POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Deep Learning-based Reduced Order Models for PDEs: Multi-fidelity Strategies for Transfer Learning

Tesi di Laurea Magistrale in
Mathematical Engineering - Ingegneria Matematica

Author: **Daniel Fraulin**

Student ID: 945020
Advisor: Prof. Andrea Manzoni
Co-advisors: Dott. Nicola Rares Franco
Academic Year: 2020-2021

Abstract

Deep Learning-based Reduced Order Models (DL-ROMs) are a recently developed framework for the efficient approximation of the parameter-to-solution map associated to parametric Partial Differential Equations (PDEs) exploiting Machine Learning techniques. Taking advantage of a set of PDEs snapshots and deep neural networks, after an offline training DL-ROMs enable the inexpensive online evaluation of the PDE solution for any new parameter instance. Thanks to a rigorous theoretical setting, approximation capabilities of DL-ROMs and network architecture complexity have been recently investigated. This thesis extends these results to the case of PDE parameterized by infinite dimensional objects like random fields, confirming the error estimates with numerical experiments. Moreover, this work presents several strategies, all based on the concept of Transfer Learning, to alleviate the computational burden of the DL-ROM offline stage. In particular, we designed a multi-level training algorithm, that takes advantage of snapshots at lower resolution, and a Hybrid DL-ROM, that inherits the internal representation of the solution manifold from a simpler DL-ROM and enhances it with further details through a suitable re-training. Both these strategies have been assessed on a set of model problems involving linear elliptic PDEs. Finally, Hybrid DL-ROMs have been exploited to solve a Bayesian inverse problem for parameter estimation by means of a revisited Monte-Carlo Markov-Chain algorithm in the framework of Uncertainty Quantification of PDEs.

Keywords: Partial Differential Equations, Reduced Order Models, Machine Learning, Artificial Neural Networks, Transfer Learning, Multi-Fidelity models

Abstract in lingua italiana

I modelli di ordine ridotto basati su tecniche di Deep Learning (Deep Learning-based Reduced Order Models, DL-ROM) sono metodi che sfruttano tecniche di Machine Learning per approssimare in maniera efficiente la mappa parametro-soluzione associata a Equazioni a Derivate Parziali (EDP) parametriche. Attraverso l'impiego di Deep Neural Networks e di un insieme di snapshots della EDP, dopo una fase di allenamento offline i DL-ROM permettono una valutazione online computazionalmente efficiente della soluzione della EDP per ogni nuovo valore dei parametri. Grazie ad un rigoroso setting teorico, le capacità di approssimazione dei DL-ROM e la complessità, in termini di architettura, delle reti che li compongono sono state di recente investigate. Questa tesi estende tali risulti al caso di EDP parametrizzate da un oggetto infinito dimensionale come un campo stocastico, confermando la stima ottenuta attraverso esperimenti numerici. Inoltre, questo lavoro sviluppa diverse strategie, tutte basate sul concetto di Transfer Learning, per ridurre l'onere computazionale della fase di costruzione offline dei DL-ROM. Specificatamente, presentiamo un algoritmo di allenamento multi-livello, che sfrutta l'utilizzo di soluzioni a bassa risoluzione come snapshots, e DL-ROM Ibridi, che possono acquisire da altri DL-ROM più semplici una rappresentazione interna della varietà delle soluzioni per poi arricchirla di ulteriori dettagli attraverso un opportuno ri-allenamento. Entrambe le strategie proposte sono corredate di un'analisi dei costi computazionali che permette di valutarne l'efficacia su dei problemi modello associati a EDP ellittiche. Infine, i DL-ROM Ibridi sono stati impiegati nella soluzione di un problema di stima di parametri in ambito Bayesiano, attraverso la versione rivisitata di un algoritmo Markov-Chain Monte-Carlo, nel contesto della Uncertainty Quantification per le EDP.

Parole chiave: Equazioni a Derivate Parziali, Modelli di Ordine Ridotto, Machine Learning, Reti Neurali Artificiali, Transfer Learning, modelli Multi-Fedeltà

Contents

Introduction

Parametrized Partial Differential Equations (PDEs) are fundamental tools for modeling the behaviour of physical, biological and mechanical systems. Supposing the problem is well-posed for any value of the input parameters within a suitable parameter space, PDEs are generally solved by means of high-fidelity Full-Order Models (FOMs), such as the Finite Element Method (FEM) [26]. FOMs guarantee an accurate numerical solution but might entail a non-negligible computational cost. The latter may become easily unbearable when dealing with many-queries applications, such as optimal control problems or Bayesian inversion and uncertainty quantification. In these contexts we are interested in exploring a possibly wide portion of the discrete *solution manifold*, by solving the PDE for a potentially large ensemble of parameter instances. One possible strategy consists in replacing the FOMs with Reduced Order Models (ROMs) that approximate in a highly efficient way the *parameter-to-solution* map, while maintaining adequate levels of accuracy.

In this thesis, we focus on a particular class of ROMs, namely Deep Learning-based Reduced Order Models (DL-ROMs), recently introduced in [7, 9, 10]. DL-ROMs are a Machine Learning (ML) framework that exploits extensively Artificial Neural Network (NN) to learn the parameter-to-solution map starting from a dataset of solution snapshots. These latter are generated using the FOM in an expensive off-line stage, that also includes the NN model training. The high popularity of this kind of surrogates is due to the completely effortless evaluation of PDE queries for new input parameters during the following on-line stage. Indeed, DL-ROMs are non-intrusive methods, which do not involve the assembling and the resolution of any linear system associated to a reduced order differential problem. What distinguishes DL-ROM from others ML-based approximation algorithms (e.g. [14, 21]) is that, first, a low dimensional representation of the solution manifold is determined through a deep auto-encoder (AE). This allows to compress the information associated to the solution manifold in a so-called *latent space*. Afterward, the learning problem is rephrased into the much more manageable approximation of the low dimensional relation between parameter instances and the latent representation of

the PDE solution. Furthermore, DL-ROMs were developed on a deep theoretical basis that plays an essential role in designing the NN architecture. In particular, recent results [7] prescribe, depending on properties of the parameter-to-solution map, the minimal dimension of the latent space to ensure that the error entailed by compressing the solution manifold can be made arbitrarily small. This information is in turn used to fix the width of the deep auto-encoder bottleneck, with a huge impact on the number of degrees of freedom (dof) of the NN and, as a consequence, on the computational time required by its training.

Starting from this background the work follows two main paths. The first one regards the extension of DL-ROMs theoretical basis to the case of stochastic PDEs. Indeed, the available results holds for the case of PDE parametrized by a finite number of coefficients belonging to a compact subset of $\mathbb{R}^n$. We instead proof an analogous error estimate for PDEs depending on a random field and, thus, on an infinite number of random inputs. This task was accomplished through two steps. The first one allows to remove the hypothesis of compactness of the parameter space by exploiting classical inequalities from Real Analysis. The second steps follows immediately thanks to a dimensionality reduction approach based on the Karhunen-Loeve expansion of the random field. Moreover, the theoretical result is provided with rigorous numerical experiments that confirm its validity for three different test-cases within the class of elliptic stochastic PDEs.

The second path is instead related to more practical issues. Indeed, we design several strategies to alleviate the burden of the expensive DL-ROM off-line stage, all based on the concept of Transfer Learning. With this term we indicate the possibility of storing the information gained while solving a problem, and applying it to a different but related one. From the Deep-Learning perspective, this implies the reuse of an already trained NN within possibly more expressive learning framework to accomplish new approximation tasks. In particular:

- first, we deal with the reduction of the computational effort for the dataset generation. Indeed, the possibility to rely on a large training set is of crucial importance to make the DL-ROM able to generalize well, but this entails the extensive use of the FOM. We therefore propose a Multi-level training algorithm, suitable for any kind of parametrized PDE, that leverages snapshots at different level of resolution.
- Then, we develop a class of Hybrid DL-ROMs for stochastic PDEs. This new approach allows to reduce the training time, thanks to its ability of inheriting, from simpler DL-ROMs, an internal representation of the solution manifold and enhance

it with further details thanks to a suitable re-training procedure.

The capabilities of Hybrid DL-ROMs are further investigated by employing them for the efficient solution of a Bayesian inverse problem for a stochastic PDE. Specifically, we combine the flexibility provided by the Hybrid DL-ROM implementation with a multi-stage Metropolis algorithm, with the double goal of increasing the surrogate model accuracy during the simulation and of tackling the dimensionality curse that affects the inference on random fields.

The thesis is structured as follows. In Chapter 1 we first recall some basic notions about the NNs structure and their learning process. Then, we focus on the DL-ROMs by describing their implementation and reporting the most important theoretical results. In Chapter 2 we propose the Multi-level training algorithm, evaluating its performances on the case of linear elliptic PDEs including however highly nonlinear parameter dependencies. The theoretical results are developed in Chapter 3 together with the supportive numerical experiments. Chapter 4 is devoted to the design of Hybrid DL-ROMs, from the implementation issues, to the setting optimization and the performance analysis. Finally, Chapter 5 focuses on the solution of Bayesian inverse problems through Monte Carlo Markov Chain algorithms, reporting numerical results obtained on two test-cases through the use of Hybrid DL-ROMs. The work then is concluded by a brief summary of the achieved results and of possible further steps in several directions.

1 | From Neural Networks to DL-ROMs

This chapter briefly recalls how Deep Learning-based Reduced Order Models (DL-ROMs) work, by presenting the main theoretical results that describe their approximation properties. In order to provide a complete presentation on the subject, we first devote a section to what a neural networks is and in what the learning process consists. In particular, we analyze the various types of errors that the use of a NN entails. This, together with the description of some specific tool used throughout this work, is fundamental to clarify the reason behind the design choices made in the following.

1.1. Artificial Neural Networks

An Artificial Neural Network (NN) is a mathematical model inspired by biological nervous system in which the fundamental unit is the perceptron [28, 33]. A NN receives information from multiple sources, that might be external to the network or be the output of another perceptron, elaborates it and then transmits it. More specifically it applies to the input vector an affine transformation $\boldsymbol{x} \to \boldsymbol{w} \cdot \boldsymbol{x} + b$, where $\boldsymbol{w}$ is the weight vector and b is an additive degree of freedom called *bias*. Then, the result is possibly mapped through a nonlinear activation function ρ, which is usually continuous and monotone. Among the most common used, there are the sigmoid function $\rho(x) = 1/(1+\exp(-x))$, the hyperbolic tangent $\rho(x) = \tanh(x)$ and the ReLu activation function $\rho(x) = \max(0, x)$. The most appropriate choice strongly depends on the specific application. A schematic representation of the perceptron is depicted in figure 1.1.

The architecture of a Neural Network is determined by the number of perceptron and by the way they are linked among themselves. When dealing with function approximation the most natural choice consists in using *Multilayer Feedforward Perceptrons* (MFP). In this particular kind of NN, perceptrons with the same activation functions are collected in layers; every layer receives information from the previous one and transmits it to the

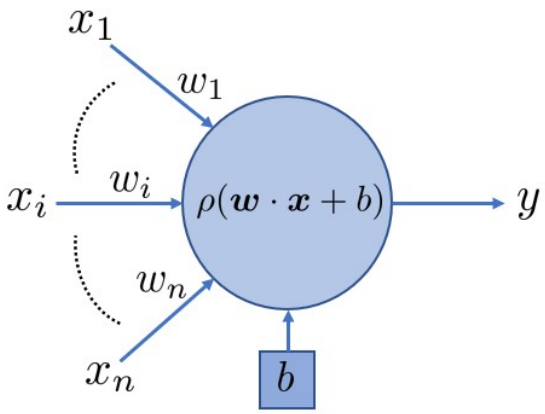


Figure 1.1: A schematic representation of the Perceptron which is the fundamental unit of a Neural Network. The Perceptron makes the weighted sum of the inputs, adds the bias b and transmits the result, after the application of a nonlinear function ρ.

following, except for the first one, the *input* layer, which take the information from the outside, and the last one, which provides the output of the computation. By collecting weights and biases of the k-layer respectively in the matrix W_k and in the vector $\boldsymbol{b}_k$, the layer output can be computed as $\boldsymbol{x}_k = \rho_k(W_k\boldsymbol{x}_{k-1} + \boldsymbol{b}_k)$, where the application of the function ρ_k is intended componentwise. Therefore, the neural network output can be obtained by the composition of linear maps and scalar nonlinear activation function, so that the evaluation of an input vector is extremely fast independently from the neural network dimensions. This fact clearly explains why NNs gained an extreme popularity as surrogate models, together with their capability in learning efficiently any kind of relation. Figure 1.2 represents a three layer MFP, showing through the colors how the different NN components play a role in producing the output. The absence of the activation function in the output layer is common [17] and due to the fact that, generally speaking, the output components take values in $\mathbb{R}$.

1.1.1. Approximation Properties and Learning Procedure

A large variety of theoretical results (for instance [17, 32, 35]) describes the capabilities of this networks in approximating with arbitrary accuracy any kind of function, if the architecture is sufficiently expressive, namely if the NN posses an appropriate number of layers and neurons. This kind of results, available for functions belonging to different functional spaces and stated in different norms, often prescribe the minimum number of layers and dof that are needed to create a correct internal representation of the function to approximate. Anyway, their use is often impractical and the architecture is fixed using empirical or semi-empirical approaches, like the *ensemble training procedure* [22].

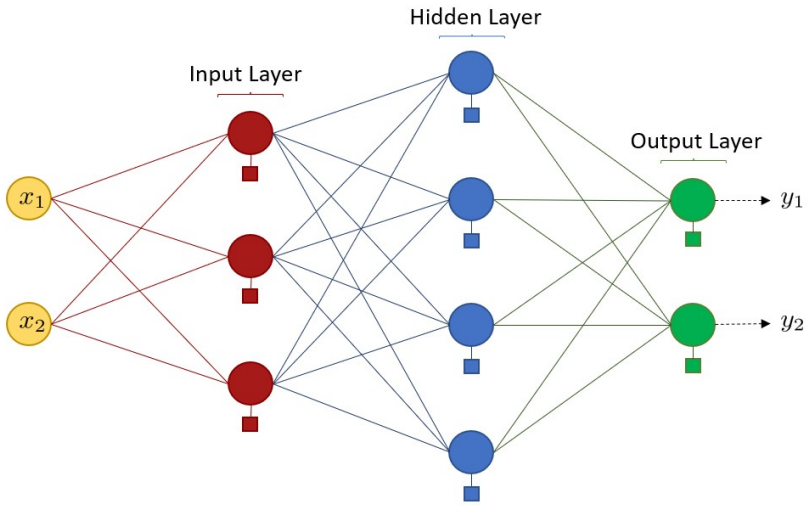


Figure 1.2: A three layer MFP mapping 2D inputs into 2D outputs. The layers are connected by linked weights, whose values are collected in the corresponding matrices W_i, whereas the biases (represented through squares) are collected in the vectors $\boldsymbol{b}_i$. The relation below shows how the NN functioning is based on the composition of each layer transformation, with the last one which is usually linear.

In order to explain the reason behind this fact, it is necessary to understand in what the learning process consists and how it is performed. With this aim, let us introduce some notation. Let λ be a probability measure on $X \subset \mathbb{R}^n$, $\mathcal{L} : X \subset \mathbb{R}^n \to \mathbb{R}^m$ be the map to approximate, Ξ be the set of all the possible configuration of the NN inner parameters (i.e. weights and biases) and $\mathcal{L}_\theta : X \subset \mathbb{R}^n \to \mathbb{R}^m$ be the NN for a specific choice of $\theta \in \Xi$. The aforementioned theoretical results state, generally speaking, that, for any $\epsilon > 0$, there exists a a NN architecture Ξ and a dof setting $\theta^* \in \Xi$ such that

$$LF(\theta^*) = \mathbb{E}_\lambda ||\mathcal{L} - \mathcal{L}_{\theta^*}|| < \epsilon \tag{1.1}$$

where we indicate by $\mathbb{E}_\lambda$ the expected value with respect to the measure λ, and by $||\cdot||$ the norm through which measuring the discrepancy, whereas LF is the so-called *Loss Function*, and ϵ is sometimes referred to *approximation error* [12]. However, finding this minimum is an unreachable goal from the numerical point of view, because of the strong non-convexity and the high dimensionality of the loss surface associated to the simultaneous optimization of thousands degrees of freedom.
Then, in the context of supervised learning, the standard way [19] to proceed consists in leveraging data, in the form of input-output couples generated by the function to

approximate, in order to perform a regression. More specifically, the values of the NN weights and biases are optimized during an iterative process called *training* in which, through gradient-descent like algorithms, an empirical version of the Loss Function is minimized. Being $S = \{(x_i, \mathcal{L}(x_i))\}$, with $i = 0, ..., N_{tr}$, the set of the training data, we can define the Empirical Loss Function $\widehat{LF}$ as

$$\widehat{LF}(\theta) = \frac{1}{N_{tr}} \sum_{i=1}^{N_{tr}} ||\mathcal{L}(x_i) - \mathcal{L}_\theta(x_i)||. \tag{1.2}$$

The training continues for a number of *epochs* (that is the unit time in which the whole set of data is "examined" by the NN) sufficient to reach a sub-optimal solution, represented by a local minimum $\theta^\#$ on the loss surface, which guarantees an adequate level of accuracy. Coming back to the initial issue, theoretical estimates are not frequently used because it is easier and less expensive, in terms of computational cost, to find a good local minimum using networks that are way more complex than the ones prescribed by the theory, even if this has an impact.

Indeed, the training process introduces two further sources of error. First, the fact that we are able to find only a sub-optimal minimum, gives raise to the so-called *training error*. As is customary, since from the user point of view they are not distinguishable, the approximation error is incorporated in the training error. In this way it possible to quantify it through

$$E_{tr} = \widehat{LF}(\theta^\#)$$

and it can be reduced by augmenting the NN expressiveness or improving the learning process. The second issue is related to the finite amount of data available for the training. In particular, this is a consequence of the replacement of the Loss Function by its empirical version during the optimization. For this reason, in addition to the training error, the *generalization gap* G has to be considered. The latter can be roughly estimated through the central limit theorem as

$$G \sim \frac{\text{var}(||\mathcal{L} - \mathcal{L}_{\theta^\#}||)}{\sqrt{N_{tr}}}.$$

As observed in [25] in fact, if the sample $\{x_i\}_{i=1}^{i=N_{tr}}$ are independently drawn from λ, then the comparison between ((1.1)) and ((1.2)) makes it evident that $\widehat{LF}$ is the Monte Carlo approximation of LF. Despite being of impractical use, this fact sheds light on the (usual) very slow decay with respect to the training sample size which becomes highly problematic in small data regime or, as in our contest, when the dataset is generated with high computational costs. Moreover, it allows a better understanding of what is

over-fitting, from a mathematical perspective. If the NN architecture is decisively more complex than necessary and the data used for the training are too few, the training process might generate an unusable NN surrogate that perfectly interpolates the function being approximated at the training points, but is not able to make accurate predictions on new samples, or in other words, to *generalize* properly. For this reason, the more important indicator of a NN based surrogate is not the training error, but the *test error* which is defined through

$$E_{te} = E_{tr} + G$$

and can be estimated by evaluating the Empirical Loss Function on a different set of points, the *test set*, from the ones used in the training.

1.1.2. Back-propagation and L-BFGS optimizer

The implementation of the gradient descent algorithm in the context of the deep learning translates into the *back-propagation algorithm* [30]. In its basic version, during every epoch k, the Empirical Loss Function is evaluated with the current configuration of the NN on all the training set and the values of the dof are updated according to

$$\theta_{k+1} = \theta_k - \alpha \frac{\partial \widehat{LF}}{\partial \theta}\Big|_{\theta=\theta_k}; \tag{1.3}$$

α is the hyper-parameter that regulates the step length on the error surface and is usually called *learning rate*. Here the main issue is clearly the computation of the partial derivative, that in this context is commonly called sensitivity, in particular for the nodes that are not immediately adjacent to the output layer. The solution provided by the back-propagation algorithm consists in updating before the weights and biases of the output layer and then proceeding backwards, layer by layer. By exploiting the chain rule of differential calculus, sensitivities can be expressed through linear combination of first order derivatives of linear functions, activation functions and already computed sensitivities.

Despite the numerical efficiency of the described procedure, many variants of the updating rule ((1.3)) have been proposed in order to overcome some performance issues of the optimizer. Among them, slow convergence and stability issues might arise, due to the choice of the learning rate, as well as the risk of remaining trapped in a bad local minimum during the early stages of the training process. Despite the vast majority of the literature employs first order optimizer [29], at the most combined with "mini-batch" approaches like for ADAM [15], in the context of this work great effectiveness has been shown by

the quasi-Newton second order optimizer Limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm (L-BFGS). In this version only the most recent part of the descent history is used to produce an approximation of the hessian matrix inverse, but this was enough to make a big difference in terms of accuracy and time implied by the training. For further details, see e.g. [4].

1.2. DL-ROMs

As already stated in the Introduction, this work deals with much more complex relations than just low dimensional vectorial functions. Indeed, the goal is to approximate the map $\boldsymbol{\mu} \to u_{\boldsymbol{\mu}}$, where $\boldsymbol{\mu}$ is a vector of coefficients parameterizing a PDE, while $u_{\boldsymbol{\mu}}$ is the corresponding solution belonging to some Hilbert space $(\mathcal{V}, ||\cdot||)$. Deep Learning based Reduced Order Models are an original Machine Learning framework, recently developed in [7, 9, 10] with the aim to efficiently accomplish this task. The innovative idea consists in splitting the approximation process in two phases. First, a low dimensional representation of the solution manifold is determined through a deep auto-encoder: this operation allows to compress the massive amount of information coming from a functional space, which is theoretically infinite-dimensional, into a much smaller *latent space*. Afterward, the learning problem can be rephrased into the approximation of the low dimensional relation between the parameter vector and the latent representation of the solution. The latter can be therefore tackled with classical Machine Learning techniques.

1.2.1. Theoretical foundations

Before entering into the implementation details, we report the main theoretical results that support this approach, which were developed by *Franco N.R., Manzoni A., Zunino P.* in [7], with minor modifications. In particular, the Theorems are presented in a simplified and less general version, tailored for the specific problems tackled in this work. These results play an essential role when it comes to designing the NN architecture, as they prescribe the minimal dimension of the latent space at which one may carry out an arbitrarely accurate compression of the solution manifold. This information is in turn used to fix the width of the deep auto-encoder bottle-neck, with a huge impact on the NN number of dof and, as a consequence, on the computational time requested by its training.

