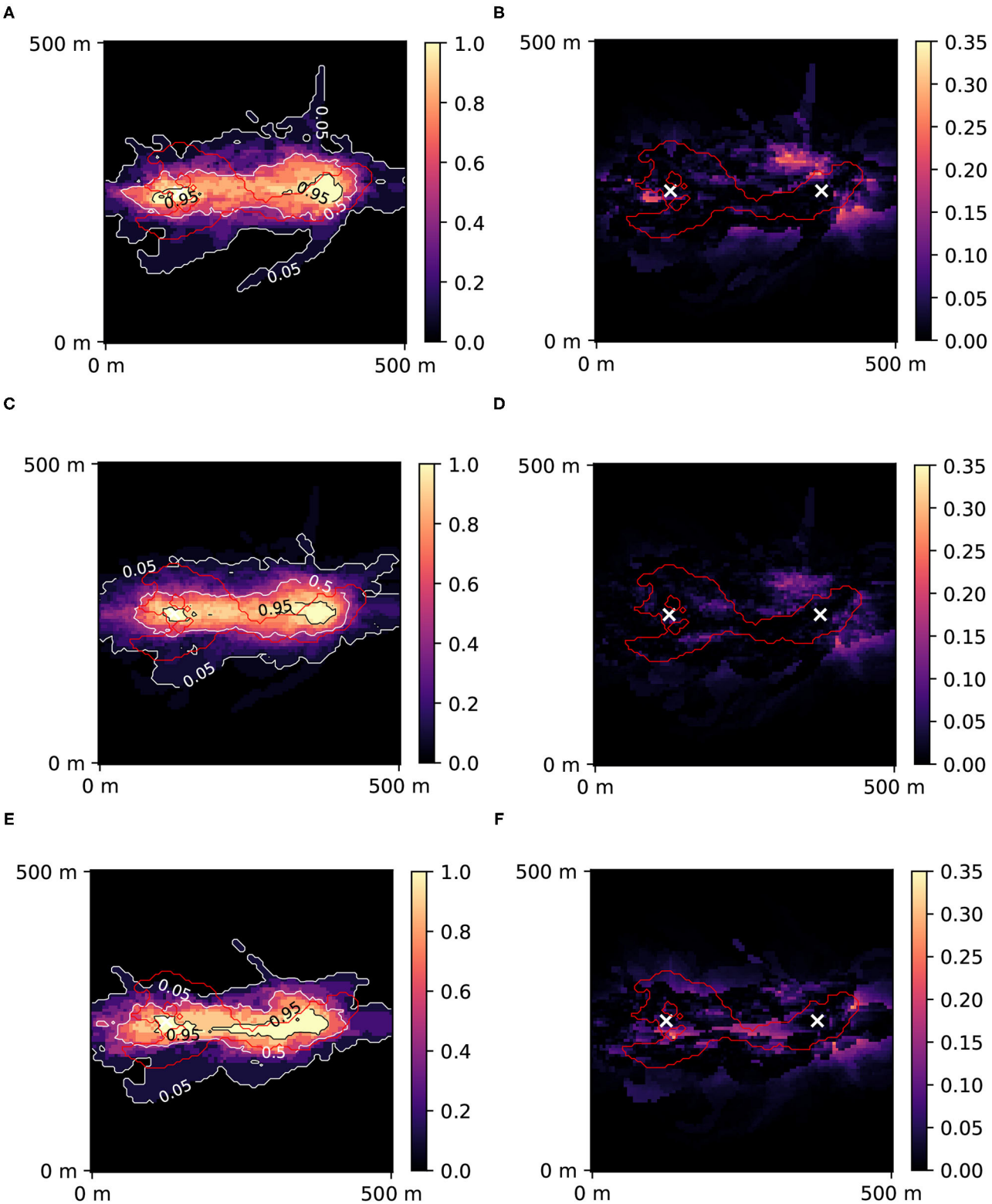

FIGURE 9 | Posterior probability maps of 10 days zone with Jensen-Shannon error maps for standard PoPEx solution with $N = 2{,}000$ **(A,B)**, for RF-accelerated PoPEx with $N = 4{,}000$ and $r = 0.4$ **(C,D)**, and for AdaBoost-accelerated PoPEx with $N = 4{,}000$ and $r = 0.2$ **(E,F)**. The red contour corresponds to the reference 10 days zone. White crosses indicate positions of the injection well (left) and the pumping well (right).

recommend labeling a relatively high ratio of models as significant based on their likelihood—the lower variations and results in more training suffers from stable models. The obtained speedups are close to 2 for this problem, which can represent a significant gain of computing time and resources for large problems.

