MoMoWo Siymposium 2018-Women's Creativity since the Modern Movement (1918-2018). Toward a New Perception and Reception

geosciences

MDPI

Article

Performance Analysis of Multi-Task Deep Learning Models for Flux Regression in Discrete Fracture Networks

Stefano Berrone [1,2] and Francesco Della Santa [1,2,*]

[1] Dipartimento di Scienze Matematiche (DISMA), Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Turin, Italy; stefano.berrone@polito.it
[2] SmartData@PoliTO Center, Politecnico di Torino, 10138 Torino, Italy
* Correspondence: francesco.dellasanta@polito.it

Abstract: In this work, we investigate the sensitivity of a family of multi-task Deep Neural Networks (DNN) trained to predict fluxes through given Discrete Fracture Networks (DFNs), stochastically varying the fracture transmissivities. In particular, detailed performance and reliability analyses of more than two hundred Neural Networks (NN) are performed, training the models on sets of an increasing number of numerical simulations made on several DFNs with two fixed geometries (158 fractures and 385 fractures) and different transmissibility configurations. A quantitative evaluation of the trained NN predictions is proposed, and rules fitting the observed behavior are provided to predict the number of training simulations that are required for a given accuracy with respect to the variability in the stochastic distribution of the fracture transmissivities. A rule for estimating the cardinality of the training dataset for different configurations is proposed. From the analysis performed, an interesting regularity of the NN behaviors is observed, despite the stochasticity that imbues the whole training process. The proposed approach can be relevant for the use of deep learning models as model reduction methods in the framework of uncertainty quantification analysis for fracture networks and can be extended to similar geological problems (for example, to the more complex discrete fracture matrix models). The results of this study have the potential to grant concrete advantages to real underground flow characterization problems, making computational costs less expensive through the use of NNs.

Keywords: discrete fracture networks; neural networks; deep learning; uncertainty quantification

MSC: 65D40; 68T07; 68T37; 76-10; 76-11

Citation: Berrone, S.; Della Santa, F. Performance Analysis of Multi-Task Deep Learning Models for Flux Regression in Discrete Fracture Networks. *Geosciences* **2021**, *11*, 131. https://doi.org/10.3390/geosciences11030131

Academic Editors: Chaoshui Xu and Jesus Martinez-Frias

Received: 18 January 2021
Accepted: 9 March 2021
Published: 12 March 2021

1. Introduction

Analysis of underground flows in fractured media is relevant in several engineering fields, e.g., in oil and gas extraction, in geothermal energy production, or in the prevention of geological or water-pollution risk, to mention a few. Many possible approaches exist for modeling fractured media, and among the most used is the Discrete Fracture Network (DFN) model [1–3]. In this model, fractures in the rock matrix are represented as planar polygons in a three-dimensional domain that intersect each other; through the intersection segments (called "traces"), a flux exchange between fractures occurs while the 3D domain representing the surrounding rock matrix is assumed to be impermeable. On each fracture, the Darcy law is assumed to characterize the flux and head continuity and flux balance are assumed to characterize all traces.

Underground flow simulations using DFNs can be, however, a quite challenging problem in the case of realistic networks, where the computational domain is often characterized by a high geometrical complexity; in particular, fractures and traces can intersect, forming very narrow angles, or can be very close to each other. These complex geometrical characteristics make the creation of the mesh a difficult task, especially for numerical

methods requiring conforming meshes. Therefore, new methods using different strategies have been proposed in the literature to avoid these meshing problems. In particular, in [4–7], the mortar method is used, eventually together with geometry modifications, while in [8–10], lower-dimensional problems are introduced in order to reduce the complexity. Alternatively, a new method that allowed the meshing process be considered an easy task was illustrated in [11–17]; in this case, the problem was reformulated as a Partial Differential Equation (PDE) constrained optimization one; thanks to this reformulation, totally non-conforming meshes are allowed on different fractures and the meshing process can be independently performed on each fracture. The simulations used in this study are performed with this approach. Other approaches can be found in [18–21].

In real-world problems, full deterministic knowledge of the hydrogeological and geometric properties of an underground network of fractures is rarely available. Therefore, these characteristics are often described through probability distributions, inferred by geological analyses of the basin [22–25]. This uncertainty about the subsurface network of fractures implies a stochastic creation of DFNs, sampling the geometric features (position, size, orientation, etc.) and hydrogeological features from the given distributions; then, the flux and transport phenomena are analyzed from a statistical point of view. For this reason, Uncertainty Quantification (UQ) analyses are required to compute the expected values and variances (i.e., the momentums) of the Quantity of Interests (QoI), e.g., the flux flowing through a section of the network. However, UQ analyses typically involve thousands of DFN simulations to obtain trustworthy values of the QoI momentums [26,27] and each simulation may have a relevant computational cost (both in terms of time and memory). Then, it is worth considering some sort of complexity reduction techniques, e.g., in order to speed up the statistical analyses, such as the multi-fidelity approach [28] or graph-based reduction techniques [29].

Machine Learning (ML), and in particular Neural Networks (NNs), in recent years has been proven to be a potential useful instrument for frameworks related to complexity reduction due to their negligible computational cost in making predictions. Some recent contributions involving ML and NNs applied to DFN flow simulations or UQ analysis are proposed in [30–35]. To the best of the authors' knowledge, other than [35–37] there are no works in the literature that involve the use of NNs as a model reduction method for DFN simulations. In particular, in [35], multi-task deep neural networks are trained to predict the fluxes of DFNs with fixed geometry, given the fracture transmissivities. A well-trained Deep Learning (DL) model can predict the results of thousands of DFN simulations in the order of a second and, therefore, lets a user estimate the entire distribution (not only momentums) of the chosen QoI; the simulations that must be run to generate the training data are the actual computational cost. The results of [35] showed not only that NNs can be useful tools for predicting the flux values in a UQ framework but also that the quality of the flux approximation is very sensitive to some training hyperparameters. In particular, a strong dependence of the performance was observed when the training set size varied.

In this paper, we deeply investigate the dependence of the performances of a trained NN and the size of the training set required for good flux prediction on variance in the stochastic parameter of fracture transmissivities. When variability in the phenomenon increases, good training of an NN requires more and more data. If the data are generated by numerical models, a large number of simulations are necessary for the creation of the dataset involved in training the NN. Then, it can be useful to have a tool that provides an estimate of NN performances for different amounts of training data and for different values of variance in the stochastic input parameters. This issue is relevant to predict the convenience of the approach in real-world applicative situations. Indeed, we recall that the DFN simulations required to generate the training data are the only nonnegligible cost for training NNs on these problems. Therefore, it is important to provide a rule that estimates the number of simulations needed to train an NN model with good performances: if the number of simulations for NN training is less than the number required by a standard UQ

analysis, it is convenient to use an NN reduced model; otherwise, other approaches can be considered.

In this work, we take into account the same flux regression problem described in [35] and we explicitly analyze the performance behavior of the NNs trained for DFN flux regression. The analysis is applied to a pair of DFN geometries and to multiple NNs with different architectures and training configurations, showing interesting behaviors that let us characterize the relationship between the number of training data, the transmissivity standard deviation, and the NN performances. From this relationship, we determine a "UQ rule" that provides an estimate of the minimum number of simulations required for training an NN with a flux approximation error less than an arbitrary $\varepsilon > 0$. The rule is validated on a third DFN, proving concrete efficiency and applicability to real-world problems.

The paper is organized as follows. In Section 2, we start with a brief description of the DFN numerical models and their characterization for the analyses of this work; then, we continue with a short introduction on the framework of NNs in order to better describe the concepts discussed in Section 2.2.3 that concerns the application of deep learning models for flux regression in DFNs. In Section 2.3, the performance analysis procedure used in this work is described step by step. In Section 3, we show the application of the analysis described in the previous section and the results obtained for the two test cases considered; in particular, here, we introduce interesting rules that characterize the error behaviors and that are useful for estimating the minimum number of data required for good NN training. We conclude the work with Sections 4 and 5, where the main aspects of the obtained results are commented upon and discussed.

2. Methods

2.1. Discrete Fracture Networks

We recall here, for the reader's convenience, the model problem of Discrete Fracture Networks (DFNs). The model is described briefly, and for full details, we point the interested reader to [3,10,11,14]. After the model description, we also introduce the main characteristics of the DFNs used for the performance analysis of Neural Networks (NNs) trained for flux regression.

2.1.1. Numerical Model and Numerical Solution

A DFN is a discrete model that describes an underground network of fractures in a rock medium. In a DFN, the network of fractures is represented as a set of intersecting two-dimensional polygons in a three-dimensional space (see Figure 1).

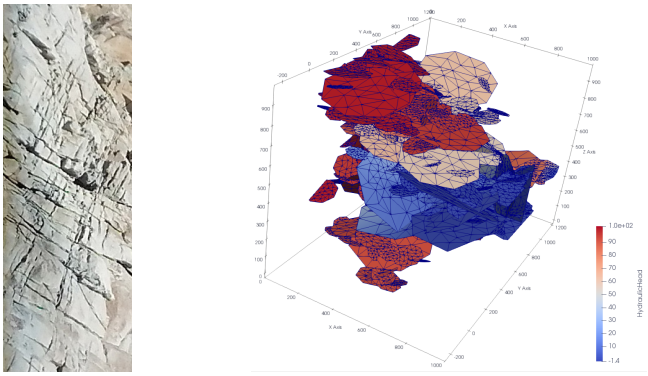

Figure 1. External surface of a natural fractured medium (**left**) and a Discrete Fracture Network (DFN) (**right**).

Each one of the polygons that stands for the network fractures was labeled with an index in a set I, and each fracture was denoted by $\mathcal{F}_i$, with $i \in I$; then, a DFN was given by the union $\bigcup_{i \in I} \mathcal{F}_i$ of all the fractures. The intersections between fractures of the DFN are called *traces*, and through them, the flow crosses the network.

The flow model we assumed in the network was a simple Darcy model, and the DFN numerical simulation consisted in finding for each $i \in I$ the hydraulic head H_i of fracture $\mathcal{F}_i$. In our simulations, the flow was induced by a fixed pressure difference ΔH between two arbitrary surfaces of the 3D domain of the DFN. In order to solve the problem, some matching conditions were imposed at each trace of the network: we assumed the hydraulic head continuity and the flux balance. In this work, the DFN numerical simulations were computed following the approach described in [11,12], reformulating the problem as a PDE-constrained optimization one and using the finite elements as the space discretization method.

The flow simulation in a DFN was characterized by the geometry and by hydrogeological properties. In particular, for each $i \in I$, the transmissivity parameter κ_i of the fracture $\mathcal{F}_i$ characterized the flow facilitation through the fracture. In the most general case, the fracture transmissivity κ_i can be a function of the fracture points, but in this work, we considered it as a constant parameter for each fracture $\mathcal{F}_i$.

In this work, we trained the NN regression models to predict the fluxes exiting from the DFN given the set of transmissivities of its fractures for a fixed geometry and ΔH.

2.1.2. DFN Characterization

In the problem addressed in this work, we considered two DFN geometries and the transmissivities were modelled as random variables with a known distribution.

In particular, each DFN considered in this paper was characterized by the following properties:

- A fixed geometry with $n \in \mathbb{N}$ fractures $\mathcal{F}_1, \dots, \mathcal{F}_n$ in a cubic domain $\mathbb{D} = [0\,,\,\ell]^3 \subset \mathbb{R}^3$ with edge length $\ell = 1000\,\mathrm{m}$. The fractures were assumed to be octagons and randomly generated as in [22,23], i.e., with geometrical features such that we had the following:
  - The fracture radii were sampled with respect to a truncated power law distribution with exponent $\gamma = 2.5$, upper cutoff $r_u = 560$, and lower cutoff $r_0 = 50$.
  - The fracture orientations were sampled from a Fisher distribution with mean direction $\boldsymbol{\mu} = [0.0065, -0.0162, 0.9998]^\top$ and dispersion parameter $\delta = 17.8$.
  - The mass center of fractures were sampled with respect to a uniform distribution in $\mathbb{D}$.
- The Darcy flow model on each fracture was induced by a fixed head difference $\Delta H = 10\,\mathrm{m}$ between the two surfaces of $\mathbb{D}$ corresponding to $x = 0$ and $x = \ell$. More specifically, we imposed a Dirichlet boundary condition $H = 10\,\mathrm{m}$ on fracture edges obtained intersecting the DFN with the plane $x = 0$ and we imposed $H = 0\,\mathrm{m}$ on the fracture edges obtained intersecting the network with the plane $x = \ell$.
- Transmissivities $\kappa_1, \dots, \kappa_n \in \mathbb{R}$ of the n fractures were assumed to be isotropic parameters, modeled as random variables with respect to a log-normal distribution [24,25], i.e., such that
$$\log_{10}(\kappa_i) \sim \mathcal{N}(-5, \sigma)\,, \quad \text{for each } i = 1, \dots, n\,, \tag{1}$$
where $\sigma \in \mathbb{R}$ is an arbitrary parameter that characterizes the standard deviation of the transmissivity distribution.
- Due to the fixed geometry of the DFN, the fracture positions in the space did not change. Then, given the fixed pressure difference ΔH, we had that the boundary fractures with flux entering and exiting the network were the ones intersecting the plane $x = 0$ and the plane $x = \ell$, respectively, independently from the transmissivity values. We denoted with $m \in \mathbb{N}$ the number of boundary fractures with exiting flux.