We temporarily leave the Machine Learning context to set up some notation, useful to the development of the theory. We define an auto-encoder as the composition of two maps,

Ψ' and Ψ. The former is the so called *encoder* $\Psi^{'} : \mathcal{V} \to \mathbb{R}^n$ which receives as an input a solution instance and maps it into the latent space, operating the compression. The latter is the *decoder* $\Psi : \mathbb{R}^n \to \mathcal{V}$, which does the opposite work reconstructing the entire solution from the compressed version.
With the only requirement of being continuous, encoder and decoder are chosen in order to minimize the error committed in the auto-encoding process. The latter is quantified by using the continuous version of the Nonlinear Kolmogorov n-width, as defined in [5], namely

$$\delta_n(\mathcal{S}) = \inf_{\substack{\Psi^{'} \in \mathcal{C}(\mathcal{S}, \mathbb{R}^n) \\ \Psi \in \mathcal{C}(\mathbb{R}^n, \mathcal{V})}} \sup_{u \in \mathcal{S}} ||u - \Psi \circ \Psi^{'}(u)|| \ ,$$

where we denote by $\mathcal{S}$ the solution manifold. $\delta_n(\mathcal{S})$ is clearly nonincreasing in n, and better results may be obtained by increasing the dimension of the latent space, from now on called *latent dimension*. However, as anticipated, from a computational point of view it is extremely important to keep the latent dimension as low as possible. For this reason the authors were interested in determine the *minimal latent dimension*, namely

$$n_{\min}(\mathcal{S}) = \min\{n \in \mathbb{N} \mid \delta_n(\mathcal{S}) = 0\},$$

for which the existence of a couple $(\Psi, \Psi^{'})$ allowing a nearly perfect auto-encoding of the solution manifold is guaranteed. With this premise, we may now report one of the results in [**?**] that will be our starting point in Chapter 3.

Theorem 1.1. *Let $\boldsymbol{\mu} \to u_{\boldsymbol{\mu}}$ be a map from a compact set $\Theta \subset \mathbb{R}^p$, with nonempty interior, to some Hilbert space $(\mathcal{V}, ||\cdot||)$. Define the sets $\mathcal{S} = \{u_{\boldsymbol{\mu}}\}_{\boldsymbol{\mu} \in \Theta}$. We have the following:*

(i) if the map $\boldsymbol{\mu} \to u_{\boldsymbol{\mu}}$ is Lipschitz continuous, then $n_{min}(\mathcal{S}) \leq 2p + 1$;

(ii) if the map $\boldsymbol{\mu} \to u_{\boldsymbol{\mu}}$ is continuous and injective, then $n_{min}(\mathcal{S}) = p$.

Furthermore, in both cases the infimum is attained, meaning that there exists a pair (Ψ', Ψ), with latent dimension respectively 2p+1 and p, such that $u = \Psi \circ \Psi'(u)$ for all $u \in \mathcal{S}$.

This result, which holds for every kind of map satisfying the hypotheses without referring solely to the PDEs context, serves as a base for the second one, which is specifically designed for the kind of problems faced in this work: diffusion-advection equations. In order to state it, we need to provide some notation. We denote by D a bounded domain in $\mathbb{R}^d$. We then define the sets of all admissible conductivity tensor-fields $\Sigma(D) \subset L^{\infty}(D, \mathbb{R}^{d \times d})$, such that $\sigma \in \Sigma(D)$ if and only if it is uniformly elliptic, and the sets of all the admissible

transport fields $B(D) \subset L^\infty(D, \mathbb{R}^{d\times d})$, such that $\boldsymbol{b} \in B(D)$ if and only if it is differentiable and divergence free. Finally, we denote by $H^1(D)$ the Sobolev space of those $L^2(D)$ functions whose partial derivatives are in $L^2(D)$, by $H^{1/2}(\partial D)$ the associate space of traces and by $H^{-1}(D)$ the dual space of $H_0^1(D) = \{v \in H_0^1(D) \text{ such that } v|_{\partial D} = 0\}$. Now we are ready to state the following:

Theorem 1.2. *Let $D \subset \mathbb{R}^d$ be a bounded domain with Lipschitz boundary, and let $\Theta \subset \mathbb{R}^p$ be a compact subset with nonempty interior. Moreover, let $\boldsymbol{\mu} \to \sigma_{\boldsymbol{\mu}} \in \Sigma(D)$, $\boldsymbol{\mu} \to b_{\boldsymbol{\mu}} \in B(D)$, $\boldsymbol{\mu} \to f_{\boldsymbol{\mu}} \in H^{-1}(D)$ be parameter dependent coefficients and $\boldsymbol{\mu} \to g_{\boldsymbol{\mu}} \in H^{1/2}(\partial D)$ be the boundary data. For each $\boldsymbol{\mu} \in \Theta$, we define $u_{\boldsymbol{\mu}} \in H^1(D)$ as the unique solution to the following second order elliptic PDE*

$$
\begin{aligned}
&u \in H^1(D): \\
&\quad u_{|\partial D} = g_{\boldsymbol{\mu}} \text{ and } \int_D \sigma_{\boldsymbol{\mu}} \nabla u \cdot \nabla w + \int_D (b_{\boldsymbol{\mu}} \cdot \nabla u)\, w = \int_D f_{\boldsymbol{\mu}} w \quad \forall w \in H_0^1(D).
\end{aligned}
$$

Consider the solution manifold $\mathcal{S} = \{u_{\boldsymbol{\mu}}\}_{\boldsymbol{\mu}\in\Theta}$ as a subset of $\mathcal{V} = L^2(D)$. The following hold true:

(i) if the dependence of $\sigma_{\boldsymbol{\mu}}$, $b_{\boldsymbol{\mu}}$, $f_{\boldsymbol{\mu}}$, $g_{\boldsymbol{\mu}}$ on $\boldsymbol{\mu}$ is Lipschitz continuous, then $n_{min}(\mathcal{S}) \leq 2p + 1$.

(ii) if $\sigma_{\boldsymbol{\mu}}$, $b_{\boldsymbol{\mu}}$, $f_{\boldsymbol{\mu}}$, $g_{\boldsymbol{\mu}}$ depend continuously on $\boldsymbol{\mu}$ and the solution map $\boldsymbol{\mu} \to u_{\boldsymbol{\mu}}$ is injective, then $n_{min}(\mathcal{S}) = p$.

Obviously, the results above still hold when dealing with the discretized solution manifold $\mathcal{S}_h = \left\{u_{\boldsymbol{\mu}}^h\right\}_{\boldsymbol{\mu}\in\Theta}$, where $u_{\boldsymbol{\mu}}^h$ is the solution of PDE by means of an high fidelity FOM, which lives in some finite dimensional subspace $(\mathcal{V}_h, ||\cdot||)$. As a consequence, by fixing $n = n_{\min}$ and exploiting the aforementioned approximation results for the MFP, the authors proved the following: for any $\epsilon > 0$, there exist two NN, $\Psi_{NN}^{'} : \mathcal{V}_h \to \mathbb{R}^n$ and $\Psi_{NN} : \mathbb{R}^n \to \mathcal{V}_h$, playing respectively the role of the encoder and of the decoder, such that

$$
\sup_{\boldsymbol{\mu}\in\ \Theta} \ ||u_{\boldsymbol{\mu}}^h - \Psi_{NN} \circ \Psi_{NN}^{'}(u_{\boldsymbol{\mu}}^h)|| < \epsilon \ .
$$

1.2.2. Training the model

Coming to the actual implementation, the first issue is the definition of the three NNs involved: not only for the encoder and the decoder but also for the NN $\phi_{NN} : \Theta \subset \mathbb{R}^p \to \mathbb{R}^n$, which approximates the relation between the parameter vector and the reduced

order solution. Concerning the auto-encoder, apart from the bottle-neck width, which is surely the most thorny design choice, the rest was determined through a semi empirical approach, based upon some authors' recommendations. These included the importance of using sparse convolutional layers to deal with high dimensional objects like FOM solutions and the fact that, in order to maximize the accuracy, the decoder should be richer than the encoder. A performing architecture for ϕ_{NN} is instead easy to find by using few dense layers, since $p, n \ll N_h = \dim(\mathcal{V}_h)$. For what concerns the training set $\{(\boldsymbol{\mu}_i, u^h_{\boldsymbol{\mu}_i})\}_{i=1}^{N_{tr}}$ (and the test set), as it is common when dealing with dimensionality reduction techniques, it is synthetically generated by sampling the parameter space and by invoking the FOM. The latter is used to compute, for every instance of the parameter vector, the corresponding snapshot. For the two-steps training phase we can finally refer to Figure 1.3, which depicts how a two dimensional solution depending on p parameters is elaborated. In the first phase, the auto-encoder is fed with the instances of the discretized solution manifold until a sufficiently accurate internal representation of the reduced solution manifold is formed. Good results have been obtained by considering, as a measure of discrepancy, the relative error. As a consequence, the Empirical Loss Function for the auto-encoder can be written as

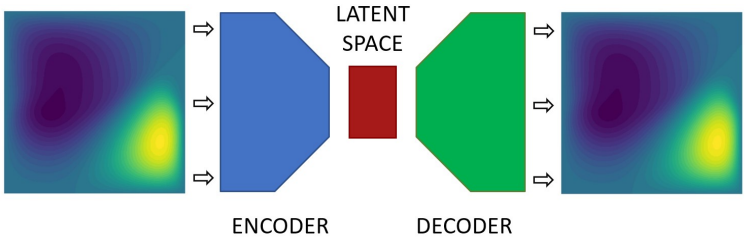
$$\widehat{LF}(\Psi_{NN}, \Psi'_{NN}) = \frac{1}{N_{tr}} \sum_{i=1}^{N_{tr}} \frac{||u^h_{\boldsymbol{\mu}_i} - \Psi_{NN} \circ \Psi'_{NN}(u^h_{\boldsymbol{\mu}_i})||}{||u^h_{\boldsymbol{\mu}_i}||} \ . \tag{1.4}$$

In the second one, decoder and encoder are separated and the latter is used to generate the reduced order version of the dataset, $\{(\boldsymbol{\mu}_i, \Psi'(u^h_{\boldsymbol{\mu}_i}))\}_{i=1}^{N_{tr}}$. This is used to train and test ϕ_{NN}, whose dof are optimized by minimizing the following loss function

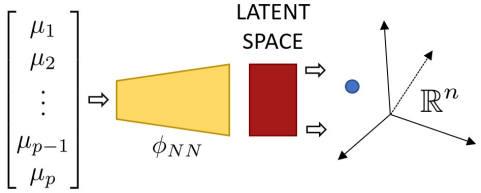
$$\widehat{LF}(\phi_{NN}) = \frac{1}{N_{tr}} \sum_{i=1}^{N_{tr}} ||\Psi'(u^h_{\boldsymbol{\mu}_i}) - \phi_{NN}(\boldsymbol{\mu}_i)|| \ . \tag{1.5}$$

The relative error cannot be chosen as discrepancy measure anymore, since the encoder may represent an instance of the solution manifold as the null vector in the latent space. Finally the decoder is connected to ϕ_{NN}, so that the resulting network maps a vector of parameters into the corresponding PDE solution and the DL-ROM is full operative. As already mentioned, both the minimization were run using L-BFGS optimizer for a number of epochs which depended on the problem complexity and on the desired accuracy. However, we can already anticipate that the training of the auto-encoder was considerably more demanding than the one of ϕ_{NN}. This is the reason why in Chapter 3, Chapter 4 and Chapter 5 the analysis is concentrated on the properties and the computational efficiency of the auto-encoder without paying the same attention at the role of ϕ_{NN}.

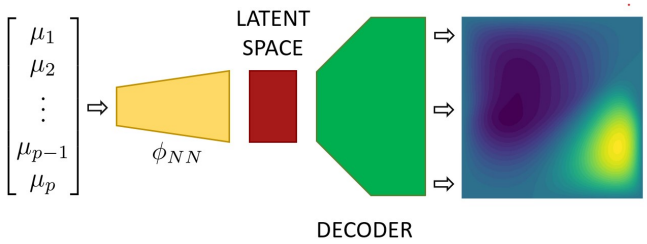
We conclude by highlighting some of the reasons that make worth adopting the DL-ROM approach. The first one is clearly related to the underlying theoretical basement that guarantees a good interpretability of what is happening inside the Neural Network, partially opening the *black-box*. The second reason is associated to computational costs: when the output space is so highly dimensional, being able to split the work in two different stages is certainly useful to lower the computational power and the memory needed to complete the training. Finally, as showed by the authors, through DL-ROM, we can exploit intrinsic regularities of the solution manifold, even if the parameter-to-solution map is just Lipschitz-continuous, as for the case of diffusion problems.



(a) The auto-encoder is trained to learn the identity map between the solution manifold and itself. This allows to operate a dimensionality reduction of the solution manifold.



(b) The encoder is used to compress the solution instances. These latter are then implied in the training of ϕ_{NN}.



(c) The surrogate model is finally formed by connecting ϕ_{NN} and Ψ.

Figure 1.3: The three-step process of the DL-ROM building.

1.3. Technical setup

All the numerical experiments ran throughout this work were implemented in Python 3 and run over GPUs. In particular, the high-fidelity snapshots used to train and test the NNs were generated through the FEniCS library [1], whereas the construction and the training of the DL-ROM exploited Pytorch [2].

2 | Mesh independence and multilevel training

After introducing a first model problem useful as a testbed for DL-ROMs design and training, this Chapter faces two key related issues. First, we concentrate on the surrogate flexibility with respect to the choice of the mesh in the FOM discretization and, then, on the high computational cost of generating the training set during the DL-ROM off-line stage. We address the former thanks to a recently developed type of layer and design a strategy to overcome the latter by drawing inspiration from one of the most commonly used approaches in the field of statistical learning, namely a multi-fidelity approach.

2.1. A first model problem

As already mentioned, this work focus on parametrized elliptic PDEs an, specifically, on advection-diffusion equations. In particular, in this Chapter the parameter-to-solution map is determined through the following PDE, set on the unit square domain $D = (0, 1)^2$, with Dirichlet boundary conditions:

$$\begin{cases} -\nabla \cdot (\sigma_\mu \nabla u) + \boldsymbol{\beta_\mu} \cdot \nabla u = 100(xy - y^2) & \text{in } D \\ u = 0.01 & \text{on } \partial D \end{cases} \tag{2.1}$$

It depends on four input parameters $\boldsymbol{\mu} = [\mu_1, ..., \mu_4]$ which belong to $\Theta = [0, 1]^4$ in a highly nonlinear way that was chosen in order to make the approximation problem particularly challenging. More specifically, three parameters play a role in the expression of the diffusion field, by determining the shape of a curve that crosses the domain and divides it in two areas with very different but homogeneous diffusivity properties. The latter parameter tunes instead the direction of the transport term.

The formal expression of $\sigma_{\boldsymbol{\mu}}$ and β_{μ} is given, respectively, by

$$\begin{cases} \sigma_{\boldsymbol{\mu}} = 6 + 5\tanh\left(20(y + 10\mu_1 x(x-1)(x-\mu_2)(x-\mu_3) - 0.5)\right) \\ \boldsymbol{\beta_{\mu}} = 10(\cos(2\pi\mu_4), \sin(2\pi\mu_4))^T \ , \end{cases}$$

whereas some instances of the diffusion field, together with the corresponding solutions, are depicted in Figure 2.1.

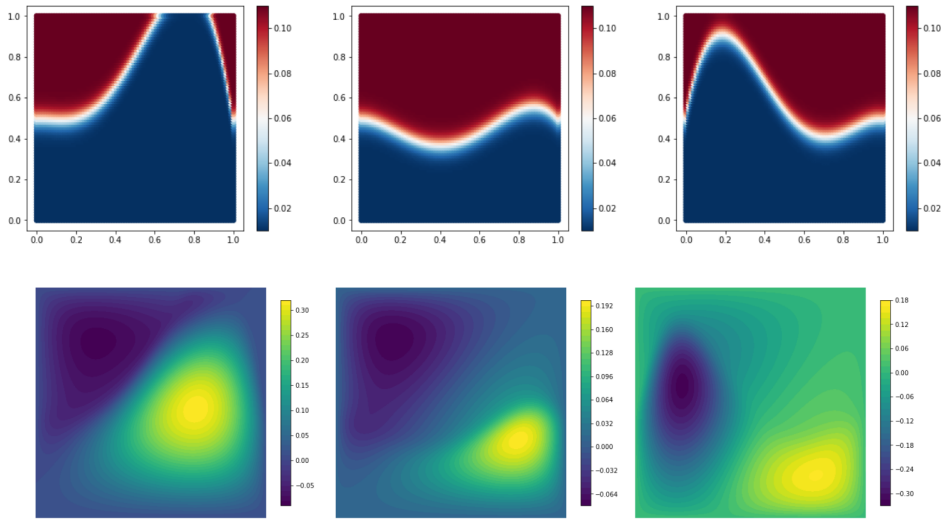


Figure 2.1: Some diffusion field instances (top) and the associated solutions (botton).

2.1.1. DL-ROM performances

We obviously started by generating the dataset, obtained by sampling uniformly the parameter space Θ since we do not assume any prior knowledge about the parameter distribution. We then solved the PDE for each sampled value of the parameter vector, exploiting the Galerkin - FE method. We used, as basis functions, first order polynomials on a structured triangular mesh, with mesh size $h = 0.01$; in this case, the output space has dimension $N_h = 10201$. We fixed the training set dimension at 1200 elements, whereas the test set was composed by 5000 instances. Having a test set which is considerably greater than the training set ensures indeed a much more reliable estimate of the test error. For this reason, the same proportion has been maintained for every experiment in this work. Concerning the three NNs design, once fixed the parametric architectures reported in Table 2.1, we ran various test to determine the best value for m and k. Good results in terms of training error have been reached using $m = 16$ and $k = 4$, resulting in $\approx 3.5 \times 10^6$ dof for the auto-encoder and $\approx 5 \times 10^5$ for ϕ. All the layers, except for the last one, are

equipped with the 0.1-leaky ReLu activation function, namely

$$\rho(x) = 0.1x\mathbb{1}_{\{x<0\}} + x\mathbb{1}_{\{x>0\}} \ .$$

	Layer	Input	Output	Kernel	Stride	Dof
Ψ'	Dense	10201	4	-	-	51005
Ψ	Dense	4	$100m$	-	-	$400(m+1)$
	Deconv.	$5 \times 5 \times 4m$	$19 \times 19 \times 2m$	11	2	$968m^2 + 4m$
	Deconv.	$19 \times 19 \times 2m$	$46 \times 46 \times m$	10	2	$200m^2 + 2m$
	Deconv.	$47 \times 47 \times m$	101×101	11	2	$121m + 1$
ϕ	Dense	4	$50k$	-	-	$200k$
	Dense	$50k$	$50k$	-	-	$2500k^2$
	Dense	$50k$	4	-	-	$200k$

Table 2.1: The parametric architecture used for these experiments. By choosing $m = 16$ and $k = 4$, the model has more than 4×10^6 dof, 75% of them belonging to the decoder.

Both the auto-encoder and ϕ_{NN} were trained for 200 epochs, but for the auto-encoder the training took three time because of its more complex architecture. Figure 2.2 reports the error paths, followed by the auto-encoder and by ϕ_{NN} during the training, obtained using different seed for the random initialization of the dofs value. Highlighted through a wider line, there is the mean path. In order to obtain more interpretable results, we depicted in the same Figure the AE mean relative error and also the $E_{\phi,\text{TOT}}$, defined through

$$E_{\phi,\text{TOT}} = \mathbb{E}\left[\frac{||u^h_{\boldsymbol{\mu}} - \Psi_{NN} \circ \phi_{NN}(\boldsymbol{\mu})||}{||u^h_{\boldsymbol{\mu}}||}\right] \ .$$

The latter indeed, on the contrary to the error minimized during the ϕ_{NN} training (see eq. (1.5)), allows to monitor how accurately we are approximating the parameter-to-solution map. Despite the fact that the model problem is scalar, solved in a two-dimensional domain and not even dominated by the transport term, the learning problem is still difficult to tackle. In particular the training error is above the 2% threshold, even though the implemented architecture is quite complex and we are using a powerful optimizer such as L-BFGS. The real issue is, however, the generalization gap: the test error is just barely below 5% despite the high number of snapshots used for the training.

We take finally advantage of this picture to highlight the great variability in the error

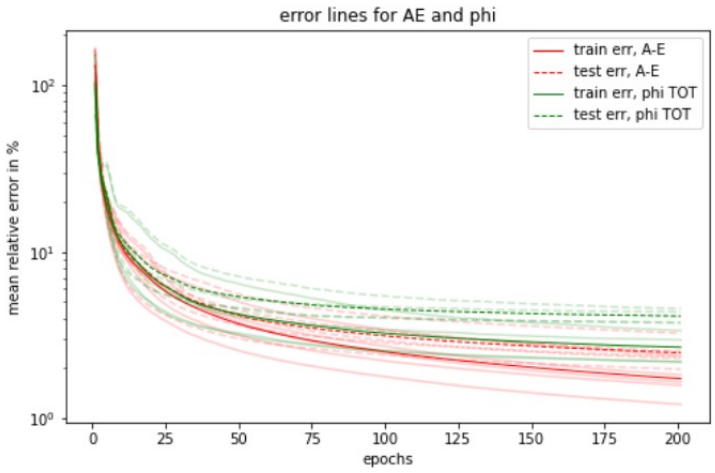


Figure 2.2: Model problem error paths

paths due to the random initialization of weights and biases. In order to make the results of the numerical experiments more reliable every training process has been repeated from 5 to 10 times. The analyses were then carried out on the averaged error values. We adopted this methodology for the whole work.

2.2. Mesh independence

The identification of an architecture that is sufficiently expressive to learn the map correctly, but without too many degrees of freedom (in order to avoid over-fitting), is a difficult task carried out with semi-empirical approaches. Once the latter has been accomplished, the DL-ROM must be trained with a non negligible computational effort. These facts motivate the development of a strategy to make the DL-ROM independent of the specific mesh on which it was trained. Strictly speaking, this is not feasible without substantially modifying the learning framework: the auto-encoder has indeed to be defined for a specific choice of the mesh, typically the same one used to generate the snapshots with the FEM. However, for two different discretization steps h_1 and h_2, $u_{\boldsymbol{\mu}}^{h_1}$ and $u_{\boldsymbol{\mu}}^{h_2}$ are the Galerkin projections of the same PDE solution on two different FE spaces. As a consequence, a DL-ROM trained over snapshots generated with a coarse mesh should contain useful information also about the parameter-to-solution map obtained with a more refined mesh. In particular, let us assume that the involved meshes are suitable to solve the problem without giving rise to stability issues and qualitatively catching all the solution features. The Lipschitz-continuity (and possibly the injectivity) of the PDE solution holds independently of the mesh choice. Then, by applying Theorem 1.2, we can determine the same minimal latent dimension for both the Galerkin-FE problems.

This suggests that the low dimensional representation of the solution manifold should be compatible with both the choices of the mesh. This means that the auto-encoding procedure shall be actually done once and for all, possibly for the coarser mesh and, as a consequence, with a smaller architecture. The same kind of reasoning applies to the neural network ϕ_{NN}, mapping the values of the parameters to the latent space: the choice of its architecture should not be affected by the mesh resolution, therefore, once trained on the coarser mesh, it can be used also for the finer one.