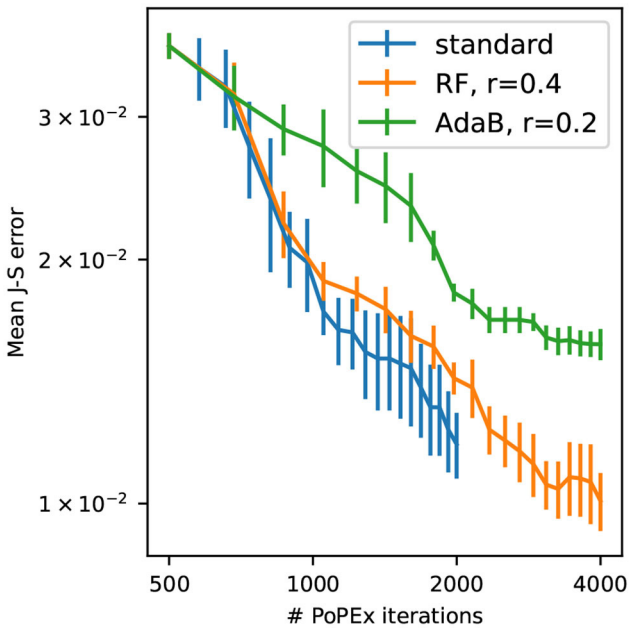

FIGURE 10 | Convergence of standard PoPEx method compared with RF-accelerated PoPEx with ratio $r = 0.4$ and AdaBoost-accelerated with $r = 0.2$. Each point is an average over 4 PoPEx runs, and bars represent the standard error of the mean.

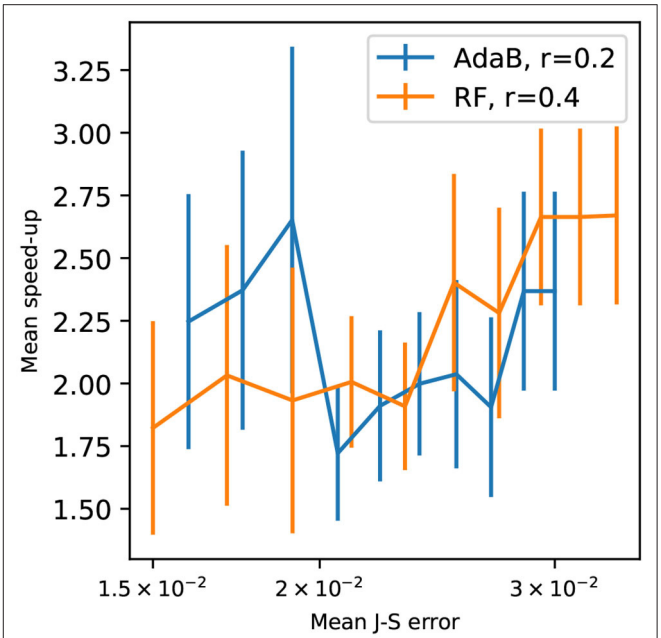

FIGURE 12 | Speed-up with respect to the mean Jensen-Shannon error standard PoPEx : for RF-accelerated PoPEx with $r = 0.4$, and AdaBoost-accelerated with $r = 0.2$. Each point is an average over 4 PoPEx runs, and bars represent the standard error of the mean.