2.2. Neural Networks

Neural Networks are powerful ML models that were first introduced more than fifty years ago (see, e.g., [38–40]). In the last decade, the usage of NNs has been characterized by

incredible growth, thanks to new and more powerful computer hardware and the increase in available data (see Chapter 1 in [41]).

In this section, we recap briefly the main properties of NNs, and in Section 2.2.3, we describe the multi-task architecture adopted for the flux regression problem in DFNs.

2.2.1. General Concepts about Neural Networks and Learning Models

An NN, similar to most other ML models, is a parametric model $\widehat{\boldsymbol{F}}_{\boldsymbol{w}} : \mathbb{R}^n \to \mathbb{R}^m$, where the vector of parameters (also called *weights*) $\boldsymbol{w}$ is adjusted through an error-minimization process in order to approximate a fixed target function $\boldsymbol{F} : \Omega \subseteq \mathbb{R}^n \to \mathbb{R}^m$, i.e., looking for a final vector of parameters $\boldsymbol{w}_{fin}$ such that $\widehat{\boldsymbol{F}}_{\boldsymbol{w}_{fin}}(\boldsymbol{x}) \approx \boldsymbol{F}(\boldsymbol{x})$ for each $\boldsymbol{x} \in \Omega$. In typical ML problems, the function $\boldsymbol{F}$ is unknown or requires high computational efforts to be executed, justifying the need to find a computationally cheap approximation of it. The parameters $\boldsymbol{w}_{fin}$ are computed starting from a dataset of pairs:

$$\mathcal{D} = \{(\boldsymbol{x}_d\,,\, \boldsymbol{y}_d = \boldsymbol{F}(\boldsymbol{x}_d)) \in \Omega \times \mathbb{R}^m \,|\, d = 1\,, \ldots, D\}\,, \tag{2}$$

where $D \in \mathbb{N}$ and $\mathcal{D}$ are obtained from observation of real data (when $\boldsymbol{F}$ is unknown) or built after random sampling of D elements $\boldsymbol{x}_d$ from the domain Ω and evaluating $\boldsymbol{y}_d$ when $\boldsymbol{F}$ is known but computationally expensive. The search for an optimal weight vector, such that $\boldsymbol{F}$ is approximated by $\widehat{\boldsymbol{F}}_{\boldsymbol{w}}$, is based on a subset $\mathcal{T} \subseteq \mathcal{D}$ called the *training set*. The idea is to find a weight vector that minimizes an arbitrary error measure, e.g., the square of the euclidean norm $\parallel \widehat{\boldsymbol{F}}_{\boldsymbol{w}}(\boldsymbol{x}) - \boldsymbol{y} \parallel^2$ for each pair $(\boldsymbol{x}, \boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{y}) \in \Omega \times \mathbb{R}^m$; to do so, in NNs typically, we solve a stochastic optimization problem of the following form:

$$\min_{\boldsymbol{w}} \Big\{ \text{Loss}(\boldsymbol{w}) = L(\widehat{\boldsymbol{F}}_{\boldsymbol{w}}\,,\, \mathcal{B}) \,|\, \mathcal{B} \subseteq \mathcal{T} \Big\}\,, \tag{3}$$

where the subset $\mathcal{B}$, also called *minibatch*, is a set of $B \in \mathbb{N}$ pairs randomly sampled from $\mathcal{T}$ and $L(\widehat{\boldsymbol{F}}_{\boldsymbol{w}}\,,\, \mathcal{B})$ is the *loss function* on $\mathcal{B}$, e.g., the sum of Mean Square Errors (sMSE) over the pairs of the minibatch:

$$sMSE(\widehat{\boldsymbol{F}}_{\boldsymbol{w}}\,,\, \mathcal{B}) = \sum_{j=1}^{m} \left(\frac{1}{B} \sum_{(\boldsymbol{x},\boldsymbol{y}) \in \mathcal{B}} \left(\widehat{y}_j - y_j\right)^2 \right)\,, \tag{4}$$

where $\widehat{\boldsymbol{y}} := \widehat{\boldsymbol{F}}_{\boldsymbol{w}}(\boldsymbol{x})$ for each $\boldsymbol{x} \in \mathbb{R}^n$, and $\widehat{y}_j$ and y_j are the jth components of $\widehat{\boldsymbol{y}}$ and $\boldsymbol{y}$, respectively, for each $j = 1\,, \ldots, m$ (vectors are denoted by boldface symbols).

The problem (3) is solved with arbitrary variants of the stochastic gradient descent method (see Chapter 8.3 in [41]), i.e., variants of the gradient iterative method that perform a minimization iteratively on a random minibatch $\mathcal{B} \subset \mathcal{T}$ of fixed size B at each step. In particular, the sampling of $\mathcal{B}$ at each step occurs without repetition until all the pairs $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{T}$ have been extracted; once the pairs $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{T}$ are finished, an *epoch* of the training is completed and a new one starts, until it reaches a stopping criterium. An NN model can be trained for many epochs to improve the approximation of $\boldsymbol{F}$.

The remaining pairs of data, i.e., the pairs $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D} \setminus \mathcal{T}$, are used for monitoring and evaluating the quality of the NN training. Usually, the set $\mathcal{D}$ is split in three disjoint subsets:

- the training set $\mathcal{T}$ (already introduced above),
- the *validation set* $\mathcal{V}$, and
- the *test set* $\mathcal{P}$.

The validation set $\mathcal{V}$ is often the smallest of the three subsets, and it is used to measure the approximation error of $\widehat{\boldsymbol{F}}_{\boldsymbol{w}}$ at the end of each training epoch; the test set $\mathcal{P}$, on the contrary, is often bigger than $\mathcal{V}$ (and sometimes also bigger than $\mathcal{T}$), and it is used to measure the approximation quality of the trained NN on new data that the model never used during the training phase. Then, $\mathcal{P}$ is useful to understand if the NN is a good approximation of the target function for new input data, whereas $\mathcal{V}$ is useful to monitor the training activity

and to define the proper stopping criteria for avoiding underfitting/overfitting problems (see Chapter 5.2 in [41]).

2.2.2. Neural Network Structure

The structure of an NN is what actually makes it different from other ML models. An NN is a learning algorithm that can be described through an oriented weighted graph (U, A), where U is the set of nodes and A is the set of edges, i.e., the set of ordered pairs of nodes $a_{ij} = (u_i, u_j) \in A \subseteq U \times U$n each one endowed with a weight $w_{ij} \in \mathbb{R}$.

Each node of an NN, also called a *unit* or a *neuron*, can receive signals either from other nodes (through the edges) or from one external source; in the latter case, the unit is defined as the *input unit* of the NN and it returns as output the same signal received as an input. Each non-input unit $u_j \in U$ performs arbitrary fixed operations on the input signals $x_{i_1}, \ldots, x_{i_N}$ (sent by units $u_{i_1}, \ldots, u_{i_N} \in U$) and returns an output signal x_j. Typically, a unit is characterized as in Figure 2, i.e., with output signal x_j such that

$$x_j = f\left(\sum_{i \in \{i_1, \ldots, i_N\}} x_i w_{ij}\right), \tag{5}$$

where f is an arbitrary function called the *activation function*.

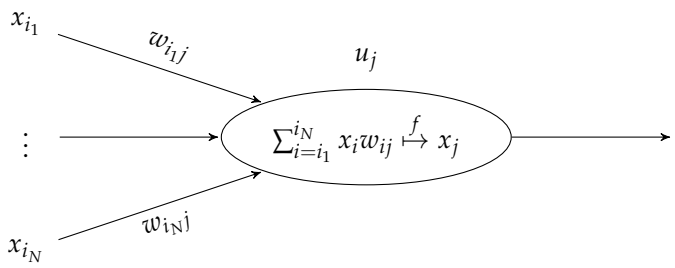

Figure 2. Scheme of a typical Neural Network (NN) unit operation.

If the output x_j of u_j is sent only to other units of the NN, then u_j is defined as a *hidden unit*; otherwise, if the signal x_j is sent "outside" as a general output of the NN model, then u_j is defined as an *output unit*.

One special type of hidden unit is the *bias* units that are characterized by a fixed unit output. For this reason, Equation (5) is rewritten with a different notation that highlights the action of the bias unit:

$$x_j = f\left(\sum_{\substack{i \in \{i_1, \ldots, i_N\} \\ i \neq bias}} x_i w_{ij} + b_j\right), \tag{6}$$

where b_j is the weight of the edge (u_{bias}, u_j) and $u_{bias} \in U$ is a bias unit connected to u_j. Even if it is not forbidden, in practice, no more than one bias unit is connected to one hidden or output unit.

We conclude this brief description of NN structure by introducing the concept of "layers" for NNs. NN architectures organize their units in subsets that interact with each other; these subsets are called *layers* and can be divided in three main types (as the units): the *input layers*, the *hidden layers*, and the *output layers*.

The simplest type of hidden and output layers are the so-called *fully connected* layers. A fully connected layer L that receives signals from a layer I is characterized by the fact that each unit in L is connected to all the units in I and that all the units in L have the

same activation function f; then, assuming that all the units in L are characterized by (6), the layer action of the output signals from I can be described as the function $\mathcal{L}$ such that

$$\boldsymbol{x}^{(L)} = \mathcal{L}\left(\boldsymbol{x}^{(I)}\right) = \boldsymbol{f}\left(W^{\top}\boldsymbol{x}^{(I)} + \boldsymbol{b}\right), \tag{7}$$

where

- $\boldsymbol{x}^{(L)} \in \mathbb{R}^M$ and $\boldsymbol{x}^{(I)} \in \mathbb{R}^N$ are the vectors of the output signals of L and I, respectively;
- $\boldsymbol{f} : \mathbb{R}^M \to \mathbb{R}^M$ is the element-wise application of the activation function f of the layer units;
- $W \in \mathbb{R}^{N \times M}$ is the weight matrix with entry weights w_{ij}, corresponding to the edge that connects unit u_i of I to unit u_j of L; and
- $\boldsymbol{b} \in \mathbb{R}^M$ is the vector of bias weights for the M units of L.

See Figure 3 for a graphical representation of a fully connected layer.

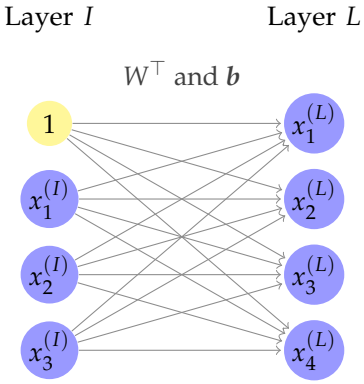

Figure 3. Example of layers I and L fully connected, made of 3 and 4 units, respectively. The bias unit is highlighted in yellow.

Layer formulation in NNs is extremely useful when better describing the function $\widehat{\boldsymbol{F}}_{\boldsymbol{w}}$ as a composition of layer functions and for NN implementation in computer programs. The representation of $\widehat{\boldsymbol{F}}_{\boldsymbol{w}}$ as a composition of functions is used to build a computational graph that is extremely useful in speeding up the computation of both NN outputs and the gradient of the loss function during the training phase. For more details about these properties, see Chapter 6 in [41].

2.2.3. Deep Learning Models for Flux Regression in DFNs

Let us consider a DFN with a geometry defined as in Section 2.1.2, where n is the number of fractures and m is the number of boundary fractures with exiting flux; we recall that the case has fixed geometry and that only the fracture transmissivities change (see (1)). For each vector of transmissivity samples $\boldsymbol{\kappa} = [\kappa_1, \ldots, \kappa_n]^{\top}$, we can run a flow simulation for the given DFN and compute the m fluxes $\boldsymbol{\varphi} = [\varphi_1, \ldots, \varphi_m]^{\top}$ of the m boundary outflowing-flux fractures. Let $\boldsymbol{F} : \mathbb{R}^n \to \mathbb{R}^m$ be a function related to the given DFN such that

$$\boldsymbol{F}(\boldsymbol{\kappa}) = \boldsymbol{\varphi}, \tag{8}$$

for each $\boldsymbol{\kappa} \in \mathbb{R}^n$ sampled following (1); then, we want to define a multi-task architecture for Deep Neural Network (DNN) models able to approximate $\boldsymbol{F}$, i.e., able to predict the corresponding fluxes $\boldsymbol{\varphi} \in \mathbb{R}^m$ for each transmissivity vector $\boldsymbol{\kappa} \in \mathbb{R}^n$ in the input. The DFN flow simulations are performed using the GEO++ software [42], based on a PDE-constrained reformulation of the problem, using finite elements for the spatial discretization. For further details, we point the interested reader to [11,14,42].