We can take advantage of these comments to design a strategy to adapt DL-ROMs to more refined meshes and by relying on the possibility to *freeze* the layers. This operation consists in fixing the values of weights and biases of a layer, preventing further modification during subsequent trainings. In conclusion, in order to obtain the desired architecture starting from the DL-ROM with the courser mesh, it is sufficient to freeze the whole NN made by the composition of ϕ_{NN} and the decoder Ψ_{NN} and compose the latter with a third NN χ_{NN}, initialized as usual, that interpolates the DL-ROM approximation on the finer mesh. Finally, $\Psi_{NN} \circ \phi_{NN} \circ \chi_{NN}$ has to be fed with some snapshots generated on the finer mesh to optimize the additional weights and biases. This process is depicted in Figure 2.3 as an ideal continuation of the standard DL-ROM implementation in Figure 1.3.

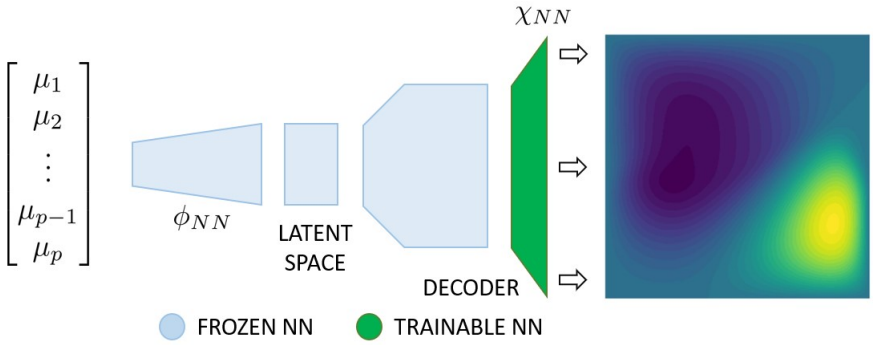


Figure 2.3: DL-ROM mesh adaptation

2.2.1. Technical aspects

Although following the presented strategy might seem straightforward, numerical tests have actually shown that choosing in a proper way χ_{NN} is a difficult task. In particular, dense and convolutional layers, namely the two principal blocks of DL-ROMS, are not suitable tools in this context, due to different reasons.

Regarding dense layers, they add too many degrees of freedom, making the resulting NN hardly trainable. For instance, we can consider the elliptic PDE (2.1) of the previous section, solved on $(0,1)^2$ using two structured triangular meshes, with mesh size $h = 0.02$ and $h/2 = 0.01$ respectively. The associated finite element spaces have dimensions $\dim(V_h) \simeq 2.5 \times 10^3$ and $\dim(V_{h/2}) \simeq 10^4$. For the approximation of this problem with the finer mesh, an architecture with 4×10^6 dofs (see Table 2.1) guarantees an acceptable level of accuracy. However, a single dense layer χ_{NN} for this case contains $\simeq 2.5 \times 10^7$ parameters to be trained.
For what concerns convolutional layers, the problem is instead the opposite: they are characterized by a very limited number of dofs. The only way to overcome the subsequent lack of expressiveness is to greatly increase the number of channels. Furthermore, a large number of layers is needed in order to pass from a certain degree of resolution of the "solution-image" to another arbitrarily detailed, with particular choices for the shape of the windows and for the stride. The resulting very deep NN is once again hard to train and might potentially show bad performances.
Very good results, in terms of accuracy and trainability of the model, have been achieved using *Mesh Informed* layers, recently introduced in [8]. In these layers, neurons are represented as nodes in the mesh: then, when passing from a layer to another one, only nearby nodes are allowed to communicate. This introduces a certain level of sparsity in the weights matrix which can be regulated through a *support* hyper-parameter, that quantifies the nodes vicinity. The resulting layer is much more expressive and elastic than a convolutional one. On the other hand, the weights matrix sparsity allow to reduce the training time and the generalization gap with respect to dense networks, by letting the spatial correlation play a role. A graphical representation of a mesh informed NN in comparison with a dense one is depicted in Figure 2.4.

Instead of showing the result of any numerical experiment at this stage, we postpone to the next section the validation of the proposed strategy efficacy. Indeed, the *mesh-independent* strategy, once supplied with the mesh-informed layer for the implementation of the NN χ_{NN}, performed so well that we were able to develop a multi-level training approach based on it.

2.3. Multilevel training

As emerged clearly from the first section of this Chapter, the NN architecture involved in the approximation of the parameter-to-solution map by means of a DL-ROM approach are quite big. The high number of dofs allows to reach adequate level of training error

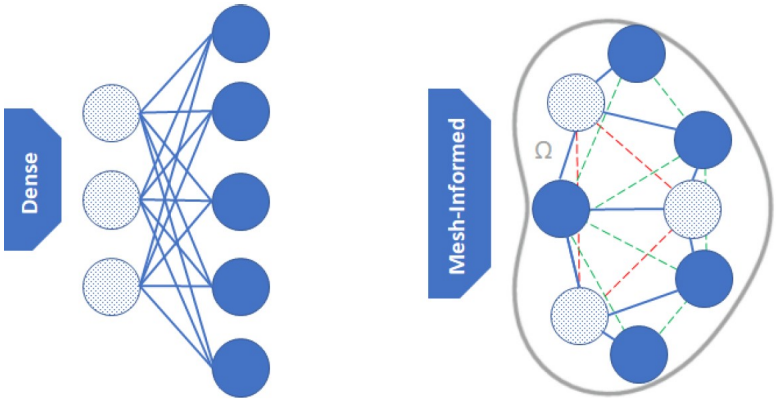


Figure 2.4: Comparison between a Mesh Informed NN (right) and a dense one (left), taken from [8]. The continuous lines represent the links between the NN nodes, while the dashed ones identify two (oversimplified) meshes on the domain.

but also increases the risk of overfitting. We then need a large amount of training samples to lower the generalization gap, since its decay rate satisfies a Monte Carlo-like bound, decreasing as $1/\sqrt{N_{tr}}$, with N_{tr} being the dimension of the training set. However, we recall that in the ROM context, data (namely, snapshots) are generated through high-fidelity FOMs. Therefore, the computational cost for the construction of a sufficiently large training set may easily became unaffordable.
Various attempts to regularize the learning process can be found in the literature for the classical deep feed-forward neural networks, following several different paths. First of all, the commonly used L^p-penalization, in which the Loss Functions takes into account also the total number of active weights. Other more recent examples, in the field of Scientific Machine Learning, are provided by the PINNs [3], which leverage the problem physics, the use of low discrepancy sequences for the sampling of the parameter space [25] and also of a multi-fidelity framework [20]. The latter work, in particular, implements a series of NN based approximators where the first one operates in the standard way, the next one learns the map between the input and the error committed by the network and so on.

Here we develop and test another kind of multi-fidelity strategy, that we call Multi-level Training, since it exploits multiple discretization levels to generate training data with various fidelity and, as a consequence, different costs. The main idea is then not to lower the number of required training data demanded but to reduce the computational time needed for each snapshot generation, by exploiting a "low-fidelity" FOM. This strategy is entirely based on the learning framework presented in the previous section. However, in this case, we initially aim to build a DL-ROM working on a high-fidelity mesh and, for this reason, we implement an auxiliary DL-ROM for a coarser mesh. The algorithmic

procedure can be summarized as follows:

1. use the FOM to generate a big dataset with a coarse mesh and a small dataset with the refined one;
2. design and train in the standard way a small DL-ROM working on the former;
3. freeze the obtained NN and connect to the latter a further network χ_{NN} that maps the solution of the coarser mesh to a more refined one;
4. re-train the multi-fidelity model $\phi_{NN} \circ \Psi_{NN} \circ \chi_{NN}$ on a few solution instances for the refined mesh.

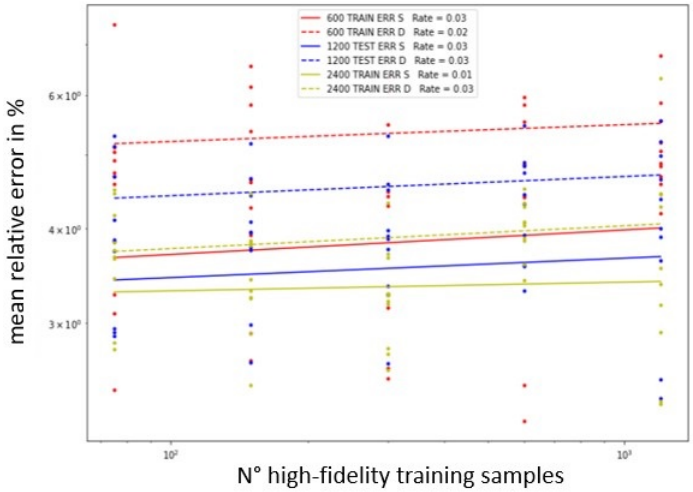
Thanks to its simpler architecture and the great amount of data available, the "low-fidelity" DL-ROM is able to generalize extremely well. As a consequence, the success of the whole strategy clearly depends on whether or not this property is inherited by the multi-fidelity model. We assessed this fact by a series of numerical experiments.

2.3.1. Numerical experiments

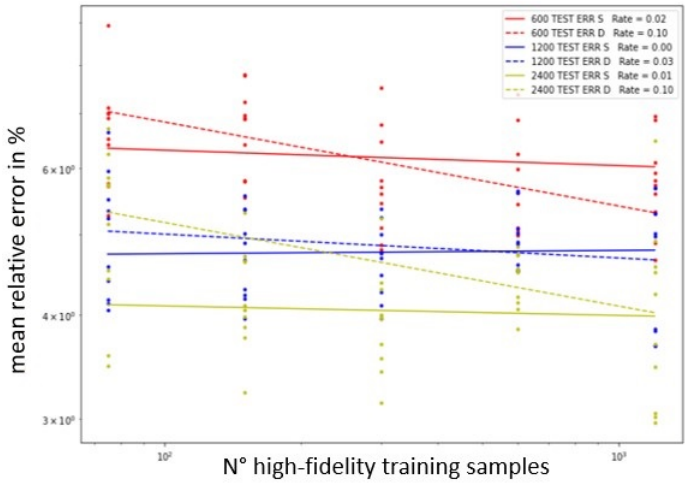
The numerical experiments were carried out on Problem (2.1), using a 100×100- elements high-fidelity mesh and a 50×50-elements coarse mesh. By using our technical setup, the time required for solving the PDE for a fixed value of the parameters is approximately 0.18 s on the fine mesh and 0.05 on the coarse mesh. The experiments were run comparing the performances of the proposed Multi-Level Training strategy with the ones obtained in the standard way. This was done by varying the cardinalities of both the low and the high-fidelity training sets. For what concerns the implied architecture, the high-fidelity DL-ROM against which we compare the new Multi-level strategy is the same of the first section, described in Table 2.1. The low-fidelity DL-ROM has been designed with the same ϕ_{NN} and a much more lighter auto-encoder with just one third of the dofs of the high-fidelity one. The fairness of the experiment is guaranteed by the fact that the DL-ROMs share the same expressiveness, having reached similar levels of training error. After the first training, we connected the low-fidelity DL-ROM to a shallow Mesh-Informed NN χ_{NN}, mapping the coarse mesh onto the fine one.

Before analysing the strategy efficiency, we shall answer a preliminary question. It is indeed unclear what the best way to sample the parameter space is when generating the training set for the re-training. The are two obvious possibilities: to sample new parameter instances with the aim of further increase the generalization capabilities or to use a high fidelity version of the snapshots already involved in the first training phase, obtained for

the same parameter values. Note that there is no difference in the computational cost entailed by the two strategies. However, according to our numerical experiments, the first choice does not necessarily entail improvements in the performances. Figure 2.5 depicts the training error (on the top) and the test error (on the bottom) when varying the number of high-fidelity training samples with the two possible strategies. We used continuous lines for the case of same parameters instances and dashed lines for the different ones. The experiment was repeated for three sizes of the low fidelity training set.



(a) Mean relative train error



(b) Mean relative test error

Figure 2.5: Multi-level training errors obtained by using the same parameter instances and new ones, on varying the dimension of the high-fidelity training set, for different cardinialites of the low-fidelity training set.

The dots represent the single training result (every training is repeated five times), while the lines are obtained through regression and the rate refers to the decay in the number of high-fidelity snapshots $\propto N_{tr}^{-R}$. As we can clearly see from the plot on the right, the use of a different set of parameters for the re-training allows the generalization gap to decrease further. Anyway, the simultaneous increase in the training error gives rise to a trade-off. This can be heuristically explained by the fact that the DL-ROM more easily adapts to a refined mesh if we feed it with already encountered instances of the solution. From the figure on the bottom emerges that when the number of high-fidelity training points is lower than the one generated through the coarse mesh the test error is lower if we use the same parameter instances. As a consequence, the latter is the strategy that we adopted, since this is the data regime for every Multi-level approach.
Finally, we can compare the test error obtained with the Multi-Level Training with the standard one, exploiting a dataset containing only high fidelity snapshots. As showed in Figure 2.6, the latter decays precisely as predicted by the Monte-Carlo like estimate, decreasing from 20% with 75 samples to below 3% with 2400 samples. The performance of the Multi-Level Training is instead not affected by the amount of high-fidelity data used for the re-training, as demonstrated by a decay rate equal to 0. The accuracy depends solely on the amount of row data involved in the first training phase. This fact confirms that the Multi-Fidelity DL-ROM is actually able to inherit the generalization capabilities of the low-fidelity DL-ROM and just a few high-fidelity solution instances are needed in order to calibrate the added dof.

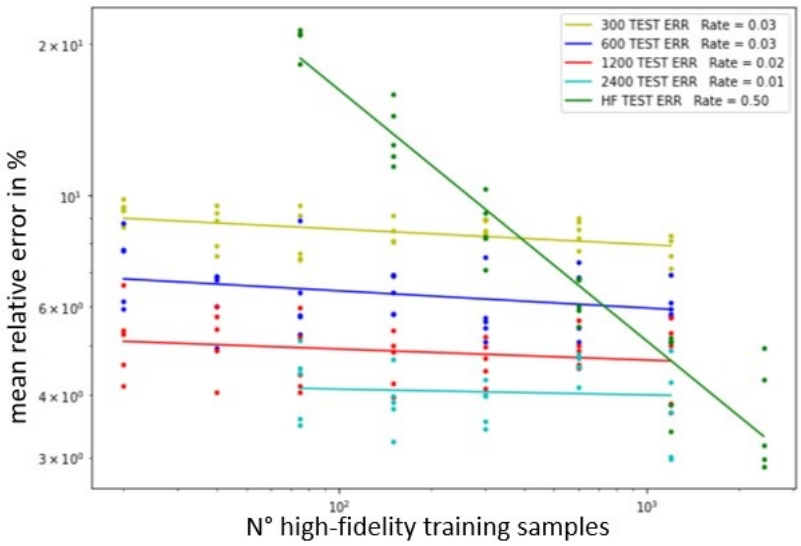


Figure 2.6: Comparison between the decay in the test error obtained with a standard training strategy and with the Multi-level approach, on varying the cardinality of the training set.

We summarize the obtained results in Table 2.2 by showing, besides the test error, the off-line computational time implied for three different combinations of coarse and high-fidelity samples, respectively denoted by (C) and (H). The time is divided in *total training time*, accounting the duration of the whole learning procedure, and total *dataset generation time*. This latter is drastically reduced up to 65%. Furthermore, since we are splitting the optimization of a very large number of dofs in two steps, the strategy makes also the training time decrease, at least for large training sets. The only drawback is a slight decrease in the accuracy levels.

Model	**Test err**	**Samples N°**	**Tot. training T.**	**Data. Gen. T.**
High-F.	8.61%	C:0/H:300	4m 58s	57s
Multi-F.	8.24%	C:300/H:75	5m 37s (+13%)	30s (-47%)
High-F.	6.19%	C:0/H:600	5m 56s	1m 52s
Multi-F.	6.41%	C:600/H:75	5m 50s (-1.7%)	46s (-59%)
High-F.	4.51%	C:0/H:1200	7m 50s	3m 46s
Multi-F.	4.76%	C:1200/H:75	6m 23s (-18%)	1m 18s (-64%)
High-F.	3.65%	C:0/H:2400	11m 38s	7m 32s
Multi-F.	3.90%	C:2400/H:150	7m 35s (-35%)	2m 36s (-65%)

Table 2.2: Performances of the 2-Level Training vs the standard high-fidelity one on varying the cardinality of the training sets.

We conclude this chapter by reporting similar results obtained by adding a third discretization level through a 25×25-elements mesh. As showed by Table 2.3, the Multi-level training strategy still performs quite well even if the highest levels of accuracy are hardly achievable. The introduction of a third learning phase fixes the training time at an almost constant (but quite high) level. These considerations make the 2-Level Training preferable for this specific problem. However, the chance to take advantage of even coarser mesh might be of key importance when dealing with more realistic three-dimensional problems for which the cost of the snapshots generation is higher.

Model	Test err	Samples N°	Tot. training T.	Data. Gen. T.
High-F.	8.61%	CC:0/C:0/H:300	4m 58s	57s
Multi-F.	8.05%	CC:600/C:75/H:75	7m 13s (+45%)	30s (-47%)
High-F.	6.19%	CC:0/C:0/H:600	5m 56s	1m 52s
Multi-F.	6.46%	CC:1200/C:75/H:75	7m 20s (+24%)	46s (-70%)
High-F.	4.51%	CC:0/C:0/H:1200	7m 50s	3m 46s
Multi-F.	4.96%	CC:2400/C:150/H:75	7m 35s (-3%)	1m 18s (-70%)

Table 2.3: Performances of the 3-Level Training vs the standard high-fidelity one on varying the cardinality of the training sets. CC denotes the snapshots for the coarsest mesh.

3 | DL-ROMs for stochastic PDEs

This chapter aims to extend the use of DL-ROMs to a larger class of problems, in particular stochastic PDEs, in which the parametrization involves a countable number of random inputs that generate a stochastic field. A classical example in this context is the Darcy problem to describe the fluid motion in a porous media, in which the ground permeability is modeled by means of a random field with a certain level of regularity, prescribed through the choice of a suitable covariance kernel. The extension is first of all theoretical and involves the proof of two Theorems regarding the DL-ROMs approximation capabilities in the aformentioned context. Finally, the result of numerical experiments are reported to confirm the validity of the theoretical estimates in different scenarios.

3.1. Theoretical aspects

The theoretical result on the approximation properties of the auto-encoder for the case of stochastic PDEs is developed starting from Theorem 1.1 in two steps. The first one allows to remove the hypothesis of compactness on the parameter space by introducing a suitable probability measure on it, and changing accordingly the norm in which the approximation property holds. The second step follows immediately through dimensionality reduction techniques, under very common technical assumptions on the covariance kernel of the random field.

Theorem 3.1. *Let $\boldsymbol{\mu} \sim \mathcal{P}_{\boldsymbol{\mu}}$ be a random variable which takes values in $\Theta \subset \mathbb{R}^p$ and such that $\mathbb{E}\left[|\boldsymbol{\mu}|^2\right]$ is finite. Moreover, let $u : \Theta \to \mathcal{V}$ be a Lipschitz-continuous function, with $(\mathcal{V}, ||\cdot||)$ Hilbert space, and let $\mathcal{S} \subset \mathcal{V}$ be the manifold obtained by mapping Θ through u. Then,*

(i) if $n \geq 2p+1$, or

(ii) if $n \geq p$ and u is injective,

it holds that

$$\inf_{\substack{\Psi^{'}\in\ \mathcal{C}(\mathcal{S},\mathbb{R}^n)\\ \Psi\in\ \mathcal{C}(\mathbb{R}^n,\mathcal{V})}} \mathbb{E}\ ||u_{\boldsymbol{\mu}} - \Psi\circ\Psi^{'}(u_{\boldsymbol{\mu}})|| = 0 \quad .$$

Proof. First of all, without loss of generality, we assume that $\mathbf{0}\in\Theta$. Then, for any $R>0$, we define the closed ball of radius R, $B_R=\{\boldsymbol{\mu}\in\Theta|\ |\boldsymbol{\mu}|\leq R\}$, so that we can rewrite the expectation as

$$\mathbb{E}\ ||u_{\boldsymbol{\mu}} - \Psi\circ\Psi^{'}(u_{\boldsymbol{\mu}})|| = \mathbb{E}\ \left[\mathbb{1}_{B_R}||u_{\boldsymbol{\mu}} - \Psi\circ\Psi^{'}(u_{\boldsymbol{\mu}})||\right] + \mathbb{E}\ \left[\mathbb{1}_{B_R^C}||u_{\boldsymbol{\mu}} - \Psi\circ\Psi^{'}(u_{\boldsymbol{\mu}})||\right].$$

Furthermore, taking advantage of the triangular inequality, we obtain

$$\begin{aligned}\mathbb{E}\ ||u_{\boldsymbol{\mu}} - \Psi\circ\Psi^{'}(u_{\boldsymbol{\mu}})|| \leq \mathbb{E}\ \left[\mathbb{1}_{B_R}||u_{\boldsymbol{\mu}} - \Psi\circ\Psi^{'}(u_{\boldsymbol{\mu}})||\right] + \mathbb{E}\ \left[\mathbb{1}_{B_R^C}||u_{\boldsymbol{\mu}}||\right] + \\ + \mathbb{E}\ \left[\mathbb{1}_{B_R^C}||\Psi\circ\Psi^{'}(u_{\boldsymbol{\mu}})||\right].\end{aligned}$$

We can bound each integral separately, using trivial inequalities and the Lipschitz-continuity of the map u:

$$\begin{aligned}\mathbb{E}\ \left[\mathbb{1}_{B_R}||u_{\boldsymbol{\mu}} - \Psi\circ\Psi^{'}(u_{\boldsymbol{\mu}})||\right] &\leq \sup_{\boldsymbol{\mu}\in B_R}||u_{\boldsymbol{\mu}} - \Psi\circ\Psi^{'}(u_{\boldsymbol{\mu}})||\mathcal{P}_{\boldsymbol{\mu}}(B_R)\\ &\leq \sup_{\boldsymbol{\mu}\in B_R}||u_{\boldsymbol{\mu}} - \Psi\circ\Psi^{'}(u_{\boldsymbol{\mu}})||\ ;\end{aligned}$$

$$\begin{aligned}\mathbb{E}\ \left[\mathbb{1}_{B_R^C}||u_{\boldsymbol{\mu}}||\right] &\leq \mathbb{E}\ \left[\mathbb{1}_{B_R^C}||u_{\boldsymbol{\mu}} - u_{\mathbf{0}}||\right] + \mathbb{E}\ \left[\mathbb{1}_{B_R^C}||u_{\mathbf{0}}||\right]\\ &\leq \mathcal{P}_{\boldsymbol{\mu}}^{1/2}(B_R^C)\ \mathbb{E}^{1/2}\left[||u_{\boldsymbol{\mu}} - u_{\mathbf{0}}||^2\right] + \mathcal{P}_{\boldsymbol{\mu}}(B_R^C)\ ||u_{\mathbf{0}}||\\ &\leq L\ \mathcal{P}_{\boldsymbol{\mu}}^{1/2}(B_R^C)\ \mathbb{E}^{1/2}\left[|\boldsymbol{\mu}|^2\right] + \mathcal{P}_{\boldsymbol{\mu}}(B_R^C)\ ||u_{\mathbf{0}}||\ ;\end{aligned}$$

$$\mathbb{E}\ \left[\mathbb{1}_{B_R^C}||\Psi\circ\Psi^{'}(u_{\boldsymbol{\mu}})||\right]\ \leq\ \mathcal{P}_{\boldsymbol{\mu}}(B_R^C)\ \sup_{\boldsymbol{\mu}\in B_R}||\Psi\circ\Psi^{'}(u_{\boldsymbol{\mu}})||\leq\ \mathcal{P}_{\boldsymbol{\mu}}(B_R^C)\ \sup_{\boldsymbol{x}\in\mathbb{R}^n}||\Psi(\boldsymbol{x})||\ ,$$

where $\mathcal{P}_{\boldsymbol{\mu}}(A)$, for a generic measurable set $A\subset\mathbb{R}^p$, is by definition the probability of the event $(\boldsymbol{\mu}\in A)$. Let us now recall that if $u: E\to\mathcal{V}$ is Lipschitz-continuous with E compact and $n\geq 2p+1$, or equivalently, $n\geq p$ and u is injective,then Theorem 1.2 applies so that

$$\inf_{\substack{\Psi^{'}\in\ \mathcal{C}(\mathcal{S},\mathbb{R}^n),\\ \Psi\in\ \mathcal{C}(\mathbb{R}^n,\mathcal{V})}}\ \sup_{\boldsymbol{\mu}\in E}||u_{\boldsymbol{\mu}} - \Psi\circ\Psi^{'}(u_{\boldsymbol{\mu}})|| = 0\ ,$$

and the infimum id attained. We then denote by $(\Psi_R,\Psi_R^{'})$ the couple of functions for

which the minimum is attained in the case $E = u(B_R)$, that is a compact set, since B_R is compact and u is Lipschitz-continuous. Therefore in our proof we can choose Ψ^* and $\Psi^{*'}$ such that

- they are equal to Ψ_R and $\Psi_R^{'}$ on $\Psi_R^{'}(u(B_R))$ and $u(B_R)$, respectively;
- they are the continuous extensions of Ψ_R and $\Psi_R^{'}$ on $u(\Theta)$ and $\mathbb{R}^n$, respectively.

