Doctoral Dissertation
Doctoral Program in Chemical Engineering (35^{th}cycle)

Deep neural networks as data-driven models for flow and transport in porous media

By

Agnese Marcato

******

Supervisor(s):
Prof. Daniele Marchisio, Supervisor
Prof. Gianluca Boccardo, Co-Supervisor

Doctoral Examination Committee:
Prof. Antonio Bertei, Referee, Università di Pisa
Prof. Massimiliano Villone, Referee, Università degli Studi di Napoli Federico II
Prof. Antonio Buffo, Politecnico di Torino
Prof. Tania Cerquitelli, Politecnico di Torino
Dr. Yacine Haroun, IFP Energies Nouvelles

Politecnico di Torino
2023

Declaration

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Agnese Marcato
2023

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

I would like to dedicate this thesis to my grandmother Maddalena.

Vorrei dedicare questa tesi a mia nonna Maddalena.

Acknowledgements

I want to thank several people who, in various forms, have contributed to the work presented in this dissertation.

The work presented in Chapter 6 was done while I was at the University of Texas at Austin, where I was hosted in the research group of Prof. Maša Prodanović.

The subsequent Chapter 7 presents the work done in the context of the Horizon 2020 project BIGMAP (GA 957189), in collaboration with Prof. Alejandro Franco, who hosted me in the Laboratoire de Reactivité et Chimie des Solides (LRCS, Amiens).

I want to thank my mentors at the Los Alamos National Laboratory Dr. Javier Estrada Santos and Dr. Eric Guiltinan who gave me the chance to spend the summer working in the Earth and Environmental Science division. Thanks Javier for being there all the way.

I sincerely would like to thank my supervisors Prof. Gianluca Boccardo and Prof. Daniele Marchisio. I want to thank Prof. Roberto Pisano and Prof. Antonello Barresi for the collaboration, and all the research group in Politecnico di Torino. Lastly, I want to thank Prof. Matteo Icardi at University of Nottingham for the fruitful discussions. The effort of the reviewers of this manuscript is also very gratefully acknowledged.

Thank you Lorenzo Vasquez Giuliano and all the other members of both the Mulmopro team and the other research groups I've visited in my travels!

The computational resources for this work have been provided by HPC@polito and by CINECA awards under the ISCRA initiative.

Abstract

Porous media systems are relevant in many research fields of chemical engineering: packed bed catalytic reactors, filters, subsurface applications like carbon capture and storage, and batteries. The microscale modelling, i.e. pore-scale, of a representative volume of the porous structure is a state-of-the-art methodology to obtain the accurate evaluation of transport related properties, such as reaction rates or filtration efficiencies. Computational fluid dynamics (CFD) has been widely employed to this end. Nevertheless, these microscale simulations are computationally expensive, in fact, high performance computing (HPC) systems and long computation times are usually required to numerically solve the transport equations. Thus, these models can hardly be integrated in multiscale modelling, or optimization workflows, where fast response models are needed. Machine learning, and in particular deep learning techniques, can be employed to train data-driven models as surrogates of CFD simulations. Starting from a CFD based dataset, neural networks can be trained to obtain accurate predictions of the quantities of interest, resulting in a fast surrogate model to employ in the above-mentioned examples.

This methodological dissertation is a benchmark for the use of neural networks as surrogate models for flow and transport in porous media. Two main applications have been studied: the filtration of colloids in packed beds, and the discharge of the cathode side of lithium-ion batteries. Even though these applications are governed by very different transport equations, both reactive problems were successfully modelled by neural networks, which is the main novelty of this work. In fact, main efforts of this kind of machine learning coupling in the literature are addressed to the prediction of the permeability of porous media, which is a geometry related factor, but fewer have tackled more complex problems of common interest in chemical engineering, with machine learning techniques. For the filtration case study two categories of data-driven models have been developed: networks for the prediction of integral properties, and networks for the prediction of local fields. The dataset for the

training of those models has been created by the solution of the transport equation on bi-dimensional and three-dimensional sphere packings by means of a finite volume method. In the first case, simple fully connected neural networks have been trained for the prediction of both the permeability and the filtration rate starting from hand selected geometrical parameters, and operating conditions. Another approach implemented for the prediction of the same objectives is the use of convolutional neural networks (CNN), which are appropriate models for porous media applications since they allow images of the porous structure as inputs, in this way the choice of most relevant geometrical features is performed by the network itself. Instead, multiscale convolutional neural networks have been employed for the prediction of local concentration fields in porous media. In this case the dimensionality of the model is not reduced, resulting in a complete surrogation of the CFD simulations, thus the trained model gains interpretability compared to the previous ones.

The lithium-ion batteries case study allowed us to face a transient problem: in order to do that, the time-dependent discharge of the cathode side of a lithium-ion battery has been modelled by a modified multiscale neural network. For this application, an autoregressive approach has been conceived and an appropriate training strategy has been tailored in order to obtain accurate fields of concentration and potential in the solid phase of the electrodes, both of which exhibit an evolution in time during the discharge process. This method turned out to be a reliable and accurate surrogate model for the prediction of discharge curves, also on new cathodes geometries not originally investigated with the physics-based CFD model.

This dissertation proves the feasibility of training robust neural networks models for different applications of flow and reactive transport in porous media. The guidelines presented in this work are meant to be employed to build accurate surrogate models for multiscale models and optimization workflows not only for porous media as proven here, but in general for complex reactive and transport systems of interest for chemical and process engineering.

Contents

List of Figures

List of Tables

Nomenclature

Roman Symbols

$\mathbf{g}$	Gravitational acceleration
$\mathbf{u}$	Superficial velocity
$\mathbf{v}$	Velocity vector
$\mathscr{D}$	Molecular diffusion coefficient
$\mathscr{L}$	Characteristic length of the fluid
A	Area of the cross-section
C	Scalar concentration
d	Characteristic size of the grains of the porous medium
d_C	Diameter of the colloid
E_{eq}	Equilibrium potential
F	Flux
i	Current flux
k	Permeability
k	Reaction rate coefficient
k_1	Inertial permeability
k_B	Boltzmann constant

K_f Filtration rate

Kn Knudsen number

L Dimension of the porous medium in the main direction of flow

N_{data} Number of samples in the dataset

p Pressure

Pe Péclet number

Q Volumetric flux of fluid

q Flux of fluid per unit area of the cross-section

R Perfect gas constant

R_{SEI} Resistance to charge transport due to SEI

Re Reynolds number

T Temperature

t_+ Transport number of the lithium ions

U_∞ Velocity of the fluid far from the collector

V_{solid} Volume of the solid phase

V_{total} Volume of a REV

w, b Weight and bias of neural networks

x Processed input features array of a batch

x' Raw input features array of a batch

D Dispersion coefficient

Greek Symbols

α Attachment efficiency

α_a Anodic transfer coefficient

α_c	Cathodic transfer coefficient
$\boldsymbol{\tau}$	Stress tensor
η	Overpotential
η	Total collector efficiency
η_0	Collector efficiency
η_B	Brownian contribution of the collector efficiency
η_G	Inertial contribution of the collector efficiency
η_I	Steric interception contribution of the collector efficiency
λ	Mean free path
μ	Dynamic viscosity
ν	Kinematic viscosity
Ω	Collector volume
ϕ	Electric potential
ρ	Fluid density
ρ_P	Collector density
σ	Electrical conductivity
σ	Standard deviation
ε	Porosity

Acronyms / Abbreviations

AM	Active Material
CBD	Carbon Binder Domain
CFD	Computational Fluid Dynamics
CNN	Convolutional Neural Network

FCNN Fully Connected Neural Network

GPU Graphical Processing Unit

HPC High Performance Computing

LIBs Lithium-ion batteries

NMC Nickel Manganese Cobalt

REV Representative Elementary Volume

SEI Solid Electrolyte Interface

TLU Threshold Logic Unit

Chapter 1

Introduction

Machine learning, in particular deep learning models, recently gained a lot of interest in the research community. The application of these techniques in physics, engineering, and, in general, in the computational field is increasingly catching up attention in the machine learning community [8]. This methodological thesis is a benchmark for the employment of deep learning models for systems of interest in chemical engineering: flow and transport in porous media, specifically filtration in packed beds, and lithium transport in lithium-ion batteries. From simple input-output models to time dependent deep neural networks, this dissertation provides the workflows, the computational details, and a critical discussion on the employment of these models in the computational chemical engineering related to porous media.

In this introductory chapter the perspectives of machine learning in chemical engineering are discussed in Section 1.1, while a literature review for the application of deep learning in porous media modelling and the thesis outline are presented in Section 1.2.

1.1 Machine learning in chemical engineering

Computational modelling is huge in chemical engineering. In the investigation of (molecular) transport phenomena a plethora of methods and tools are involved ranging from ab initio quantum chemistry methods, classical full atom and coarse-

Portions of the content of this chapter appear, in a modified form, in Marcato et al. (2023)[7]

grained molecular dynamics methods, mesoscopic methods, continuum models, among which computational fluid dynamics (CFD), chemical reaction engineering models and process models. Each of these tools is characterized by a certain degree of empiricism, which increases when moving from ab initio calculations to process modelling. The computational models available nowadays are more complex than the ones employed in the past, but it is still possible to distinguish between first-principle models and data-driven models. In first-principle models, when a certain degree of disagreement with experimental data is observed, the theory and the hypotheses are improved. In data-driven models, in case of disagreement, the dataset is improved. Indeed, also in first-principle models, there are unknown parameters, that are identified through fitting, but still the difference persists.

In this context it is also useful to reflect on why models are employed. One reason is because by building the model a deeper understanding of the problem is gained. In this first case first-principle models are usually employed. Another reason is because quantitative reliable predictions are needed for designing a process, for its optimization, or for scaling purposes. In this context the main target is an accurate prediction, not the deeper understanding of the physical problem. In this second case a data-driven model can indeed be employed. In many cases one does computational modelling for both reasons.

An example of data-driven models are machine learning models. The main issue in facing a problem with machine learning is the necessity of a large amount of data, usually referred to as “big data”. In many contexts huge datasets are available, as for example, for social media and search engines. In chemical engineering this “big data” is not always there, except if the data is produced already within a computer, as it is the case with computer simulations obtained from computational models. This is very easy to realize now, thanks to high performance computing, software orchestrators, and high throughput workflows, which allow to produce a huge amount of simulation data in a short time and at little cost.

High performance computing (HPC) represents a good reason why machine learning is now so popular. Especially with graphical processing unit (GPU) architectures, a large amount of data can be collected, stored and employed to train data-driven models. Another reason for the recent increase in popularity of machine learning is the availability of libraries that are easy to use and integrated into complex workflows. In fact, even though the fundamentals of the deep learning theory

have been introduced before the 90s, only with the advent of the CUDA platform in 2007, the training of those algorithms became computationally feasible for the practitioners of machine learning (i.e. data and computer scientists). Then, in 2015 and 2016 Google and Facebook released two open-source Python libraries: TensorFlow and PyTorch, respectively. From that moment, deep learning tools became concretely available to a wider community by greatly reducing their barrier to entry, and chemical engineers as well.

The availability of these libraries, together with an outstanding open community online and the study of deep learning, has given to traditional modelling researchers, already familiar with statistics and coding, the opportunity to move quickly through the learning curve and take an active role in the machine learning research. Open-source physics-based toolboxes (like OpenFOAM for computational fluid dynamics, LAMMPS for molecular dynamics and the different codes for discrete element methods available) made chemical engineers developers, rather than simple users. Nowadays in machine learning their contribution can go beyond being the creators of the dataset or passive end-users of the simplest codes.