For the DNN models of this work, we adopt a multi-task architecture (see Chapter 7.7 in [41]) of the same structure described in [35]. Given a fixed hyperparameter $\alpha \in \mathbb{N}$ and the target function $\boldsymbol{F}$, we define the DNN multi-task architecture $\mathcal{A}_{\alpha}$ (see Figure 4) such that

- $\mathcal{A}_\alpha$ has one input layer L_0 of n units;
- the input layer L_0 is fully connected to the first layer L_1 of a sequence of α fully connected layers: $(L_1, \ldots, L_\alpha)$. All the layers of the sequence have n units characterized by the *softplus* activation function (i.e., $f(x) = \log(1 + e^x)$);
- let us consider m sequences of fully connected layers $(L^{(j)}_{\alpha+1}, \ldots, L^{(j)}_{2\alpha})$, for each $j = 1, \ldots, m$; then, the layer L_α is fully connected to each one of the first layers $L^{(j)}_{\alpha+1}$ of these sequences. All the layers $(L^{(j)}_{\alpha+1}, \ldots, L^{(j)}_{2\alpha})$, for each $j = 1, \ldots, m$, have n units characterized by the softplus activation function; and
- for each $j = 1, \ldots, m$, the layer $L^{(j)}_{2\alpha}$ is fully connected to an output layer $L^{(j)}_{2\alpha+1}$ made of only one linear unit.

We can easily see that the characteristic function of a DNN model with architecture $\mathcal{A}_\alpha$ accepts n inputs and returns m outputs; then, it is a function $\widehat{\boldsymbol{F}}_{\boldsymbol{w}} : \mathbb{R}^n \to \mathbb{R}^m$, and the NN can be trained for approximating the function $\boldsymbol{F}$.

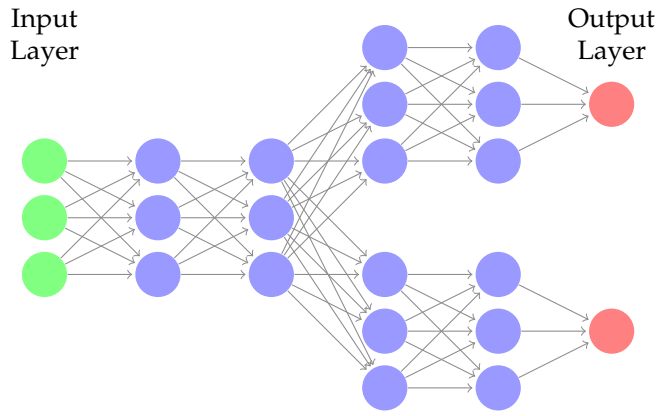

Figure 4. Example of a multi-task architecture $\mathcal{A}_\alpha$ with $n = 3$, $m = 2$, and $\alpha = 2$. For simplicity, bias units are not represented.

In order to evaluate the quality of the approximation, for each outflow fracture, we consider an error measure that evaluates the relative distance between the flux estimated by the NN and the real flux. Let us assume that we have a trained DNN with function $\widehat{\boldsymbol{F}}_{\boldsymbol{w}}$ approximating $\boldsymbol{F}$; then, the chosen error measure is the following:

- **relative error:** for each $\boldsymbol{\kappa} \in \mathbb{R}^n$, we can measure the vector of prediction errors normalized by the actual total exiting flux, i.e., the vector

$$\boldsymbol{e}(\boldsymbol{\kappa}) := \frac{1}{\sum_{j=1}^{m} \varphi_j} \left[\, |\varphi_1 - \widehat{\varphi}_1|, \ldots, |\varphi_m - \widehat{\varphi}_m| \,\right]^\top, \tag{9}$$

  where $\widehat{\boldsymbol{\varphi}} := \widehat{\boldsymbol{F}}_{\boldsymbol{w}}(\boldsymbol{\kappa}) = [\widehat{\varphi}_1, \ldots, \widehat{\varphi}_m]^\top$ is the flux prediction for $\boldsymbol{\kappa}$. Then, given the input $\boldsymbol{\kappa}$, for each $j = 1, \ldots, m$, the jth element of error (9) tells us how much the prediction for the jth exiting flux differs from the original value, represented as a fraction of the total flux outflowing from the DFN.

The errors introduced above, are very useful in studying the prediction ability of a flux regression NN. However, in order to directly compare the predictions of different NNs, we need to define some cumulative scalar values that summarize the approximation quality of the NN models. Then, we introduce a quantity that is obtained from the aggregation of (9) once an arbitrary test set $\mathcal{P}$ of pairs $(\boldsymbol{\kappa}, \boldsymbol{\varphi})$ is given. This quantity is

- **average mean relative error:**

$$E(\mathcal{P}) := \frac{1}{|\mathcal{P}|} \sum_{(\boldsymbol{\kappa}, \boldsymbol{\varphi}) \in \mathcal{P}} \frac{1}{m} \sum_{j=1}^{m} e_j(\boldsymbol{\kappa}), \tag{10}$$

where $e_j(\boldsymbol{\kappa})$ is the jth element of the vector $\boldsymbol{e}(\boldsymbol{\kappa})$. For simplicity, from now on, we will call the quantity $E(\mathcal{P})$ simply *average error* instead of average mean relative error.

2.3. Performance Analysis of Deep Learning Models for Flux Regression

In Sections 2.1 and 2.2, we introduced all the notions needed to analyze the performances of multi-task Deep Learning (DL) models trained to predict the fluxes exiting from a DFN. In [35], the NN sensitivity to the cardinality of the training set $\mathcal{T}$ was noticed, but in those cases, a fixed value $\sigma = 1/3$ for (1) was chosen. In this work, we deeply investigate this sensitivity, quantifying it through the measurement of average errors varying the available number of training data and the sparsity of the data itself.

The analysis performed in this paper compares the average errors $E(\mathcal{P})$ of a set of NNs with a common architecture $\mathcal{A}_\alpha$ and the same configuration of training hyperparameters and functions but trained on different training data; in particular, the differences between the training data are characterized by two hyperperparameters:

- the parameter $\sigma \in \mathbb{R}_{\geq 0}$, characterizing the standard deviation of the transmissivity distribution (see (1)). This parameter varies among a discrete and finite set of values Σ, arbitrarily chosen;
- the number $\vartheta \in \mathbb{N}$ of data $(\boldsymbol{\kappa}, \boldsymbol{\varphi})$ used for training the NN, i.e., the sum of the training set and validation set cardinalities:

$$\vartheta := |\mathcal{T}| + |\mathcal{V}|\,. \tag{11}$$

  Similar to σ, this parameter varies among a discrete and finite set of values Θ, arbitrarily chosen;

In other words, the analysis procedure consists of training $(|\Sigma| \cdot |\Theta|)$ times a fixed untrained NN, each time with respect to a set of training data characterized by a different combination of hyperparameters $(\sigma, \vartheta) \in \Sigma \times \Theta$ (i.e., with different size ϑ and different sparsity, dependent from σ); then, the performances (average errors) of all the new $(|\Sigma| \cdot |\Theta|)$ trained NNs are measured on test sets of the same size and compared, searching for the behavior of average errors with respect to values of σ and ϑ.

2.3.1. Performance Analysis: Method Description

Let us consider a DFN of the type described in Section 2.1.2 and let

$$\Sigma = \{\sigma_1, \ldots, \sigma_s\} \subset \mathbb{R}_{\geq 0} \tag{12}$$

be the set of values for the distribution parameter σ that we want to consider for the NN performance analysis; similarly, let

$$\Theta = \{\vartheta_1, \ldots, \vartheta_t\} \subset \mathbb{N} \tag{13}$$

be the set of values for the number of training data ϑ considered, and let $\rho \in \mathbb{N}$ be the chosen cardinality of the test set $\mathcal{P}$ used for measuring the average errors and the mean divergences.

The method that we use in this work in order to measure and compare the NN performances in flux regression problems for DFNs is characterized by the following steps:

1. Select a DFN, generated as described in Section 2.1.2, with n fractures and where m of them have exiting fluxes.
2. Define the arbitrary sets of values Σ and Θ for the hyperparameters that characterize the analysis.
3. Choose the cardinality $\rho = |\mathcal{P}|$ of the test set.
4. For each $\sigma \in \Sigma$, generate a set of $\delta \in \mathbb{N}$ transmissivity vectors

$$K_\sigma = \{\boldsymbol{\kappa}_1, \ldots, \boldsymbol{\kappa}_\delta\} \subset \mathbb{R}^n\,, \tag{14}$$

where $\delta := (\max(\Theta) + \rho)$ and, for each $\boldsymbol{\kappa} \in K_\sigma$, the transmissivity vector elements have been sampled following (1).

5. For each $\sigma \in \Sigma$, for each $\boldsymbol{\kappa} \in K_\sigma$, compute the corresponding flux vector $\boldsymbol{\varphi} = \boldsymbol{F}(\boldsymbol{\kappa})$ running a DFN flow simulation. Then, for each $\sigma \in \Sigma$, we obtain a dataset

$$\mathcal{D}_\sigma = \{(\boldsymbol{\kappa}_i, \boldsymbol{\varphi}_i) \in K_\sigma \times \mathbb{R}^m \mid \boldsymbol{\varphi}_i = \boldsymbol{F}(\boldsymbol{\kappa}_i)\,,\ \forall\, i = 1, \ldots, \delta\}\,. \tag{15}$$

6. For each $\sigma \in \Sigma$, create a test set $\mathcal{P}_\sigma$ with a random sampling of ρ pairs from $\mathcal{D}_\sigma$.
7. Choose a value $\alpha \in \mathbb{N}$, and build a not trained NN $\mathcal{N}$ with architecture $\mathcal{A}_\alpha$.
8. For each $\sigma \in \Sigma$, for each $\vartheta \in \Theta$, sample randomly a number ϑ of pairs $(\boldsymbol{\kappa}, \boldsymbol{\varphi})$ from $\mathcal{D}_\sigma \setminus \mathcal{P}_\sigma$ (i.e., the dataset without the test set). We decides to use 20% of the ϑ sampled pairs as the validation set $\mathcal{V}_\sigma^\vartheta$ and the remaining 80% as the training set $\mathcal{T}_\sigma^\vartheta$.
9. For each pair $(\sigma, \vartheta) \in \Sigma \times \Theta$, train the untrained NN $\mathcal{N}$ using the data of $\mathcal{T}_\sigma^\vartheta$ and $\mathcal{V}_\sigma^\vartheta$, obtaining a trained NN $\mathcal{N}_\sigma^\vartheta$. For all the cases, the training is characterized by the same hyperparameters and functions arbitrarily chosen.
10. For each pair $(\sigma, \vartheta) \in \Sigma \times \Theta$, measure the quantity E_σ^ϑ that is the average error $E(\mathcal{P}_\sigma)$ computed for the NN $\mathcal{N}_\sigma^\vartheta$.
11. Analyze the set of points

$$\mathrm{E} := \left\{(\sigma, \vartheta, E_\sigma^\vartheta) \in \Sigma \times \Theta \times \mathbb{R}\right\}, \tag{16}$$

and then find the best fitting function $\widehat{E} : \mathbb{R}^2 \to \mathbb{R}$ with respect to the points in E such that they characterize the average error as a function of the parameters σ and ϑ.

2.3.2. Training Hyperparameters and Functions

For step 9 of the method described above, we talk about a fixed and arbitrary configuration of the hyperparameters and functions of the training phase. In particular, in this work, we perform the analysis considering two cases with two fixed configurations that are different only for the minibatch size adopted; we have that the first configuration is characterized by a minibatch size $|\mathcal{B}| = 10$ while the second one has a minibatch size $|\mathcal{B}| = 30$.

Let β be a parameter denoting which training configuration we choose; then, $\beta = 1$ represents the choice for the first configuration ($|\mathcal{B}| = 10$) and $\beta = 2$ represents the choice for the second configuration ($|\mathcal{B}| = 30$). All the other properties of the training configurations are the same for both the cases $\beta = 1$ and $\beta = 2$ and are as follows:

- **input data preprocessing:** the input data are transformed applying first the function $\log_{10}$ (element-wise) and then the z-normalization [43];
- **output data preprocessing:** the output data are rescaled by a factor equal to 10^6;
- **layer-weights initialization:** Glorot uniform distribution [44];
- **biases initialization:** zeroes;
- **maximum number of epochs:** 1000;
- **regularization methods:** early stopping, Chapter 7.8 in [41] (patience parameter $p = 150$);
- **optimization algorithm:** Adam [45] (learning rate $\epsilon = 0.001$, first moment decay parameter $\gamma_1 = 0.9$, and second moment decay parameter $\gamma_2 = 0.999$);
- **loss function:** sMSE (see (4)).

The shared hyperparameters and functions chosen for the configurations $\beta = 1, 2$ consist mainly in the default options provided by most of the frameworks for NN implementation; indeed, in this analysis, we focus our attention on the effects of the parameters σ and ϑ on the NNs and, therefore, we choose standard training configurations that should grant a reasonable training quality.