The existence of such extensions is ensured by Theorem 4.1 of [6], which furthermore states that the image of the extensions is contained in the convex hull of the original function image. Thanks to this property and to the perfect reconstruction of $u_{\boldsymbol{\mu}}$ by the composition $\Psi^* \circ \Psi^{*'}$ in B_R, we can observe that

$$\Psi^*(\mathbb{R}^n) \subseteq \text{conv}\left[\Psi^* \circ \Psi^{*'}(u(B_R)))\right] \subseteq \text{conv}\left[u(B_R)\right] \subseteq B(u_{\mathbf{0}}, \text{diam}(u(B_R)))$$

where $B(u_{\mathbf{0}}, \text{diam}(u(B_R)))$ is the ball centered in $u_{\mathbf{0}}$ with radius given by $\text{diam}(u(B_R))$. Exploiting once more the Lipschitz-continuity it follows immediately that

$$\sup_{\boldsymbol{x}\in\mathbb{R}^n} ||\Psi^*(\boldsymbol{x})|| \leq ||u_{\mathbf{0}}|| + \text{diam}(u(B_R)) \leq ||u_{\mathbf{0}}|| + 2LR \ .$$

In order to conclude the proof, we now estimate $\mathcal{P}(B_R^C)$ thanks to the Markov's inequality:

$$\mathcal{P}_{\boldsymbol{\mu}}(B_R^C) = \mathcal{P}_{\boldsymbol{\mu}}(|\boldsymbol{\mu}| > R) = \mathcal{P}_{\boldsymbol{\mu}}(|\boldsymbol{\mu}|^2 > R^2) \leq \frac{\mathbb{E}\left[|\boldsymbol{\mu}|^2\right]}{R^2} \ .$$

Finally, collecting all the previous results, we can obtain

$$\begin{aligned}
&\inf_{\substack{\Psi^{'}\in\mathcal{C}(\mathcal{S},\mathbb{R}^n),\\ \Psi\in\mathcal{C}(\mathbb{R}^n,\mathcal{V})}} \mathbb{E}||u_{\boldsymbol{\mu}} - \Psi \circ \Psi^{'}(u_{\boldsymbol{\mu}})|| \leq \mathbb{E}||u_{\boldsymbol{\mu}} - \Psi^* \circ \Psi^{*'}(u_{\boldsymbol{\mu}})|| \\
&\quad \leq \sup_{\boldsymbol{\mu}\in B_R} ||u_{\boldsymbol{\mu}} - \Psi^* \circ \Psi^{*'}(u_{\boldsymbol{\mu}})|| + L\ \mathcal{P}_{\boldsymbol{\mu}}^{1/2}(B_R^C)\ \ \mathbb{E}^{1/2}\left[|\boldsymbol{\mu}|^2\right] + \mathcal{P}_{\boldsymbol{\mu}}(B_R^C)\ ||u_{\mathbf{0}}|| + \\
&\qquad\qquad + \mathcal{P}(B_R^C) \sup_{\boldsymbol{x}\in\mathbb{R}^n} ||\Psi^*(\boldsymbol{x})|| \leq \frac{\mathbb{E}\left[|\boldsymbol{\mu}|^2\right]}{R}\left(\frac{2||u_{\mathbf{0}}||}{R} + 3L\right)
\end{aligned}$$

where the last term can be made arbitrarily small with a proper choice of R. □

Remark 1. The hypothesis of global Lipschitz-continuity on the map $\boldsymbol{\mu} \mapsto u_{\boldsymbol{\mu}}$ is certainly strong. It is possible to weaken it by assuming instead that:

(i) $\forall R > 0, \qquad \sup_{|\boldsymbol{\mu}^1|,|\boldsymbol{\mu}^2|\leq R} \dfrac{||u_{\boldsymbol{\mu}^1} - u_{\boldsymbol{\mu}^2}||}{|\boldsymbol{\mu}^1 - \boldsymbol{\mu}^2|} < \infty \ ;$

(ii) $\lim_{R\to+\infty} \mathcal{P}_{\boldsymbol{\mu}}(B_R^C) \sup_{|\boldsymbol{\mu}|\leq R} ||u_{\boldsymbol{\mu}}|| = 0$.

These two hypotheses may be more easily satisfied. For example, let us consider the case where $u_{\boldsymbol{\mu}}$ is the solution of an elliptic problem in a bounded Lipschitz domain, with a permeability field described in term of the linear combination of n random parameters:

$$\begin{cases} \nabla \cdot (\sigma_{\boldsymbol{\mu}} \nabla u) = f \text{ in } D, \\ u = 0 \qquad\qquad\quad \text{on } \partial D \end{cases}$$

where $\sigma_{\boldsymbol{\mu}} = \exp\{\sum_{i=1}^N b_i(x)\mu_i\}$, $\{b_i(x)\}_{i=1}^N$ are uniformly bounded in D by γ and $\{\mu_i\}_{i=1}^N$ independent random variable with standard Gaussian distribution. If the source belongs to the dual space H^{-1} and $\boldsymbol{\mu} \in B_R$ (so that the bilinear form of the associated weak formulation is coercive), then we can ensure the existence of a unique weak solution $u_{\boldsymbol{\mu}} \in (H_0^1, ||\cdot||)$, by the Lax-Milgram theorem [31]. Furthermore, the following estimate holds

$$||u_{\boldsymbol{\mu}}|| \leq \frac{1}{\min_{x\in D} \sigma_{\boldsymbol{\mu}}} ||f||_*.$$

Thanks to the latter, it is now possible to prove that the problem satisfies the second assumption. First of all, by the monotonicity of the exponential function we have

$$\sup_{|\boldsymbol{\mu}|\leq R} ||u_{\boldsymbol{\mu}}|| \leq \exp\left\{ \max_{x\in D} \sup_{\boldsymbol{\mu}\leq R} \sum_{i=1}^N b_i(x)\mu_i \right\} ||f||_* \leq \exp\{\gamma N R\} ||f||_*.$$

Furthermore, because of the assumption on the distribution of the random parameters, we have that $|\boldsymbol{\mu}|^2 \sim \mathcal{X}^2(N)$. Then, denoted by $F_{|\boldsymbol{\mu}|^2}(x; N)$ its cumulative distribution function, we can exploit the Chernoff bound for the case $x > N$ (condition that is clearly satisfied since since R is arbitrarily large) in order to obtain

$$\mathcal{P}_{\boldsymbol{\mu}}(B_R^C) = \mathcal{P}_{\boldsymbol{\mu}}(|\boldsymbol{\mu}| > R) = \mathcal{P}_{\boldsymbol{\mu}}(|\boldsymbol{\mu}|^2 > R^2) = 1 - F_{|\boldsymbol{\mu}|^2}(R^2; N) \leq \left(\frac{R^2}{N}\right)^{N/2} \exp\left\{1 - \frac{R^2}{N}\right\}.$$

Finally, we can observe that

$$\mathcal{P}_{\boldsymbol{\mu}}(B_R^C) \sup_{|\boldsymbol{\mu}|\leq R} ||u_{\boldsymbol{\mu}}|| \leq \left(\frac{R^2}{N}\right)^{N/2} \exp\left\{1 - \frac{R^2}{N} + \gamma N R\right\} \to 0 \text{ if } R \to \infty.$$

For what concerns the first hypothesis, under the condition of uniqueness and existence of the solution $u_{\boldsymbol{\mu}}$ for every $\boldsymbol{\mu} \in B_R$ and because of the smoothness of the map $\boldsymbol{\mu} \mapsto \sigma_{\boldsymbol{\mu}}$, we can apply Lemma C.2 of [7] to ensure the Lipschiz-continuity of the map $\boldsymbol{\mu} \mapsto u_{\boldsymbol{\mu}}$ in B_R for any $R > 0$.

Theorem 3.2. *Let $\mu : D \times \Omega \to \mathbb{R}$ be a mean square integrable random field, where D is a compact subset of $\mathbb{R}^d$ and Ω is the sample space. Let $Cov_\mu : D \times D \to \mathbb{R}$ be its symmetric, non-negative definite and continuous covariance kernel. Moreover let $\{\lambda_i\}_{i=1}^{\infty}$ be the countable non increasing sequence of eigenvalues associated to the Karhunen-Loeve (KL) expansion of μ and let $\mu^{(p)}$ be the random field obtained by truncating the expansion at order p. Let Θ be the space containing all the possible realizations of the random field. Finally let $u : \Theta \to \mathcal{V}(D)$ be a Lipschiz-continuous function (with Lipschiz constant L), with $\mathcal{V}$ and $\mathcal{S}$ defined as in the previous theorem. Then, if $n \geq 2p+1$ or $n \geq p$ but u is injective, it holds*

$$\inf_{\substack{\Psi' \in \mathcal{C}(\mathcal{S},\mathbb{R}^n),\\ \Psi \in \mathcal{C}(\mathbb{R}^n,\mathcal{V})}} \mathbb{E}\, ||u_\mu - \Psi \circ \Psi'(u_{\mu^{(p)}})|| \leq L \sum_{i=p+1}^{\infty} \sqrt{\lambda_i}. \tag{3.1}$$

Proof. Thanks to the hypothesis made on the covariance kernel of the random field, Mercer's theorem [11] applies and we are able to express it in terms of its KL expansion, namely

$$\mu(x,\omega) = \mathbb{E}\left[\mu\right](x) + \sum_{i=1}^{\infty} \sqrt{\lambda_i}\; b_i(x)\; \mu_i(\omega)$$

where $x \in D$, $\omega \in \Omega$, $\{b_i\}_{i=1}^{\infty}$ are the orthonormal eigenfunctions of the covariance kernel belonging to $L^2(D)$ and $\{\mu_i\}_{i=1}^{\infty}$ are uncorrelated random variables with zero mean and unit variance. Then the result follows directly from the application of the triangular inequality and of Theorem 3.1, indeed

$$\begin{aligned}\inf_{\substack{\Psi' \in \mathcal{C}(\mathcal{S},\mathbb{R}^n),\\ \Psi \in \mathcal{C}(\mathbb{R}^n,\mathcal{V})}} & \mathbb{E}\, ||u_\mu - \Psi \circ \Psi'(u_{\mu^{(p)}})|| \\ & \leq\ \mathbb{E}\, ||u_\mu - u_{\mu^{(p)}}|| + \inf_{\substack{\Psi' \in \mathcal{C}(\mathcal{S},\mathbb{R}^n),\\ \Psi \in \mathcal{C}(\mathbb{R}^n,\mathcal{V})}} \mathbb{E}\, ||u_{\mu^{(p)}} - \Psi \circ \Psi'(u_{\mu^{(p)}})||.\end{aligned}$$

For what concerns the first term, we can exploit the Lipschitz-continuity of the map u, moreover, thanks to the Cauchy-Schwarz inequality and to the properties of the KL expansion terms, we obtain

$$\begin{aligned}\mathbb{E}\, ||u_\mu - u_{\mu^{(p)}}|| \leq L\, \mathbb{E}\ \ ||\mu - \mu^{(p)}|| &\leq L\, \mathbb{E}\, || \sum_{i=p+1}^{\infty} \sqrt{\lambda_i}\; b_i(x)\; \mu_i(\omega)|| \\ &\leq L \sum_{i=p+1}^{\infty} \sqrt{\lambda_i}\; \mathbb{E}\ \left[|\mu_i|\right]\ ||b_i|| \leq L \sum_{i=p+1}^{\infty} \sqrt{\lambda_i}\; \mathbb{E}^{1/2}\left[|\mu_i|^2\right] \leq L \sum_{i=p+1}^{\infty} \sqrt{\lambda_i}.\end{aligned}$$

Regarding the second term, we can introduce the $p-dimensional$ vector of the random coefficients in the KL expansion $\boldsymbol{V}_\mu^{(p)}$ and the linear function $\phi : \boldsymbol{V}_\mu^{(p)} \mapsto \mu^{(p)}$ which maps the random vector to the associated truncated random field, so that $u_{\mu^{(p)}} = (u \circ \phi)_{\boldsymbol{V}_\mu^{(p)}}$. Then, the composition of u with ϕ is in turn Lipschitz-continuous and, as a consequence, we are under the hypothesis of Theorem 3.1. It follows that, if $n \geq 2p+1$, or equivalently $n \geq p$ and u is injective, we have

$$\inf_{\substack{\Psi^{'} \in \mathcal{C}(\mathcal{S}, \mathbb{R}^n) \\ \Psi \in \mathcal{C}(\mathbb{R}^n, \mathcal{V})}} \mathbb{E}|| \, (u \circ \phi)_{\boldsymbol{V}_\mu^{(p)}} - \Psi \circ \Psi^{'}((u \circ \phi)_{\boldsymbol{V}_\mu^{(p)}})|| = 0.$$

□

Remark 2. The effectiveness of the result above is due to the fast decay of the eigenvalues, which depends on the smoothness of the covariance kernel and on the correlation length of the random field. Nevertheless, for a standard choice like the Gaussian one, the exponential decay leads to a very good approximation with just a small order of truncation and consequently a small minimal latent dimension. For theoretical results applicable to the kind of problems this work deals with, with compact and multidimensional domain we can refer to [16]. Furthermore, estimate (3.1) highlights an advantage in modeling efficiently the *parameter-to-solution* map using DL-ROMs, instead of linear projections methods relying on POD. With these latter indeed, the error depends on the decay of the spectrum elements of the covariance kernel push-forward measure. This decay rate may be much slower, as demonstrated in Lemma 3.15 of [18] for the case of a Gaussian measure on the parameter space, even if the map $\mu \to u_\mu$ is Lipschitz-continuous. On the contrary, for the cases in which the decay of the manifold eigenvalues is faster, we are not harmed by the use of DL-ROMs. It is indeed sufficient to seek the encoder-decoder couple in a linear space to recover a POD-like error estimate.

3.2. Numerical experiments

3.2.1. Problem definition

In order to empirically validate the previous results, we now present the outcome of a numerical analysis obtained by using the DL-ROM approach on a stochastic Poisson equation, namely an elliptic PDE parameterized by a random field, equipped with Robin's

boundary conditions and solved on the spatial domain $D = (0,1)^2$:

$$\begin{cases} -\nabla \cdot (e^{\mu} \nabla u) = 1 & \text{in } D \\ \nabla u = u & \text{on } \partial D \end{cases} \tag{3.2}$$

Here μ denotes a centered Gaussian random field, meaning that all its finite distributions are Gaussian i.e for any $\boldsymbol{x}_1, ..., \boldsymbol{x}_n \in D$, the random vector $\boldsymbol{V}_\mu = (\mu(\boldsymbol{x}_1), ..., \mu(\boldsymbol{x}_n)$ has a multivariate Gaussian distribution. For what concernes the choice of the kernel, the same experiment has been repeated for a Gaussian (or squared exponential) kernel

$$\mathrm{Cov}_\mu(||x-y||) = \exp\left(-\frac{||x-y||^2}{2l^2}\right),$$

which generates an analytic almost surely field (with two different choices for the correlation length l) and for an exponential kernel

$$\mathrm{Cov}_\mu(||x-y||) = \exp\left(-\frac{||x-y||}{l}\right),$$

for which the field is α-Holder continuous almost surely, with $\alpha < 1/2$. Both kernels clearly satisfy the hypothesis of Mercer's theorem. Moreover, their isotropy makes the PDE easily solvable and the solution manifold learning relatively feasible: these are desirable conditions when running a significant number of tests. Nevertheless, this choice allow to confirm the theoretical result for different decay rates of the eigenvalues. For the purpose of showing the predicted trends, we used five order of truncation: p=5,10,20,40,100. The last one approximates the whole, not truncated, field playing the role of the benchmark.

The snapshots are generated using as high fidelity FOM the Galerkin F-E Method, with first order polynomial basis functions, on a structured triangular mesh with $N_h = 10201$ degrees of freedom. Some instances of the snapshots for the Gaussian kernel case, with the associated random field realization, are depicted in Figure 3.1, whereas the decay of the eigenvalues of its K-L decomposition is reported in Figure 3.2.

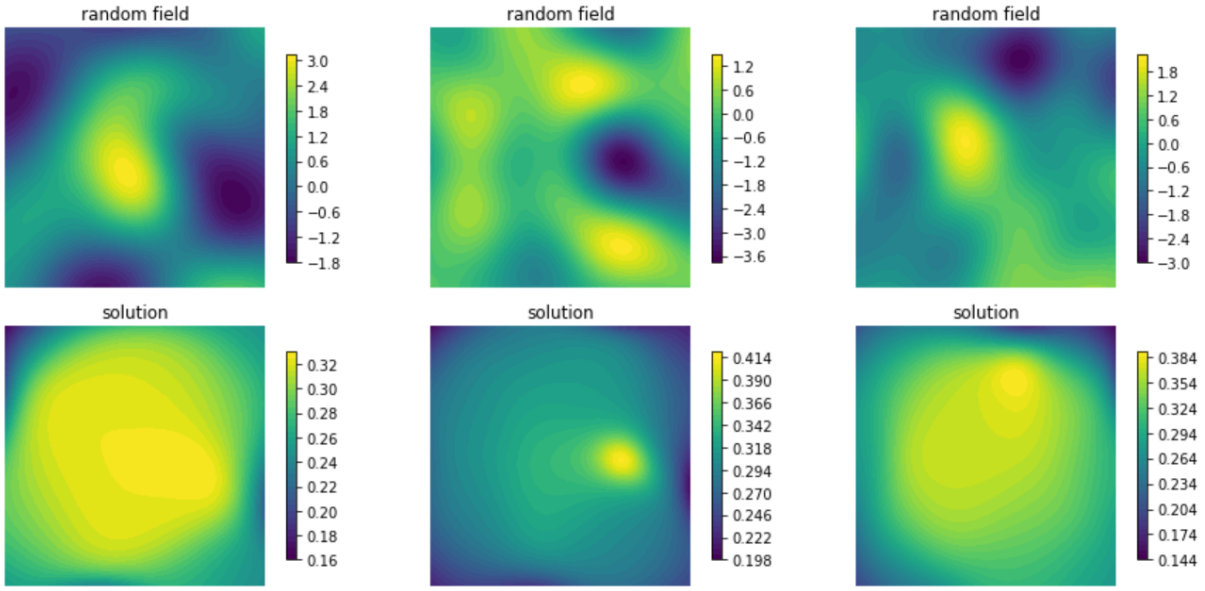


Figure 3.1: Some instances of the random field and associated PDE solution for the case of Gaussian kernel with $l^2 = 1/10$. The smoothing effect of the diffusion problem is clearly visible.

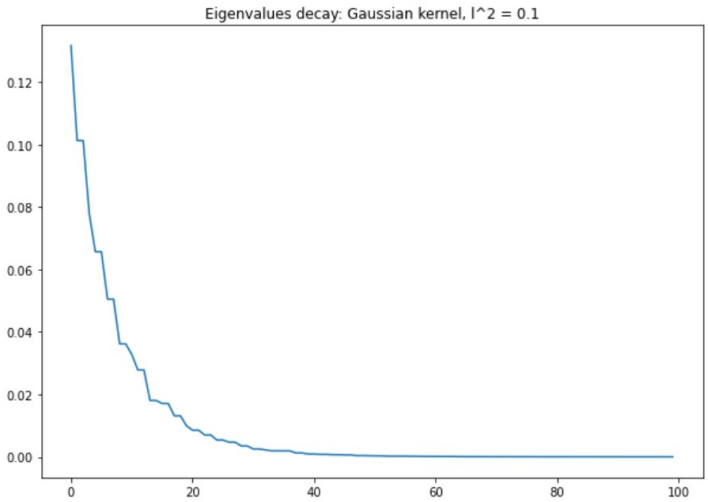


Figure 3.2: If the kernel is highly regular, as for the Gaussian case, or the correlation length is high, the decay of the eigenvalues in its K-L decomposition is fast and the approximation problem can be solved in a particularly efficient way through the use of DL-ROM, according to Theorem 3.2

.

3.2.2. Experimental design

Despite the fact that Theorem 3.1, combined with classical NN approximation theorems, ensures the existence of an auto-encoder that can learn the solution manifold at any level of accuracy, when dealing with numerical experiments the presence of the test error must be taken into account. This is due to the high dimensionality and non-convexity of the error surface (training error) and to the finite amount of data used in the NN training (generalization gap), as already mentioned in chapter 1. Consequently, in order to make the results of the analysis more robust, we considered three groups of AEs, with different levels of accuracy. The latter was determined by choosing different values of a structural parameter m, which fixes the number of channels of the convolutional layers. This is indeed an efficient way to increase the AE expressiveness. Furthermore, four auto-encoders fall in each group, one for every truncation order, $p = 5, 10, 20, 40$, of the random field. They only differ in the latent dimension, which was chosen depending on p, in such way to satisfy the hypothesis of Theorem 3.2, that is $n = 2p + 1$.
The parametric architecture is described in Table 3.1, without considering the NN ϕ, since the analysis concerns the auto-encoding process only.

