POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

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

Laurea Magistrale in Mathematical Engineering - Ingegneria Matematica

Author: Daniel Fraulin

Advisor: Prof. Andrea Manzoni

Co-advisor: Dott. Nicola Rares Franco

Academic year: 2020-2021

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 Galerkin-Finite Element Method (FEM). 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, Bayesian inversion or 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 [1, 3]. 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 FOMs 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. [4, 5]) 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 [1] 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 (dofs) 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 is theoretical and consists in the extension of the aforementioned approximation results to the case of stochastic PDEs, which are parametrized by random fields, and thus by infinite dimensional objects. The provided error estimate is then confirmed by numerical experiments. The second path is instead related to more practical issues. Indeed, we design several strategies to alleviate the burden of the expensive DL-ROM offline stage, all based on the concept of Transfer Learning. In particular, a multi-level training algorithm, suitable for any kind of parametrized PDE, has been proposed taking advantage of snapshots at lower resolution to reduce the dataset generation time. Moreover, coming back to the case of stochastic PDEs, a class of Hybrid AEs has been designed. The latter allow to reduce the training time, thanks to their ability of inheriting, from simpler AEs, an internal representation of the solution manifold and enhance it with further details through a suitable re-training procedure.

1. DL-ROMs implementation

We take advantage of this first section to introduce the adopted notation before briefly reporting some technical details regarding the DL-ROMs implementation. We denote by $\Theta \subset \mathbb{R}^p$ the set of the parameter configurations, by $\boldsymbol{\mu}$ the parameters vector and by $u^h_{\boldsymbol{\mu}}$ the associated high-fidelity FOM solution, that we assume belonging to some Hilbert space $\mathcal{V}_h, ||\cdot||$). Concerning the NN, the auto-encoder is made by the composition of an encoder $\Psi^{'} : \mathcal{V}_h \to \mathbb{R}^n$, which receives as an input an instance of the solution and maps it into the latent space, and of a decoder $\Psi : \mathbb{R}^n \to \mathcal{V}_h$, operating the reconstruction. Finally, we denote by $\phi : \Theta \to \mathbb{R}^n$ the NN mapping the parameters vector into the latent space. In this context, n is called *latent dimension* and the aforementioned theoretical results allows to chose $n = p$ or $n = 2p + 1$, depending on the hypotheses satisfied by $u^h_{\boldsymbol{\mu}}$, introducing an arbitrarily small compression error. Consequently, as desired when dealing with reduced order models, $n \ll \dim(\mathcal{V}_h)$, as long as p is small. Coming to the actual implementation, the architectures are determined through a semi-empirical approach, using both dense and convolutional layers. The dataset $\{(\boldsymbol{\mu}_i, u^h_{\boldsymbol{\mu}_i})\}_{i=1}^N$, which is later divided into training and test set, is generated synthetically by sampling from the parameter space and exploiting the FOM. The DL-ROM is then trained according to the steps in Figure 1: in phase 1a the auto-encoder is fed with the instances of the discretized solution manifold until an accurate internal representation of the reduced solution manifold is formed. This is achieved by minimizing the Loss Function below

$$LF(\text{AE}) = \frac{1}{N_{tr}} \sum_{i=1}^{N_{tr}} \frac{||u^h_{\boldsymbol{\mu}_i} - \Psi \circ \Psi^{'}(u^h_{\boldsymbol{\mu}_i})||}{||u^h_{\boldsymbol{\mu}_i}||} \ .$$

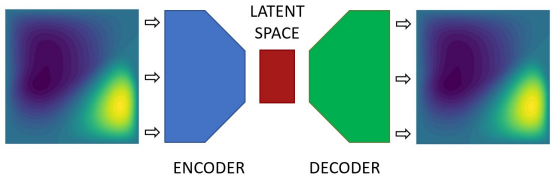
The encoder and the decoder are then separated in phase 1b and the former is used to generate the reduced order version of the dataset, $\{(\boldsymbol{\mu}_i, \Psi^{'}(u^h_{\boldsymbol{\mu}_i}))\}_{i=1}^N$. The third network, ϕ, is then trained over this dataset by minimizing objective function below

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

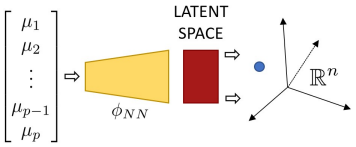
Finally, the decoder is connected to ϕ in phase 1c, so that the resulting networks maps a vector of parameters into the corresponding PDE solution and the DL-ROM is fully operative.