3. Results

Here, we show the application of the performance analysis method described in Section 2.3 on two test cases. In particular, we consider two DFNs, DFN158, and DFN395, generated with respect to the characterization of Section 2.1.2; the total number of fractures n is equal to 158 and 395 for DFN158 and DFN395, respectively, and the number of outflux fracture m is equal to 7 and 13 for DFN158 and DFN395, respectively.

For each of these two DFNs, we train three different NNs with architectures $\mathcal{A}_\alpha$ (see Section 2.2.3) for each $\alpha = 1, 2, 3$ and with respect to the two training configurations $\beta = 1$ and $\beta = 2$ (see Section 2.3.2); then, we have a total number of six trained NNs, one for each (α, β) combination, for both DFN158 and DFN395. Moreover, we fixed the values $\rho = 3000$ for the test set cardinality and the set $\Theta = \{500, 1000, 2000, 4000, 7000\}$ of training-validation set cardinalities ϑ. For the two DFNs considered, we define the set of distribution parameters σ such that

DFN158: $\Sigma = \{1/5 = 0.20,\ 1/4 = 0.25,\ 1/3 \approx 0.33,\ 2/5 = 0.40,\ 1/2 = 0.50,\ 0.70\}$.

DFN395: $\Sigma = \{1/5 = 0.20,\ 1/3 \approx 0.33,\ 1/2 = 0.5\}$.

In total, for the following analyses, we trained 180 NNs for DFN158 (30 for each (α, β) case) and 90 NNs for DFN395 (15 for each (α, β) case); the reason for the smaller set Σ and, therefore, a smaller number of trainings for DFN395 depends on the more expensive DFN simulations (with respect to the ones of DFN158) that are needed for the creation of the dataset $\mathcal{D}_\sigma$ (see step 5 of the method in Section 2.3). The results found for the two DFNs are in very good agreement.

The analysis was performed for different combinations of the parameters α and β in order to show that the results found are general for the family of NNs. After these performance analyses, in Section 3.3, we describe the rules for the best choice of ϑ value given a σ value.

3.1. DFN158

Given the 180 trained NNs with respect to the datasets $\mathcal{T}_\sigma^\vartheta$ and $\mathcal{V}_\sigma^\vartheta$ of DFN158, we analyzed the set of points E (see (16)) for any fixed combination $(\alpha, \beta) \in \{1, 2, 3\} \times \{1, 2\}$; this set of points is described in Table 1 and illustrated in Figure 5.

Table 1. DFN158. Table of the E_σ^ϑ values, varying $(\sigma, \vartheta) \in \Sigma \times \Theta$, for each NN of architecture $\mathcal{A}_\alpha$ trained with configuration β, for each $\alpha = 1, 2, 3$, and for each $\beta = 1, 2$.

E_σ^ϑ		$\alpha = 1$					$\alpha = 2$					$\alpha = 3$				
	σ/ϑ	500	1000	2000	4000	7000	500	1000	2000	4000	7000	500	1000	2000	4000	7000
$\beta = 1$	0.20	0.0097	0.0076	0.0066	0.0063	0.0055	0.0104	0.0074	0.0065	0.0046	0.0028	0.0106	0.0077	0.0055	0.0047	0.0036
	0.25	0.0132	0.0097	0.0086	0.0080	0.0073	0.0129	0.0099	0.0085	0.0068	0.0044	0.0134	0.0103	0.0081	0.0068	0.0051
	∼0.33	0.0190	0.0148	0.0129	0.0119	0.0109	0.0190	0.0148	0.0128	0.0109	0.0081	0.0205	0.0154	0.0130	0.0107	0.0084
	0.40	0.0248	0.0196	0.0171	0.0160	0.0136	0.0263	0.0184	0.0166	0.0150	0.0119	0.0264	0.0193	0.0166	0.0146	0.0120
	0.50	0.0494	0.0423	0.0369	0.0335	0.0288	0.0515	0.0410	0.0355	0.0308	0.0267	0.0539	0.0434	0.0345	0.0307	0.0262
	0.70	0.2366	0.2066	0.1837	0.1687	0.1538	0.2434	0.1942	0.1787	0.1515	0.1451	0.2406	0.1991	0.1629	0.1477	0.1438
$\beta = 2$	0.20	0.0103	0.0077	0.0064	0.0055	0.0047	0.0103	0.0072	0.0062	0.0049	0.0033	0.0106	0.0066	0.0057	0.0042	0.0032
	0.25	0.0136	0.0102	0.0085	0.0074	0.0063	0.0131	0.0097	0.0078	0.0068	0.0047	0.0139	0.0096	0.0077	0.0063	0.0043
	∼0.33	0.0199	0.0150	0.0128	0.0115	0.0093	0.0195	0.0147	0.0126	0.0010	0.0079	0.0206	0.0140	0.0116	0.0103	0.0074
	0.40	0.0258	0.0195	0.0172	0.0157	0.0133	0.0262	0.0184	0.0162	0.0149	0.0113	0.0259	0.0199	0.0158	0.0137	0.0115
	0.50	0.0512	0.0428	0.0366	0.0336	0.0278	0.0506	0.0408	0.0334	0.0296	0.0247	0.0523	0.0412	0.0345	0.0318	0.0242
	0.70	0.2650	0.2115	0.2221	0.1740	0.1534	0.2304	0.1942	0.1689	0.1518	0.1293	0.2329	0.1871	0.1591	0.1436	0.1315

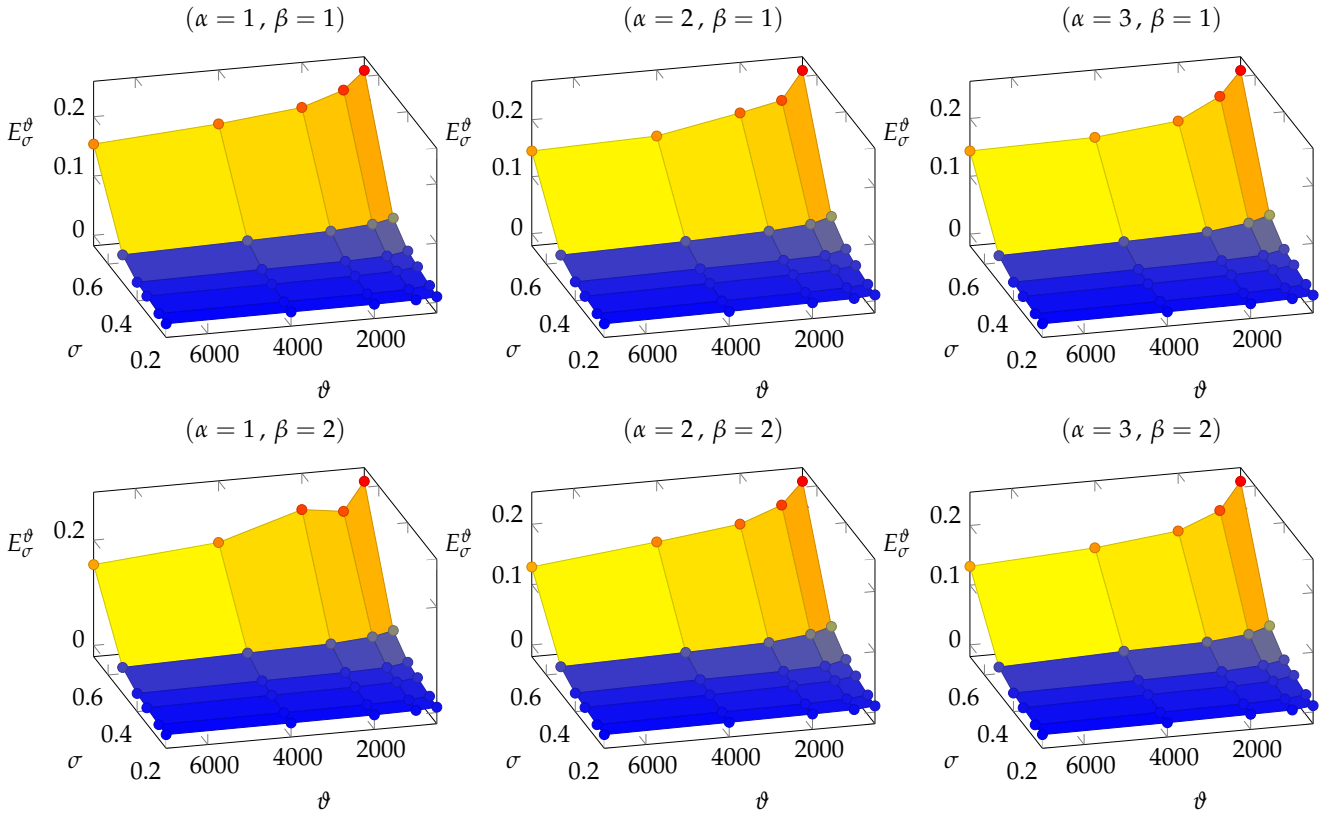

Figure 5. DFN158. Three-dimensional plot of the E^{ϑ}_{σ} values, varying $(\sigma, \vartheta) \in \Sigma \times \Theta$, for each NN of architecture $\mathcal{A}_{\alpha}$ trained with configuration β, for each $\alpha = 1, 2, 3$, and for each $\beta = 1, 2$.

For the average errors E^{ϑ}_{σ}, we observed the following behavior characteristics:

1. The general trend of E^{ϑ}_{σ} decreases with respect to ϑ and increases with respect to σ. Indeed, higher values of ϑ provide more data for better training the NN whereas higher values for σ mean a larger variance for input data and, therefore, a more difficult target function $\boldsymbol{F}$ to be learned.
2. Keeping the value of σ fixed, we observed that, in the logarithmic scale, the values of E^{ϑ}_{σ} are inversely proportional to ϑ (see Figure 6-left).
3. Keeping the value of ϑ fixed, we observe that, in the logarithmic scale, the values of E^{ϑ}_{σ} increase with respect to σ (see Figure 6-right), with an almost quadratic behavior with respect to σ.

The numerical results and these observations actually suggest that the performances of an NN for flux regression seem to be characterized by well-defined hidden rules. Therefore, as proposed at the end of step 11 of the method, we sought a function $\widehat{E}(\sigma, \vartheta)$ such that

$$\widehat{E}(\sigma, \vartheta) \approx E^{\vartheta}_{\sigma}\,, \tag{17}$$

for each $(\sigma, \vartheta) \in \Sigma \times \Theta$.

Taking into account the observations at items 2 and 3, we decided to look for $\widehat{E}(\sigma, \vartheta)$ among the set of exponential functions characterized by exponents inversely proportional to ϑ and proportional to σ with linear or quadratic behavior, i.e., functions with the following expressions:

$$g_1(\sigma, \vartheta) = e^{\left(c_1 + \frac{c_2}{\vartheta} + c_3\sigma + c_4\sigma^2\right)} \quad \text{and} \quad g_2(\sigma, \vartheta) = e^{\left(c_1 + \frac{c_2}{\vartheta} + c_3\sigma\right)}\,, \tag{18}$$

where $c_1, c_3 \in \mathbb{R}$ and $c_2, c_4 \in \mathbb{R}_{\geq 0}$ are parameters of the functions.

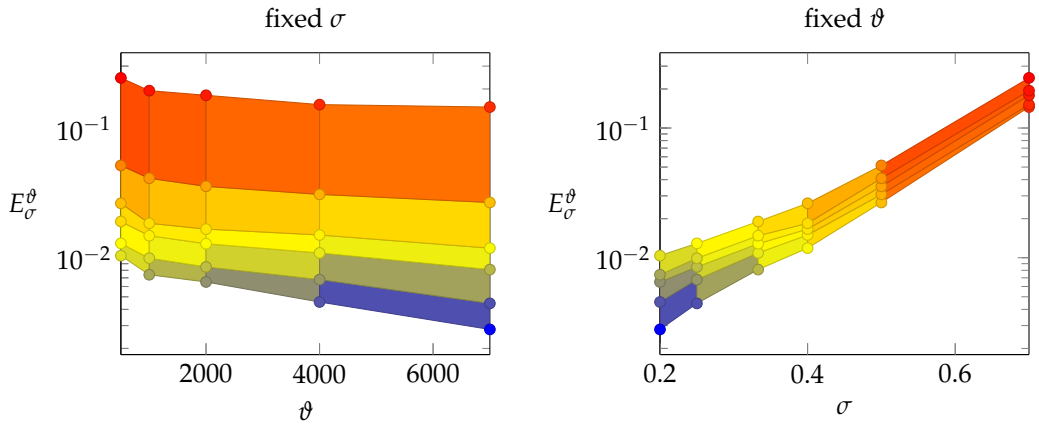

Figure 6. DFN158. Example for the case $(\alpha = 2, \beta = 1)$. Three-dimensional plot projections in order to observe the behavior of E_σ^ϑ while maintaining fixed σ (**left**) or ϑ (**right**). E_σ^ϑ points are reported in logarithmic scale.