	Layer	Input	Output	Kernel	Stride	Dof
Ψ'	Dense	10201	n	-	-	$10201(n+1)$
Ψ	Dense	n	$144m$	-	-	$144(m+1)n$
	Convol.	$6 \times 6 \times 4m$	$20 \times 20 \times 2m$	10	2	$800m^2 + 4m$
	Convol.	$20 \times 20 \times 2m$	$47 \times 47 \times m$	9	2	$162m^2 + 2m$
	Convol.	$47 \times 47 \times m$	101×101	9	2	$81m + 1$

Table 3.1: The parametric architecture used for experiments related to Problem (3.2). The parameter m fixes the number of channels of the convolutional layer, whereas n is the latent dimension. All the layers have the 0.1-leaky ReLu activation function, except the last one, which is equipped with the linear one.

In order not to introduce a bias in the error decay estimate, the test error must be homogeneous within each group. The simplest way to satisfy this requirement is to modify properly the training duration: the auto-encoders with greater latent dimension need more epochs to reach the same level of accuracy on their respective test set. This is due to the higher complexity of the solution manifold that has to be learned and to the higher number of dof that increases the number of optimization steps that are needed. More precisely, the experiments showed that it was sufficient to double the number of epochs from one

order to the following, passing from 50 to 400 epochs. Figure 3.3 shows the mean relative error of every auto-encoder (for the case of Gaussian kernel with $l^2 = 1/10$) by indicating with different color the belonging to a certain group. The horizontal lines indicate the mean error associated to the group and the low level of dispersion around them guarantees the aforementioned homogeneity. The training of every NN was independently repeated ten times and the results were averaged, with the aim to reduce the influence of the NN parameters initialization on the analysis.

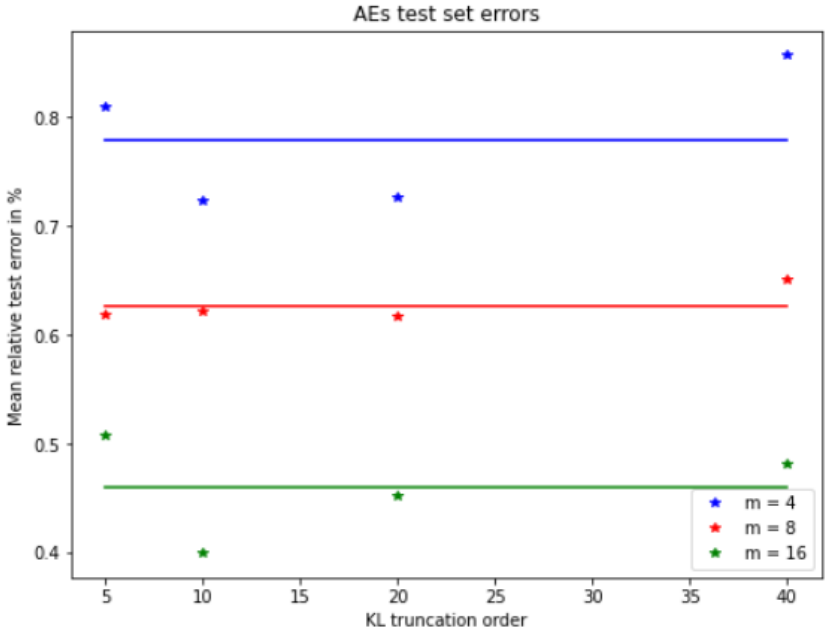
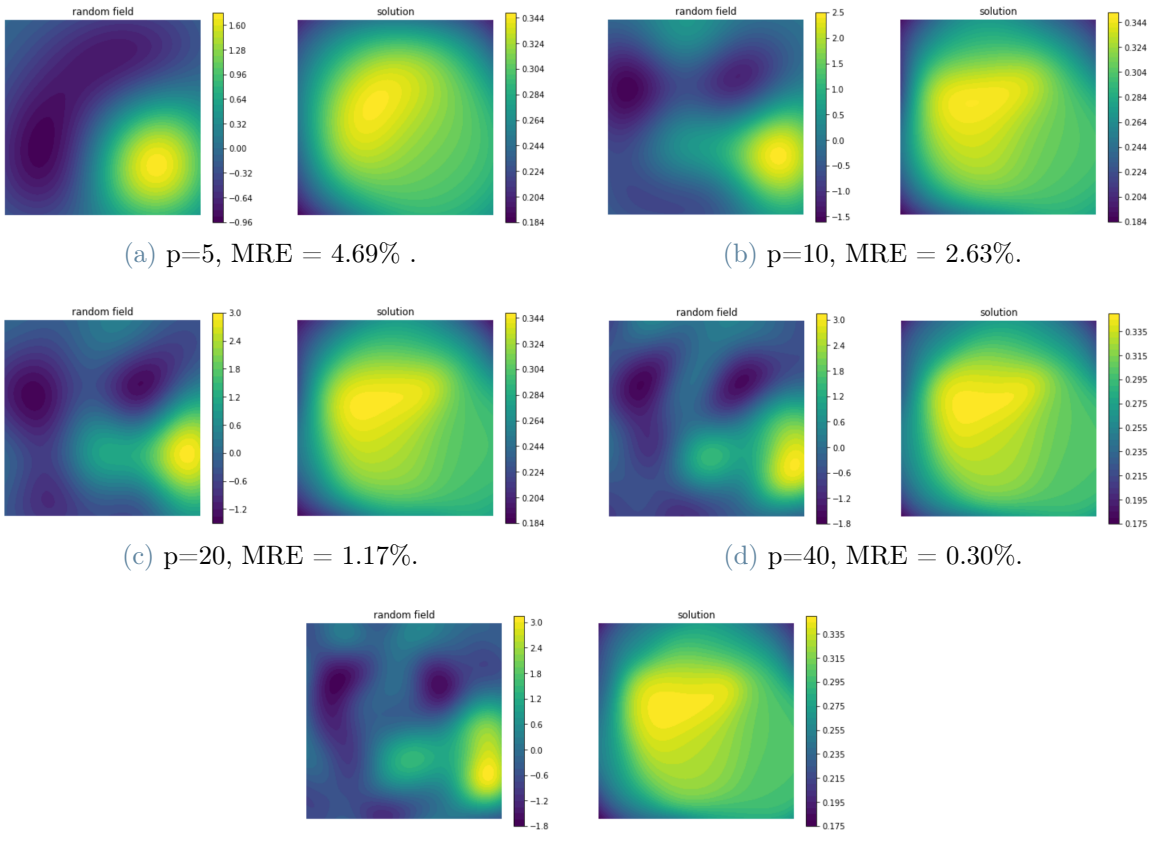


Figure 3.3: Mean relative error on the test set of the 12 AEs involved in the experiments, subdivided in 3 different groups, according to their number of channels. Within every group there are 4 AEs, one for every random field truncation order. They share a similar accuracy in order not to introduce a bias in the experiment.

In this part of the work we are not interested in analysing the computational cost of the training phases; however, it varied approximately from one up to ten minutes, depending both on the number of epochs and on the total number of degrees of freedom of the auto-encoder, and so on the parameter m.

A crucial issue in designing the experiment was the construction and the subdivision of the data sets implied in the training and in the evaluation of the auto-encoders. This task was accomplished through the following steps:

- the first one hundred eigenvalues and eigenfunctions of the covariance kernel were numerically determined;
- 10^4 one hundred-dimensional vectors of random coefficients were sampled from the Normal distribution, since we chose to work with a Gaussian random field;
- depending on the desired order of truncation of the random field, only the first p K-L bases were linearly combined to generate the random field;
- the latter was then passed to the FE solver and the resulting PDE solution stored according to the truncation order;
- the data sets were subdivided reserving 1000 solutions instances for the training, 5000 for the test and 4000 for comparing the performances of different auto-encoders on new samples.



(a) p=5, MRE = 4.69% .

(b) p=10, MRE = 2.63%.

(c) p=20, MRE = 1.17%.

(d) p=40, MRE = 0.30%.

(e) p=100.

Figure 3.4: A particular instance of the Gaussian field (with squared exponential kernel and $l^2 = 1/10$) and associated numerical PDE solutions for various orders of truncation. The mean relative error between the truncated solution and the benchmark one (e) is besides reported.

In this manner, we can have perfect matching among the various datasets for auto-encoders with different latent dimensions: the solutions sharing the same index in the two datasets are generated from the same instance of the diffusion field, which is just truncated at different orders. This allows us to run the experiments without introducing a bias due to a particular choice of the solution manifold points used to evaluate the expected value in Theorem 3.2. Figure 3.4 shows an instance of the random field and of the associated solution for different order or truncation, together with the mean relative error between the "truncated" solution and the benchmark one, with $p = 100$.

3.2.3. Results

Figure 3.5 summarizes the results obtained through the aforementioned experiment for the case of Gaussian kernel and correlation length $l = 1/\sqrt{10}$. The plotted data (coloured circles) show the performances of the three groups of auto-encoders in terms of the mean L^2 error between the benchmark solution and the auto-encoded "truncated" one, namely:

$$MLE = \mathbb{E}\ ||u_\mu - \Psi_{NN} \circ \Psi'_{NN}(u_{\mu^{(p)}})||.$$

The black dots correspond to the values of the function

$$f(p) = L \sum_{i=p+1}^{\infty} \sqrt{\lambda_i}$$

with $L = 1/100$ and $p = 5, 10, 20, 40$. The regression lines and the associated decay rates are obtained assuming an exponential trend $\propto e^{\alpha p}$ (which is guaranteed by the theory, for this case).

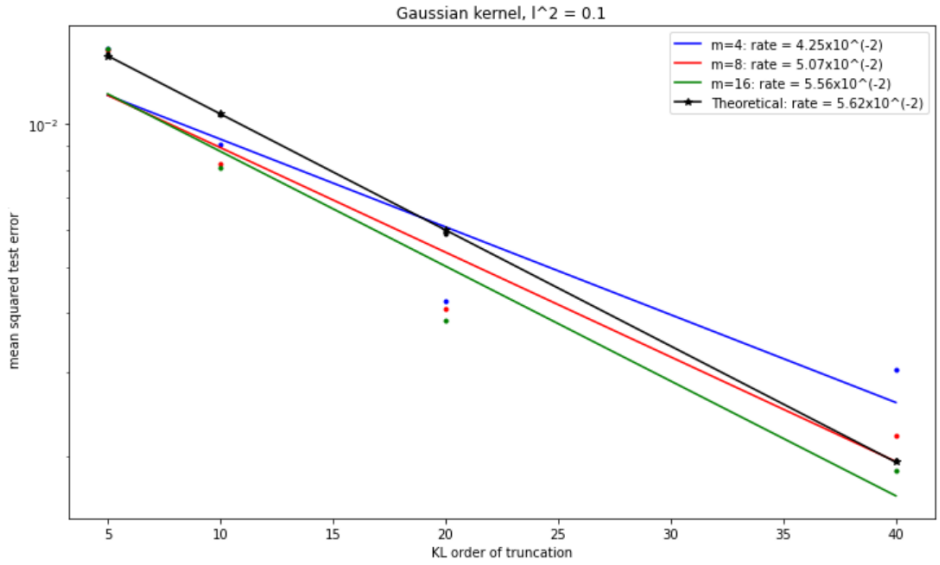


Figure 3.5: The results of the experiment, showing the validity of the theoretical estimate when improving the AEs expressiveness.

As we can clearly see, the error decay rate increases with the parameter m, finally obtaining for $m = 16$ an almost perfect match with the theoretical estimate. This is due to the fact that the approximation by means of NN of the encoder-decoder couple, whose existence is ensured in 3.1, generates a not negligible error, as described in Section 1.1.1. By increasing m and improving the overall accuracy of the approximation, we were then able to recover the predicted decay.

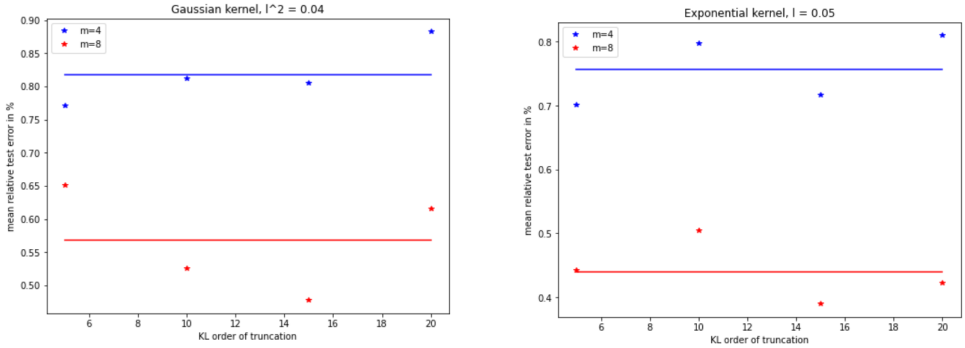


Figure 3.6: Mean test error of the AEs used in these experiments, divided in two groups according to their accuracy. The absence of a trend appearing while varying the order of truncation ensures the fairness of the experiment.

The same experiment was carried out also in two other scenarios: for a Gaussian kernel with a different correlation length $l = 1/5$ and for an Exponential kernel. For these

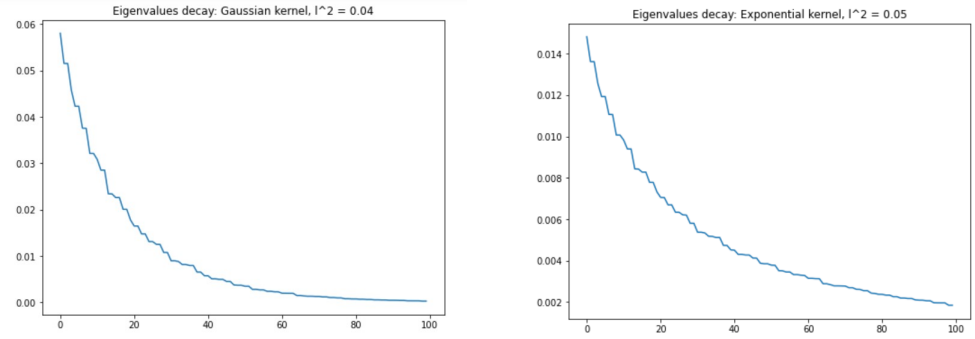
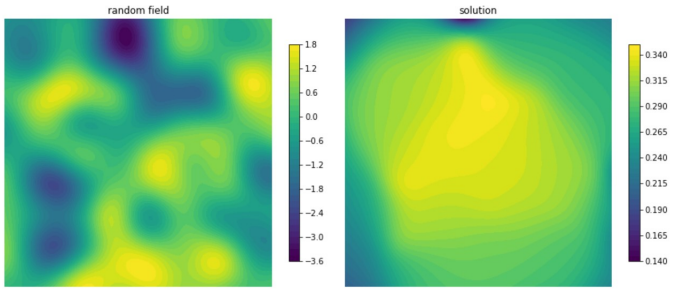
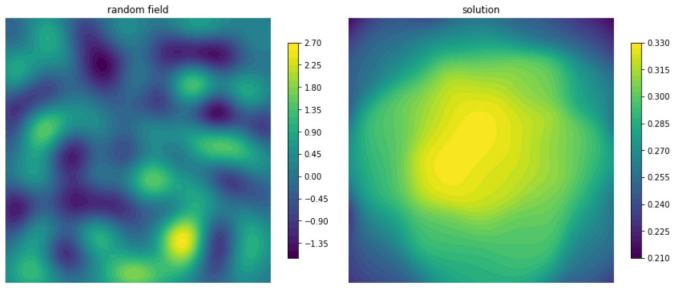


Figure 3.7: Eigenvalue decay for the other kernels

cases only two levels of accuracy were taken into account so that eight auto-encoders were involved in the experiment. The homogeneity of their test error is again confirmed through the plots in Figure 3.6. As shown by the graphics in Figure 3.7, both the decreasing in correlation length of the random field and of the level of regularity of the kernel slow down the eigenvalue decay in the K-L decomposition. As a consequence, the number of modes playing an important role in the K-L decomposition increases and the random field instances have a more complex shape. This can be verified through a comparison between the random fields in Figure 3.1 and in Figure 3.8.



(a) Gaussian kernel, $l = 1/5$



(b) Exponential kernel

Figure 3.8: Random field and solution instance for the other kernels

Finally, Figure 3.9 shows how the theoretical estimate is confirmed in both cases already for $m = 4$, despite having a level of accuracy comparable to the one of the previous experiment. This fact might be due to the intrinsic regularity that the smoothing effect of the Laplace operator confers to the solution manifold. The latter is apparent from the solutions in Figure 3.8 and from the rapid decay of the eigenvalues associated to the solution manifold, for which we refer to [7]. The exploitation of the manifold regularity during the learning process is indeed one of the main advantages in using a reduced order model based on auto-encoders, as already mentioned in Section 1.2.

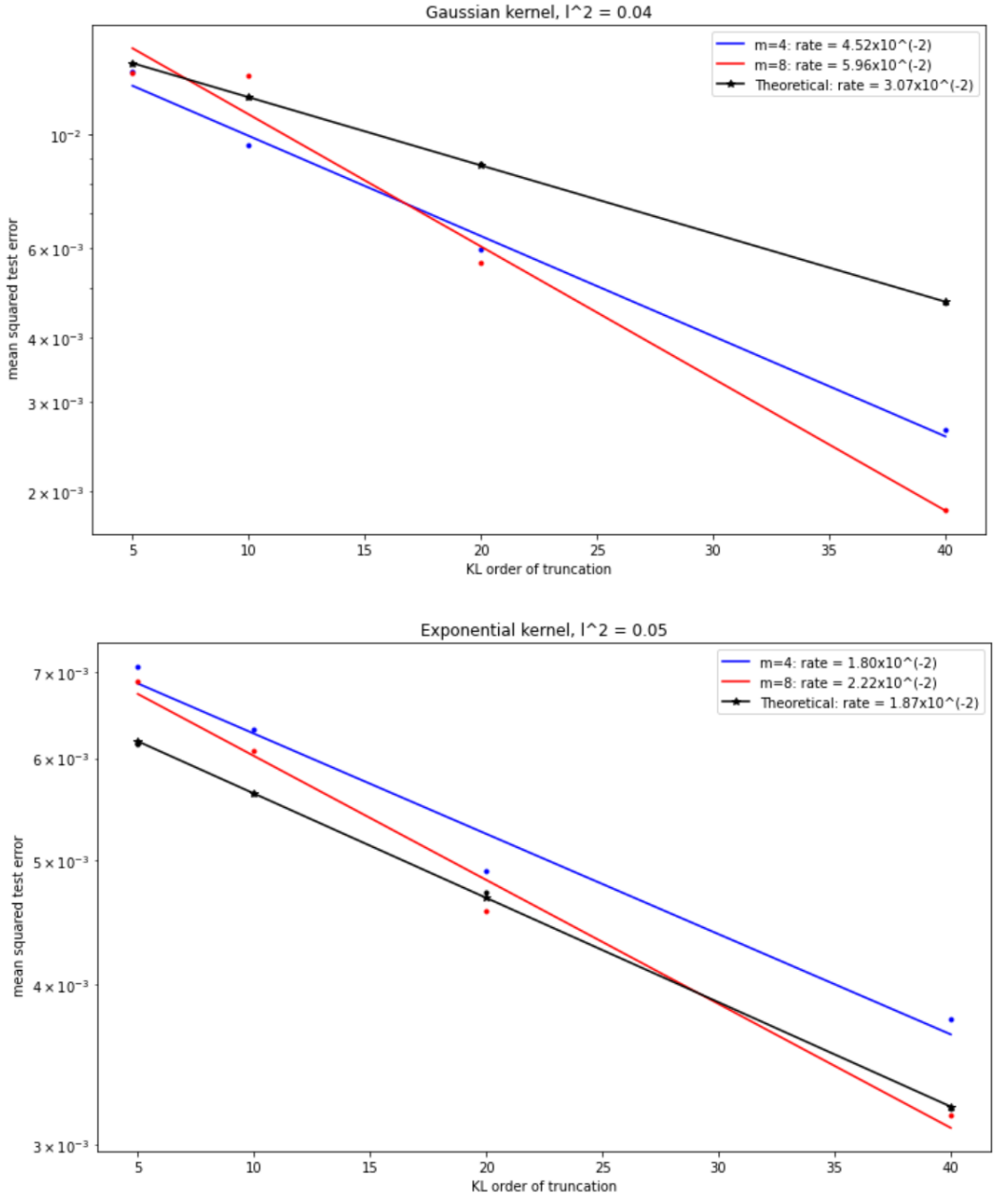


Figure 3.9: Error decay for the other kernels

4 | Transfer learning

In this chapter we present a Transfer learning strategy involving DL-ROMs, specifically for the case of PDEs parametrized by random fields. The goal consists in designing a DL-ROM able to inherit the information contained in another one, already trained for a simpler problem in terms of random field modes involved and, as a consequence, with a smaller latent dimension. We start this chapter by explaining why this idea might be successful and beneficial also from a computational perspective. Then, we introduce the original technical tool developed for its realization, namely the *hybrid* dense layers. Finally, after a section devoted to the experimental tuning of the hyper-parameters characterizing the proposed strategy, we report the results obtained on Problem (3.2) with a Gaussian kernel.

4.1. Idea development

While carrying out the experiments in the previous chapter, we wondered how an already trained auto-encoder would respond when receiving as input the PDE solution corresponding to a random field which is more complex with respect to those considered during training. For a fixed order of truncation, it is possible to interpret the additional modes of the random field as noise that propagates through the *parameter-to-solution* map. In this perspective, it appears reasonable that the auto-encoder performs poorly or, in the best case scenario, that it filters the noise out without exploiting the additional information provided. Figure 4.1 shows that this is not what happens. The picture displays a comparison between the mean relative error committed by the 12 AEs (subject of the previous experiments) in approximating the solutions for the full random field when the input is the solution associated respectively to the full (circle) or truncated (triangle) random field. The error is always smaller in the first case, independently of the NN expressiveness, and the gap reduces gradually with the growth of the truncation order. This means that the auto-encoder is able to process richer solutions than the ones it is considered during training and it can take advantage of the additional details. The same occurred for each of the covariance kernels tested.