The objective of the deep learning model has to be clear and bounded. In fact, the dataset creation and repartition, the training strategy and the model architecture must be set in order to target the level of generalization required. The level of generalization achieved by a machine learning model constitutes the basis of a study about its reliability when it is employed in end-applications, such as optimization or multi-scale modelling. It has been shown for some type of machine learning tools, such as neural networks, that tailoring the model in terms of architecture, loss function and input features to the prediction objective can improve its generalization efficacy and reduce significantly the amount of data required during training. This activity can be performed only knowing in depth the physics of the problem, that is why the research process cannot be cleanly split into chemical engineering competence, and data science competence.

Machine learning can have an impact on chemical engineering in a variety of fields, such as:

- **Molecular Dynamics force fields.** Ab initio quantum chemistry methods are very accurate, but the codes do not scale as efficiently as classical molecular dynamics codes. It is possible to run a large number of small ab initio quantum chemistry calculations, to train an artificial neural network and generate

interatomic potentials. This allows the investigation of large systems, for long times, with classical molecular dynamics simulations, characterized however by quantum accuracy [9, 10].

- **Multiphase CFD model closures.** Similarly to what just discussed, in the simulation of polydisperse multiphase systems with CFD, direct numerical simulations methods, with fully-resolved interfaces, are very accurate, but extremely computationally expensive. On the contrary, modelling approaches in which the interface is not resolved (such as Lagrangian tracking, or Eulerian multi-fluid models), are extremely cheap and can be employed to investigate large systems for long times. Also in this case, direct numerical simulations can be used to formulate closures for less detailed simulations. One typical closure is for the drag force, or for dense polydisperse multiphase systems [11, 12].

- **Multiscale modelling.** Most of the systems of interest in chemical engineering have an intrinsic multiscale nature. It is necessary to rely on constitutive equations to consider lower-scale phenomena. Constitutive equations are a simplification of the real microscale behavior, and they fail outside of the range of their validity hypotheses. Detailed microscale simulations could be called by macroscale simulations in order to consider the actual complexity among scales, however this approach can be unfeasible when physics-based models are computationally expensive. In this case, machine learning models can be trained on a dataset of microscale simulations to convey the information to a higher scale. This approach has been proposed for complex rheologies [13] and porous media problems [14–16].

- **Identifying physics-based model parameters.** In many cases in continuum models the transport equations include terms not directly representing the physical reality of the phenomenon at the molecular scale, but an appropriately calibrated model: for example in the case of population balance equations [17], this happens for the parameters of the aggregation or breakage kernels. Some of the parameters are often only obtainable by repeated runs of CFD simulations and an iterative comparison with available experimental data. In this case, if one is able to train a reasonably accurate data-driven model which is able to give a fast response, it is possible to conduct this parameter optimization

much faster, and create a finely-tuned CFD model which is coherent with the experimental ground truth.

- **Process optimization and control.** When an accurate data-driven model of the process is available (such as a neural network surrogating the full simulation results [18]), it is possible to use it to *optimize the design* of a process [19, 20], be it its geometrical definition or the operating conditions at which the process performance is maximum. Moreover the fast response model that has been used upstream can also be used downstream for its in-line control, where advanced model predictive control system can then be developed to smoothly and accurately keep a process inside of the desired parameters of operation.

- **Simulation on the loop: building a data-driven model with sparse experimental data.** One of the key features of machine learning is the necessity of a dataset with which to train the model. Experimental points are usually limited to small campaigns of experiments and the operating conditions are usually explored in a rigid structured way. In this case the dataset can be enriched by physics-based simulations, validated on the available experimental points. This validated computational tool allows a flexible operating conditions exploration and the possibility to obtain a dataset wide enough for the generalization purposes of the machine learning model. Finally, the obtained data-driven model can be easily called to address new experimental campaigns.

- **Artificial intelligence in the laboratory.** Machine learning models can find employment in laboratory routines, when there is lack of fast models for decision making. In this case computationally expensive simulations can hardly be used, so fast response models can find application. First works in this perspective are found in the energy storage field [21].

Given all of these examples, the place where machine learning and chemical engineering modelling meet can be considered as a valuable ground for research and investigation.

1.2 Deep learning models for flow and transport in porous media

Amongst the points just presented, in this dissertation we focus on the application of machine learning models for the construction of surrogate models of flow and transport in porous media. The study of flow and transport in porous media is of the utmost importance in chemical engineering [22]. Many fields of application in the traditional chemical industry are impacted by the understanding of these phenomena, such as packed beds catalytic reactors [23, 24] and filtration devices [25, 26]. Other examples can be found in large scale environmental applications, such as groundwater extraction and remediation [27, 28], or enhanced oil recovery [29].

Apart from the importance of porous media in these established sectors, it is apparent how the study of transport phenomena in dispersed and random structures will play an increasingly important role in the transition towards a sustainable and carbon neutral economy. The study of flow in porous media is essential for the modelling of carbon capture and storage processes [30, 31]. One other prominent example of the pivotal importance in the energy transition is the growing attention to the study of energy storage systems, and specifically batteries (both in their current Li-ion and future beyond-lithium incarnations). Indeed, electrodes, in their microscale reconstruction, are modelled as porous media impregnated by the electrolyte. Both long-term safety of battery operation, energy density and battery cycle life, (which constitute the main areas of improvement inasmuch their limits constitute the main barriers to overcome towards a more pervasive grid electrification) are being investigated by means of detailed studies of the transport of electrochemical species inside the electrodes [32–34].

Traditional modelling is based on fundamental laws obtained from analytical and experimental methods resulting in expressions of dimensionless numbers and macroscale parameters, that characterize the geometry of the porous media. The confidence of these laws is affected by uncertainties due to the non-linear correlation between the defining features of the geometries and the resulting macroscale parameters, respectively the input and output features of these models [35]. As a way to improve these models confidence, and spurred on by the increasing availability of computational resources, in recent years a host of microscale models

have been developed that completely describe porous media behaviour at the pore scale [36–39].

Computational fluid dynamics is a well established modeling approach for flow and transport in porous media. Depending on the complexity of the physical problem, and on heterogeneity and size of the domains, both meshing and the subsequent simulation can be highly time consuming and computationally expensive to run. For this reason the use of high-performance computing (HPC) systems is often required to solve the simulations in parallel.

When numerous simulations are needed (like in optimization algorithms and in multiscale modeling [13, 40, 41]) or when real-time predictions are necessary (like in-line plant control [42]), it would be extremely useful to have fast and accurate models to predict the system behaviour (or performance) based on locally changing microscopic-scale conditions. Given the clear multi-scale nature of porous media transport phenomena, historically a lot of effort went into the development of such tools, differing widely in approach, from theoretical upscaling approaches to the development of phenomenological constitutive equations.

An example of the first class of solutions, aside from well known averaging procedures [43] is the analytical development of models by means of asymptotic homogenization, which has enjoyed great success in obtaining closed forms of macroscale transport equations [44, 45] but which suffer (due to the complicated analysis involved) in limits to its applicability both in treatable geometrical structures [46] and transport regimes [47]. Other approaches are based on building constitutive equations from both empirical or computational results and while they have been vastly employed in many different fields [48–50], these relations are still prone to fail when the geometries become random [51] and are hardly parametrizable [52]. Many solid research works have thus focused on simpler models [53, 54] and even slightly more complicated pore/collector geometries have been found to noticeably complicate things in terms of obtaining upscaled laws [55].

Then, one alternative to the mentioned approaches, which is gaining momentum in the last few years, is to train specific neural network models in order to obtain these fast response surrogate models. However, as of now, problem-specific design choices have to be made to identify the most suitable neural network architecture for each problem to be solved. In the porous media research, different kinds of neural networks have been trained on datasets of physics-based simulations. Fully connected

neural networks (FCNN) have been employed for the prediction of integral quantities from effective hand-picked features [56–58, 16]. These techniques are effective and their training is easy to carry on, but the resulting model is very sensitive to the choice of input parameters, since the domain is described via upscaled parameters that have no way of relaying spatial heterogeneity.

The increasing availability of GPUs and open-source deep learning libraries have made the training of more complex neural networks computationally feasible. The use of deep learning techniques such as convolutional neural networks (CNN) ease the choice of the right integral descriptors of the porous media since the entire system geometry is fed to the network as an image, and the network autonomously detects the most effective features for the prediction of the objective output. It has to be remarked that this does not result just in an automatic choice of relevant integral features, but in a trained network that operates by "seeing" the system geometry and is then able to make predictions based on the experience thus acquired [59, 60, 14].

CNN with encoding and decoding architectures have been used to train surrogate models able to predict the flow field in different microscale porous media systems [61–63] and for uncertainty quantification in macroscale subsurface applications [64–68] Multi Scale Neural Network (MSNet) [69] came out to be a preferred alternative to the previous ones, from both the computational point of view and, notably, its generalization capability. In fact, it is possible to train this network with larger geometric samples than what more classical approaches allow, which is fundamental when dealing with representative elementary volumes of heterogeneous geometries and/or complicated transport phenomena.

Nonetheless, as the main effort in the last years was addressed to the development of neural networks models for the prediction of the permeability, or the microscale prediction of flow fields, little was done to expand these methodologies to more complex physical systems, which rely on the heterogeneity of both geometries and operating conditions, and are of common experience in the chemical engineering field.

In this dissertation flow and transport in porous media are modelled by means of CFD tools and neural networks models are trained for the sake of obtaining accurate and fast surrogate models. Two case studies are investigated: the filtration of colloids by porous media and the microscale lithium reactive transport in cathodes of lithium-ion batteries. Different neural networks models have been trained and tested as

surrogate models of the filtration process in porous media: at first a simple fully connected neural network has been trained for the prediction of integral descriptors, then the use of convolutional neural networks is tested in order to avoid the choice of the input integral parameters, and finally a complete surrogate of the microscale models has been proposed by using multiscale convolutional neural networks. For the lithium-ion batteries case study instead, an autoregressive multiscale convolutional neural network has been conceived in order to obtain a time-dependent surrogate model.

So, as a brief overview and reading guide, this dissertation is organized as follows.

Chapter 2 gives an overview of the theoretical background of flow and transport in porous media. In particular, both microscale governing equations of transport and macroscale approaches are presented for the single fluid phase flow in laminar conditions. Then, the basic theory of colloid filtration is proposed. After, the microscale governing equations of charge and mass balance for lithium-ion batteries are reported and discussed.

In Chapter 3 a review on neural networks (FCNN and CNN) theory is proposed along with the basics of the training procedure.

In Chapter 4 the workflow for the construction of data-driven models is presented and applied for the training of fully convolutional neural networks for the prediction of integral quantities, namely the permeability and the filtration rate in bi-dimensional porous media.

In Chapter 5 the construction of a three-dimensional dataset of geometries and subsequent simulations are detailed. Convolutional neural networks are trained on the dataset for the prediction of the permeability, and the filtration rate. The results of FCNN and CNN are compared and discussed.

In Chapter 6 a multiscale convolutional neural network has been adapted and modified to predict the microscale concentration fields in filtration applications. The choice of the most appropriate input features is explored in detail in the chapter. This work has been done in collaboration with the research group of Prof. Maša Prodanović (University of Texas at Austin, USA).