2. Multi-fidelity training

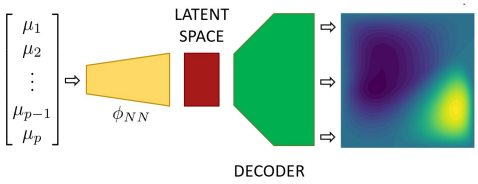
A multilevel strategy has been considered in order to reduce the off-line cost entailed by the generation of the training set. Indeed, a large amount of snapshots is required to obtain DL-ROMs that can generalize well, but the computational cost of solving the PDE using the FOM becomes particularly high when working on refined meshes.
However, we can take advantage of the fact that



(a)



(b)



(c)

Figure 1: DL-ROM implementation process.

the minimal latent dimension does not depend on the discretization: as a consequence, the low dimensional representation of the solution manifold shall be compatible with any choice of the mesh size. This means that the auto-encoding process can be run once and for all on data set on a coarser mesh (and using a smaller architecture). The same kind of reasoning applies also to the NN ϕ, 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 exploited this idea to propose the following framework for a multi-level training:

1. generate the dataset for a coarse mesh;
2. implement and train the DL-ROM following Figure 1;
3. freeze the NN $\phi \circ \Psi$ and connect a further network χ that maps the solution of the coarser mesh to a more refined one;
4. re-train the model $\phi \circ \Psi \circ \chi$ on a few instances of the solution on the refined mesh.

Here, by *freezing*, we denote the operation of fixing the values of the NN dofs, preventing further modifications during subsequent training sessions and reducing the computational cost of these latter.
Concerning the design of χ, both dense and convolutional layers, namely the two principal blocks of DL-ROMs, revealed to be unsuitable for this task. Dense layers contain too many weights, making the resulting NN hardly trainable, whereas convolutional layers have the exact opposite problem, since they are characterized by a very limited number of dofs. Good results have been achieved instead by using *Mesh Informed* layers, recently introduced in [2]. 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.
The success of the whole strategy depends anyway on the way the generalization properties of the coarser model are inherited from the multi-fidelity one. This has been tested on the following PDE, on the domain $D = (0,1)^2$ and, as parameter space, $\Theta = [0,1]^4$:

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

where $\boldsymbol{\beta}_{\boldsymbol{\mu}} = 10^{-1}(\cos(2\pi\mu_4), \sin(2\pi\mu_4))^T$ and $\sigma_{\boldsymbol{\mu}}$ depends on 3 parameters that determine the shape of a "discontinuity" line in the diffusion field (see Figure 2).

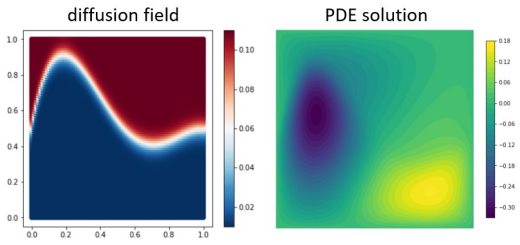


Figure 2

During the experiment, we compared the performances of a standard DL-ROM, working on a 100×100 elements mesh, with a 2-levels DL-ROM that exploits a coarse mesh made by 50×50 elements. The results reported in Table 1 show test error and off-line computational time for three different combinations of coarse and high-fidelity samples, respectively denoted by (C) and (H). The proposed method allows to reduce both the total training time, due to the splitting of the dofs to be simultaneously optimized, and, drastically, the time implied by the dataset generation, implying only a slight decrease in the accuracy levels.

Model	Test err	Samples N°	Tot. training T.	Data Gen. T.
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 1: Multi-level training vs standard one on varying the cardinality of the training sets.