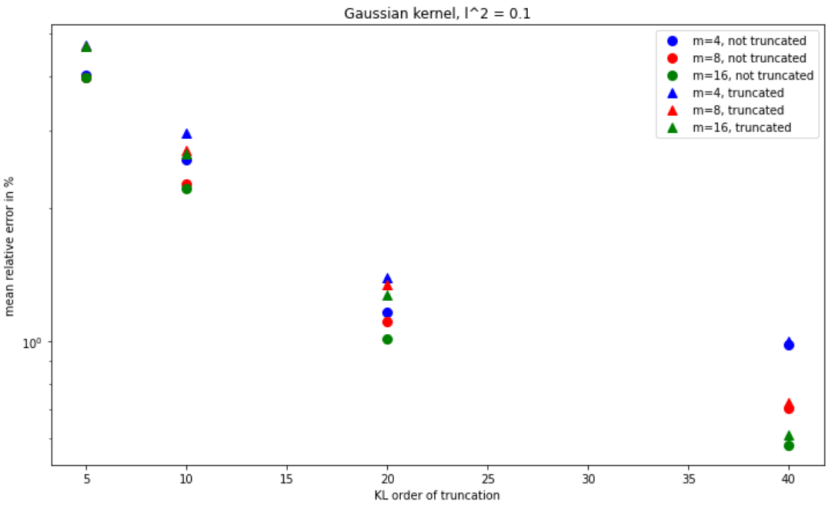


Figure 4.1: Auto-encoders errors in processing the truncated and the whole solution manifold.

This result suggests that a DL-ROM implemented and trained for the manifold associated to a certain truncation order, may be used as a basis for a richer DL-ROM, built for a more complex approximation problem, in terms of involved modes. The purpose of this chapter is then to understand if it is possible to reach an arbitrarily low approximation error starting from a fixed internal representation of the solution manifold and by increasing, *a posteriori*, the latent dimension. The term "fixed" here highlights the fact that the value of weights and biases of the smaller auto-encoder remain *frozen*, therefore do not serve solely as an educated initial guess where to start a new training.* A graphical representation of the proposed Transfer learning strategy is depicted in Figure 4.2.

There would be also a practical impact of this great flexibility: the training of a very large NN could indeed be divided in more steps. This would allow to deal with only a portion of the degrees of freedom at each step, making the optimization feasible even when fixing all the NN parameters at once would be impossible, because of the lack of computational resources.

*The aforementioned strategy was initially tested but it did not show any efficacy: the initial guess was actually quite accurate (initial MRE $\approx 5\%$ passing from truncation order 5 to 10), but the value of the NN parameters was not preserved at all after the re-train. In particular it changed only the 25% less than with the usual initialization strategy. This is not sufficient to claim that the internal representation obtained by training the smaller auto-encoder is maintained. Moreover, there was no gain in terms of computational efficiency: L-BFGS, despite being a second order optimizer, did not show the necessity of a good initial guess in order to easily converge.

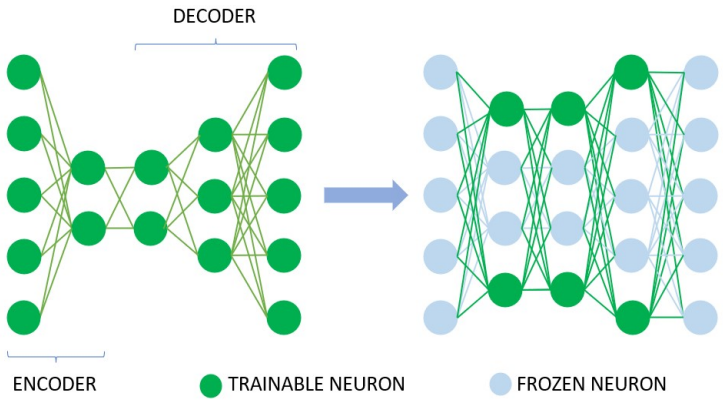


Figure 4.2: The process of freezing an already trained auto-encoder and growing it, by adding new degrees of freedom and in particular by increasing the latent dimension, before learning a more complex solution manifold.

4.2. Hybrid layers

Practically speaking, a possible strategy could consist in copying the values of weights and biases of an already trained auto-encoder in another one, with a greater latent dimension and eventually more degrees of freedom also in other layers. These parameters are then *frozen*, meaning that they cannot be optimized during further training with a decisive impact on the cost of each epoch. The remaining part of the NN is finally initialized and trained. The starting point was the creation of a Hybrid dense layer, with a frozen part and another one with a normal functioning, without any loss in terms of efficiency during the training. Since the available implementations only allow to freeze the whole layer, the Hybrid layer is built as a collection of three of them: one is the starting point layer, which is frozen; the second one generates the new outputs; the third one, with no biases, contains the weights to deal with the new inputs. In order to ensure that the different layers work together to produce the desired outcome, great attention should be paid in the implementation of the *forward* function: this maps the inputs onto the outputs and, as a consequence, is involved in the back-propagation. Figure 4.3 provides a sketch of how a Hybrid dense layer is structured, showing the effect of increasing the number of inputs and outputs of a layer on the weight matrix and on the bias vector.

It is not clear how to extend the proposed strategy also to convolutional layers, both from a theoretical and a technical point of view. For this reason, the remaining part of the work was developed using NNs which are composed uniquely of dense layers. This did not have a major impact on our experiments, mainly because of the problems and meshes considered, except for a slight increase in the gap error and in the computational time.

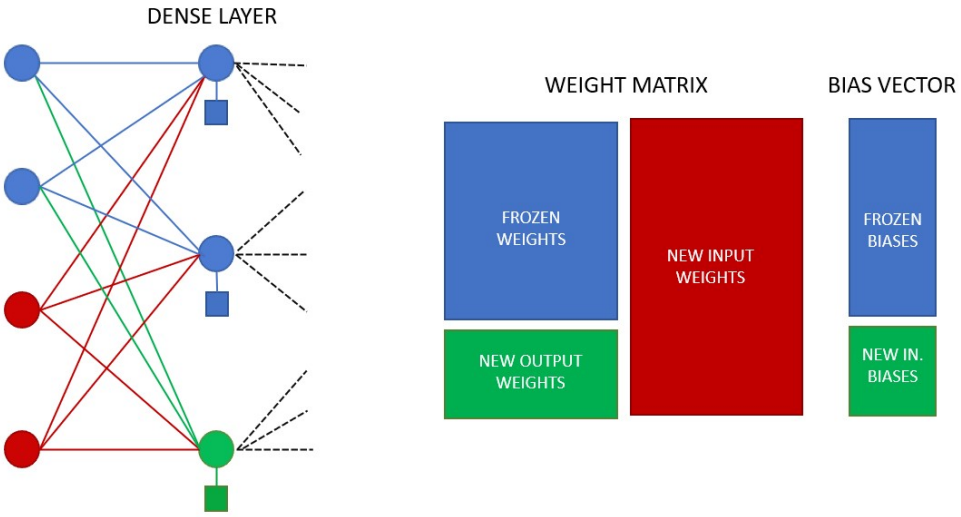


Figure 4.3: Diagram of a Hybrid layer, relating its architecture to the weight matrix and the bias vector.

Both issues are due to the larger amount of dofs characterizing dense layer with respect to sparse ones. The parametric architecture used in this chapter is reported in Table 4.1: the total number of dofs increased by an order of magnitude, even if $1 \leq k \leq 4$.

	Layer	Input	Output	Dofs
Ψ'	Dense	10201	n	$10201(n+1)$
Ψ	Dense	n	$100k$	$100(k+1)n$
	Dense	$100k$	$200k$	$2 \times 10^4 k(k+1)$
	Dense	$200k$	10201	$\approx 2 \times 10^6 k$

Table 4.1: Parametric NN architecture used for the experiment on Transfer learning.

4.3. Setting choices

Before analyzing the performances of the proposed learning strategy, we must clarify the role of some design choices, like the distribution of the additional degrees of freedom, and the importance of some hyper-parameters, such as the choice of a scaling factor for the weight initialization. An answer to all these questions can be given through numerical experiments, comparing the NN error trends during the training phase. In order to make the results more robust with respect to the weight random initialization, the error lines depicted in this chapter are the results of the average between five different trainings.

In addition to the training error and the test error, it might be useful to look at some performance indicator which not depend on the specific order of truncation of the random field. For this reason, we define the *true error* as

$$E_{true} = \mathbb{E}\ ||u_\mu - \Psi_{NN} \circ \Psi'_{NN}(u_\mu)||$$

which measures the accuracy in the reconstruction of the benchmark solution manifold. A note on terminology: in the following, the original DL-ROM with a latent space of dimension $2p+1$ will be denoted as "Base-p", whereas the one which inherits the information and it is then retrained will be referred to as "Hybrid-p".

4.3.1. Architecture

The first practical issue consists in understanding how to augment the dofs from the Base-p_1 to a Hybrid-p_2 DL-ROM, with $p_2 \geq p_1$. A partial answer is given through Figure 4.4, depicting the re-training of three different Hybrid-10 DL-ROMs with $k = 1, 2, 3$, starting from the same Base-5 with $k = 1$. The latter was trained for 200 epochs on 200 snapshots, generated with a truncation order 5 on the KL expansion of the input random field. The Hybrid DL-ROMs were then trained with the correspondent snapshots in the dataset with truncation order equal to 10.

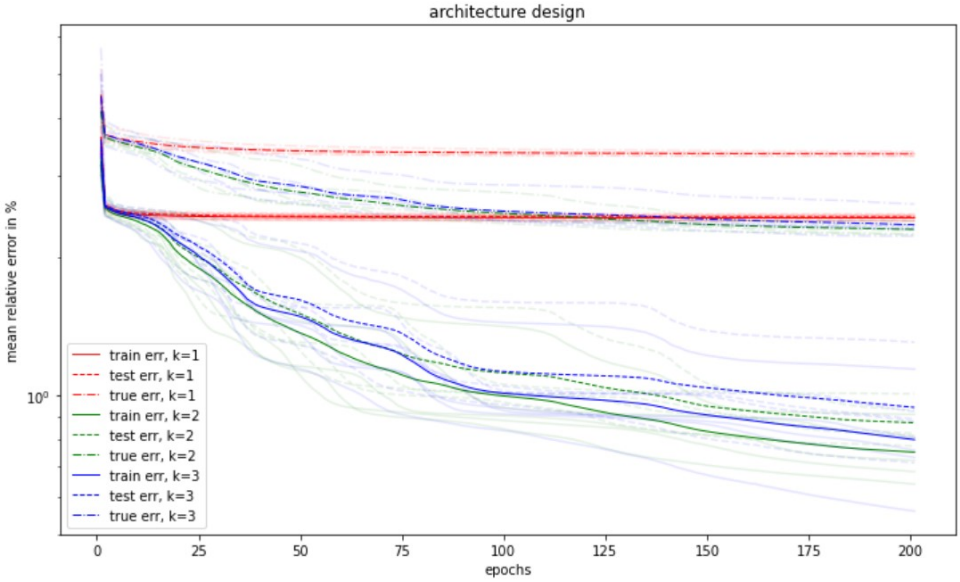


Figure 4.4: Error monitoring of three different Hybrid-10 DL-ROMs, with an increasing number of additional dofs, starting from the same Base-5 DL-ROM. The error lines are obtained by averaging between 5 instances, reported in transparency.

As we can clearly see from the flat error lines, corresponding to the case $k = 1$, it is not sufficient to increase only the latent dimension in order to recover the full NN expressiveness: the other layers must also be given some additional dofs. On the other hand, also adding too many degrees of freedom does not guarantee any advantage. Besides requiring higher computational resources, the Hybrid-10 DL-ROM with $k = 3$ performs worse in all the accuracy indicators, with respect to the one with $k = 2$. If a greater generalization gap was expected (this is related to the NN complexity, see Section 1.1.1), a lower training error is instead surprising and supportive of the chapter driving idea: the Hybrid DL-ROM is able to enhance its internal representation of the solution manifold with further details, starting from a fixed simpler structure and adding minor corrections thanks to a limited number of additional dofs.

4.3.2. Weight initialization

The second issue is related to the initialization of the additional degrees of freedom. At the extremes of the solution range there are:

- the usual He initialization: this is the most obvious choice, since it guarantees the correct exploration of the configuration space and the training stability.
- the initialization at 0 of the weights value: this is optimal from the point of view of the initial guess (since on the contrary the internal manifold representation is compromised by random noise) but impracticable because the weights value would remain at zero, being their sensitivities null.

In order to find the balance point between these two alternatives we looked for the optimal order of magnitude of a scaling factor ϵ, by which the weights, after the He initialization, are multiplied. The goal was achieved by running an experiment similar to the previous one; average results are depicted in Figure 4.5.

The three types of error are not highly sensitive to the value of ϵ, in contrast to the initial condition. Anyway, the existence of the aforementioned trade-off seems to confirmed: among the three, the intermediate value $\epsilon = 10^{-1}$ outperforms the other choices and was chosen for the following of the work.

4.3.3. Choice of the training set

The last design choice regards the opportunity, during the training of the Hybrid DL-ROM, of using a new dataset. In particular, we wonder if it is beneficial to use solutions

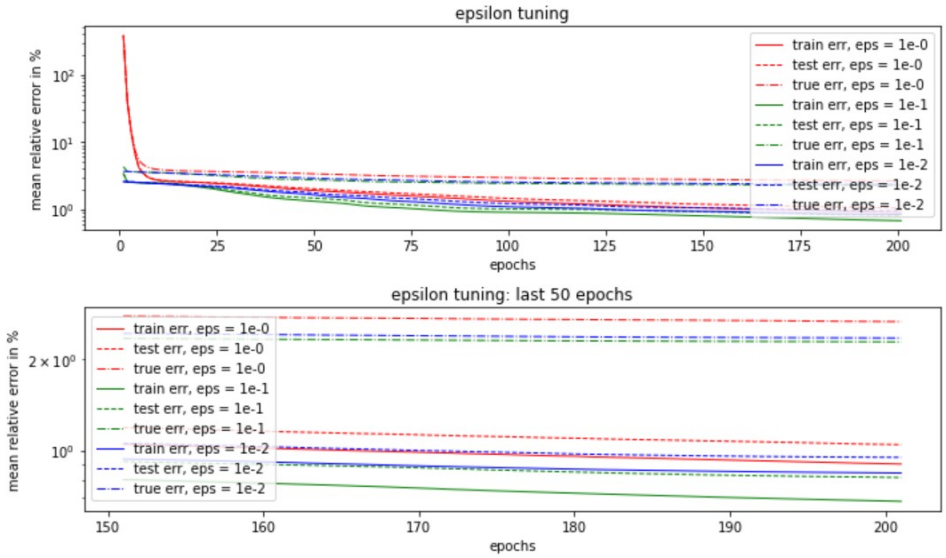


Figure 4.5: Error monitoring of three Hybrid-10 DL-ROMs, with different scaling of the weights initial value, starting from the same Base-5 DL-ROM.

associated to other instances of the random field in place of the same ones moreover characterized by a higher order of truncation. In Figure 4.6, obtained by training a Base-10 DL-ROM with $k = 2$ whose dofs are then inherited by a Hybrid-20 DL-ROM with $k = 4$, emerges a trend similar to the one already encountered when dealing with the Multi-level training in Chapter 2. Indeed, the use of the same solution instances allows to reach a lower training error: the Hybrid DL-ROM is able to take full advantage of the inherited information on the solution manifold. On the contrary, changing the dataset improves the performances in terms of generalization gap.

What the best strategy in term of test and true error is, it might depend on the problem, however, we can draw a general conclusion: if the Transfer learning process has to be repeated many times, changing the dataset every time is a successful strategy to maximize the accuracy.

4.4. Final results

The first goal of the numerical experiments is to verify whether this strategy allows to reach an arbitrarily high accuracy level. The answer is given by comparing the errors committed by the auto-encoders built and trained in the standard way. Figure 4.7 collects the results in three different scenarios: from a Base-5 DL-ROM to a Hybrid-10 DL-ROM, from a Base-10 DL-ROM to a Hybrid-20 DL-ROM and from a Base-20 DL-ROM to a Hybrid-40 DL-ROM. Concerning the architectures, independently of the fact that the

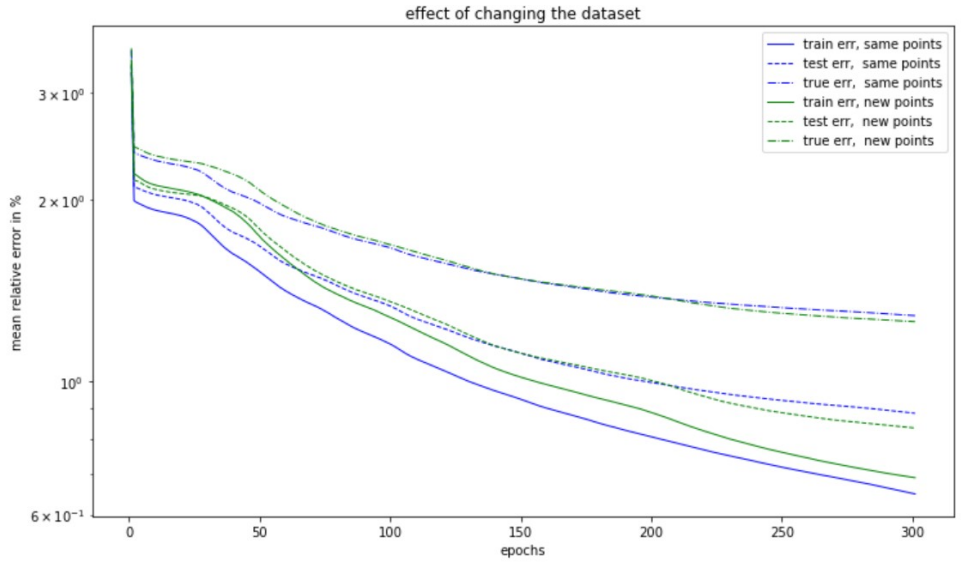


Figure 4.6: Error monitoring of two Hybrid-20, trained with the same solution instances or other ones, starting from the same Base-10.

auto-encoder was of Hybrid or Base type, we fixed $k = 1/2/3/4$ for $p = 5/10/20/40$.[†] Finally, the dimension of the training set was fixed at 200 samples and the number of epochs was varied between 200 and 800, depending on the complexity of the solution manifold. The results show that the Transfer learning strategy outperforms the standard training in terms of training error. This guarantees lower values also for test and true errors even though the generalization gap is higher. Note that it would be sufficient to increase the dimension of the training set to reach a level of accuracy that was unavailable with the standard training, especially when high orders of truncation are involved.

[†]This choice guarantees the fairness of the comparison, since it optimizes the Base auto-encoders performances, according to various experiment in which smaller and bigger architecture were considered, once all the other hyper-parameter were fixed. The Hybrid performances might be even better for another choice of k.

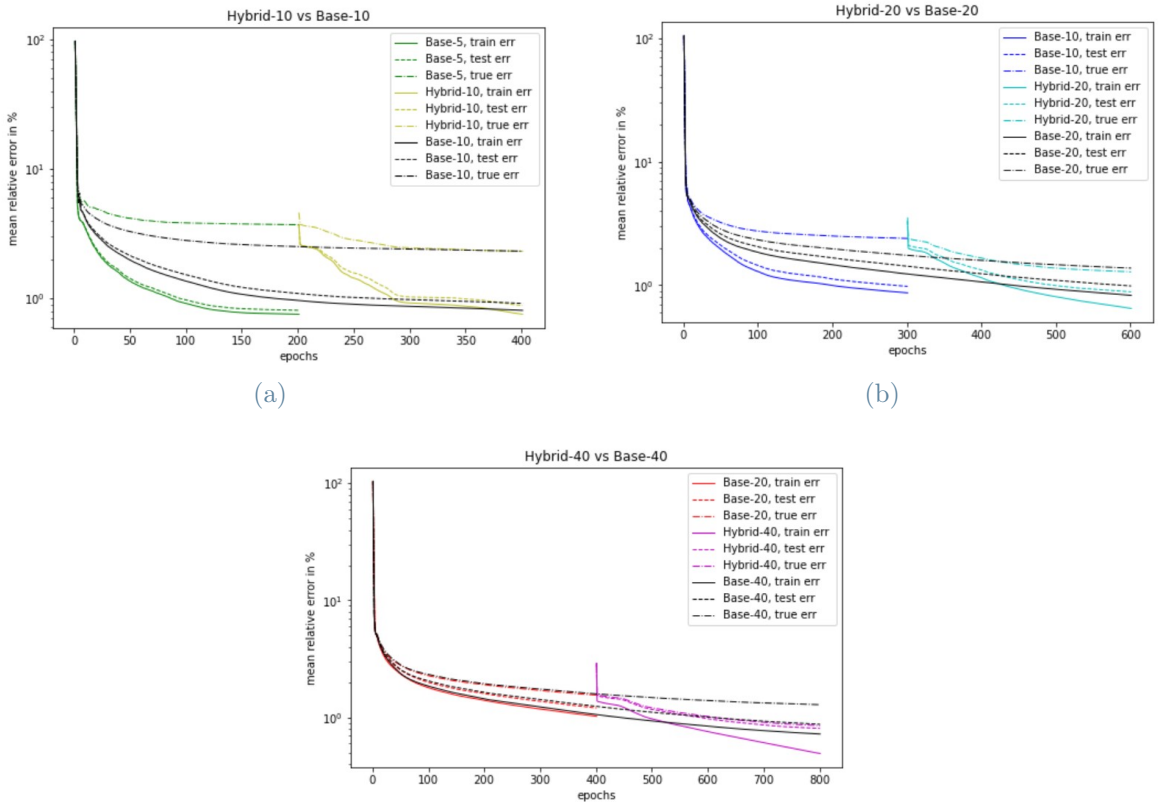


Figure 4.7: Comparison between the mean error path obtained through the standard training and the one obtained through the Transfer learning strategy. The latter outperforms independently of the random field truncation order.

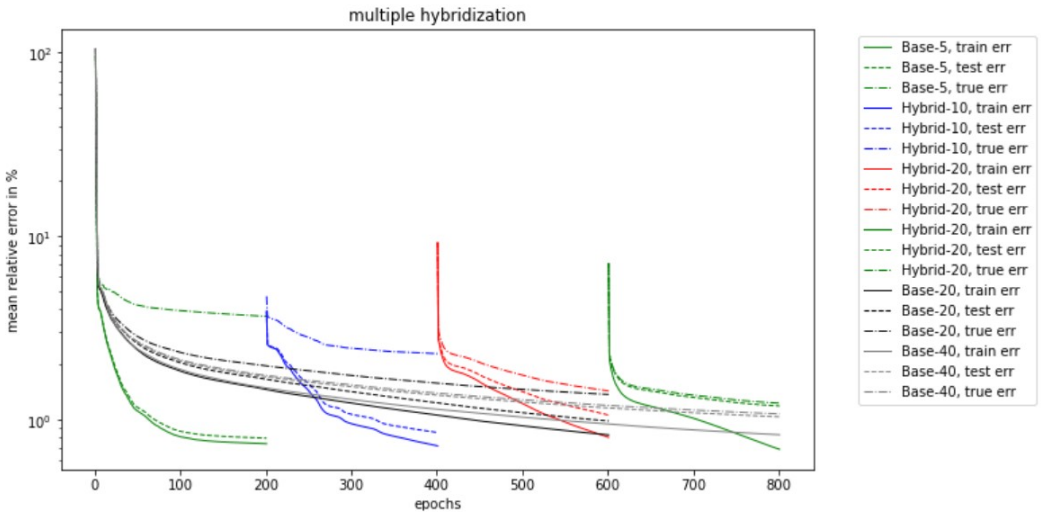


Figure 4.8: Error lines obtained exploiting for three times the Transfer learning procedure, using the Hybrid DL-ROMs as Base DL-ROMs.