In Chapter 7 a transient problem is faced in a dataset of discharge simulations of lithium-ion batteries, and an autoregressive multiscale neural network is proposed as

a surrogate model. The comparison between autoregressive and classic approch is proposed and discussed. This work has been done in collaboration with the research group of Prof. Alejandro A. Franco (Laboratoire de Réactivité et Chimie des Solides - Amiens, France).

Chapter 2

Theoretical background of flow and reactive transport in porous media

As mentioned in the Introduction, data-driven models obtained by machine learning techniques can be very useful when a fast response model is required. The training of these models should be performed on a dataset tailored on the final application in order to obtain a model capable of generalization toward new cases, so a deep understanding of the physics is essential to this end. To reach this detailed understanding, and to create a large enough dataset for the training, accurate CFD simulations can be performed. In this chapter the theoretical background of flow and transport in porous media is summarized, both to describe the equations solved in the CFD simulations, and also in regards to the wider context of these equations related to other oft-used laws for porous media analysis. In the first Section 2.1 the basic definitions of porosity, representative elementary volume, microscale and macroscale approaches are provided. Then, the theory and the governing equations of momentum transport are summarized for both the microscopic and the macroscopic scale, in Section 2.2. A review regarding the transport of a scalar quantity follows, with reference to the colloid deposition theory, in Section 2.3. Finally in section 2.4, the basics governing equations of transport with electrochemical reaction are introduced with reference to the lithium-ion batteries application.

2.1 Porous media modelling fundamentals

2.1.1 Porosity and representative elementary volume

A porous medium is a portion of space filled by at least two phases: a solid and a non-solid phase. The domain filled by a non-solid phase is the void space and is constituted by a collection of pores. In the work conducted in the thesis project a single-phase fluid flows in the void space and the pores are all interconnected (neither unconnected nor dead-end pores are present). Thus the porosity, ε, can be defined as:

$$\varepsilon = 1 - \frac{V_{solid}}{V_{tot}}, \tag{2.1}$$

where V_{solid} is the volume of the solid and V_{tot} is the total volume of a representative elementary volume (REV) portion of the domain. A sketch of this representative elementary volume can be found in Fig. 2.1.

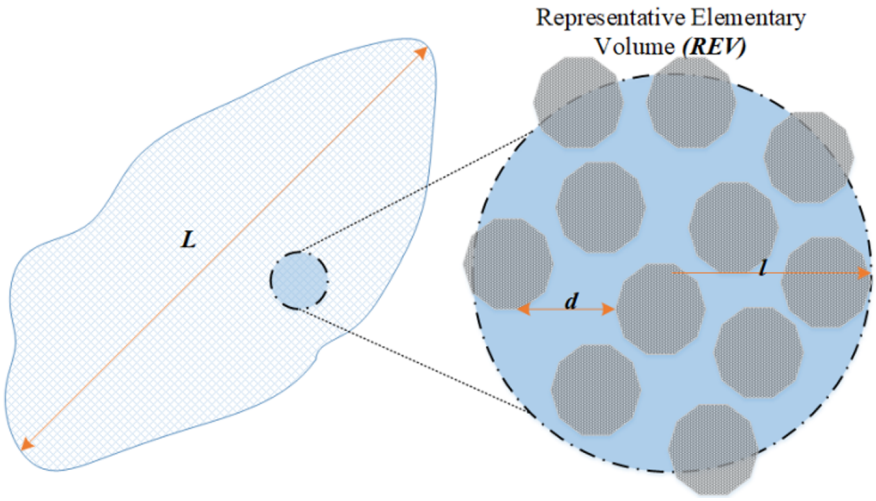

Fig. 2.1 Sketch of a representative elementary volume of a porous medium [1].

A REV is the smallest portion of the domain whose properties (i.e. the porosity in a geometrical analysis) are statistically representative of the entire domain. Let's consider a point in the porous medium domain, P, and ΔV a portion of volume centered in P [2], the porosity of the sub-domain i is:

$$\varepsilon_i = 1 - \frac{\Delta V_{solid,i}}{\Delta V_{tot,i}}, \tag{2.2}$$

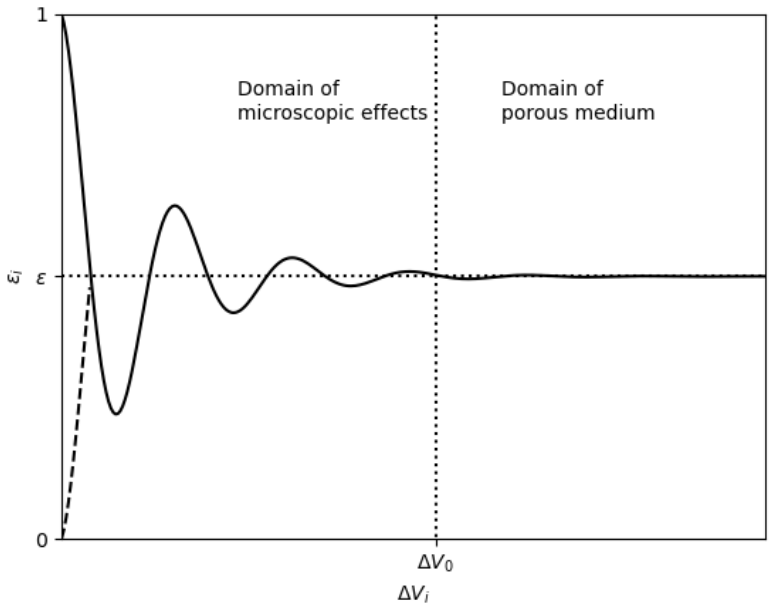

Fig. 2.2 Definition of porosity and representative elementary volume [2].

The porosity depends on the volume of the sub-domain as it is qualitatively outlined in Fig. 2.2.

When the volume, $\Delta V_{tot,i}$, approaches 0, the porosity is 0 or 1 depending on the phase (solid or non-solid) in which P is placed. Increasing the volume, $\Delta V_{tot,i}$, the porosity, ε_i, oscillates until a constant value is reached: the volume of the REV is obtained, ΔV_0. This behaviour holds for homogeneous porous media, which is the case of this work, so when the porous medium properties do not significantly change by considering different macroscopic samples of it.

2.1.2 Nomenclature for multiscale analysis

In the modelling terminology of transport phenomena, the term microscale (or nanoscale) is usually employed when a fluid is modeled at its molecular level, in contrast to the term macroscale, used when the fluid is modelled as a continuum. It is possible to model a fluid as a continuum if its Knudsen number, Eq. 2.3 is lower

than 0.01. It is defined as follows:

$$Kn = \frac{\lambda}{\mathscr{L}}, \tag{2.3}$$

where λ is the mean free path of the fluid molecules, $\mathscr{L}$ is the representative length of the system.

The terminology usually employed in the modelling of porous media systems is different, in fact, another scale should be taken into account: the scale of the actual process or system under investigation (the macro scale), whose spatial scale may be separated from the continuum scale by orders of magnitude. Therefore in this field, the microscale is the scale of the pores (hence, pore scale) and not the molecular scale: in this definition of microscale the continuum hypothesis holds and the geometry of the porous medium is explicitly employed as a boundary condition for the solution of the transport equations. In the macroscale instead, a coarsened representation of the porous medium is considered averaging the fluid and solid properties above the REV.

2.2 Single phase flow in porous media

2.2.1 Microscale models - Equation of motion

Under the continuum hypothesis the fluid flowing in the pore space of a porous medium follows the equation of motion [70]:

$$\frac{\partial}{\partial t}\rho\mathbf{v} = -\left[\nabla\cdot\rho\mathbf{v}\mathbf{v}\right] - \nabla p - \left[\nabla\cdot\boldsymbol{\tau}\right] + \rho\mathbf{g}, \tag{2.4}$$

where the first term is the rate of momentum increase per unit volume (ρ is the density and $\mathbf{v}$ is the velocity vector), the second term is the rate of momentum addition by convection per unit volume, the third and fourth terms are the rate of momentum addition by molecular diffusion per unit volume (p is the pressure, $\boldsymbol{\tau}$ is the stress tensor), the fifth term is the external force per unit volume ($\mathbf{g}$ is the acceleration vector, that can be the gravitational acceleration).

The equation of motion becomes the well-known Navier-Stokes equation under the hypotheses of Newtonian fluid and constant density:

$$\rho \frac{\partial}{\partial t}\mathbf{v} = -\rho \left[\nabla \cdot \mathbf{vv}\right] - \nabla p - \mu \left[\nabla^2 \mathbf{v}\right] + \rho \mathbf{g}, \tag{2.5}$$

where μ is the dynamic viscosity of the fluid.

In this work, no external forces have been taken into account, and, since it was of interest having a steady state solution, the accumulation term has not been taken into account. Finally, the Navier-Stokes equation that was solved (together with the continuity equation) is:

$$\nabla \cdot \mathbf{vv} + \nabla \frac{p}{\rho} + \nu \left[\nabla^2 \mathbf{v}\right] = 0. \tag{2.6}$$

where ν is the kinematic viscosity of the fluid.

In microscale simulations of flow in porous media the porous structure geometry is a boundary for the solution of the motion equation. A no-slip velocity on the solid surface is usually set given its impermeability to fluid penetration. Among the possible boundary conditions, it is possible to set either the inlet velocity, or the pressure drop across inlet and outlet. The main advantage of using a pressure drop as boundary condition is that a non-constant velocity profile will arise at the inlet boundary which is more realistic for the bulk portion of a porous medium, as it corresponds to a developed flow profile.

Even though the motion equation can be numerically solved, the spatial discretization required by those methods make it difficult to solve large porous media domains. What is usually done with the computational resources available nowadays is to model at the pore scale a bulk core of the porous structure. It is advisable to solve the motion equation on a REV in order to obtain statistically meaningful parameters as a result of the post-processing of the microscale solution.

The most common methods for the solution of the motion equation are the ones of common experience in the fluid dynamics modelling, i.e. finite element and finite volume methods, the latter of which is the one employed in this work. A body fitted mesh is needed as computational grid in those methods, but, being the specific surface of porous media large the number of cells/elements can rapidly increase when refinements on the walls are applied. Lattice Boltzmann simulations are thus

also widely employed, even though the motion equation is not directly solved. The main advantage of this method is the use of a regular lattice as a computational grid. The lattice is Cartesian, so real segmented (i.e. voxelized) images of porous media (from tomographies or electron microscopy) can be employed without the need for a meshing step. This obviously also contains a drawback, as the real geometry may indeed have smooth surfaces, which a non-cartesian mesh can effectively help to describe better and result in a discretization closer to the real geometry.

2.2.2 Macroscale models - Permeability

The first macroscale law that has been introduced for the motion of fluids in porous media is Darcy's law which correlates the flux with the pressure drop across a porous medium by means of a coefficient of proportionality [71]:

$$q = \frac{k}{\mu}\frac{\Delta p}{L}, \tag{2.7}$$

where q is the flux per unit area of the cross section (with units in $m \cdot s^{-1}$), L is the length of the porous medium in the main direction of flow, and k is the permeability. Darcy's law was at first demonstrated experimentally, but it is possible to derive it from first principles by averaging the Navier-Stokes equation (Eq. 2.6, or rather Stokes' equation) over the cross-section [2]. Darcy's law can be extended to three-dimensional flow in porous media as well:

$$\mathbf{u} = -\frac{k}{\mu}\nabla p, \tag{2.8}$$

where $\mathbf{u}$ is the superficial velocity vector. The permeability remains a constant if the porous medium is homogeneous and isotropic, otherwise, in the case of anisotropic porous media it becomes a tensor.