Through a least square error minimization process, we found the best-fitting coefficients for the functions (18) with respect to the data points E_σ^ϑ (see Table 2). Looking at the results, we see that the observation made at item 3 concerning the quadratic behavior of E_σ^ϑ with respect to σ is confirmed; indeed, the approximation error of g_1 is always smaller than the one of g_2 (with a nonzero coefficient c_4). Then, we have that a good function $\widehat{E}(\sigma, \vartheta)$ for the characterization of the average errors is

$$\widehat{E}(\sigma, \vartheta) := e^{(\widehat{c}_1 + \frac{\widehat{c}_2}{\vartheta} + \widehat{c}_3\sigma + \widehat{c}_4\sigma^2)}, \tag{19}$$

where $\widehat{c}_1, \ldots, \widehat{c}_4$ are the fixed parameters obtained with the least square minimization.

Table 2. DFN158. Mean Square Error (MSE) and coefficients of the least square minimization with respect to E_σ^ϑ.

		$\alpha = 1$					$\alpha = 2$					$\alpha = 3$				
	E_σ^ϑ	**MSE**	c_1	c_2	c_3	c_4	**MSE**	c_1	c_2	c_3	c_4	**MSE**	c_1	c_2	c_3	c_4
$\beta = 1$	g_1	1.848×10^{-2}	-5.678	206.3	1.869	5.139	1.802×10^{-2}	-5.928	261.0	2.551	4.535	1.778×10^{-2}	-5.969	276.3	2.843	4.129
	g_2	2.255×10^{-2}	-7.188	206.3	7.649	-	2.109×10^{-2}	-7.262	260.9	7.628	-	2.054×10^{-2}	-7.177	276.2	7.457	-
$\beta = 2$	g_1	4.014×10^{-2}	-5.652	233.1	1.515	5.742	2.521×10^{-2}	-5.884	262.3	2.342	4.653	1.709×10^{-2}	-6.084	285.8	3.171	3.785
	g_2	4.262×10^{-2}	-7.356	233.0	7.966	-	2.757×10^{-2}	-7.248	262.2	7.545	-	1.936×10^{-2}	-7.190	285.5	7.399	-

We conclude this section with a visual example (Figure 7) of the fitting quality of $\widehat{E}(\sigma, \vartheta)$ for the values E_σ^ϑ of the case $(\alpha = 2, \beta = 2)$.

3.2. DFN395

Given the 90 trained NNs with respect to the datasets $\mathcal{T}_\sigma^\vartheta$ and $\mathcal{V}_\sigma^\vartheta$ of DFN395, we analyzed the set of points E for any fixed combination $(\alpha, \beta) \in \{1, 2, 3\} \times \{1, 2\}$. This set of points is described in Table 3 and illustrated in Figure 8.

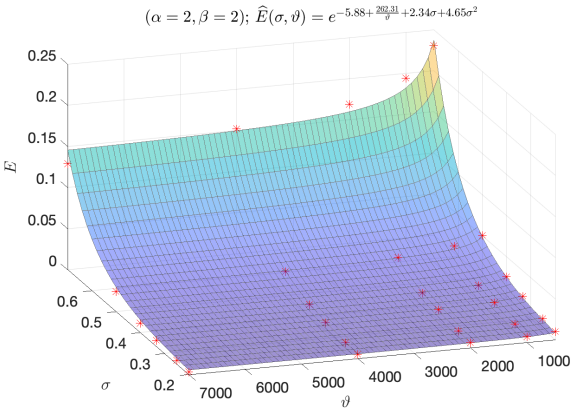

Figure 7. DFN158. Example for the case $(\alpha = 2, \beta = 2)$. Plot of the function $\widehat{E}(\sigma, \vartheta)$ and the data points E^{ϑ}_{σ} (red stars).

Table 3. DFN395. Table of the E^{ϑ}_{σ} values, varying $(\sigma, \vartheta) \in \Sigma \times \Theta$, for each NN of architecture $\mathcal{A}_{\alpha}$ trained with configuration β, for each $\alpha = 1, 2, 3$, and for each $\beta = 1, 2$.

E^{ϑ}_{σ}		$\alpha = 1$					$\alpha = 2$					$\alpha = 3$				
	σ/ϑ	**500**	**1000**	**2000**	**4000**	**7000**	**500**	**1000**	**2000**	**4000**	**7000**	**500**	**1000**	**2000**	**4000**	**7000**
$\beta = 1$	0.20	0.0132	0.0079	0.0055	0.0047	0.0046	0.0143	0.0110	0.0067	0.0050	0.0037	0.0145	0.0110	0.0073	0.0056	0.0040
	∼0.33	0.0226	0.0140	0.0102	0.0087	0.0078	0.0238	0.0176	0.0122	0.0090	0.0072	0.0243	0.0185	0.0128	0.0102	0.0083
	0.50	0.0673	0.0500	0.0389	0.0334	0.0288	0.0729	0.0594	0.0418	0.0350	0.0297	0.0761	0.0614	0.0467	0.0399	0.0318
$\beta = 2$	0.20	0.0124	0.0077	0.0052	0.0041	0.0039	0.0141	0.0096	0.0063	0.0044	0.0041	0.0135	0.0102	0.0060	0.0045	0.0043
	∼0.33	0.0217	0.0139	0.0103	0.0086	0.0075	0.0230	0.0158	0.0106	0.0087	0.0075	0.0235	0.0175	0.0120	0.0094	0.0080
	0.50	0.0692	0.0490	0.0389	0.0343	0.0290	0.0738	0.0569	0.0419	0.0349	0.0312	0.0744	0.0558	0.0449	0.0379	0.0320

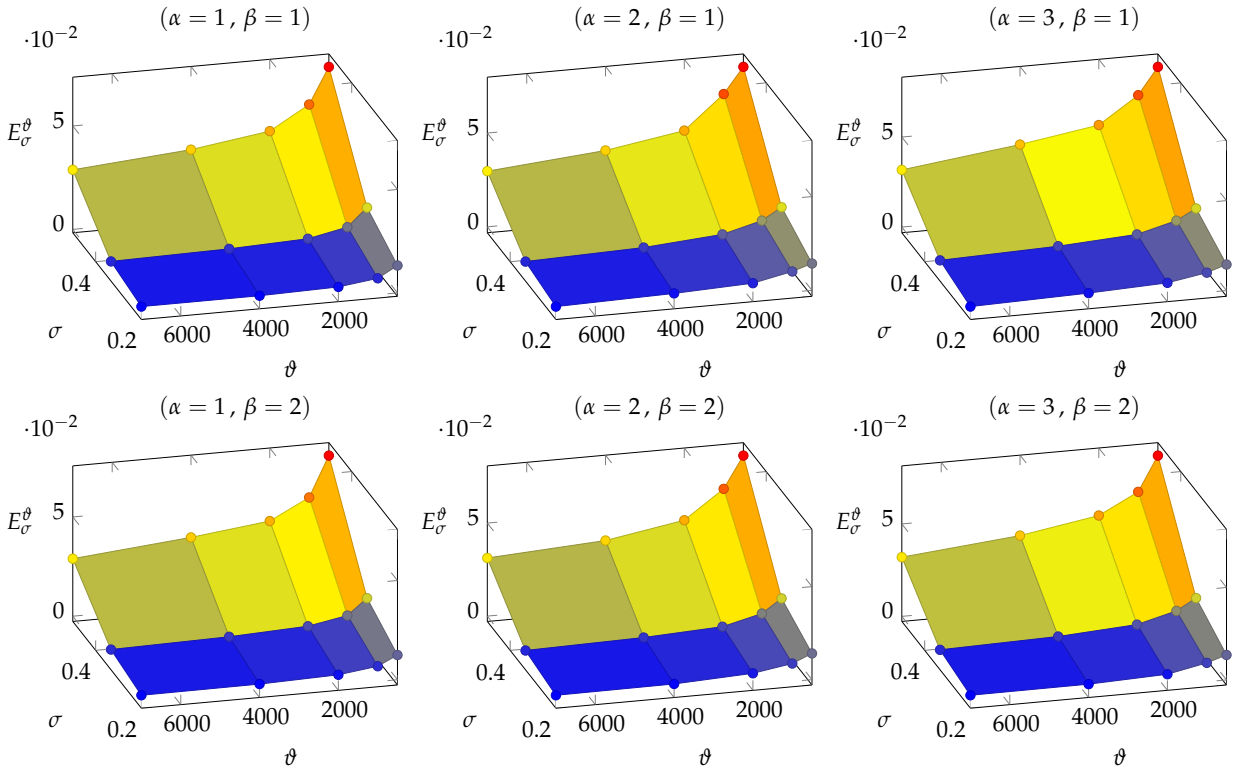

Figure 8. DFN395. Three-dimensional plot of the E^{ϑ}_{σ} values, varying $(\sigma, \vartheta) \in \Sigma \times \Theta$, for each NN of architecture $\mathcal{A}_{\alpha}$ trained with configuration β, for each $\alpha = 1, 2, 3$, and for each $\beta = 1, 2$.

Looking at the results in Table 4, it is very interesting to observe that the average errors E_σ^ϑ are characterized by the same behaviors observed for DFN158 and, therefore, they can be described by a functions $\widehat{E}(\sigma, \vartheta)$ with the same expressions deduced for DFN158.

Table 4. DFN395. MSE and coefficients of the least square minimization with respect to E_σ^ϑ.

		$\alpha = 1$					$\alpha = 2$					$\alpha = 3$				
	E_σ^ϑ	MSE	c_1	c_2	c_3	c_4	MSE	c_1	c_2	c_3	c_4	MSE	c_1	c_2	c_3	c_4
$\beta = 1$	g_1	7.057×10^{-3}	-5.638	417.8	0.0002	8.544	1.289×10^{-2}	-5.531	422.0	0.0001	8.459	1.236×10^{-2}	-5.465	391.0	$\sim 10^{-5}$	8.596
	g_2	8.431×10^{-3}	-6.839	417.9	6.665	-	1.402×10^{-2}	-6.718	422.0	6.596	-	1.358×10^{-2}	-6.675	390.9	6.711	-
$\beta = 2$	g_1	5.874×10^{-3}	-5.719	426.4	0.0006	8.872	9.826×10^{-3}	-5.624	425.1	$\sim 10^{-5}$	8.817	8.934×10^{-3}	-5.548	400.6	0.0002	8.716
	g_2	7.283×10^{-3}	-6.974	426.3	6.941	-	1.137×10^{-2}	-6.875	425.0	6.903	-	1.015×10^{-2}	-6.777	400.5	6.808	-

We remark that the values of E_σ^ϑ increase faster with respect to σ than in Section 3.1.

3.3. Error Characterization with Training Data

In Proposition 1 of this section, assuming that the average error of an untrained NN is characterized by the function $\widehat{E}(\sigma, \vartheta)$ described in (19), we can identify the minimum value of ϑ (i.e., the minimum number of training data) required to obtain an average error smaller than an arbitrary quantity $\varepsilon > 0$ for each fixed $\sigma \in \mathbb{R}_{\geq 0}$; in brief, for each fixed $\sigma \in \mathbb{R}$, the proposition tells which is the minimum $\vartheta \in \mathbb{N}$ such that $\widehat{E}(\sigma, \vartheta) \leq \varepsilon$.

We conclude this introduction to Proposition 1 by making a few remarks to its assumption on the coefficients $\widehat{c}_1, \ldots, \widehat{c}_4$. By construction, it holds that $\widehat{c}_2, \widehat{c}_4 \in \mathbb{R}_{\geq 0}$ and $\widehat{c}_1, \widehat{c}_3 \in \mathbb{R}$ but, looking at the coefficients in Tables 2 and 4, we observe that $\widehat{c}_1$ is always negative, $\widehat{c}_2$ is always positive, and $\widehat{c}_3$ is always nonnegative; then, in the proposition, we assume $\widehat{c}_1 < 0$, $\widehat{c}_2 > 0$ and $\widehat{c}_3 \geq 0$.