Secondly, we wonder if the Transfer learning strategy can be used more than once, namely if we can use a Hybrid DL-ROM as a Base DL-ROM. Figure 4.8 shows the error lines obtained starting from a Base-5 DL-ROM, training a Hybrid-10 DL-ROM and then using it as a base for a Hybrid-20 DL-ROM, and finally exploiting the latter in turn for a Hybrid-40 DL-ROM. The architecture and training specifications are the same used before. The comparison with the Base-20 DL-ROM and the Base-40 DL-ROM shows that a further subdivision of the training is possible and guarantees adequate performances.

Finally, we analysed the advantage in terms of computational burden of the preoposed strategy. As already mentioned, the possibility to split the train in two steps allows to lower the requirements in terms of memory storage: this has a great advantage since we are dealing with millions of dofs and we are using a second-order optimizer that exploits also their values in the previous epochs. Beyond this, we wonder whether there is a gain also in terms of training times. As reported in Table 4.2, splitting the training in two stages allows to save approximately 30% of time, simultaneously increasing the accuracy. The use of further intermediate Hybrid DL-ROMs guarantees even higher improvements, keeping constant the time per epoch, entailing just a slight worsening of the test error.

AE type	Test err	Epochs	Time	T/e	Gain
B5+H10	0.75%	200×2	3m 15s	0.49 s/e	30%
B10	0.91%	400	4m 39s	0.70 s/e	
B10+H20	0.88%	300×2	6m 29s	0.65 s/e	29%
B5+B10+H20	1.06%	200×3	5m 5s	0.50 s/e	45%
B20	0.98%	600	9m 12s	0.92 s/e	
B20+H40	0.80%	400×2	10m 2s	0.76 s/e	34%
B5+H10+H20+H40	1.19%	200×4	7m 2s	0.52 s/e	56%
B40	0.87%	800	15m 45s	1.18 s/e	

Table 4.2: Computational time saved using the Transfer learning strategy with two or more hybridations.

It might be argued that the need of generating new synthetic data for the simplified version of the random field might cancel off any computational gain. This issue can be easily overcome since we can use the training set associated to the richer solution manifold also for the training of the Base DL-ROM, without any impact on the final Hybrid performances. This fact is proved by the results in Figure 4.9. Here, we compare

the error lines obtained by training a Base-5 DL-ROM on a 5-truncation order dataset with the ones associated to the same architecture trained on a 10-truncation order dataset. Furthermore, we also analyse the behaviour of the Hybrid-10 DL-ROM, built upon those Base-5 DL-ROMs.

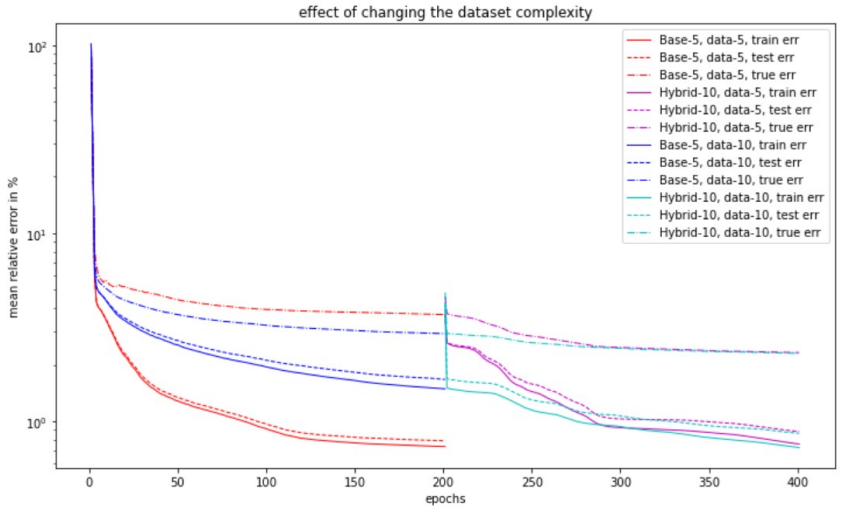


Figure 4.9: Error monitoring of two Base-5 DL-ROMs (and of the Hybrid-10 DL-ROMs built upon them), trained respectively with a dataset generated truncating at order 5 of the random field and directly with the one generated for order 10.

As expected, test and train errors are lower using the 5-dataset, for which the Base-5 DL-ROM is specifically designed, whereas the true error is higher, since clearly the 10-dataset gives more information about the manifold associated to the whole random field. These clear differences in the various performance indicators disappear when observing the Hybrid DL-ROMs during the last 200 epochs.

In conclusion, the presented results show that transferring information from a DL-ROM to another is not only possible without undermining the expressiveness of the latter, but it might be used as a strategy when dealing with complex random fields. Besides reducing the requirements in terms of memory, the Transfer learning strategy could be used to lower drastically the time implied by the training or even to reach higher accuracy, depending on the number of implemented steps and on the availability of synthetic data.

5 | An Inverse UQ application

This Chapter is devoted to the development of an original method for the solution of Bayesian inverse problems when the parametrization involves complex random fields. More specifically, we propose a revisited version of the Metropolis algorithm (MA) for the exploration of the posterior probability distribution, which we build entirely upon the use of Hybrid DL-ROMs introduced in the previous Chapter. Before passing to the actual implementation and to the presentation of the experimental results, we devoted some sections to introducing the concept of inverse problem, its solution within the Bayesian framework and the standard version of the Metropolis algorithms, highlighting its main issues.

5.1. The solution of a Bayesian inverse problem

5.1.1. Definition of the inverse problem

In order to formalize the concept of inverse problem, we start by introducing the forward problem and the associated terminology, limiting ourselves to the contest of elliptic PDEs. Given a generic system, whose state is compliant to a parametric PDE, the forward problem consists in predicting the value of a certain Quantity of Interest (QoI) associated to the state of the system for a particular value of the parameters. These latter may play a role in the definition of the differential operator, or influence other components of the problem, such as the source, the boundary conditions or the domain geometry. However, in this work we only deal with the former case. The QoI is defined as a functional (often linear and continuous) of the solution, but may as well explicitly depend on the parameters. As always, we denote by Θ the parameters space, by $\boldsymbol{\mu}$ a particular parameter instance, by $u_{\boldsymbol{\mu}}$ the corresponding state of the system, which is modeled as the solution to a PDE defined over a domain D. Furthermore, we denote by $\mathcal{L}$ a generic differential operator, by f the source and by q the QoI. Then, we can write the generic forward

problem as follows:

$$\begin{cases} \mathcal{L}(u_{\boldsymbol{\mu}}; \boldsymbol{\mu}) = f \text{ in } D \\ +\text{BCs} \\ q(\boldsymbol{\mu}) = \mathcal{Q}(u_{\boldsymbol{\mu}}; \boldsymbol{\mu}) \end{cases}$$

where BCs stands for *boundary conditions*. We work under the assumption that for all $\boldsymbol{\mu} \in \Theta$ the forward problem is well-posed. Typical examples of quantities of interest are the solution itself $q(\boldsymbol{\mu}) = u_{\boldsymbol{\mu}}$, its mean over the domain $q(\boldsymbol{\mu}) = \int_D u_{\boldsymbol{\mu}}(\boldsymbol{x})d\boldsymbol{x}/|D|$ and, above all, a collection of solution values at selected points of the domain $\boldsymbol{q}(\boldsymbol{\mu}) = [u_{\boldsymbol{\mu}}(\boldsymbol{x_1}), ..., u_{\boldsymbol{\mu}}(\boldsymbol{x_n})]$. The solution of the forward problem requires, generally speaking, the evaluation of the parameter-to-solution map and the successive computation of the associated QoI. The inverse problem, instead, consists in determining the value of the parameter $\boldsymbol{\mu}$ given a measurement q^* of the QoI. This kind of problems commonly suffers of an intrinsic ill-posedness caused by the fact that the parameter-to-QoI map is non-injective. For this reason, deterministic approaches, that seek the parameter values by minimizing the discrepancy between predicted and observed QoI, always need regularization strategies. The latter consists in including some prior information about the parameter in a rather "artificial" way, by introducing further constraints. A much more natural approach, that allows to exploit the prior information and to take into account experimental errors in the QoI measurement, within a rigorous theoretical framework, is the Bayesian one.

5.1.2. The Bayesian framework

Compared to the frequentist perspective, within the Bayesian paradigm parameters are considered to be random variables with an associated probability density. The key idea is that our prior knowledge, which is fundamental for the inverse problem well-posedness, is condensed into a *prior* distribution $\pi_0(\boldsymbol{\mu})$. The prior distribution is then updated on the basis of observed values of the QoI q^*, which we here assume to be a vector. The final result of this operation, which is actually the solution of the inverse problem according to the Bayesian paradigm, is the *posterior* distribution $\pi(\boldsymbol{\mu}|\boldsymbol{q}^*)$. Accordingly to Bayes' formula, the latter can be expressed as

$$\pi(\boldsymbol{\mu}|\boldsymbol{q}^*) = \frac{\pi(\boldsymbol{q}^*|\boldsymbol{\mu})\pi_0(\boldsymbol{\mu})}{\pi_q(\boldsymbol{q}^*)}. \tag{5.1}$$

As already mentioned, $\pi_0(\boldsymbol{\mu})$ reflects our prior knowledge on the parameters, which may come from the existing literature, other models or physical constraints. If none of this sources is available also uninformative uniform prior may be used. $\pi(\boldsymbol{q}^*|\boldsymbol{\mu})$ is the so

called *likelihood* function, that quantifies the probability of observing $\boldsymbol{q}^*$ for a specific instance of the parameters. In the context of Bayesian inverse problems the likelihood is determined by modelling the measurement error. The latter is assumed to take into account both the experimental noise and the discrepancy between the real phenomenon and the mathematical representation through the PDE. Since this Chapter has the first aim to develop a new strategy, we adopt the simplest possible error model (which is, nevertheless, a quite widespread choice in the literature), namely the additive Gaussian model. With this choice, the measurement of a (vectorial) QoI can be expressed as

$$\boldsymbol{q}^* = \boldsymbol{\mathcal{Q}}(u_{\boldsymbol{\mu}}; \boldsymbol{\mu}) + \boldsymbol{\varepsilon} = \boldsymbol{q}(\boldsymbol{\mu}) + \boldsymbol{\varepsilon} \tag{5.2}$$

where $\boldsymbol{\varepsilon}$ is random Gaussian vector with $\mathbf{0}$ mean and a (symmetric and positive definite) covariance matrix C. Usually, one assumes that the errors affect the components of the QoI independently, and C is chosen to be diagonal. Let $\pi_{\boldsymbol{\varepsilon}}$ be the corresponding probability density function: by taking advantage of the error model in (5.2), we can express the likelihood as

$$\pi(\boldsymbol{q}^*|\boldsymbol{\mu}) = \pi_{\boldsymbol{\varepsilon}}(\boldsymbol{q}^* - \boldsymbol{q}(\boldsymbol{\mu})) \ , \tag{5.3}$$

so that $\boldsymbol{q}^*|\boldsymbol{\mu} \sim \mathcal{N}(\mathbf{0}, C)$ for the Gaussian case.
By representing the marginal densisty $\pi_{\boldsymbol{q}}(\boldsymbol{q}^*)$ as the integral over possible joint densities and using again the Bayes formula, we obtain

$$\pi_{\boldsymbol{q}}(\boldsymbol{q}^*) = \int_\Theta \pi(\boldsymbol{\mu}, \boldsymbol{q}^*) d\boldsymbol{\mu} = \int_\Theta \pi(\boldsymbol{q}^*|\boldsymbol{\mu}) \pi_0(\boldsymbol{\mu}) d\boldsymbol{\mu} \ .$$

so that we are finally able, exploiting the error model, to rewrite expression (5.1) in terms of all known probability densities

$$\pi(\boldsymbol{\mu}|\boldsymbol{q}^*) = \frac{\pi(\boldsymbol{q}^*|\boldsymbol{\mu})\pi_0(\boldsymbol{\mu})}{\int_\Theta \pi(\boldsymbol{q}^*|\boldsymbol{\mu})\pi_0(\boldsymbol{\mu}) d\boldsymbol{\mu}} = \frac{\pi_{\boldsymbol{\varepsilon}}(\boldsymbol{q}^* - \boldsymbol{q}(\boldsymbol{\mu}))\pi_0(\boldsymbol{\mu})}{\int_\Theta \pi_{\boldsymbol{\varepsilon}}(\boldsymbol{q}^* - \boldsymbol{q}(\boldsymbol{\mu}))\pi_0(\boldsymbol{\mu}) d\boldsymbol{\mu}}$$

However, the latter formula does not generally allow to determine the posterior density in a closed form. Indeed, the normalizing integral at the denominator can be analytically computed only under very specific hypothesis (e.g. prior and likelihood both Gaussian and QoI linear in the parameters) and numerically integrated with classical quadrature rule only when the number of parameters is limited. Most of the applications regard instead large-scale Bayesian inversion problems and high-dimensional posterior distribution, that are hardly visualizable. As a consequence, it is often preferable to investigate the posterior

by estimating some quantities related to its distribution, such as the *conditional mean*

$$\boldsymbol{\mu}_{CM} = \int_{\Theta} \boldsymbol{\mu}\, \pi(\boldsymbol{\mu}|\boldsymbol{q}^*) d\boldsymbol{\mu} \tag{5.4}$$

or the *conditional covariance*

$$C_C = Cov(\boldsymbol{\mu}|\boldsymbol{q}^*) = \int_{\Theta} (\boldsymbol{\mu} - \boldsymbol{\mu}_{CM}) \otimes (\boldsymbol{\mu} - \boldsymbol{\mu}_{CM})\, \pi(\boldsymbol{\mu}|\boldsymbol{q}^*) d\boldsymbol{\mu}. \tag{5.5}$$

In order to estimate these latter, it is not necessary to have a complete knowledge of the posterior distribution, but just to be able to sample from it. This is why we devote the next subsection to the Metropolis algorithm, a simulation technique that allows to explore the posterior while avoiding the evaluation of the marginal density. For further details on the Bayesian framework for the solution of inverse problems we refer to [13].

5.1.3. Metropolis algorithm

The Metropolis algorithm (MA) belongs to a large class of methods, the Monte Carlo Markov Chain (MCMC) methods, all based on the same common idea: given a target distribution they aim to generate a Markov chain (an homogeneous and discrete stochastic process, in which the probability of being in a certain state at step t depends only on the state at step $t-1$) whose ergodic distribution is the target distribution [27]. In our case, the target distribution is the posterior, so that the sequence of random samples generated by the method $(\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_n, ...)$ approaches the posterior when as n becomes sufficiently large, regardless of the initial guess and under minimal assumption on the target distribution. We chose the Metropolis algorithm because it is a very simple example of MCMC, and thus a good starting point for our generalization to stochastic PDEs. The main idea of the MA is to generate a random walk in which every step is sampled from a *proposal* symmetrical distribution. The latter might be accepted or not according to a simple rule, based on what we know about the target distribution. The detailed procedure can be found in Algorithm 5.1 The key point is that the target distribution only appears in the form of a ratio within the algorithm. As a consequence, we do not need anymore to determine the normalizing constant at the denominator of our posterior target, indeed

$$\frac{\pi_T(\boldsymbol{\mu}_\#)}{\pi_T(\boldsymbol{\mu}_{i-1})} = \frac{\pi(\boldsymbol{\mu}_\#|\boldsymbol{q}^*)}{\pi(\boldsymbol{\mu}_{i-1}|\boldsymbol{q}^*)} = \frac{\pi(\boldsymbol{q}^*|\boldsymbol{\mu}_\#)\pi_0(\boldsymbol{\mu}_\#)}{\pi(\boldsymbol{q}^*|\boldsymbol{\mu}_{i-1})\pi_0(\boldsymbol{\mu}_{i-1})}\ .$$

It is easy to observe that the new candidate $\boldsymbol{\mu}_\#$ is always accepted if the chain is moving towards a region in the parameter space where the density is higher. The fact that it

might be accepted anyway (through the *coin tossing* in line 7) makes the Markov chain reversible and, thus, ergodic.

Algorithm 5.1 Metropolis algorithm

1: INPUT: the target π_T, a proposal σ, a initial guess $\boldsymbol{\mu}_0$, the chain length N
2: **for** $i = 1 : N$ **do**
3: Draw a step $\Delta\boldsymbol{\mu} \sim \sigma$
4: Define a new candidate $\boldsymbol{\mu}_\# = \boldsymbol{\mu}_{i-1} + \Delta\boldsymbol{\mu}$
5: Compute $\alpha = \min\left(1, \pi_T(\boldsymbol{\mu}_\#)/\pi_T(\boldsymbol{\mu}_{i-1})\right)$
6: Draw from $\rho \sim Be(\alpha)$
7: **if** $\rho == 1$ **then**
8: Assign $\boldsymbol{\mu}_i = \boldsymbol{\mu}_\#$
9: **else**
Assign $\boldsymbol{\mu}_i = \boldsymbol{\mu}_{i-1}$
10: **end if**
11: **end for**
12: OUTPUT: the chain $(\boldsymbol{\mu}_0, ..., \boldsymbol{\mu}_N)$

Despite its effectiveness, the MA presents several critical points. First of all, the incidence of the initial guess can bias the simulation. For this reason, it is common to eliminate the first part of the chain through the called *burn-in*. Second, the choice of the proposal: if it is too wide too narrow, then it does not allows the chain to explore the posterior. Moreover, if its shape is too different from the target one the acceptance rate might be very low, slowing down the convergence. Finally, the latter is also hampered by the *curse of dimensionality*: when the number of parameters grows, the number of samples needed to accurately simulate the posterior increases exponentially. All the described issues contribute to exacerbate the biggest problem affecting not only the Metropolis algorithm but every MCMC method, namely the computational cost. Indeed, at every step of the random walk, specifically at line 5 of Algortithm 5.1, we need to evaluate the likelihood for a new instance of the parameters and this implicates the evaluation of the forward map, as emerges from expression (5.3). Practical applications require the generation of tens of thousands of samples, with an unsustainable computational cost if the state equation is a PDE to be solved numerically by means of high-fidelity FOMs.
To face this issue, developing every sort of surrogate model able to make the simulation feasible is therefore essential. After the training offline stage, the surrogate ability in providing fast evaluations for every new parameter candidate, allows to reach convergence in a reasonable amount of time even though, depending on the model accuracy, this

might introduce a bias in the simulation. A lot of works go further this objective by design strategies in which the interplay between the surrogate and the MCMC method guarantees even better performances. Just to make some examples, we can consider the use of the Reduced Basis method to approximate the PDE solution combined with reduction error models [24], of an adaptive Polynomial Chaos Expansion informed by the posterior to surrogate the parameters-to-QoI map [34], and finally, of a deep NN to do the same, enhancing a Delayed Rejection Metropolis algorithm [23]. The algorithm proposed in this Chapter follows the same path.

5.2. An adaptive algorithm for stochastic PDEs

We develop an adaptive version of the Metropolis algorithm with the goal of reconstructing complex random fields from a noisy measurement of the associate stochastic PDE solution. The proposed strategy not only makes an extensive use of DL-ROMs as a surrogate to speed-up the simulation, but takes advantage of the Hybrid DL-ROMs presented in Chapter 4 to deal with the curse of dimensionality. The latter is a particularly critic issue in this context, since the random field might be characterized by tens of input parameters for which determine a posterior distribution.

5.2.1. Our meta-algorithm

We design a way to tackle the curse of dimensionality by a "divide and conquer" strategy that infers the posterior distribution of some random coefficients at a time, starting from the most influential ones and enriching the picture during successive steps. This allows to improve our surrogate model between two successive runs of the Metropolis algorithm by exploiting the information resulting from the partial exploration of the posterior.

Let $\mu(\boldsymbol{x}) = \sum_{i=1}^{p} b_i(\boldsymbol{x})\ \mu_i$ be a random field truncated at order p, where b_i are the spatial modes and μ_i the random coefficients characterizing its expansion. Finally, let π_0 be the prior distribution that we assume for the vector of the random coefficients $\boldsymbol{\mu}^p = [\mu_1, ..., \mu_p]$. We first select a group of p_1 relevant parameters and we generate a dataset S_{T1} using the FOM and sampling the parameter space according to the prior distribution $\pi_0(\boldsymbol{\mu}^{p_1})$ where $\boldsymbol{\mu}^{p_1} = [\mu_1, ..., \mu_{p_1}, 0, ..., 0]$.
Then, we train a Base-p_1 DL-ROM to use online as a surrogate for a first run of the MA, with standards choices for what concerns the initial guess and the proposal distribution.

In particular, we have

$$\boldsymbol{\mu}_0^{p_1} = \mathbb{E}_{\pi_0}\left[\boldsymbol{\mu}^{p_1}\right] \quad \text{and} \quad \sigma_1 \sim \mathcal{N}(\mathbf{0}, s_\sigma^2 \mathbb{I}) \tag{5.6}$$

where s_σ is a scaling factor that ensure the correct exploration of the parameter space. At the end of the simulation, once the first k samples are eliminated in the burn-in procedure, we obtain the sample

$$\left(\boldsymbol{\mu}_{k+1}^{p_1}, ..., \boldsymbol{\mu}_N^{p_1}\right) \sim \pi(\boldsymbol{\mu}^{p_1}|\boldsymbol{q}^*).$$

At this point, we augment the number of parameters in the estimation process, passing from p_1 to p_2. Thanks to the transfer learning strategy developed in the Chapter 4, there is no need to devise another surrogate from scratch. We can indeed use an Hybrid-p_2 DL-ROM to inherit the information about the rough knowledge of the solution manifold associated to the state equation from the Base-p_1, used previously. The goal of the following MA run is then to simulate the "higher-dimensional version" of the posterior distribution, namely $\pi(\boldsymbol{\mu}^{p_2}|\boldsymbol{q}^*)$. This whole strategy can obviously be repeated more than one time, until the posterior distribution of all the random coefficient has been inferred.