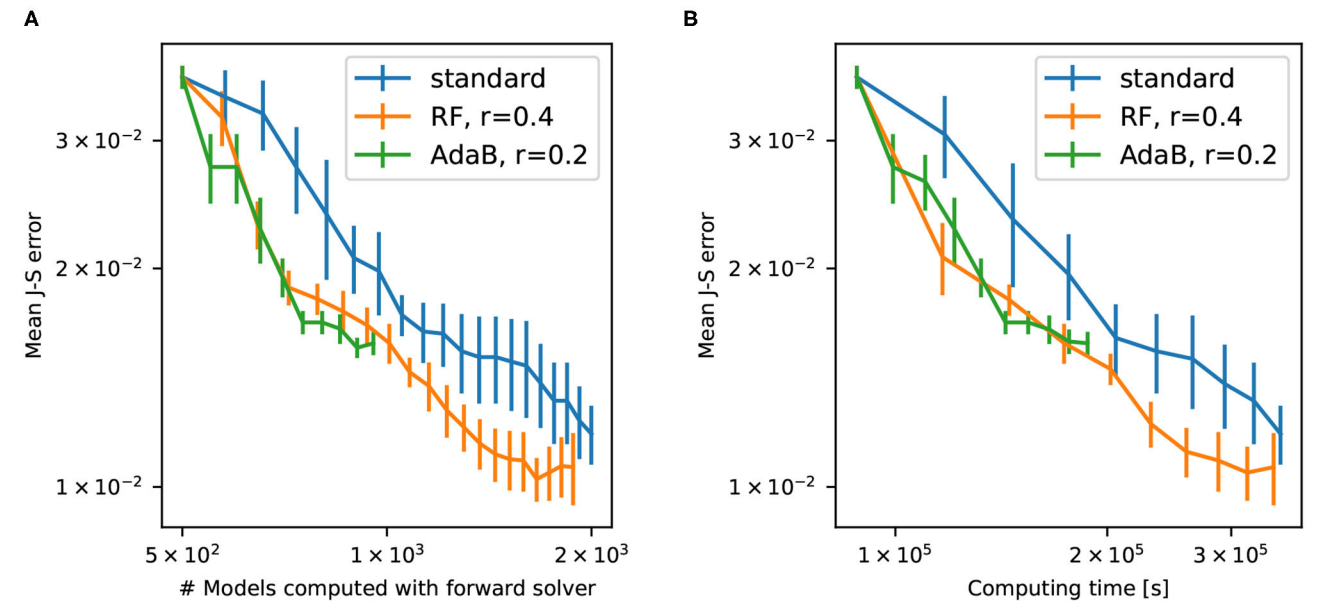

FIGURE 11 | Error with respect to the number of models computed by the forward solver **(A)** and with respect to the computing time **(B)** for standard PoPEx, RF-accelerated PoPEx with $r = 0.4$, and AdaBoost-accelerated with $r = 0.2$. Each point is an average over 4 PoPEx runs, and bars represent the standard error of the mean.

Random Forest and AdaBoost performance were comparable and superior to CNN in the classification task. Kelleher et al. (2015) pointed out that Random Forests perform well when the input has many features, as is the case in our study; AdaBoost performed well in the imbalanced case. The poor performance of CNN is surprising, as it benefits from information about the spatial arrangement of the input data. However, while CNN is very effective in recognizing shapes which are invariant by translation or rotation, in our specific case the absolute position of the features is important to predict the solute transport processes. The fact that simplified CNN architectures (which were reduced to feed-forward neural nets) performed generally better than the more complicated architectures, supports this claim. Further research is needed to determine if it is possible to design a specific Neural Network architecture outperforming the methods presented in this paper. Another important aspect is that the training dataset including only 500 elements is probably too small for the neural network to achieve good performance.

Another challenge for the ML methods is that the labels are arbitrary and depend on the relative values of the likelihood. The generated models are not "good" or "bad" in an absolute sense; they are "better" or "worse" than the others. It is possible that the same model receives different labels depending on other models in the ensemble. The imbalanced nature of the dataset makes it harder to train, especially for low values of the ratio r. We therefore recommend using the ratio $r = 0.4$, as it makes training easier and the performance of the classifier is more robust. Lower ratios are attractive not only from the point of view of the estimated speed-up, but also because lower ratios correspond to fewer good models, which is more relevant in terms of likelihood statistics (few models represent the majority of posterior weights). However, lower ratios suffer from the discrepancy between the estimated speed-up and the actual speed-up. This observation can be explained by the fact that rejecting a good model slows down the convergence of PoPEx. If a classifier missclassifies a good model, the core PoPEx algorithm misses an important information which could guide the sampling. PoPEx then continues sampling from a less-informed distribution, proposing worse models. This is why using a classifier with a low recall (low ratio value) is more penalizing than when using one with a higher recall (higher ratio values).