Proposition 1. *Let $\widehat{E}(\sigma, \vartheta)$ be a function defined as in (19), such that $\widehat{c}_1 < 0$, $\widehat{c}_2 > 0$ and $\widehat{c}_3, \widehat{c}_4 \geq 0$. Then, for each $\varepsilon > 0$ and $\sigma \in \mathbb{R}_{\geq 0}$, the set of natural solutions $\Theta^* \subset \mathbb{N}$ of the inequality*

$$\widehat{E}(\sigma, \vartheta) \leq \varepsilon \tag{20}$$

is characterized by the following:

1. *any $\vartheta \in \mathbb{N}$ such that*

$$\vartheta \geq \vartheta_\varepsilon := -\frac{\widehat{c}_2}{(C_\sigma - \log \varepsilon)}, \tag{21}$$

   if $\varepsilon > \varepsilon_\sigma := e^{C_\sigma}$, where $C_\sigma := \widehat{c}_1 + \widehat{c}_3 \sigma + \widehat{c}_4 \sigma^2$;
2. *no $\vartheta \in \mathbb{N}$ (i.e., $\Theta^* = \emptyset$), if $\varepsilon \leq \varepsilon_\sigma$.*

Proof. Inequality (20) has the same solutions as inequality

$$\frac{\widehat{c}_2}{\vartheta} + C_\sigma \leq \log \varepsilon, \tag{22}$$

that can be rewritten as $\vartheta(C_\sigma - \log \varepsilon) \leq -\widehat{c}_2$. Therefore, (22) has no solutions if $(C_\sigma - \log \varepsilon) \geq 0$ and solution $\vartheta \geq \vartheta_\varepsilon$ if $(C_\sigma - \log \varepsilon) < 0$; then, both the assertions of Proposition 1 are proven. □

The threshold value $\varepsilon_\sigma = e^{C_\sigma}$ of Proposition 1 is actually the infimum of $\widehat{E}(\sigma, \vartheta)$, assuming a fixed σ:

$$\inf_{\vartheta \in \mathbb{N}} \widehat{E}(\sigma, \vartheta) = \lim_{\vartheta \to +\infty} e^{\frac{\widehat{c}_2}{\vartheta} + C_\sigma} = e^{C_\sigma}.$$

Thanks to Proposition 1, we can define a rule-of-thumb "*UQ rule*" for users who need to perform UQ on a DFN, with a number of fractures n in the order of magnitude around

158–395 generated by similar laws (Section 2.1.2) and who want to understand whether it is convenient to train an NN as a reduced model. This rule is based on the regular behavior characterizing the coefficients $\widehat{c}_1, \dots, \widehat{c}_4$ of $\widehat{E}(\sigma, \vartheta)$, varying the hyperparamenters α and n for each fixed β. Indeed, for each fixed β and $i = 1, \dots, 4$, we observe that the values of the coefficient $\widehat{c}_i$ with respect to (α, n) are well-approximated by the function $\widehat{c}_i^{(\beta)}(\alpha, n)$ defined in Tables 5 and 6 and illustrated in Figures 9 and 10. The expression of the function $\widehat{c}_i^{(\beta)}$ was chosen by looking at the positions of the points $(\alpha, n, \widehat{c}_i)$ in the space $\mathbb{R}^3$, for each $i = 1, \dots, 4$ and each $\beta = 1, 2$; future analyses, involving more DFNs (i.e., more cases for n), may surely help find better-fitting functions to describe the behavior of the coefficients $\widehat{c}_i$.

Table 5. Parametric expressions of the functions $\widehat{c}_i^{(\beta)}(\alpha, n)$ fitting the coefficients $\widehat{c}_1, \dots, \widehat{c}_4$.

i	Function $\widehat{c}_i^{(\beta)}(\alpha, n)$
1	$\widehat{c}_1 = (d_1 + d_2 n)\alpha + (d_3 + d_4\alpha)n + d_5$
2	$\widehat{c}_2 = (d_1 + d_2/n)\alpha + d_3 n + d_4$
3	$\widehat{c}_3 = (d_1 + d_2/n)\alpha$
4	$\widehat{c}_4 = (d_1 + d_2/n)\alpha + d_3 n + d_4$

Table 6. Coefficient values for parametric expressions of the functions $\widehat{c}_i^{(\beta)}(\alpha, n)$ fitting the coefficients $\widehat{c}_1, \dots, \widehat{c}_4$.

i	β	d_1	d_2	d_3	d_4	d_5
1	1	−0.3002	0.1372	−0.0006	−0.1362	−5.467
	2	−0.417	−0.3173	−0.0015	0.3185	−5.201
2	1	−45.67	12 750	1.094	5.067	-
	2	−39.07	10 340	0.9935	50.72	
3	1	−0.738	291.5	-	-	-
	2	−0.748	295.5	-	-	-
4	1	0.38	−139.8	0.0121	3.698	-
	2	0.52	−237.1	0.0096	5.168	-

Given the functions $\widehat{c}_1^{(\beta)}(\alpha, n), \dots, \widehat{c}_4^{(\beta)}(\alpha, n)$, for each fixed $\beta = 1, 2$, we can define a function

$$\widehat{E}_\beta(\sigma, \vartheta, \alpha, n) = e^{\left(\widehat{c}_1^{(\beta)}(\alpha,n) + \frac{\widehat{c}_2^{(\beta)}(\alpha,n)}{\vartheta} + \widehat{c}_3^{(\beta)}(\alpha,n)\sigma + \widehat{c}_4^{(\beta)}(\alpha,n)\sigma^2\right)} \tag{23}$$

that returns estimates of the average errors E_σ^ϑ for any NN with architecture $\mathcal{A}_\alpha$ trained with respect to a number ϑ of simulations (and configuration β) to approximate the fluxes of a DFN with n fractures (see Section 2.1.2) and transmissivity variation characterized by σ. Then, the UQ rule exploits (23) and Proposition 1, and it is outlined by the following steps:

1. Let $n \in \{159, \dots, 394\} \subseteq \mathbb{N}$ be the number of fractures of a given DFN with fixed geometry generated with respect to the characterization of Section 2.1.2, and let σ be the parameter characterizing the standard deviation of the transmissivity distribution (see (1));
2. For each $(\alpha, \beta) \in \{1, 2, 3\} \times \{1, 2\}$ and each arbitrary $\varepsilon \in (e^{C_\sigma^{(\beta)}(\alpha,n)}, 1) \subset \mathbb{R}$, following the results of Proposition 1, compute the values

$$\vartheta_\varepsilon^{(\alpha,\beta)} = -\frac{\widehat{c}_2^{(\beta)}}{(C_\sigma^{(\beta)}(\alpha, n) - \log\varepsilon)}, \tag{24}$$

where $C_\sigma^{(\beta)}(\alpha, n) := \widehat{c}_1^{(\beta)}(\alpha, n) + \widehat{c}_3^{(\beta)}(\alpha, n)\sigma + \widehat{c}_4^{(\beta)}(\alpha, n) + \sigma^2$. Then, the values $\vartheta_\varepsilon^{(\alpha,\beta)}$ represent the estimates of the minimum number of simulations required by the NNs $\mathcal{A}_\alpha$, trained with respect to configuration β, in order to return an average error E_σ^ϑ less than or equal to ε.

The reliability of the values $\vartheta_\varepsilon^{(\alpha,\beta)}$ depends strictly on the reliability of $\widehat{E}(\sigma, \vartheta)$ representing the values E_σ^ϑ and on the reliability of the functions $\widehat{c}_i^{(\beta)}(\alpha, n)$ representing the coefficients $\widehat{c}_i$. Therefore, we conclude this section by testing the efficiency of the UQ rule and, consequently, the reliability of the expressions chosen in this work for the functions $\widehat{E}(\sigma, \vartheta), \widehat{c}_1^{(\beta)}(\alpha, n), \ldots, \widehat{c}_4^{(\beta)}(\alpha, n)$.

We validate and test the UQ rule with respect to DFN202, a DFN with $n = 202$ fracture ($m = 14$ outflow fractures) and transmissivity distribution characterized by $\sigma = 1/3$. We train an NN $\mathcal{A}_\alpha$ with configuration $\beta \in \{1, 2\}$ on a number of simulations ϑ_{act} equal to (24) rounded up to the nearest multiple of five for each $\varepsilon \in \{0.01, 0.02, 0.03\}$ and each $\alpha \in \{1, 2, 3\}$. For the case $(\alpha, \beta) = (3, 1)$, we do not use a value $\varepsilon = 0.01$ but a value $\varepsilon = 0.011$ because 0.01 is too close to the infimum error value $e^{C_\sigma^{(\beta)}(\alpha,n)} = 0.0099$ and, indeed, in this case, $\vartheta_\varepsilon^{(\alpha,\beta)}$ is approximately equal to 20,000; since we do not have enough simulations available to test $\varepsilon = 0.01$, we adopt $\varepsilon = 0.011$.

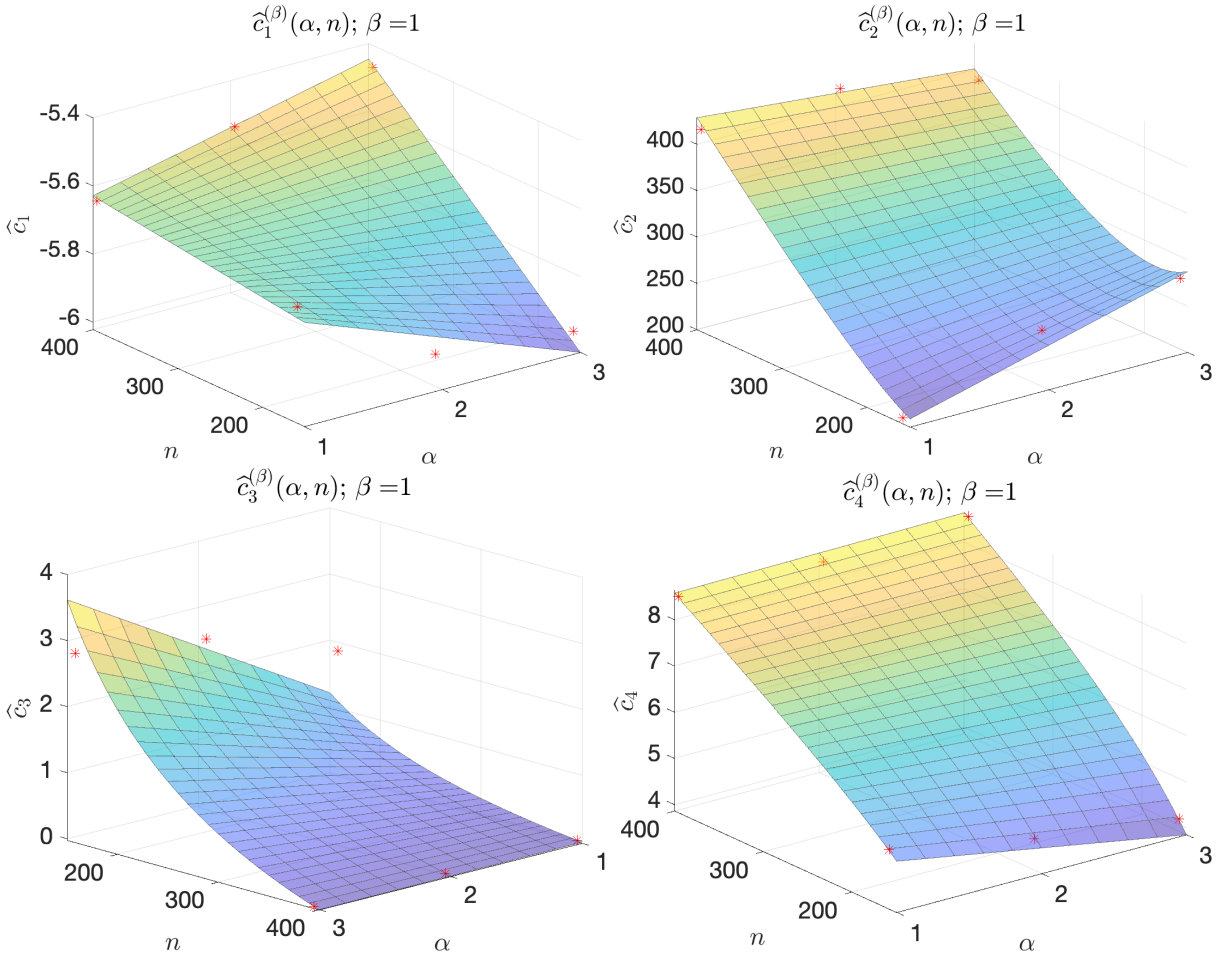

Figure 9. Plots of the functions $\widehat{c}_i^{(\beta)}(\alpha, n)$ fitting the coefficients $\widehat{c}_1, \ldots, \widehat{c}_4$ (from **left** to **right**), for $\beta = 1$.

The average errors obtained for the test on DFN202 are reported in Table 7. For each (α, β), we report the minimum error value $e^{C_\sigma^{(\beta)}(\alpha,n)}$, the chosen target error $\varepsilon > e^{C_\sigma^{(\beta)}(\alpha,n)}$, the estimated minimum number of simulations $\vartheta_\varepsilon^{(\alpha,\beta)}$, the number of simulations $\vartheta_{\text{act}} \sim \vartheta_\varepsilon^{(\alpha,\beta)}$ performed for the training of the NN, and the final average error E_σ^ϑ returned by the

trained NN on a test set $\mathcal{P}_\sigma$ (with $|\mathcal{P}_\sigma| = 3000$). In all the cases, with $\vartheta_{\text{act}} \sim \vartheta_\varepsilon^{(\alpha,\beta)}$, the error E_σ^ϑ is very close to the target error ε.