At this point it is relevant to distinguish between the effective velocity of the fluid, $\mathbf{v}$, the flux per unit area, q, and the superficial velocity, $\mathbf{u}$. The effective velocity of the fluid is the velocity of the fluid at the microscale, as it results from the solution of the motion equation 2.4. The flux per unit area of the cross-section is the ratio between the volumetric flow rate, Q, and the area of the cross section in the main

flow direction, A, thus:

$$q = \frac{Q}{A}. \tag{2.9}$$

The superficial velocity, $\mathbf{u}$ is the macroscale velocity of the fluid when the domain is considered as homogeneous and no distinction between solid and fluid phase is made. The magnitude of the superficial velocity, $|\mathbf{u}|$, is equal to the flux per unit area. The volume average (over the fluid domain) of the effective velocity and the superficial velocity are related by the porosity of the porous medium, in fact, the volume average effective velocity is the volumetric flux divided by the portion of cross-section area in the fluid domain, so

$$v = \frac{Q}{A\varepsilon} = \frac{q}{\varepsilon}, \tag{2.10}$$

in fact, it is possible to demonstrate that the volumetric porosity, ε, is equal to the average areal porosity for a REV [2].

Although the most common way of calculating the permeability is by experiments or flow simulations, the permeability depends on the porous media geometry, in fact, relations have been proposed to evaluate it from geometrical parameters. The most common of these is the Kozeny-Carman equation:

$$k = Cd^2 \frac{\varepsilon^3}{(1-\varepsilon)^2}, \tag{2.11}$$

where C is a constant determined experimentally equal to $\frac{1}{150}$ [2], and d is the mean diameter of the grains/particles. For porous structures that cannot be modelled as sphere packings the law is not effective. Merging Eq. 2.7 and Eq. 2.11 the Blake-Kozeny equation for the pressure drop evaluation in porous media is obtained:

$$\frac{\Delta p}{L} = 150 \frac{\mu q}{d^2} \frac{(1-\varepsilon)^2}{\varepsilon^3}. \tag{2.12}$$

The Blake-Kozeny equation can be derived with the well-known tube bundle theory. The reader is referred to the chapter 4 and 5 of Bear (1988) [2] for an overview of the possible derivations of those laws.

Darcy's law holds for laminar flow in porous media, in fact, when the Reynolds number exceeds 1-10 the transition of flow regime starts, above 60-150 turbulence is

developed. The Reynolds number for flow in porous media can be defined as:

$$Re = \frac{\rho q d}{\mu}, \tag{2.13}$$

where d is a characteristic dimension of the porous medium domain - for spherical grains it can be their average diameter. For transition and turbulent flows Darcy-Forchheimer's law can be employed [72]:

$$\nabla p = -\frac{\mu}{k}\mathbf{u} - \frac{\rho}{k_1}\mathbf{u}\mathbf{u}, \tag{2.14}$$

where k_1 is the inertial permeability.

As for the Blake-Kozeny equation, it is possible to derive a similar equation from the bundle of tubes theory to model the pressure drop across porous media in turbulent regime, the result is the Burke-Plummer equation:

$$\frac{\Delta p}{L} = \frac{7}{4}\frac{\rho q^2}{d}\frac{1-\varepsilon}{\varepsilon^3}. \tag{2.15}$$

To take into account the transition regime it is possible to sum up the two contributions of Eq. 2.12 and Eq. 2.16 into the Ergun equation:

$$\frac{\Delta p}{L} = 150\frac{\mu q}{d^2}\frac{(1-\varepsilon)^2}{\varepsilon^3} + \frac{7}{4}\frac{\rho q^2}{d}\frac{1-\varepsilon}{\varepsilon^3}, \tag{2.16}$$

which can be rearranged into the dimensionless groups, Δp^*, and Re^*:

$$\frac{\Delta p}{L}\frac{d}{\rho q^2}\frac{\varepsilon^3}{1-\varepsilon} = 150\frac{\mu}{d}\frac{(1-\varepsilon)}{\rho q} + \frac{7}{4}, \tag{2.17}$$

$$\Delta p^* = \frac{150}{Re^*} + 1.75 \tag{2.18}$$

2.3 Transport and deposition in porous media

In this section the study of transport of a scalar quantity (i.e. a concentration) in a porous medium is detailed. The application of the modelling procedure is the filtration of colloids in porous media in laminar conditions. Both the modelling at

the microscale and macroscale are presented together with the theoretical hypotheses of validity.

2.3.1 Microscale models - Advection-diffusion equation

In a filtration process particles dispersed in a solution deposit on a collector, the dynamics of the transport is governed by three mechanisms:

- interception - a particle collides with the collector because of its steric hindrance.
- sedimentation - a particle whose density is higher than the fluid density follows a different trajectory compared to the streamlines because of the gravitational force field.
- diffusion - this mechanism occurs when the particles are subjected to Brownian motion.

Depending on the operating conditions and on the particles/fluid nature the relative predominance of the mechanisms can change, and with it the most suited modelling technique [27].

In this work the particles to be filtered are colloids, whose density is about $5000 - 10000$ kg/m^3 and whose characteristic length is smaller than 1 μm. For this system it is possible to calculate the Stokes number, that is defined as the ratio between the characteristic relaxation time of the particle and the one of the fluid. In laminar conditions it reads as follows:

$$St = \frac{\text{relaxation time of the particle}}{\text{relaxation time of the fluid}} = \frac{\rho_P d_C^2}{18\mu}\frac{U_\infty}{d}, \tag{2.19}$$

where ρ_P is the density of the colloidal particles, U_∞ is the fluid velocity far from the collector. The resulting Stokes number is lower than 1, thus the relaxation time of the particle is lower than the one of the fluid, so it is possible to assume that the particles move along the streamlines of the fluid at the same velocity of the fluid. This condition allows the modelling of the transport of the colloid as a scalar in an Eulerian framework. Under hypothesis of low concentration in the fluid

the diffusion of the colloid can be modelled by the Fick's law, thus the advection diffusion equation reads as follow:

$$\frac{\partial C}{\partial t} = \nabla \cdot (\mathscr{D} \nabla C) - \mathbf{v} \cdot \nabla C, \tag{2.20}$$

where C is the concentration and $\mathscr{D}$ is the diffusion coefficient of the colloid, which can be evaluated by the Einstein equation [73]:

$$\mathscr{D} = \frac{k_B T}{3\pi\mu d_C}, \tag{2.21}$$

where k_B is the Boltzmann constant, T is the temperature and d_C is the colloid size.

Besides the considerations on the Stokes number, it is necessary to take into account the short-range interactions of the colloid particles while approaching the collector. In fact, the particles in proximity of the collector are subjected to hydrodynamic retardation as a result of the increase of the drag force close to the collector surface, thus the velocity of the particles decreases and their paths can detach from the streamlines [74]. Another consequence is the reduction of the colloids diffusivity which becomes a function of the wall distance [75, 76].

Absent other effects, the particles would not reach the collector because of the viscous repulsion [77], but the Smoluchowski-Levich approximation can be applied to this case study [78]. This approximation states that the hydrodynamic retardation experienced by the colloid particles approaching the solid wall is balanced by the London attractive forces [79].

Moreover in this work a chemical condition favorable to attachment is considered, thus all the mesoscale forces of attraction and of repulsion are considered in a single unitary attachment efficiency. This is the most used assumption when dealing with modelling of this kind, and goes under the name of the *clean bed filtration* assumption, as it correctly describes the early stages of the filtration process when the solid load is very low and the collecting solid medium is "clean" resulting in heightened and more effective deposition/trapping efficiency. This results in a constant diffusion coefficient in space and in a perfect sink boundary condition at the collector surface, which we implement as a homogeneous Dirichlet condition ($C = 0$). It is worth noticing that the solution of the advection-diffusion equation with this boundary

condition can also represent an instantaneous heterogeneous reaction on the grains surface.

The solution of the advection-diffusion equation at the microscale is useful to evaluate the source term for macroscale models of the filtration process. This process is modelled as a reactive term so a filtration/reaction term is needed to express this at the macroscopic scale, where the distinction between solid and fluid is lost, and thus the analytical definition of a surface on which the heterogeneous reaction happens. In the following section a quick literature review is proposed to this end.

2.3.2 Macroscale models - Filtration rate

At the macroscale the partial differential equation that describes the transport and filtration of the colloid is the advection-dispersion-reaction equation:

$$\frac{\partial C\varepsilon}{\partial t} = \nabla \cdot (\varepsilon \mathrm{D} \nabla C) - \mathbf{v} \cdot \nabla C - K_f C\varepsilon, \tag{2.22}$$

where D is the dispersion coefficient, and K_f is the rate of filtration.

In the literature the first macroscale descriptor of the amount of colloid filtered by the collector was the total collector efficiency [27, 80], η, defined as:

$$\eta = \alpha \eta_0, \tag{2.23}$$

α is the attachment efficiency, ranging between 0 and 1 (considered unitary in this work), and a collector efficiency η_0. A unitary α means that all the particles colliding with the collector remain attached, a unitary η_0 would mean that all the particles flowing towards the collector collide with it. The collector efficiency is made by the contributions of the different filtration mechanisms (Brownian diffusion, η_B, sterical interception, η_I, sedimentation, η_G), and it is possible to assume that these contributions are additive [79, 27], so:

$$\eta_0 = \eta_B + \eta_I + \eta_G. \tag{2.24}$$

In the literature much effort was spent in finding analytical expressions for these efficiency terms. At first Levich [78] conceived an expression for the Brownian term under the above-mentioned Smoluchowski-Levich approximation for a single

spherical collector:

$$\eta_B = 4.04 Pe^{-\frac{2}{3}}, \tag{2.25}$$

where Pe is the Péclet number:

$$Pe = \frac{qd}{\mathscr{D}}. \tag{2.26}$$

Different relations are presented in literature for the other terms and for multiple spherical collectors. Relations to link the filtration efficiency and the filtration rate of Eq. 2.22 have been proposed as well [81, 52, 82].

The main disadvantage of using the filtration efficiency theory is its scale inconsistency, as demonstrated by Boccardo et al. [83]. They proposed a direct definition of the filtration rate by volume averaging the microscale advection-diffusion equation, resulting in:

$$K_f = \frac{F_{tot}^{in}(C) - F_{tot}^{out}(C)}{\Omega \langle C \rangle}, \tag{2.27}$$

where F is the total (advective and diffusive) flux, Ω is the total volume of the macroscopic porous medium, $\langle C \rangle$ is the volume averaged concentration. As detailed in the work cited [83] this definition of the filtration rate, unlike η, can be used directly as a reaction rate coefficient in a macroscopic scale form of the advection-diffusion-reaction equation (Eq. 2.22) and will return scale-consistent results over a wide range of Péclet numbers, whereas for diffusion-dominant problems (low Pe) or certain computational model configurations classic η estimations will be incoherent.

This dissertation wants to offer a deep-learning based tool for the reliable surrogation of pore-scale modelling with the widest possible applicability, from large ranges of operating conditions (Péclet numbers) to different sizes of geometric models. This objective calls for a robust and automatic estimation of macroscopic transport and reaction features, and as such this definition of K_f was preferred, and will be used throughout the following chapters when dealing with surface reaction/deposition.