The proposed classification method could be employed in other Monte Carlo schemes, such as rejection sampling, importance sampling, or Metropolis algorithm (Tarantola, 2005). It could even yield better results than with PoPEx algorithm, as PoPEx is iteratively expanding the ensemble of models by learning from the already sampled models. Simpler samplers without memory of previously generated models could benefit more from the learning schemes.

Finally, an interesting alternative to using classifiers could be to use regression techniques to directly estimate the likelihood value. However, this would be a challenging task, as wrong likelihood estimations could mislead the sampler. It could even lead to slowing it down if irrelevant models would receive erroneously high likelihood. In the present work, erroneous classifications only slow down the inversion but they are not a source of noise.

DATA AVAILABILITY STATEMENT

The data presented in this study and accompanying code can be found below: https://doi.org/10.5281/zenodo.4574602.

AUTHOR CONTRIBUTIONS

PJ designed and implemented the methods, carried out all the coding and numerical experiments, and wrote the paper that was edited by PR. PR supervised the work.

FUNDING

This work was partly funded by the SNF project Phenix (200020_182600/1).

ACKNOWLEDGMENTS

The authors are grateful to the numerous discussions and help received on this project from Christoph Jaggli who was involved in the first initial tests of that approach, Yasin Dagasan who supported the implementation of some early versions of the ML schemes and Julien Straubhaar for the MPS simulations using DeeSse. The authors would like to thank the editor and the reviewers, whose comments helped improve this work.

REFERENCES

Arnold, D., Demyanov, V., Rojas, T., and Christie, M. (2019). Uncertainty quantification in reservoir prediction: Part 1–model realism in history matching using geological prior definitions. *Math. Geosci.* 51, 209–240. doi: 10.1007/s11004-018-9774-6

Bakker, M., Post, V., Langevin, C. D., Hughes, J. D., White, J. T., Starn, J. J., et al. (2016). Scripting modflow model development using python and flopy. *Groundwater* 54, 733–739. doi: 10.1111/gwat.12413

Breiman, L. (2001). Random forests. *Mach. Learn.* 45, 5–32. doi: 10.1023/A:1010933404324

Chan, S., and Elsheikh, A. H. (2020). Parametrization of stochastic inputs using generative adversarial networks with application in geology. *Front. Water* 2:5. doi: 10.3389/frwa.2020.00005

Dagasan, Y., Juda, P., and Renard, P. (2020). Using generative adversarial networks as a fast forward operator for hydrogeological inverse problems. *Groundwater*. 58, 938–950. doi: 10.1111/gwat.13005

Demyanov, V., Pozdnoukhov, A., Christie, M., and Kanevski, M. (2010). "Detection of optimal models in parameter space with support vector machines," in *geoENV VII-Geostatistics for Environmental Applications* (Dordrecht: Springer), 345–358. doi: 10.1007/978-90-481-2322-3_30

Freund, Y., and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.* 55, 119–139. doi: 10.1006/jcss.1997.1504

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. Cambridge, MA: MIT Press.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York, NY: Springer Science Business Media.

Jäggli, C., Straubhaar, J., and Renard, P. (2017). Posterior population expansion for solving inverse problems. *Water Resour. Res.* 53, 2902–2916. doi: 10.1002/2016WR019550

Jäggli, C., Straubhaar, J., and Renard, P. (2018). Parallelized adaptive importance sampling for solving inverse problems. *Front. Earth Sci.* 6:203. doi: 10.3389/feart.2018.00203

Kelleher, J. D., Mac Namee, B., and D'Arcy, A. (2015). *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies*. Cambridge, MA: The MIT Press.