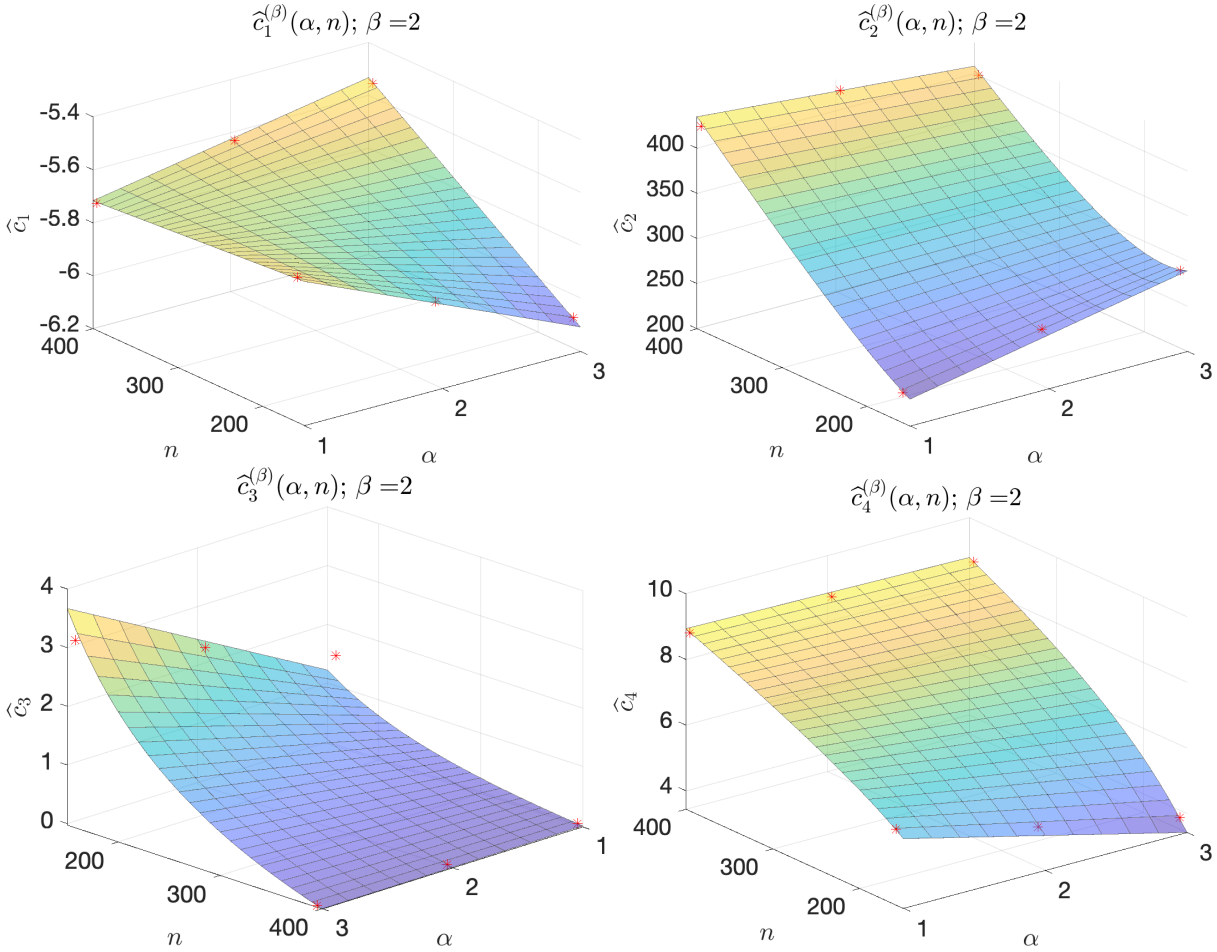

Figure 10. Plots of the functions $\widehat{c}_i^{(\beta)}(\alpha, n)$ fitting the coefficients $\widehat{c}_1, \ldots, \widehat{c}_4$ (from **left** to **right**), for $\beta = 2$.

Table 7. Values returned by validation of the Uncertainty Quantification (UQ) rule on DFN202 ($n = 202$, $\sigma = 1/3$).

		$\beta = 1$					$\beta = 2$			
α	$e^{C_\sigma^{(\beta)}(\alpha,n)}$	ε	$\vartheta_\varepsilon^{(\alpha,\beta)}$	ϑ_{act}	E_σ^ϑ	$e^{C_\sigma^{(\beta)}(\alpha,n)}$	ε	$\vartheta_\varepsilon^{(\alpha,\beta)}$	ϑ_{act}	E_σ^ϑ
1	0.0081	0.01	1162.2	1165	0.0097	0.0090	0.01	2449.3	2450	0.0080
		0.02	269.68	270	0.0221		0.02	329.09	330	0.0208
		0.03	186.09	190	0.0250		0.03	218.47	220	0.0323
2	0.0089	0.01	2339.1	2340	0.0085	0.0090	0.01	2709.7	2710	0.0071
		0.02	324.17	325	0.0226		0.02	346.75	350	0.0215
		0.03	215.55	220	0.0249		0.03	229.62	230	0.0254
3	0.0099	0.011	2555.8	2560	0.0086	0.0091	0.01	3001.9	3005	0.0079
		0.02	393.75	395	0.0195		0.02	364.66	365	0.0197
		0.03	250.20	255	0.0250		0.03	240.88	245	0.0253

4. Discussion

Some examples concerning the use of deep learning models to speed up UQ analysis can be found in [33,34]. The use of DNNs as surrogate models for UQ is still a novel approach that requires deep investigations but is very promising. To the best of the authors' knowledge, other than [35–37], there are no works in the literature that train DNNs to perform flux regression tasks on DFNs and, in particular, that use these NNs in the context of UQ as in [35]. While the results illustrated in [35] are very promising, the ones presented in Section 3 of this work concerns the use of NN reduced models as a practical possibility in the UQ framework for flow analyses of a subsurface network of fractures.

Let us assume that we deal with a natural fractured basin that can be described by a number of principal fractures in the order of $\{158, \dots, 395\}$ and probability distributions for fractures and hydrogeological properties as in Section 2.1.2, with a fixed value $\sigma \in [0.2, 0.7] \subset \mathbb{R}$. The stochastic flow analysis can be very relevant for geothermal energy exploitation and for enhanced oil and gas exploitation. Flux investigations can also be relevant in risk assessment for geological storage of nuclear waste. The approach could be extended to different situations, for example, to provide a statistical analysis of the effects of different fracturing approaches [46–48].

Uncertainty in fractures and hydrogeological properties requires the generation of an ensemble of DFNs describing the principal flow properties of the basin, and consequently, a UQ analysis of the flow properties is required. The results presented in Section 3.3 can be useful for deciding if the training of a DNN is convenient with respect to a Monte Carlo approach or the use of a different surrogate flow model. Thanks to the results in Section 3.3, we have the possibility to fix an approximation tolerance $\varepsilon > 0$ such that an NN trained on $\sim \vartheta_\varepsilon$ simulations fits the target tolerance.

Let us provide an example with DFN202, the validation DFN of Section 3.3 ($\sigma = 1/3$). During a UQ analysis for this DFN, a standard approach may need thousands of simulations to obtain good estimations of the mean value and the standard deviation of the flux exiting from the DFN. Nevertheless, the UQ rule tells us that we can train an NN with approximately 2% or 1% average error with less than 300 simulations or approximately 1000 simulations, respectively (see Table 7, $(\alpha, \beta) = (1, 1)$ case). Then, once that has been trained, a NN can return virtually infinite reliable predictions (i.e., approximations) of the DFN exiting fluxes, varying the fracture transmissivities, in the order of seconds; therefore, we can estimate the exiting flux's momentum using the NN predictions with a total cost of only the $\sim \vartheta_\varepsilon$ DFN simulations used to train the NN. If we repeat the procedure for each geometry of DFN generated for the study, the advantages are significant.

A possible drawback of our method is that a UQ rule must be defined for the family of problems and NN architectures considered. Indeed, the UQ rule defined in Section 3.3 is tailored on the multi-task architecture described in Section 2.2.3, applied to the family of DFNs defined by the probability distributions in Section 2.1.2. Moreover, the UQ rule of this work can be considered reliable at most for DFNs with few hundreds of fractures. The analysis performed here can be extended to larger DFNs and can provide useful information to wider applications.

The approach presented here is not immediately extensible to the case of DFNs with a stochastic geometry (see [49]) due to the continuous change in inflow and outflow fractures. Nevertheless, a similar approach could be extended to the case of analysis of flows through the DFN that occurs between a fixed set of wells. In that case, the NN can provide flow through fixed wells varying the DFN geometry and the hydraulic properties of the fractures. In that case, we expect that the number of training simulations increases but the proposed approach could provide information correlating the target error tolerance with the variance in the stochastic distributions and the number of fractures.

5. Conclusions

With this work, we proposed an analysis for the characterization of a family of DNNs with multi-task architecture $\mathcal{A}_\alpha$ trained to predict the exiting fluxes of a DFN given the

fracture transmissivities. The novelty of this analysis consists in characterizing these NNs, searching for rules that describe the performances, varying the available training data (ϑ) and the standard deviation of the inputs (σ). The results of our study show interesting common behaviors for all the trained NNs, providing characterization of the average error with the functions $\widehat{E}(\sigma, \vartheta)$ and $\widehat{E}_\beta(\sigma, \vartheta, \alpha, n)$ (see (19) and (23)). This result is interesting, since it shows that common characterizing formulas for NN performances exist, despite the stochastic nature of the NN training processes; thanks to these regularities, we are able to define a "*UQ rule*" that returns an estimate of the minimum number of simulations required for training an NN with an average error less than or equal to an arbitrary value $\varepsilon > e^{C_\sigma}$.

This estimate can be fruitfully exploited in real-world problems. Indeed, in the framework of UQ, it suggests whether it is convenient to train an NN as a reduced model and that a user can choose the best strategy between the use of an NN or direct simulations. In particular, the estimate returned by the UQ rule can be exploited in all "real-world" applications in which flow through a DFN with stochastic trasmissivities is recommended. The fields of interest could be oil and gas extraction, where flows through a fractured medium that occur between a fixed set of wells need to be analyzed and the possible effects of phenomena that can impact the fracture transmissivities (for example clogging) should be foreseen. Similar needs could occur in designing geothermal sites for which the performances strongly depend on the flow properties. Other application examples could be flow analysis for geological risk assessments of geological carbon dioxide or nuclear waste storage or water prevention close to other pollutant storage sites. The usage of NNs as reduced models for DFN flow simulations, optimizing the number of required numerical simulations for training through the UQ rule, can save precious time when computing an estimate of the risks and, therefore, deciding how to intervene when preventing or managing a calamity.

In general, we believe that many approaches for underground flow analysis through DFNs can be endowed with a tailored version of the method proposed in this paper; then, the method can speed up the simulation process, which is often slow and expensive, granting considerable advantages in many real-world geophysical applications.

Author Contributions: Conceptualization, S.B. and F.D.S.; data curation, S.B. and F.D.S.; formal analysis, S.B. and F.D.S.; funding acquisition, S.B.; investigation, S.B. and F.D.S.; methodology, S.B. and F.D.S.; project administration, S.B. and F.D.S.; resources, S.B. and F.D.S.; software, F.D.S.; supervision, S.B.; validation, S.B. and F.D.S.; visualization, F.D.S.; writing—original draft, S.B. and F.D.S.; writing—review and editing, S.B. and F.D.S. Both authors have read and agreed to the published version of the manuscript.

Funding: Research performed in the framework of the Italian MIUR Award "Dipartimento di Eccellenza 2018-2022" to the Department of Mathematical Sciences, Politecnico di Torino, CUP: E11G18000350001. The research leading to these results was also partially funded by INdAM-GNCS and by the SmartData@PoliTO center for Big Data and Machine Learning technologies.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets used for training and testing the Neural Networks are available at https://smartdata.polito.it/discrete-fracture-network-flow-simulations/ (accessed on 18 January 2021).

Acknowledgments: The authors acknowledge support from the GEOSCORE group (https://areeweb.polito.it/geoscore/ accessed on 18 January 2021) of Politecnico di Torino (Department of Mathematical Sciences).

Conflicts of Interest: The authors declare no conflict of interest.

Sample Availability: Samples of the datasets used for the training of neural networks are available from the authors.