2.4 Electrochemical reactions in porous media: the Li-ion batteries case study

Lithium-ion batteries (LIBs) are electrochemical systems employed as electric energy storage devices. LIBs are characterized by high specific energy and power - essential

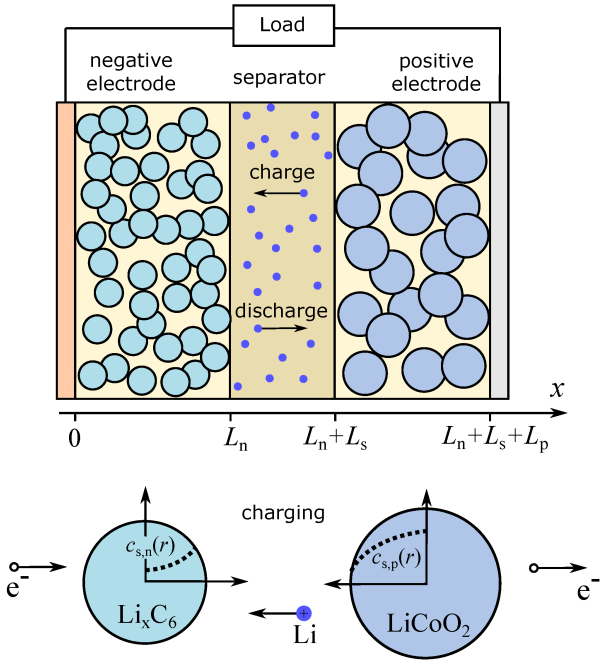

Fig. 2.3 Lithium-ion battery sketch. Image from [3].

for portable devices applications - high capacity and cyclability compared to other kinds of batteries [84]. A simplified representation of a LIB is shown in Fig. 2.3. During discharge cycles under galvanostatic conditions, the lithium ions stored in the anode de-intercalate from the electrode to the electrolyte, then they reach the cathode/electrolyte interface, and intercalate into the cathode electrode. Meanwhile the electrons flow from the anodic current collector to the cathodic current collector, because of the oxidation of the anodic material and the reduction of the cathodic material.

State-of-the-art cathodes for lithium ion batteries are made by nickel manganese cobalt (NMC) oxides, and anodes are made by graphite, so the reactions are here summarized in reference to these materials. The half-reaction taking place at the NMC cathode is:

$$\mathrm{Li_{1-x}MO_2 + xLi^+ + xe^- \rightleftharpoons LiMO_2} \tag{2.28}$$

Lithium ions oxidize the transition metal, the cobalt oxides, thus the lithium cobalt oxide is generated in the discharging mode, while the opposite reducing reaction takes place in charging mode. The half-reaction taking place at the graphite electrode

is:

$$Li_xC_6 \rightleftharpoons C_6 + xLi^+ + xe^- \tag{2.29}$$

So the full reaction is the following:

$$Li_xC_6 + Li_{1-x}MO_2 \rightleftharpoons C_6 + LiMO_2 \tag{2.30}$$

during discharge the reaction takes place from left to right, and it takes place from right to left when the battery is charging.

The modelling of batteries is crucial to ensure their optimal usage in terms of safe charging and discharging cycles; beyond their use for the continuous improvement of battery management systems, these models are also essential in researching new types of batteries [85], e.g.: new chemistries, new electrode production processes, and so on. Accurate multiscale models can help researchers to understand the effect of operating conditions on the battery performance, as well as the impact of electrode properties, or the physics beyond degradation phenomena. The main objective in this research field is to exploit models and experiments in synergy to speed up the discovery of new batteries and the understanding of the degradation phenomena that impact the life cycle of state of the art batteries [86].

Pseudo 1D and 2D models [87] were developed for the electrochemical modelling of batteries and can be easily employed since they don't require a high computational power. Many open-source implementations can be found, for example the Python library PyBaMM [88] which implements the well-known Newman model [89]. Full microscale models have been recently developed thanks to the increasing computational resources available nowadays: these 4D models (3D microscale geometry plus time dependency) do not require integral geometrical descriptors of the electrodes since the charge and transport equations are solved employing the precise microscale geometry of the electrode-electrolyte interface as boundary [90]. Pore network modelling has also been applied recently for the simulation of LIBs [91]. Geometric characterization of the electrodes is necessary in these modelling methodologies: integral parameters for pseudo 2D models, and the entire geometry in 4D models. In literature both *in-silico* reconstruction [92] and digital images were employed to this end [93].

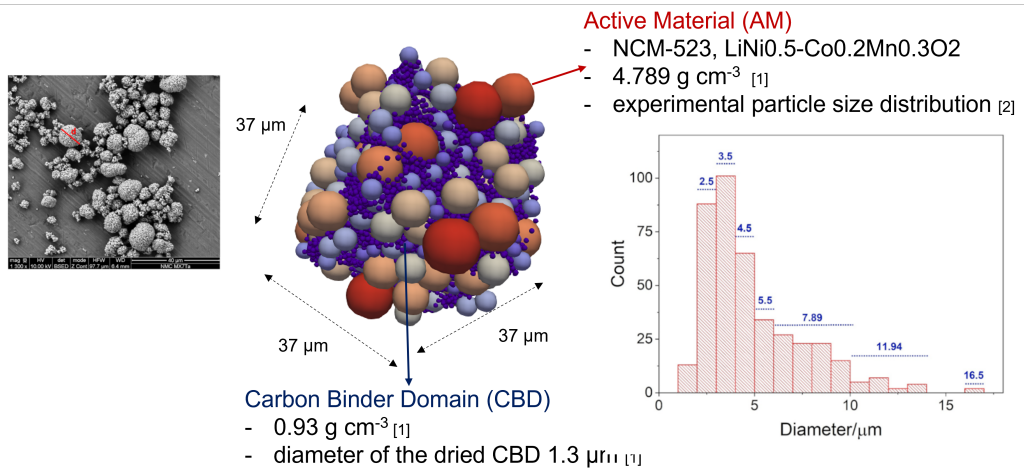

Fig. 2.4 Cathode structure reproduced by DEM. On the right the particle size distribution of the AM, and the features of the cathode [4].

2.4.1 Electrodes porous structure

Electrodes are porous media made by different solid phases immersed in the electrolyte. The components of an electrode are the active material (AM), which takes part into the electrochemical reaction, the carbon, which ensures electrical conduction throughout the electrode, and the binder, that binds the components together into a stable structure. The active material granulometry ($\sim 1-10\ \mu$m characteristic size of AM particles) constitutes a phase easily distinguishable from the carbon ($\sim 50-150$ nm) and the binder components, which instead for modelling purposes constitute a single phase called carbon binder domain (CBD) [94]. In Fig. 2.4 a cathode structure has been reproduced in-silico by means of a discrete element method (DEM), detailed in Appendix A.

2.4.2 Microscopic transport equations

In this work a dataset made by microscale simulations of half-cells (cathode side) has been employed for the training of machine learning models. In this section the transport equations numerically solved to this end are summarized. The transport equations of mass and charge balance must be solved in three domains, Fig. 2.4: AM, CBD, and electrolyte.

In the electrolyte the charge conservation reads as follows [95]:

$$\begin{cases} \nabla \cdot i_l = 0, \\ \\ i_l = -\sigma_l \nabla \phi_l + \dfrac{2RT\sigma_l}{F}(1-t_+)\left(1+\dfrac{\partial(\ln f_\pm)}{\partial(\ln C_l)}\right)\nabla \ln(C_l), \end{cases} \tag{2.31}$$

where i_l is the ionic flux in the electrolyte, σ_l is the electrical conductivity of the electrolyte, ϕ_l is the electric potential in the electrolyte, F is the Faraday constant, R is the perfect gas constant, t_+ is the transport number of the lithium ions, $f_\pm$ is the mean molar activity coefficient, C_l is the lithium ions concentration in the electrolyte. The first term is the charge conservation equation, where $\nabla \cdot i_l$ is null because of the cell neutrality. The current density is made by two terms, in fact, ions migrate because of the difference in electric potential, whose contribution depends on the electrical conductivity of the electrolyte, and because of the concentration gradient, i.e. the diffusion in the electrolyte, which is related to the diffusion coefficient of the ions by the transference number. The activity coefficient takes into account the interactions between ions in the solution, in fact, the concentration of the ions is not compatible with a dilute approach, thus the concentrated solution theory applies [96].

The transport equation for the concentration of lithium ions within the electrolyte can be expressed as

$$\frac{\partial C_l}{\partial t} + \nabla \cdot \left(-\mathscr{D}_\mathrm{l} \nabla C_l + \frac{i_l t_+}{F}\right) = 0. \tag{2.32}$$

In the NMC portion of the electrode the charge conservation equation is the Ohm's law in steady state conditions:

$$\nabla \cdot (\sigma_{s,\mathrm{AM}} \nabla \phi_s) = 0, \tag{2.33}$$

where $\sigma_{s,\mathrm{AM}}$ is the electrical conductivity of NMC, and ϕ_s is the potential of the electrode. The mass balance inside the electrode can be modelled by Fick's law, which describes the diffusion of lithium in the NMC electrode:

$$\frac{\partial C_s}{\partial t} - \nabla \cdot (\mathscr{D}_s \nabla C_s) = 0, \tag{2.34}$$

where C_s is the lithium concentration in the NMC electrode, and $\mathscr{D}_s$ is the diffusion coefficient of lithium.

The CBD is modeled as an homogeneous porous medium domain, so the transport equations are solved in the domain without taking into account its geometrical structure and the electrolyte infiltrated, the transport properties are averaged over the entire domain, so the charge transport is modelled as follows:

$$\nabla \cdot (\sigma_{s,\mathrm{CBD}} \nabla \phi_s) = 0, \tag{2.35}$$

where $\sigma_{s,\mathrm{CBD}}$ is the electrical conductivity of the CBD, and is expressed as a function of its porosity. The mass balance in the CBD is:

$$\begin{cases} \dfrac{\partial \varepsilon_{\mathrm{CBD}} C_l}{\partial t} + \nabla \cdot \left(-\mathscr{D}_{l,eff} \nabla C_l + \frac{i_l t_+}{F} \right) = 0, \\ \\ i_l = -\sigma_{l,eff} \nabla \phi_l + \frac{2RT\sigma_{l,eff}}{F}(1-t_+)\left(1 + \frac{\partial(\ln f_\pm)}{\partial(\ln C_l)}\right) \nabla \ln(C_l), \end{cases} \tag{2.36}$$

where $\sigma_{l,eff}$ and $\mathscr{D}_{l,eff}$ are the effective electrical conductivity and lithium ions diffusivity in the CBD. They are defined as a fraction f of the electrolyte properties:

$$\mathscr{D}_{l,eff} = f\mathscr{D}_l, \quad \sigma_{l,eff} = f\sigma_l. \tag{2.37}$$

The electrochemical reaction, Eq. 2.28, is modeled at the interface by means of the Butler-Volmer kinetics equation:

$$i_{se} = kF\ C_l^{\alpha_a} C_s^{\alpha_c} (C_s^{max} - C_s)^{\alpha_a} \left[\exp\left(\frac{\alpha_a F \eta}{RT} \right) - \exp\left(-\frac{\alpha_c F \eta}{RT} \right) \right], \tag{2.38}$$

where k is the reaction rate coefficient, α_a and α_c are the anodic and cathodic transfer coefficients, and η is the overpotential which is defined as:

$$\eta = \phi_s - \phi_l - E_{eq}, \tag{2.39}$$

where E_{eq} is the electrode equilibrium potential. On the surface of Li metal, which is the reference electrode for the half-cell configuration, the contribution to the overpotential of the solid electrolyte interface (SEI) formation is taken into account

using an additional resistance term R_{SEI}:

$$\eta = \phi_s - \phi_l - R_{SEI} i - E_{eq}. \tag{2.40}$$

Chapter 3

Deep learning fundamentals

In this chapter the fundamentals of neural networks modelling are summarized. As mentioned in Chapter 1, in this thesis work neural networks have been trained to obtain data-driven models for flow and transport in porous media. Thus, the aim is to provide the reader with the nomenclature that will be employed throughout the dissertation to describe those models. Specific details of the neural network architectures, and insights in the training procedure are delivered in the computational details sections of Chapters 4, 5, 6, 7. In the following sections the workflow to build a data-driven model and the classification of machine learning models are presented in Section 3.1, after the main features of fully connected neural networks (FCNN) in Section 3.2, and convolutional neural networks (CNN) in Section 3.3 are summarized. The reader is referred to the excellent literature in the field to the theory of machine learning and neural networks modelling [97, 6, 98].