3. DL-ROMs for stochastic PDEs

The use of DL-ROMs has to be extended to 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, 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 original theoretical contribution is developed in two steps. Th first Theorem deals with the case of a finite number of real random input parameters. The second one follows immediately by using a dimensionality reduction technique, based on a Karhunen-Loeve (KL) expansion of the 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 map, with $(\mathcal{V}, ||\cdot||)$ Hilbert space, and let $\mathcal{S} \subset \mathcal{V}$ be the manifold obtained by mapping Θ through u. Then, if either $n \geq 2p+1$, or $n \geq p$ but u is injective, it holds that*

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

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 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 of the random field realizations. Finally let u, $\mathcal{V}$ 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*

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

This last result is highly effective when the covariance kernel smoothness lets the eigenvalues decay in a very fast way: for instance, in the case of a Gaussian kernel the decay is known to be exponential. We confirmed the expected estimate by running a numerical experiment to learn the parameter-to-solution map associated to the following PDE:

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

where $D = (0,1)^2$ and μ denotes a centered Gaussian random field. To show the predicted trends, we used five orders of truncation: $p = 5, 10, 20, 40, 100$. The last one approximates the whole, not truncated, field playing the role of the benchmark. An instance of the benchmark field and the correspondent solution is reported in Figure 3.

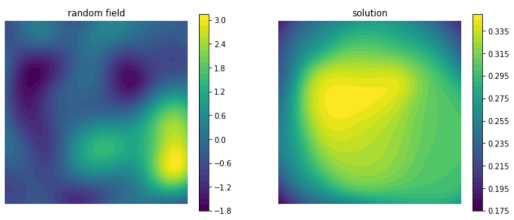


Figure 3

Despite the fact that Theorem 3.1, combined with classical NN approximation theorems, ensures the existence of an AE that can learn the solution manifold at any level of accuracy, when dealing with the model training we must consider the presence of the test error. We recall that the latter is due to the high dimensionality and non-convexity of the *loss surface* and to the finite amount of exploitable training data. Consequently, in order to make the results of the analysis robust to this source of error, we considered three groups of AEs, with different levels of accuracy. The latter was increased by augmenting the value of the parameter m, that determines the AE expressiveness by fixing the number of channels of the convolutional layers. The AE bottleneck width was instead chosen depending on the truncation order to satisfy the hypothesis of Theorem 3.2, which is $n = 2p + 1$. As a consequence, in each group there are four AEs with latent dimension $n = 11, 21, 41, 81$. From Figure 4, we can see that the error decay rate increases with the parameter m, finally obtaining for $m = 16$ an almost perfect match with the theoretical estimate.

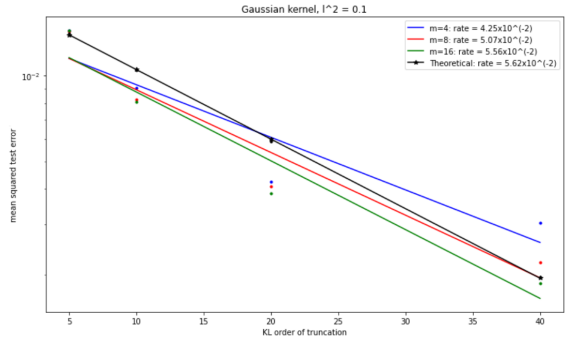


Figure 4: Predicted decay vs experimental ones.

4. Transfer learning

Then, we have considered Transfer Learning (TL) strategy for the case of PDEs parametrized by random fields. The goal consists in designing an AE able to inherit the information contained in another one, already trained for for a simpler problem admitting a smaller latent dimension (as when considering fewer modes in the random field truncation). Besides the speculative interest in understanding whether it is possible to enhance with further details the AE internal representation of the solution manifold, by starting from a fixed simpler structure, there are also practical reasons to explore this path. Indeed, splitting the training of a very large AE in more steps allows not only to lower the memory resources needed, but may also reduce the computational time implied by the training.
The proposed method consists in copying the value of weights and biases of an already trained AE, that we call Base AE, in another one, the Hybrid AE, with a greater latent dimension and a few more degrees of freedom also in the other layers. These parameters are then frozen, with a decisive impact on the cost of each training epoch. The remaining part of the NN is then initialized and re-trained. The starting point was the implementation from scratch of an efficient hybrid dense layer, that can be partially frozen. Then we run some numerical experiments to optimize the design. In particular, we clarify how to grow the Base, how to initialize the additional dofs and what the effect of changing the dataset after the hybridization process is. Finally, we compared the errors committed by the AEs built and trained normally with the Hybrid ones, keeping Problem 1 as benchmark.
Since it is useful to look at a performance indicator that is independent of the specific order of truncation for the DL-ROM construction, we define the *true error* as

$$E_{true} = \frac{1}{N_{true}} \sum_{i=1}^{N_{true}} ||u^h_{\mu_i} - \Psi \circ \Psi'(u^h_{\mu_i})||$$