Kohavi, R. (1995). "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2, IJCAI'95* (San Mateo, CA: Morgan Kaufmann Publishers Inc.), 1137–1143.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, Vol. 25, eds F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Red Hook, NY: Curran Associates, Inc.), 1097–1105.

Laloy, E., Herault, R., Jacques, D., and Linde, N. (2018). Training-image based geostatistical inversion using a spatial generative adversarial neural network. *Water Resour. Res.* 54, 381–406. doi: 10.1002/2017WR022148

Laloy, E., Herault, R., Lee, J., Jacques, D., and Linde, N. (2017). Inversion using a new low-dimensional representation of complex binary geological media based on a deep neural network. *Adv. Water Resour.* 110, 387–405. doi: 10.1016/j.advwatres.2017.09.029

Laloy, E., and Jacques, D. (2019). Emulation of cpu-demanding reactive transport models: a comparison of gaussian processes, polynomial chaos expansion, and deep neural networks. *Comput. Geosci.* 23, 1193–1215. doi: 10.1007/s10596-019-09875-y

Laloy, E., Linde, N., Ruffino, C., Herault, R., Gasso, G., and Jacques, D. (2019). Gradient-based deterministic inversion of geophysical data with generative adversarial networks: is it feasible? *Comput. Geosci.* 133:104333. doi: 10.1016/j.cageo.2019.104333

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., et al. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Comput.* 1, 541–551. doi: 10.1162/neco.1989.1.4.541

Linde, N., Renard, P., Mukerji, T., and Caers, J. (2015). Geological realism in hydrogeological and geophysical inverse modeling: a review. *Adv. Water Resour.* 86, 86–101. doi: 10.1016/j.advwatres.2015.09.019

Marçais, J., and de Dreuzy, J.-R. (2017). Prospective interest of deep learning for hydrological inference. *Groundwater* 55, 688–692. doi: 10.1111/gwat.12557

Mariethoz, G., and Caers, J. (2015). *Multiple-Point Geostatistics: Stochastic Modeling with Training Images*. Chichester: Wiley Blackwell. doi: 10.1002/9781118662953

Mariethoz, G., Renard, P., and Straubhaar, J. (2010). The Direct Sampling method to perform multiple-point geostatistical simulations. *Water Resour. Res.* 46. W11536. doi: 10.1029/2008WR007621

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.* 12, 2825–2830. Available online at: https://dl.acm.org/doi/10.5555/1953048.2078195

Rijsbergen, C. J. V. (1979). *Information Retrieval*, 2nd Edn. London: Butterworth-Heinemann.

Shen, C., Laloy, E., Elshorbagy, A., Albert, A., Bales, J., Chang, F.-J., et al. (2018). Hess opinions: incubating deep-learning-powered hydrologic science advances as a community. *Hydrol. Earth Syst. Sci.* 22, 5639–5656. doi: 10.5194/hess-22-5639-2018

Straubhaar, J., Renard, P., and Chugunova, T. (2020). Multiple-point statistics using multi-resolution images. *Stochast. Environ. Res. Risk Assess.* 34, 251–273. doi: 10.1007/s00477-020-01770-8

Tarantola, A. (2005). *Inverse Problem Theory and Methods for Model Parameter Estimation*. Philadelphia, PA: SIAM Society for Industrial and Applied Mathematics. doi: 10.1137/1.9780898717921

Tripathy, R. K., and Bilionis, I. (2018). Deep uq: Learning deep neural network surrogate models for high dimensional uncertainty quantification. *J. Comput. Phys.* 375, 565–588. doi: 10.1016/j.jcp.2018.08.036

van Leeuwen, M., te Stroet, C. B. M., Butler, A. P., and Tompkins, J. A. (1998). Stochastic determination of well capture zones. *Water Resour. Res.* 34, 2215–2223. doi: 10.1029/98WR01552

Zhou, H., Gómez-Hernández, J. J., and Li, L. (2014). Inverse methods in hydrogeology: Evolution and recent trends. *Adv. Water Resour.* 63, 22–37. doi: 10.1016/j.advwatres.2013.10.014

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Copyright © 2021 Juda and Renard. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.