3.1 Machine learning models

Data-driven models are obtained from the training of an algorithm with a dataset, and their tuning does not require the complete knowledge of the underlying physics or, in general, the deterministic relations that bind the input and output of the model. In Fig. 3.1 a workflow for the application of machine learning algorithms is sketched. At first, it is necessary to choose an appropriate dataset for the problem to be modelled; the data points of the dataset are called samples which are structured in inputs (features), and one or more output (label or target). The dataset is pre-processed

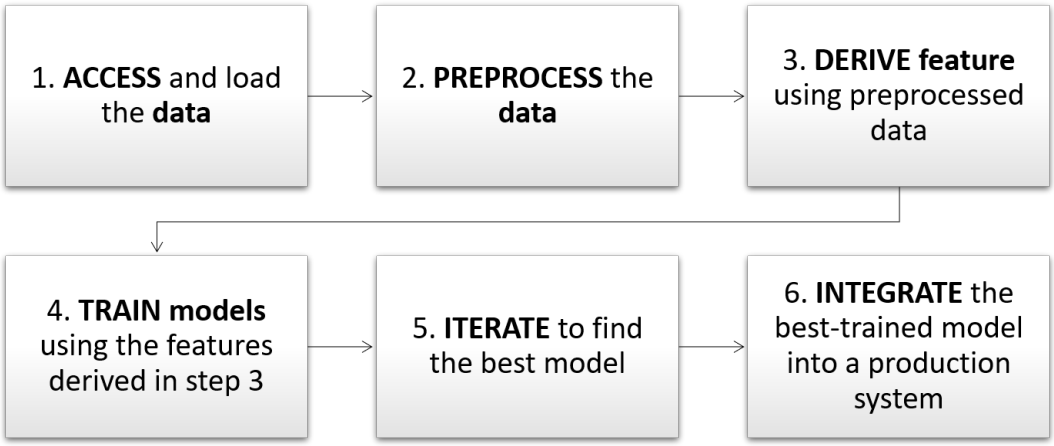

Fig. 3.1 Workflow for the application of machine learning techniques.

to improve the quality of the input to the machine learning algorithm. The most common pre-processing operations are the scaling of the features, and the control over the outliers. The following step consists in the choice of the features to use as input: in fact, effective, causal, and independent features are fundamental to have fast training and obtain accurate models. After that, it is possible to train the machine learning model, whose parameters are optimized during this process in order to minimize the loss function. Finally it is possible to improve the predictive performance of the model tuning the hyperparameters or improving the quality of the dataset and the number of samples.

Depending on the supervision type during training the machine learning algorithms can be classified as:

- *supervised learning*: each sample is labelled and the prediction objective of the training is to match the label of the samples. These techniques are employed for classification (distinction among categories) and regression tasks (prediction of numerical values);
- *unsupervised learning*: input samples do not have labels. These techniques are used for clustering tasks (split data into sub-groups with common characteristics), for visualization (to display data in a compact way and detect unpredictable patterns), for reducing the dimensionality (to simplify the dataset), or

for anomaly detection. Thus, these techniques can also be employed for the pre-processing of the dataset;

- *reinforcement learning*: the learning system, called agent, performs actions and gets positive or negative awards depending on the quality of its performance, in this way the system learns autonomously the best strategy, called policy.

In this thesis supervised techniques are employed to obtain regression models. In the next sections a brief explanation of the scaling strategies are proposed.

3.1.1 Pre-processing: scaling of the input features

The scaling step is fundamental in the pre-processing phase of the workflow. Scaling of the data is helpful when the input features have different orders of magnitude to assure an equal contribution of the features for the prediction of the output, and to speed up the training as well. Here follows some of the most common techniques, x' is the feature array, and x is the scaled array:

- min-max normalization - the values are scaled to range between a minimum a and a maximum value b.

$$x = a + \frac{(x' - \min(x)')\ (b-a)}{\max(x') - \min(x')}, \tag{3.1}$$

- average normalization - the values are scaled of the mean value, $\bar{x}'$, and normalized by the difference between the maximum and the minimum of the feature array;

$$x = \frac{x' - \bar{x}'}{\max(x') - \min(x')}, \tag{3.2}$$

- standardization - the features array is scaled as in the previous strategy, but is normalized by its standard deviation, σ:

$$x = \frac{x' - \bar{x}'}{\sigma}. \tag{3.3}$$

the standard deviation is defined as:

$$\sigma = \frac{\sqrt{\sum_{i=1}^{N_{data}} (x_i - \bar{x})^2}}{N_{data}}, \tag{3.4}$$

where N_{data} is the number of samples in the dataset.

3.2 Fully connected neural networks

Neural networks are a wide category among machine learning algorithms. These techniques are the core of deep learning, and allow to model complex non-linear problems. The inspiration in the conception of these models is the neurons functioning in a biological brain. The analogy with biological neurons can be useful to catch the principle of functioning of these models. Neurons are cells present in the cerebral cortex of animals and are made up of a cell body containing the nucleus, ramifications called dendrites and a longer extension called axon, the terminal part of which branches off into telodendrons having tiny structures called synapses which are connected to the dendrites of other neurons. Neurons exchange electrical impulses, called signals, through the synapses, when one of them receives a sufficient number of signals in the unit of time it is able to transmit an electrical signal to the neurons to which it is connected. Although they are rather simple units, neurons are able to perform complex operations as they are organized in highly articulated structures made up of billions of units, in fact, each neuron is connected to thousands of others [99].

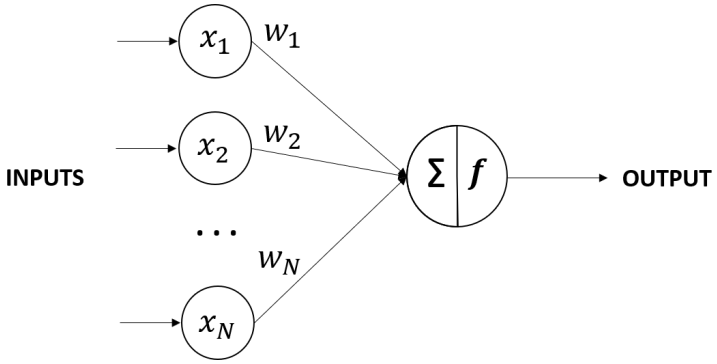

Fig. 3.2 Diagram of a threshold logic unit (TLU).

As neurons are the constitutive block of the nervous tissue, the Threshold Logic Unit (TLU, Fig. 3.2) is the basic unit of a neural network. An artificial neuron of this type is connected to numerical input values $(x_1, x_2, \ldots, x_N)$ through weights $(w_1, w_2, \ldots, w_N)$, the operations it performs are first a weighted sum of the input and

then it applies an activation function [97]:

$$y = f\left(w_1 x_1 + w_2 x_2 + \ldots + w_N x_N\right) = f\left(\mathbf{w}^T \mathbf{x}\right). \tag{3.5}$$

The traditional activation functions used, represented in Figure 3.3, are:

- the logistic function
$$\sigma(y) = \frac{1}{1+\mathrm{e}^{-y}}; \tag{3.6}$$
- the hyperbolic tangent
$$\tanh(y) = 2\sigma(2y) \; - \; 1; \tag{3.7}$$
- the Rectified Linear Unit (ReLU):
$$ReLU(y) = \max(0, y). \tag{3.8}$$

ReLU is the simplest activation function and avoids the vanishing gradient issue in deep neural networks [100], for a deeper analysis of the advantages/disadvantages in the use of ReLU and the new alternatives proposed in the last years the reader is referred to Section 6.2 of Chapter 6.

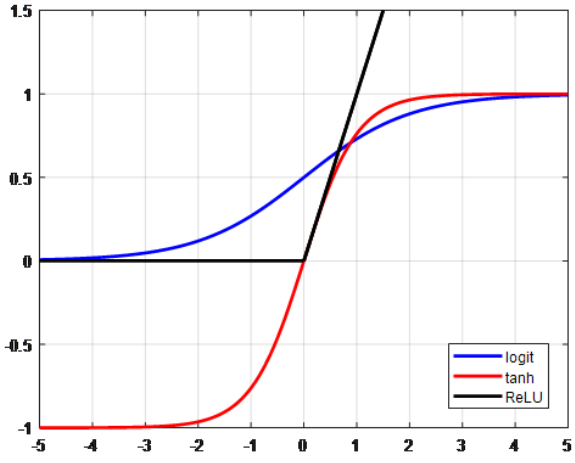

Fig. 3.3 Classical activation functions.

Several TLUs make a structure called perceptron, Fig. 3.4. Each TLU is connected to each input neuron, which outputs the input values as they are. The bias is an additional neuron of the layer that is not connected to an input feature.

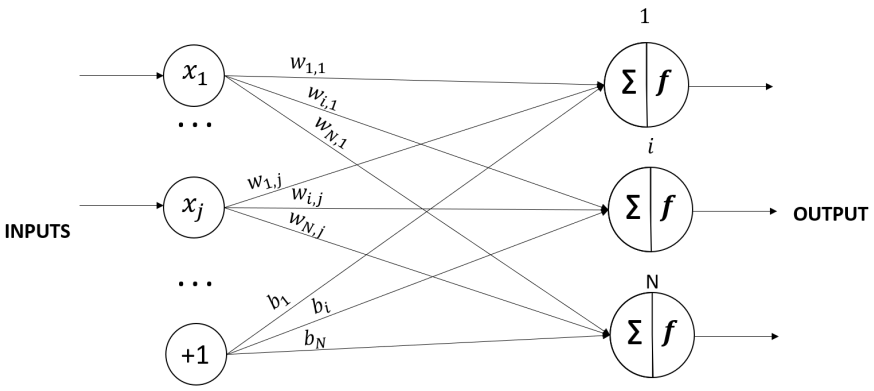

Fig. 3.4 Sketch of a perceptron.