so that it measures the accuracy in the reconstruction of the benchmark solution manifold. In order to guarantee the experiment fairness, the compared DL-ROMs share the same number of dofs and also the same training set. Figure 5 (in which we denote by Base/Hybrid-p, an AE working on a p-dimensional parameter space) shows that: i) the Transfer Learning strategy not only allows to reach an arbitrary accuracy, but can even outperform the standard training; ii) the Transfer Learning strategy can be used more than once, using an Hybrid in turn as a Base.
Finally, we analysed the advantages of the presented strategy in terms of computational cost: as reported in Table 2, splitting the training in two parts allows to save approximately 30% of time and simultaneously increases the accuracy. The use of further intermediate Hybrid auto-

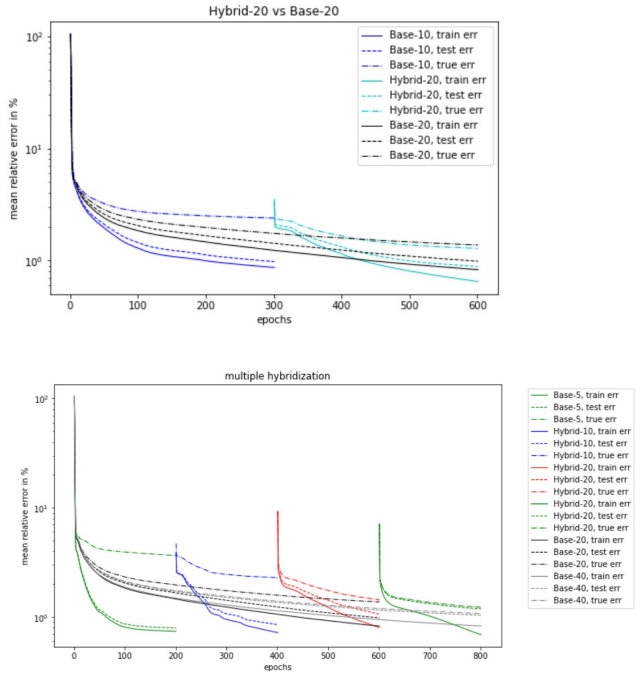


Figure 5: Mean error paths obtained through standard training and the TL strategy.

encoders guarantees even higher improvements, keeping constant the time per epoch, only implying a slight worsening in the performances.

5. 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 AE, with a prescribed latent dimension, when the parameter is a random field. We then confirmed the estimate validity by numerical experiments. We furthermore proposed two different strategies for the reduction of the off-line stage cost. The multi-level training algorithm allowed to reduce the cost of the dataset generation up to 65%, without compromising the accuracy and lowering also the training cost. This was proved on a highly nonlinear (in the parameters) elliptic PDE. The introduction of Hybrid AEs allowed instead to cut down by 30% the computational time needed for the training. This happened for the case of random field parametrization, for which we also demonstrated that an AE inheriting a fixed internal representation of the solution manifold, can enhance it with further details during a re-train, reaching higher level of accuracy than with a standard training procedure.

AE type	Test err	Time	Gain
B5:H10	0.75%	3m 15s	30%
B10	0.91%	4m 39s	
B10:H20	0.88%	6m 29s	29%
B5:B10:H20	1.06%	5m 5s	45%
B20	0.98%	9m 12s	
B20:H40	0.80%	10m 2s	34%
B5:H10:H20:H40	1.19%	7m 2s	56%
B40	0.87%	15m 45s	

Table 2: TL computational cost report.

References

[1] Nicola Rares Franco, Andrea Manzoni, and Paolo Zunino. A deep learning approach to reduced order modelling of parameter dependent partial differential equations. *ArXiv, preprint*, 2103.06183, 2021.

[2] Nicola Rares Franco, Andrea Manzoni, and Paolo Zunino. Learning Operators with Mesh-Informed Neural Networks. *ArXiv, preprint*, 2022.

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

[4] Bhattacharya Kaushik, Hosseini Bamdad, Kovachki Nikola B., and Stuart. Andrew M. Model Reduction And Neural Networks For Parametric PDEs. *The SMAI journal of computational mathematics*, 7:121–157, 2021.

[5] Lu Lu, Pengzhan Jin, Guofei Pang, Handy Zang, and George Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3:218–229, 03 2021.