The idea behind this strategy is the following: the surrogate model needs not to be accurate in an homogeneous way over the whole parameter space, but it should be more precise in evaluating the PDE solution where the posterior is concentrated. We achieve this goal through the generation of a new dataset S_{T2} for the Hybrid-p_2 training, obtained by sampling from the updated prior distribution that $\pi_0^{'}(\boldsymbol{\mu}^{p_2})$ can be expressed as follows

$$\pi_0'(\boldsymbol{\mu}^{p_2}) = \pi(\boldsymbol{\mu}^{p_1}|\boldsymbol{q}^*) \cdot \int_{\mathbb{R}^{p_1}} \pi_0(\boldsymbol{\mu}^{p_2}) d\mu_1 ... d\mu_{p_1} \ .$$

The above definition is given by the product of the simulated "partial" posterior and of the marginal prior for the remaining block of parameters. Training the Hybrid-p_2 over input parameter instances sampled in this way allow to create a model tailored for the specific inverse problem, with both low training and test error, despite the use of a limited number of new data. The main risk of this strategy consists in the fact that the first parameter estimation might be biased and a consequence the Hybrid DL-ROM might be inaccurate where the posterior is actually concentrated. This would entail an even bigger bias leading to the non-convergence of the Bayesian inversion procedure. For this reason we introduced a *spreading factor* K_{Sp} that enhances the covariance of $\pi(\boldsymbol{\mu}^{p_1}|\boldsymbol{q}^*)$ allowing, during the generation of S_{T2}, the sampling of a wider portion of the parameter space,

included the correct region (as depicted in Figure 5.1).

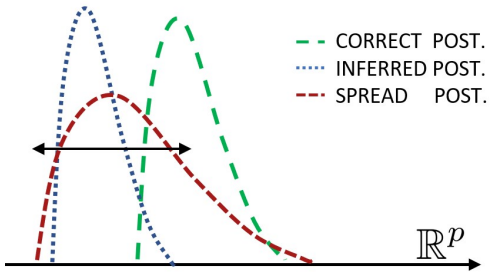


Figure 5.1: The biased inferred posterior is spread in order to sample also from the correct portion of the parameter space.

With a proper tuning of K_{Sp} we were able to increase considerably the algorithm robustness, without distorting the key idea. The proposed strategy in its two-stage basic version is formalized in Algorithm 5.2, but it can be easily generalized to an arbitrary number of stages. For what concerns the second MA run, as well as the the potential next ones, we can further exploit the information coming from the first simulation, making also the random walk adaptive.
In particular, we can exploit in the definition of the starting value $\boldsymbol{\mu}_0^{p_2}$ and of the covariance matrix of the proposal σ_2 the simulated chain and the estimation of $\tilde{C}_C^1 \approx Cov(\boldsymbol{\mu}^{p_1}|\boldsymbol{q}^*)$ by eq. (5.5). Therefore, we have

$$\boldsymbol{\mu}_0^{p_2} = \mathbb{E}_{\pi_0'}\left[\boldsymbol{\mu}^{p_2}\right] \quad \text{and} \quad \sigma_2 \sim \mathcal{N}(\mathbf{0}, C_{\sigma_2}) \quad \text{with} \quad C_{\sigma_2} = s_\sigma \begin{bmatrix} \tilde{C}_C^1 & 0 \\ 0 & \mathbb{I} \end{bmatrix} . \tag{5.7}$$

Algorithm 5.2 2-stage adaptive meta-algorithm

1: INPUT: Truncation orders p_1, p_2, the FOM, the spreading factor K_{Sp}
2: BEGIN STAGE 1: SIMULATION OF $\pi(\boldsymbol{\mu}^{p_1}|\boldsymbol{q}^*)$
3: Sample from $\pi_0(\boldsymbol{\mu}^{p_1})$ the input parameter instances
4: Generate S_{T1}, by exploiting the FOM for the PDE solution
5: Implement and train on S_{T1} a Base-p_1 DL-ROM
6: Use it as surrogate in Alg 5.1 with $\boldsymbol{\mu}_0^{p_1}$ and σ_1 as in (5.6)
7: END STAGE 1
8: BEGIN STAGE 2: SIMULATION OF $\pi(\boldsymbol{\mu}^{p_2}|\boldsymbol{q}^*)$
9: Sample from the K_{Sp}-spread updated posterior $\pi_0'(\boldsymbol{\mu}^{p_2})$ parameters instances
10: Generate S_{T2}, by exploiting the FOM for the PDE solution
11: Implement and train on S_{T2} an Hybrid-p_2 DL-ROM inheriting from the Base-p_1
12: Use it as surrogate in Alg 5.1 with $\boldsymbol{\mu}_0^{p_2}$ and σ_2 as in (5.7)
13: END STAGE 2

5.2.2. Numerical experiments

In order to evaluate the performances of the proposed strategy, we present the results obtained for two test-cases. The class of PDEs, used to prescribe the state equation for both of them, is the same of the previous Chapters, namely

$$\begin{cases} -\nabla \cdot (e^{\mu} \nabla u) = 1 & \text{in } D \\ \nabla u = u & \text{on } \partial D \end{cases},$$

with $D = (0,1)^2$. Our objective is to identify the random field μ given some pointwise measurements of the PDE solution across the domain, $Q(u_\mu; \mu) = [u_\mu(\boldsymbol{x}_1), ..., u_\mu(\boldsymbol{x}_s)]$. These values, which define the QoI according to our notation, are measured at some fixed sensor locations (see e.g. Figure 5.2).

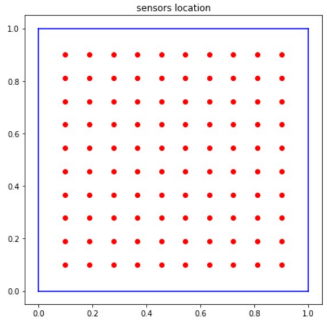


Figure 5.2: Setting of the sensors location implied for the test-cases.

Within the Bayesian framework, we assume that μ is a centered Gaussian random field truncated at a certain order p. In particular, μ can be expressed in terms of its Karhunen–Loeve decomposition as

$$\mu(x) = \sum_{i=1}^{p} \sqrt{\lambda_i}\; b_i(x)\; \mu_i$$

where λ_i are the decreasing eigenvalues of the random field covariance kernel, already plotted in Figures 3.2 and 3.7, while b_i are the spatial modes of the field, characterized by an increasing complexity clearly visible from Figure 5.3, and μ_i are the random coefficients. We assume *a priori* that the latters are independent and have a standard Gaussian distribution, therefore denoted by $\boldsymbol{\mu}^p = [\mu_1, ..., \mu_p]$ we have that

$$\boldsymbol{\mu}^p \sim \mathcal{N}(\mathbf{0}, \mathbb{I}) \quad \text{and} \quad \pi_0(\boldsymbol{\mu}^p) = \frac{1}{(2\pi)^{p/2}} \exp\left(-\frac{|\boldsymbol{\mu}^p|^2}{2}\right) \ .$$

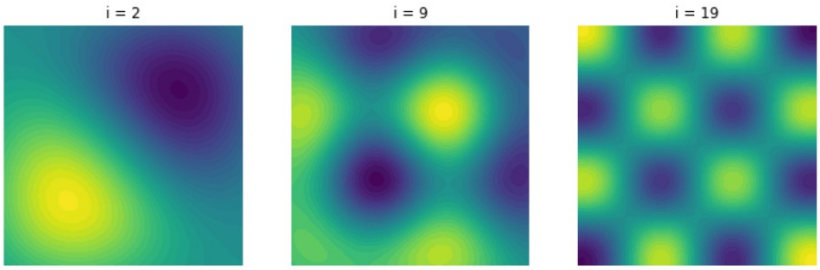


Figure 5.3: Spatial modes of increasing complexity in the KL decomposition of the random field.

Concerning the likelihood we adopt the error model (5.2) with the error distributed Normally and the further simplifying assumption of known covariance matrix. In particular, we assume that the experimental measurements of the solution at the sensors location are uncorrelated (and thus independent) with the noise variance given by 1% of the L^1-norm of the PDE solution. Considered the large number of sensors $N_s = 100$ involved in the experiment and the shape of the domain, we can approximate the latter with the average of the QoI measurements. In conclusion, we have that

$$\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, C) \quad \text{with} \quad C_{ij} = 10^{-2} \left(\frac{1}{N_s} \sum_{k=1}^{N_s} |u(\boldsymbol{x}_k)|\right) \delta_{ij} \ ,$$

where δ_{ij} denotes the Kronecker delta. We can summarize our goal with the aid of Figure 5.4. We aim to infer the random field (on the left) generating the PDE solution (at the center) by finding the posterior distribution of the random coefficients in the KL decomposition. This is performed by starting from a Gaussian prior distribution and

leveraging the experimental noisy measurement of the solution on a grid of sensors (on the right).

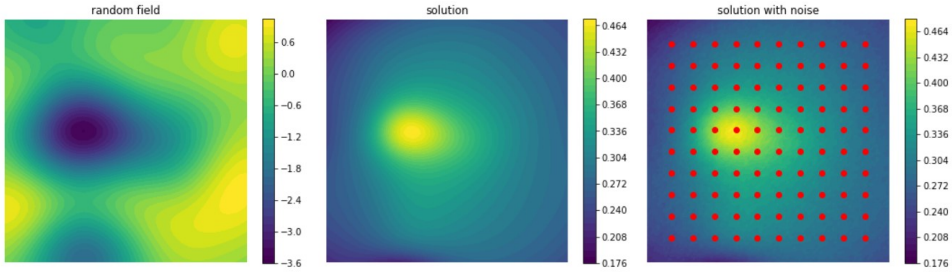
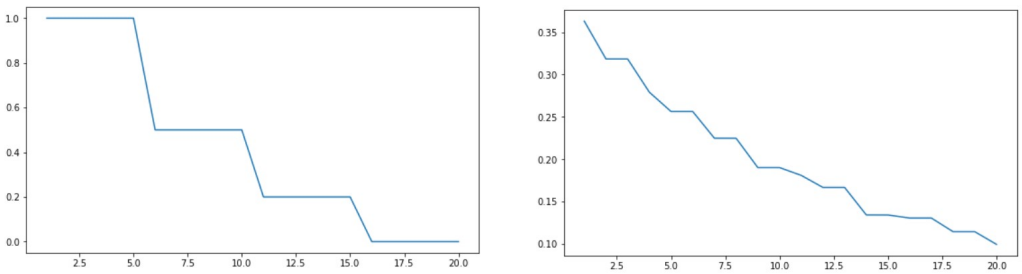


Figure 5.4: Setting of the Bayesian inverse problem.

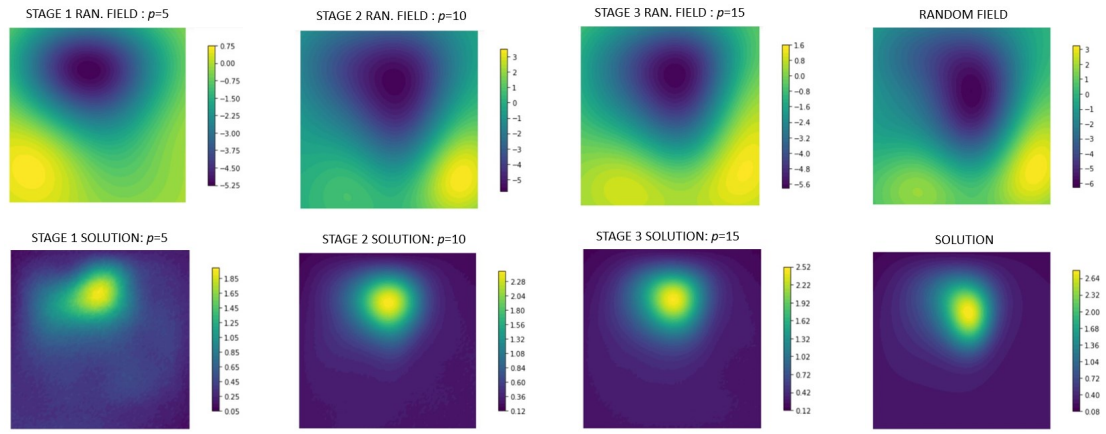
We now report the results obtained for two test-cases with our adaptive meta-algorithm. For both of them we adopted a 3-stage strategy. We quantify the improvement in the performances by estimating through the simulated posterior the conditional mean and computing the relative L^2-error (RE) between the inferred and the benchmark random field. We compute the same quantity also between the PDE solution and the one corresponding to the inferred field. We observed that our strategy performs particularly well when we are able to determine groups of random coefficients having a comparable influence on the stochastic PDE solution. This could be quantified through the eigenvalues associated at the random coefficient. Therefore, for the first case we manually fixed them according to Figure 5.5a.



(a) Test-case 1 (b) Test-case 2

Figure 5.5: Eigenvalues for the test-cases.

The corresponding results are reported in Figure 5.6, where we compare the random fields and the associated solutions obtained after every stage of the process. We can clearly see the improvements from one stage to the next one, until reaching a quite accurate result, in particular, considering that the error between the solution has the same magnitude of the error in the QoI experimental measurement.



(a) RE sol = 37.4 %, RE field = 45.7 %

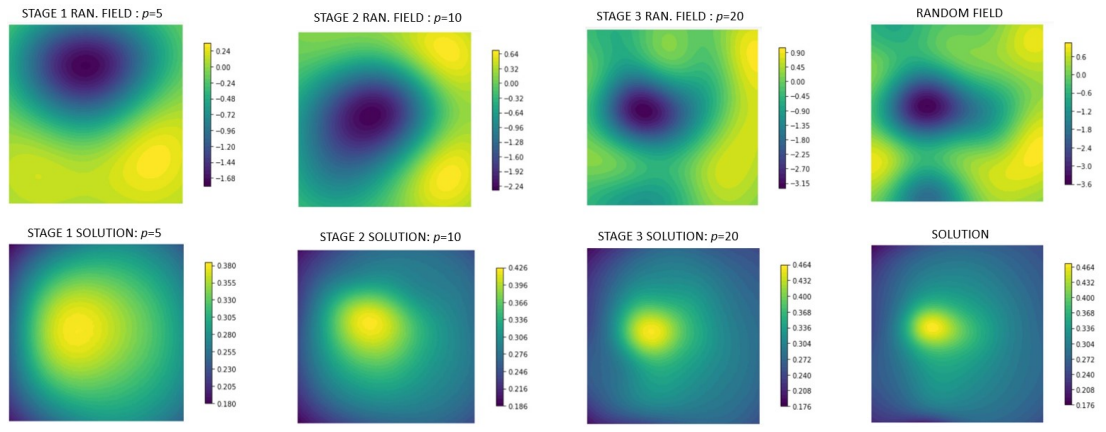
(b) RE sol = 2.8 %, RE field = 19.8 %

(c) RE sol = 1.1 %, RE field = 7.1 %

(d) Benchmark

Figure 5.6: Inferred random field and associate solution for test-case 1.

The same holds also for the more realistic second test-case, whose results are depicted in Figure 5.7. The eigenvalues here, see Fig. 5.5b, are indeed associated to a Gaussian covariance kernel.



(a) RE sol = 5.6 %, RE field = 87.15 %

(b) RE sol = 2.6 %, RE field = 50.1 %

(c) RE sol = 1.06 %, RE field = 24.8 %

(d) Benchmark

Figure 5.7: Inferred random field and associate solution for test-case 2.

The estimate of the random field is qualitatively valid since it catches all the benchmark important features. However, it is affected by a still quite high relative error that can be attributed to the smoothing properties of the stochastic PDE: the relative error in terms of the PDE solution is extremely low and of the same magnitude of the experimental noise.

Concerning the computational cost of the method, we have to consider that we generated 1000 snapshots for the first training set and 500 for each of the following ones and we performed $N = 50000$ steps for every run of the MA. The total time needed by the whole process for both the case was approximately 1200s that are allocated in the following way: training time 50%, MA simulations 30%, FOM time 20%. Just to make a comparison, assuming to be able to obtain the same posterior simulation using the FOM instead of the Hybrid DL-ROM with just one fifth of the steps, considering 0.2 s per steps, the implied time would have been more than ten times higher.

Conclusions

This work concerned Deep Learning-based Reduced Order Models, their theoretical properties and the development of strategies to improve their performances. In particular, we accomplished the extension of a theoretical result that bounds the error committed by an auto-encoder, with a prescribed latent dimension, for the case of stochastic PDEs. We then confirmed the validity of the estimate by carrying out several experiments in the case of elliptic PDEs parametrized by random fields.

We furthermore proposed two different strategies for the reduction of the computational cost entailed by the offline stage, a Multi-level training algorithm and a Hybrid DL-ROM. The 2-levels training algorithm allowed to reduce the cost of the dataset generation up to 65%, without compromising the accuracy and also lowering the training cost. This was shown empirically in the case of an elliptic PDE whose parameter dependence was highly nonlinear. Conversely, the introduction of Hybrid DL-ROM allowed us to cut down by 30% the computational time needed for the training phase of the ROM. This happened for the case of random field parametrization, for which we also demonstrated that a DL-ROM inheriting a fixed internal representation of the solution manifold, can enhance this latter with further details during a re-train, improving the performances obtained with the standard training procedure.

We finally employed Hybrid DL-ROMs in designing an adaptive Metropolis algorithm for the efficient and accurate solution of a Bayesian inverse problem. In particular, we were able to estimate with adequate levels of accuracy the structure of complex random fields despite the highly regularizing nature of the forward problem and the large number of random parameters. This goal was achieved through the use of a limited amount of computational resources (speed-up $\times 10$ with respect to the use of a FOM), showing the capabilities of DL-ROMs as accurate model surrogates.

This work touched many different aspects related to the design, the use, and the analysis of DL-ROMs and, as a consequence, further steps can be made in several directions. From a theoretical perspective, it would be interesting to proof an approximation result specifically tailored for the case of time-dependent problems, that were not considered in this work. Indeed, while time may be considered as an additional parameter, as it was done in [10], better strategies may be available. Concerning the Multi-level training, a deeper exploration of the interplay between model accuracy and the number of discretization levels could also yield significant improvements - this is still an open issue, indeed, of an otherwise very promising learning strategy. For the case of Hybrid DL-ROMs, further developments primarily regard some technical aspects: for instance, it would be of interest to design a strategy for partially freezing not only dense layers, but also convolutional and mesh-informed ones. The use of sparse layers is indeed fundamental for larger-scale problems. Finally, all the aforementioned steps might be useful for further enhancing the performances of our adaptive Metropolis algorithm. The final goal in this direction would be the development of a black-box application that is able to automatically tune its hyper-parameters, such as the number of stages and the spread factor, through prescribed criteria, in order to further enhance the accuracy of the parameter estimation.

Bibliography

[1] URL `https://fenicsproject.org/`.

[2] URL `https://pytorch.org/`.

[3] Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[4] C. C. Aggarwal. *Neural Networks and Deep Learning*. Springer, 2018.

[5] R. A. DeVore, R. Howard, and C. A. Micchelli. Optimal nonlinear approximation. *manuscripta mathematica*, 63:469–478, 1989.

[6] J. Dugundji. An extension of tietze's theorem. *Pacific Journal of Mathematics*, 1(3): 353 – 367, 1951.

[7] N. R. Franco, A. Manzoni, and P. Zunino. A Deep Learning approach to Reduced Order Modelling of Parameter Dependent Partial Differential Equations. *ArXiv, preprint*, 2021.

[8] N. R. Franco, A. Manzoni, and P. Zunino. Learning Operators with Mesh-Informed Neural Networks. *ArXiv, preprint*, 2022.

[9] S. Fresca and A. Manzoni. POD-DL-ROM: Enhancing deep learning-based reduced order models for nonlinear parametrized PDEs by proper orthogonal decomposition. *Computer Methods in Applied Mechanics and Engineering*, 388, jan 2022.

[10] S. Fresca, L. Dede, and A. Manzoni. A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized pdes. *Journal of Scientific Computing*, 87, 05 2021.

[11] J. Harlim. *Data-Driven Computational Methods: Parameter and Operator Estimations*. Cambridge University Press, 2018.

[12] P. Jin, L. Lu, Y. Tang, and G. E. Karniadakis. Quantifying the generalization error

in deep learning in terms of data distribution and neural network smoothness. *Neural Networks*, 130:85–99, Oct 2020.

[13] J. Kaipio and E. Somersalo. *Statistical and Computational Inverse Problems*. Springer, 2004.

[14] B. Kaushik, H. Bamdad, K. Nikola B., and S. Andrew M. Model Reduction And Neural Networks For Parametric PDEs. *The SMAI journal of computational mathematics*, 7:121–157, 2021.

[15] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2014.

[16] T. Kuhn. Eigenvalues of integral operators with smooth positive definite kernels. *Archiv der Mathematik*, 49(6):525 – 534, 1987.

[17] G. Kutyniok, P. C. Petersen, M. Raslan, and R. Schneider. A theoretical analysis of deep neural networks and parametric PDEs. *Constructive Approximation*, 55:73–125, 2021.

[18] S. Lanthaler, S. Mishra, and G. Karniadakis. Error estimates for DeepONets: a deep learning framework in infinite dimensions. *Transactions of Mathematics and Its Applications*, 6, 2022.

[19] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521:436–44, 05 2015.

[20] K. O. Lie, S. Mishra, and R. Molinaro. A multi-level procedure for enhancing accuracy of machine learning algorithms. *European Journal of Applied Mathematics*, 32 (3):436–469, 2021.

[21] L. Lu, P. Jin, G. Pang, H. Zang, and G. Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3:218–229, 03 2021.

[22] K. O. Lye, S. Mishra, and D. Ray. Deep learning observables in computational fluid dynamics. *Journal of Computational Physics*, 410:109339, Jun 2020.

[23] M. Lykkegaard, T. Dodwell, and D. Moxey. Accelerating uncertainty quantification of groundwater flow modelling using a deep neural network proxy. *Computer Methods in Applied Mechanics and Engineering*, 383:113895, 09 2021.

[24] A. Manzoni, S. Pagani, and T. Lassila. Accurate solution of bayesian inverse uncertainty quantification problems combining reduced basis methods and reduction error models. *SIAM/ASA Journal on Uncertainty Quantification*, 4(1):380–412, 2016.

[25] S. Mishra and T. K. Rusch. Enhancing accuracy of deep learning algorithms by training with low-discrepancy sequences. *SIAM Journal on Numerical Analysis*, 59: 1811–1834, 01 2021.

[26] A. Quarteroni. *Numerical Models for Differential Problems*. Springer, 6 edition, 2016.

[27] C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer, 1999.

[28] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychol Rev.*, 65(6):386 – 408, 1958.

[29] S. Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.

[30] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[31] S. Salsa. *Equazioni a derivate parziali. Metodi, modelli e applicazioni*. Springer, 2 edition, 2010.

[32] C. Schwab and J. Zech. Deep learning in high dimension: Neural network expression rates for generalized polynomial chaos expansions in UQ. *Analysis and Applications*, 17(01):19–55, 2019.

[33] B. Widrow and M. E. Hoff. *Adaptive Switching Circuits*, page 123–134. MIT Press, Cambridge, MA, USA, 1988.

[34] L. Yan and T. Zhou. Adaptive multi-fidelity polynomial chaos approach to bayesian inference in inverse problems. *Journal of Computational Physics*, 381:110–128, mar 2019.

[35] D. Yarotsky. Error bounds for approximations with deep ReLU networks. *Neural Networks*, 94:103–114, 2017.

List of Figures

List of Tables