The calculations performed by a perceptron can be summarized as follows:

$$y_1 = f\left(w_{1,1}x_1 + \ldots + w_{1,j}x_j + \ldots + b_1\right); \tag{3.9}$$

$$y_j = f\left(w_{i,1}x_1 + \ldots + w_{i,j}x_j + \ldots + b_i\right); \tag{3.10}$$

$$y_N = f(w_{N,1}x_1 + \ldots + w_{N,j}x_j + \ldots + b_N); \tag{3.11}$$

So:

$$\mathbf{y} = f(\mathbf{W}\mathbf{x} + \mathbf{b}), \tag{3.12}$$

where $\mathbf{W}$ is the weight matrix, $\mathbf{x}$ is the feature array, $\mathbf{b}$ is the bias vector, $\mathbf{y}$ is the output array.

The development of algorithms for the training of perceptrons has its roots in neuroscientific theories, in particular in Hebb's law (or Hebbian learning): after the interaction between two neurons their connection becomes stronger, this idea was summarized in the famous phrase by Siegrid Löwel: "*Cells that fire together, wire together*", i.e. the cells that transmit signals at the same time are linked together. In the context of neural networks, two neurons are connected in a much stronger way the greater the weight that binds them. The training algorithm can be summarized by Eq.(3.13) [97]:

$$w_{i,j}^{k+1} = w_{i,j}^{k} + \lambda\left(\hat{y}_i - y_i\right)x_j, \tag{3.13}$$

where $w_{i,j}$ is the weight linking the i^{th} input neuron and the j^{th} output neuron, x_j is the j^{th} input value of the current sample, y_i is the output of the i^{th} neuron for the current sample, $\hat{y}_i$ is the target output for the current sample. The learning rate λ of Eq. (3.13) is a parameter related to how fast the network adapts its parameters at the end of each epoch (i.e. the iterations) or after each batch (when the training is performed in mini-batches) of the training. If λ is too high the system will quickly adapt to optimize the predictions on the new data despite the contribution of previous data to the learning procedure, instead, if the learning rate is too low, the system is subject to greater inertia but is less affected by noisy, or non-representative data, Fig. 3.5. Too high or low values of learning rate are just qualitative indications, in fact, there is not a rule for the choice of this parameter. Even though the most common range is in between $10^{-2} - 10^{-5}$, depending on the problem tackled it is necessary to test the effect of different values.

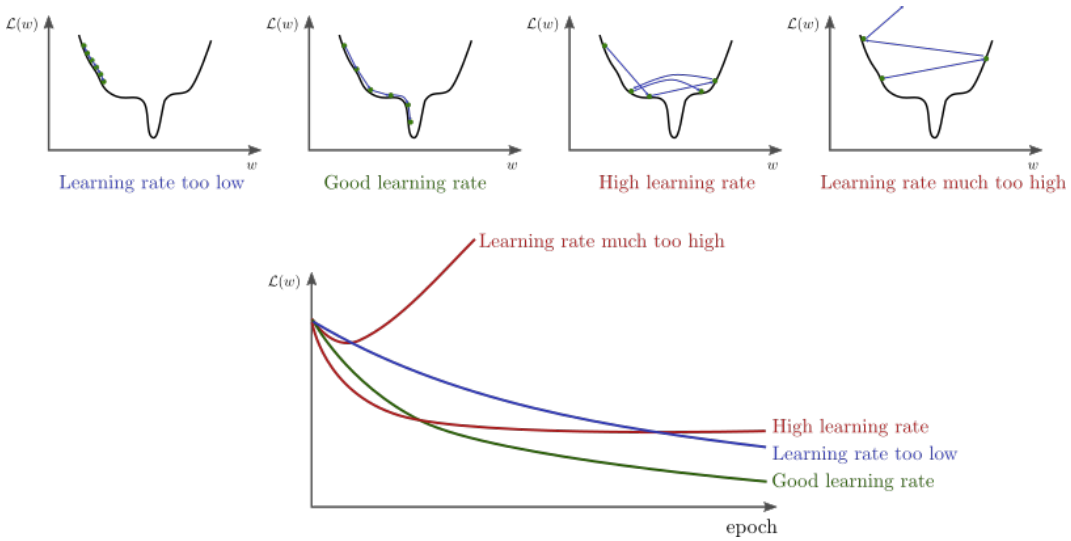

Fig. 3.5 Qualitative effect of the learning rate on the training, $\mathscr{L}$ is the loss function, the aim of the training algorithm is to reach the minimum of the loss function [5].

In order to solve increasingly complex and non-linear problems, neural networks with multiple layers have been introduced, Fig. 3.6: multi-layer perceptron, or fully connected neural networks in this thesis.

Fully connected neural networks have an input layer, an output layer and several hidden layers. The output layer contains a single neuron in regression problems, or a number of neurons equal to the number of possible categories in classification problems. The outputs of a layer are the inputs of the next layer, therefore the

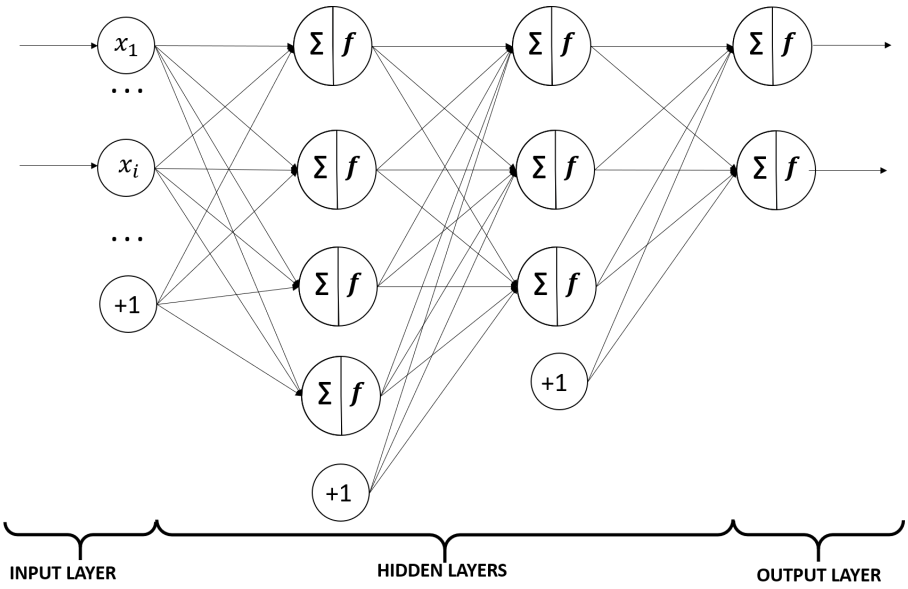

Fig. 3.6 Sketch of a multi-layer perceptron (MLP).

calculations that the network in Fig. 3.6 performs can be summarized as follows:

$$\mathbf{y}^3 = f^3\{\mathbf{W}^3 f^2\left[\mathbf{W}^2 f^1\left(\mathbf{W}^1\mathbf{x}+\mathbf{b}^1\right)+\mathbf{b}^2\right]+\mathbf{b}^3\}, \tag{3.14}$$

where the superscript indicates the layer.

The number of neurons per layer, the number of hidden layers, and the learning rate value are examples of hyper-parameters of the network, thus they are set with heuristic rules, or sensitivity studies, or by tuners available in the deep learning libraries.

3.2.1 Training of a neural network: gradient descent

The training of a neural network is performed through a backpropagation algorithm, whose aim is to minimize the loss function - a metric related to the error between the predicted and target (the true) values. The dataset available for the modelling is partitioned into three sets: the training set, which is used for the training of the neural network, the validation set, which is employed for online evaluation of the network performance, and the test set, which is employed to evaluate the generalization capability of the trained network. This subdivision is fundamental to evaluate the generalization performance of network, hence through the test set it is possible to

evaluate the capability of the network to apply the acquired knowledge on new input data, likely avoiding overfitting.

The algorithm consists of the following steps:

- forward propagation - evaluation of the output of each layer:

$$\mathbf{y}^0 = \mathbf{x}, \tag{3.15}$$

$$\mathbf{y}^{m+1} = f^{m-1}(\mathbf{W}^{m+1}\mathbf{x}^{\mathbf{m}} + \mathbf{b}^{m+1}) \text{ for m = 0, 1,..., M - 1,} \tag{3.16}$$

$$\mathbf{y} = \mathbf{x}^M, \tag{3.17}$$

  where M is the number of layers of the network;

- backward propagation - evaluation of the sensitivity, a quantity related to the gradient of the error function with respect to weights or biases. The sensitivity of the last layer M is evaluated as follows:

$$\mathbf{s}^M = -2\dot{\mathbf{F}}^M(\mathbf{n}^M)\mathscr{L}(\mathbf{y}, \hat{\mathbf{y}}) \tag{3.18}$$

  where $\mathscr{L}$ is the loss function (usually the mean square error, or the mean absolute error). Given the sensitivity in layer M it is possible to calculate the sensitivity of the previous layer:

$$\mathbf{s}^m = -2\dot{\mathbf{F}}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T\mathbf{s}^{m+1} \text{ for m = M - 1,..., 2, 1,} \tag{3.19}$$

  where $\dot{\mathbf{F}}$ is the diagonal matrix:

$$\dot{\mathbf{F}}^m(\mathbf{n}^m) = \begin{bmatrix} \dot{\mathbf{f}}^m\left(n_1^m\right) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \dot{\mathbf{f}}^m\left(n_{S^m}^m\right) \end{bmatrix}, \tag{3.20}$$

  where S^m is the number of neurons in layer m and $\dot{\mathbf{f}}^m\left(n_j^m\right)$ is defined as:

$$\dot{\mathbf{f}}^m\left(n_j^m\right) = \frac{\partial f^m\left(n_j^m\right)}{\partial n_j^m}, \tag{3.21}$$

n is a vector whose i-th element represents the input due to all neurons and the bias of the previous layer:

$$n_i^m = \sum_{j=1}^{S^{m-1}} w_{i,j}^m a_j^{m-1} + b_i^m; \tag{3.22}$$

- Updating of weights and biases - through the definition of gradient descent:

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \lambda \mathbf{s}^m \left(\mathbf{z}^{m-1}\right)^T, \tag{3.23}$$

$$\mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \lambda \mathbf{s}^m . \tag{3.24}$$

3.3 Convolutional neural networks

CNNs are a class of neural networks suited to deal with grid-like objects as input features [6], for example two-dimensional or three-dimensional images, or videos. The CNN layers implement the convolution operation:

$$\mathbf{y} = f\left(\sum_{i=1}^{F} \mathbf{x} * \mathbf{k}_i + b_i\right), \tag{3.25}$$

where $*$ denotes the convolution operation, $\mathbf{x}$ is the input, $\mathbf{y}$ is the output of the operation, f is the activation function, $\mathbf{k}_i$ is the kernel, F is the number of kernels, and $\mathbf{b}_i$ is the bias term. The kernel is a trainable array of floating point numbers, whose size and number is an hyperparameter of the network to be set, it is applied on the image as sketched in Fig. 3.7. When the input features are bi-dimensional images, the filter is two-dimensional as well, the size 3-by-3 is the most computationally efficient size for GPU computations [101].

CNNs have exhibited excellent performance in deep learning tasks compared to classical fully connected neural networks both in terms of generalization capability and computational cost of the training [102]. This is partly due to the fact that convolutional kernels share their parameters, because the same filter slides on the image and is applied in different regions, thus every output is connected just on a small portion of the input, this property is referred to as sparse connectivity. As a

consequence, CNNs are characterized by equivariance to translation, so the layers learn the influence of the images features independently from their location.

As with other deep learning algorithms, CNNs may suffer of gradient vanishing issues [100], thus the choice of an appropriate activation function is crucial to assure the contribution of each neuron to the final prediction.