Abbreviations

The following abbreviations and nomenclatures are used in this manuscript:

DFN	Discrete Fracture Network
DL	Deep Learning
DNN	Deep Neural Network
ML	Machine Learning
MSE	Mean Square Error
NN	Neural Network
PDE	Partial Differential Equation
QoI	Quantity of Interest
sMSE	sum of Mean Square Errors
UQ	Uncertainty Quantification
α, $\mathcal{A}_\alpha$	NN depth parameter, multi-task NN architecture with depth parameter α
β	NN training configuration parameter
$\mathcal{B}$	Minibatch
$\mathcal{D}$, $\mathcal{D}_\sigma$	Dataset, dataset sampled with standard deviation parameter σ
$E(\mathcal{P})$, E^ϑ_σ	Average error (measured on the test set $\mathcal{P}$), average error measured on $\mathcal{P}_\sigma$ for an NN trained on ϑ samples
$\boldsymbol{F}$, $\widehat{\boldsymbol{F}}_{\boldsymbol{w}}$	Flux simulation function of DFN, NN approximated flux simulation function
$\mathcal{F}_i$	ith DFN fracture
H	Hydraulic head
κ_i, $\boldsymbol{\kappa}$	Transmissivity of the ith DFN fracture, vector of DFN fracture transmissivities
m	Number of boundary DFN fractures with exiting flux
n	Total number of fractures of the DFN
$\mathcal{P}$, $\mathcal{P}_\sigma$	Test set, test set sampled with standard deviation parameter σ
φ_j, $\boldsymbol{\varphi}$	Exiting flux of the jth DFN boundary fracture, vector of DFN exiting fluxes
σ, Σ	parameter characterizing the transmissivity standard deviation, set of considered values for σ
$\mathcal{T}$, $\mathcal{T}^\vartheta_\sigma$	Training set, training set sampled with standard deviation parameter σ and cardinality parameter ϑ
ϑ, Θ	cardinality of training set plus validation set, set of considered values for ϑ
$\mathcal{V}$, $\mathcal{V}^\vartheta_\sigma$	Validation set, validation set sampled with standard deviation parameter σ and cardinality parameter ϑ

References

1. Adler, P.M. *Fractures and Fracture Networks*; Kluwer Academic: Dordrecht, The Netherlands, 1999.
2. Cammarata, G.; Fidelibus, C.; Cravero, M.; Barla, G. The Hydro-Mechanically Coupled Response of Rock Fractures. *Rock Mech. Rock Eng.* **2007**, *40*, 41–61. [CrossRef]
3. Fidelibus, C.; Cammarata, G.; Cravero, M. Hydraulic characterization of fractured rocks. In *Rock Mechanics: New Research*; Abbie, M., Bedford, J.S., Eds.; Nova Science Publishers Inc.: New York, NY, USA, 2009.
4. Pichot, G.; Erhel, J.; de Dreuzy, J. A mixed hybrid Mortar method for solving flow in discrete fracture networks. *Appl. Anal.* **2010**, *89*, 1629–1643. [CrossRef]
5. Pichot, G.; Erhel, J.; de Dreuzy, J. A generalized mixed hybrid mortar method for solving flow in stochastic discrete fracture networks. *SIAM J. Sci. Comput.* **2012**, *34*, B86–B105. [CrossRef]
6. de Dreuzy, J.R.; Pichot, G.; Poirriez, B.; Erhel, J. Synthetic benchmark for modeling flow in 3D fractured media. *Comput. Geosci.* **2013**, *50*, 59–71. [CrossRef]
7. Pichot, G.; Poirriez, B.; Erhel, J.; de Dreuzy, J.R. A Mortar BDD method for solving flow in stochastic discrete fracture networks. In *Domain Decomposition Methods in Science and Engineering XXI*; Lecture Notes in Computational Science and Engineering; Springer: Berlin/Heidelberg, Germany, 2014; pp. 99–112.
8. Nœtinger, B.; Jarrige, N. A quasi steady state method for solving transient Darcy flow in complex 3D fractured networks. *J. Comput. Phys.* **2012**, *231*, 23–38. [CrossRef]
9. Nœtinger, B. A quasi steady state method for solving transient Darcy flow in complex 3D fractured networks accounting for matrix to fracture flow. *J. Comput. Phys.* **2015**, *283*, 205–223. [CrossRef]
10. Dershowitz, W.S.; Fidelibus, C. Derivation of equivalent pipe networks analogues for three-dimensional discrete fracture networks by the boundary element method. *Water Resour. Res.* **1999**, *35*, 2685–2691. [CrossRef]
11. Berrone, S.; Pieraccini, S.; Scialò, S. A PDE-constrained optimization formulation for discrete fracture network flows. *SIAM J. Sci. Comput.* **2013**, *35*, B487–B510. [CrossRef]

12. Berrone, S.; Pieraccini, S.; Scialò, S. On simulations of discrete fracture network flows with an optimization-based extended finite element method. *SIAM J. Sci. Comput.* **2013**, *35*, A908–A935. [CrossRef]
13. Berrone, S.; Pieraccini, S.; Scialò, S.; Vicini, F. A parallel solver for large scale DFN flow simulations. *SIAM J. Sci. Comput.* **2015**, *37*, C285–C306. [CrossRef]
14. Berrone, S.; Pieraccini, S.; Scialò, S. An optimization approach for large scale simulations of discrete fracture network flows. *J. Comput. Phys.* **2014**, *256*, 838–853. [CrossRef]
15. Berrone, S.; Borio, A.; Scialò, S. A posteriori error estimate for a PDE-constrained optimization formulation for the flow in DFNs. *SIAM J. Numer. Anal.* **2016**, *54*, 242–261. [CrossRef]
16. Berrone, S.; Pieraccini, S.; Scialò, S. Towards effective flow simulations in realistic discrete fracture networks. *J. Comput. Phys.* **2016**, *310*, 181–201. [CrossRef]
17. Berrone, S.; D'Auria, A.; Vicini, F. Fast and robust flow simulations in Discrete Fracture Networks with GPGPUs. *GEM Int. J. Geomathematics* **2019**, to appear. [CrossRef]
18. Hyman, J.D.; Gable, C.W.; Painter, S.L.; Makedonska, N. Conforming Delaunay Triangulation of Stochastically Generated Three Dimensional Discrete Fracture Networks: A Feature Rejection Algorithm for Meshing Strategy. *SIAM J. Sci. Comput.* **2014**, *36*, A1871–A1894. [CrossRef]
19. Fumagalli, A.; Scotti, A. A numerical method for two-phase flow in fractured porous media with non-matching grids. *Adv. Water Resour.* **2013**, *62*, 454–464. [CrossRef]
20. Jaffré, J.; Roberts, J.E. Modeling flow in porous media with fractures; Discrete fracture models with matrix-fracture exchange. *Numer. Anal. Appl.* **2012**, *5*, 162–167. [CrossRef]
21. Karimi-Fard, M.; Durlofsky, L.J. Unstructured Adaptive Mesh Refinement for Flow in Heterogeneous Porous Media. In Proceedings of the ECMOR XIV-14th European Conference on the Mathematics of Oil Recovery, Sicily, Italy, 8–11 September 2014.
22. Svensk Kärnbränslehantering AB. *Data Report for the Safety Assessment, SR-Site*; Technical Report TR-10-52; SKB: Stockholm, Sweden, 2010.
23. Hyman, J.D.; Aldrich, G.; Viswanathan, H.; Makedonska, N.; Karra, S. Fracture size and transmissivity correlations: Implications for transport simulations in sparse three-dimensional discrete fracture networks following a truncated power law distribution of fracture size. *Water Resour. Res.* **2016**, *52*, 6472–6489. [CrossRef]
24. Sanchez-Vila, X.; Guadagnini, A.; Carrera, J. Representative hydraulic conductivities in saturated grqundwater flow. *Rev. Geophys.* **2006**, *44*, 1–46. [CrossRef]
25. Hyman, J.D.; Hagberg, A.; Osthus, D.; Srinivasan, S.; Viswanathan, H.; Srinivasan, G. Identifying Backbones in Three-Dimensional Discrete Fracture Networks: A Bipartite Graph-Based Approach. *Multiscale Model. Simul.* **2018**, *16*, 1948–1968. [CrossRef]
26. Berrone, S.; Canuto, C.; Pieraccini, S.; Scialò, S. Uncertainty quantification in Discrete Fracture Network models: Stochastic fracture transmissivity. *Comput. Math. Appl.* **2015**, *70*, 603–623. [CrossRef]
27. Berrone, S.; Pieraccini, S.; Scialò, S. Non-stationary transport phenomena in networks of fractures: Effective simulations and stochastic analysis. *Comput. Methods Appl. Mech. Eng.* **2017**, *315*, 1098–1112. [CrossRef]
28. Canuto, C.; Pieraccini, S.; Xiu, D. Uncertainty Quantification of Discontinuous Outputs via a Non-Intrusive Bifidelity Strategy. *J. Comput. Phys.* **2019**, *398*, 108885. [CrossRef]
29. Hyman, J.D.; Hagberg, A.; Srinivasan, G.; Mohd-Yusof, J.; Viswanathan, H. Predictions of first passage times in sparse discrete fracture networks using graph-based reductions. *Phys. Rev. E* **2017**, *96*, 013304. [CrossRef] [PubMed]
30. Srinivasan, G.; Hyman, J.D.; Osthus, D.A.; Moore, B.A.; O'Malley, D.; Karra, S.; Rougier, E.; Hagberg, A.A.; Hunter, A.; Viswanathan, H.S. Quantifying Topological Uncertainty in Fractured Systems using Graph Theory and Machine Learning. *Sci. Rep.* **2018**, *8*, 11665. [CrossRef] [PubMed]
31. Srinivasan, S.; Karra, S.; Hyman, J.; Viswanathan, H.; Srinivasan, G. Model reduction for fractured porous media: A machine learning approach for identifying main flow pathways. *Comput. Geosci.* **2019**. [CrossRef]
32. Chan, S.; Elsheikh, A.H. A machine learning approach for efficient uncertainty quantification using multiscale methods. *J. Comput. Phys.* **2018**, *354*, 493–511. [CrossRef]
33. Tripathy, R.K.; Bilionis, I. Deep UQ: Learning deep neural network surrogate models for high dimensional uncertainty quantification. *J. Comput. Phys.* **2018**, *375*, 565–588. [CrossRef]
34. Hu, R.; Fang, F.; Pain, C.C.; Navon, I.M. Rapid spatio-temporal flood prediction and uncertainty quantification using a deep learning method. *J. Hydrol.* **2019**, *575*, 911–920. [CrossRef]
35. Berrone, S.; Della Santa, F.; Pieraccini, S.; Vaccarino, F. Machine Learning for Flux Regression in Discrete Fracture Networks. Preprint (under Submission), Politecnico di Torino (PORTO@iris), 2019. Available online: http://hdl.handle.net/11583/2724492 (accessed on 18 January 2021).
36. Berrone, S.; Della Santa, F.; Mastropietro, A.; Pieraccini, S.; Vaccarino, F. Backbone Identification in Discrete Fracture Networks Using Layer-Wise Relevance Propagation for Neural Network Feature Selection. Preprint (under Submission), Politecnico di Torino (PORTO@iris), 2020. Available online: http://hdl.handle.net/11583/2844659 (accessed on 18 January 2021).

37. Berrone, S.; Della Santa, F.; Mastropietro, A.; Pieraccini, S.; Vaccarino, F. Discrete Fracture Network Insights by eXplainable AI. In *Conference Paper, Poster and Presentation, Machine Learning and the Physical Sciences, Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS), online, 11 December 2020. Neural Information Processing Systems Foundation 2020, online, 108885.* Available online: https://ml4physicalsciences.github.io/2020/ (accessed on 18 January 2021).
38. McCulloch, W.S.; Pitts, W. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.* **1943**, *5*, 115–133. [CrossRef]
39. Hebb, D.O. *The Organization of Behavior*; Wiley: New York, NY, USA, 1949.
40. Rosenblatt, F. The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain. *Psychol. Rev.* **1958**, *65*, 386–408. [CrossRef]
41. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016. Available online: http://www.deeplearningbook.org (accessed on 18 January 2021).
42. GEOSCORE Research Group. *GEO++*; Department of Mathematical Sciences, Politecnico di Torino: Turin, Italy. Available online: https://areeweb.polito.it/geoscore/software/ (accessed on 18 January 2021).
43. Nawi, N.M.; Atomi, W.H.; Rehman, M. The Effect of Data Pre-processing on Optimized Training of Artificial Neural Networks. *Procedia Technol.* **2013**, *11*, 32–39. [CrossRef]
44. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. *J. Mach. Learn. Res.* **2010**, *9*, 249–256.
45. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015.
46. Davarpanah, A.; Shirmohammadi, R.; Mirshekari, B.; Aslani, A. Analysis of hydraulic fracturing techniques: Hybrid fuzzy approaches. *Arab. J. Geosci.* **2019**, *12*, 402. [CrossRef]
47. Sun, S.; Zhou, M.; Lu, W.; Davarpanah, A. Application of Symmetry Law in Numerical Modeling of Hydraulic Fracturing by Finite Element Method. *Symmetry* **2020**, *12*, 1122. [CrossRef]
48. Zhu, M.; Yu, L.; Zhang, X.; Davarpanah, A. Application of Implicit Pressure-Explicit Saturation Method to Predict Filtrated Mud Saturation Impact on the Hydrocarbon Reservoirs Formation Damage. *Mathematics* **2020**, *8*, 1057. [CrossRef]
49. Pieraccini, S. Uncertainty quantification analysis in discrete fracture network flow simulations. *GEM Int. J. Geomath.* **2020**, *11*, 12. [CrossRef]