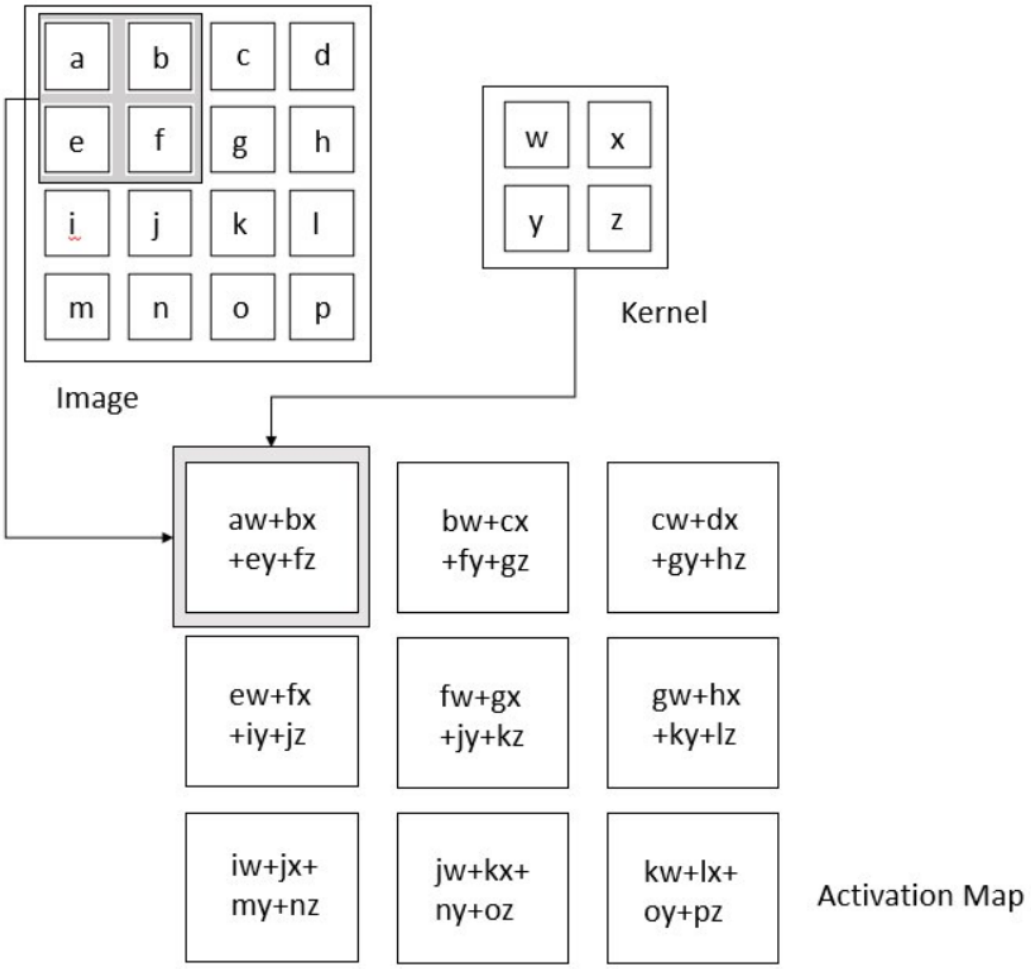

Fig. 3.7 Sketch of the operation of a convolutional layer [6]. The kernel is applied sliding on the image, and its weights are optimized during the training.

Chapter 4

Fully connected neural networks for the prediction of integral parameters

In this chapter the workflow for the creation of neural networks models from a CFD based dataset is detailed. Fully connected neural networks have been trained herein for the prediction of the permeability and the filtration rate in bi-dimensional porous media. The main novelty of this work is the construction of data-driven models for the prediction of the filtration performance. In fact, machine learning models have been widely proposed for the prediction of the permeability [59, 103, 104, 62, 69, 60], but little has been done towards the prediction of more complex integral parameters. The CFD simulations are performed using the finite volume method implemented in `OpenFoam`, and are employed to calculate the permeability, from the flow simulations, and the filtration rate, from the transport simulations.

The methodology adopted can be summarized as follows: at first the CFD model is set up, then the range of variation of the geometric parameters and the operating conditions is set, thus a number of simulations are performed with those features ranging in established boundaries. After, the permeability and the filtration rate are calculated for every simulation, in this way the dataset is created: each simulation is a sample made up by the input features and the corresponding results. The neural network is trained with a portion of the dataset and the performance is evaluated on the remaining part (following the usual split between training and testing/validation

The content of this chapter, in a modified form, has been published in Marcato et al. (2021)[16]

datasets). Finally the accuracy of the neural network in the prediction of the permeability and the filtration rate is compared with traditional analytical correlations that link those outputs with the chosen input features. The proposed workflow is entirely open-source, since both the CFD codes and the networks modelling is performed by means of open-source codes (`OpenFOAM`) and Python libraries.

4.1 Computational details

4.1.1 CFD model setup

For this work, we considered simplified bi-dimensional porous media, the geometries are made by a square containing non-overlapping circles (or grains in this work) placed randomly imposing a periodic constraint in the direction orthogonal to the main flow direction, i.e. the circles that cross the upper boundary are completed in the lower part of the domain, and viceversa; this is shown in Fig. 4.1. Both monodisperse and polydisperse grain diameter distributions have been considered. The dimension of the computational domain, i.e. the number of grains, is established as a result of a representative elementary volume (REV) study [105, 2]. To this end a wide bi-dimensional geometry is created, then areas with an increasing number of grains are considered and the porosity is calculated. As Fig. 4.2 shows, we can consider a REV constituted by 100 elements in the monodisperse case and 150 for the polydisperse one. The spatial discretization is set imposing a minimum number of cells per diameter. In case of monodisperse distributions 20 cells per diameter, in case of polydisperse distributions 10 cells per minimum diameter of the distribution, corresponding to 30 cells per mean diameter, Table (4.1). Grid independence studies for analogous physical systems can be found in previous works [51, 106]. The geometry and the mesh are built with the mesh generators `blockMesh` and `snappyHexMesh`.

The system is considered isothermal at temperature equal to 298 K, the fluid is Newtonian with density equal to 997 kg m^{-3} and kinematic viscosity equal to 0.89 $\times 10^{-6}$ m^2 s^{-1}, equal to pure water viscosity at ambient temperature. The fluid flows in laminar conditions, in fact, the Reynolds number does not exceed 0.015, so no turbulence models are required in the simulations. The gravity is not considered in this model. The solver `simpleFoam` is employed for the resolution of Eq. 2.6. The

boundary conditions set on the grains surface are no-slip condition for velocity, and zero gradient of pressure. On the upper and lower boundaries a cyclic boundary condition is set. A constant pressure at the inlet (the left boundary on Fig. 4.1) and a null pressure at the outlet (the right boundary on Fig. 4.1) are imposed, as to obtain a fully developed velocity profile entering the porous medium.

The colloid transport is modeled as a scalar transport of the normalized particle concentration C, thus the `scalarTransportFoam` solver is employed for the resolution of Eq. 2.20 in steady state conditions. A normalized inlet concentration equal to 1 and a zero gradient of the concentration are imposed as boundary conditions at the inlet and at the outlet of the porous medium respectively; a cyclic boundary condition is set at the upper and lower boundaries. On the grains surface a null concentration is imposed, representing the clean-bed filtration described earlier.

The numerical details (convergence criteria, and discretization schemes) are summarized in Appendix C.

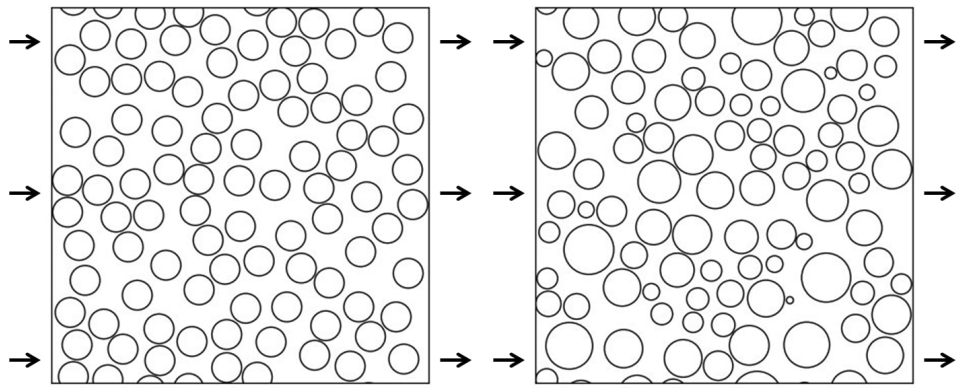

Fig. 4.1 Sample of geometries considered in the simulation of flow and transport in porous media. Left: monodisperse geometry; right: polydisperse geometry.

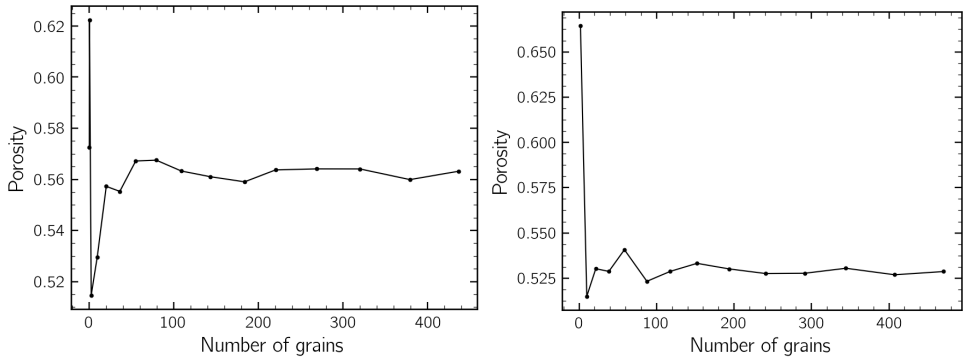

Fig. 4.2 Representative elementary volume (REV) study: evaluation of the porosity as a function of the number of grains considered in the computational domain. Left: monodisperse geometry; right: polydisperse geometry.

4.1.2 Dataset

As mentioned in the introductory section the dataset for the training of the networks is made by CFD simulations. In order to obtain a dataset wide enough for the networks training the simulations are performed in parallel on an HPC cluster equipped with 29 nodes with CPU 2x Intel Xeon E5-2680 v3 2.50 GHz 12 cores.

The CFD model described in the previous section is used as simulation template for the setup of all the simulations being part of the final dataset. Different operating conditions and geometric parameters are set, in particular: the inlet pressure of the fluid, p, the size of the colloidal particles, d_c, the diameter of the grains, d, the porosity of the medium, ε and, just for the polydisperse geometries case, the standard deviation of the grain diameter distribution, σ. In Table 4.1 the ranges of variation of those parameters are summarized. The simulations are set up with a random combination of the parameters whose values are uniformly extracted from the ranges of Table 4.1. The parameters are selected randomly, and not in a structured way, for each simulation in order to explore in the dataset the effect of a combination of parameters as wide as possible and reduce the bias in the model.

The computational domain of each simulation is the same, thus the number of grains, i.e. the circles, is set to match the corresponding porosity. The domain size has been chosen to guarantee a suitable REV in the case of both largest porosity and grain diameter, as a consequence a valid REV for all the other combinations of porosity and average diameter.

The resulting dataset is made by 962 samples for the monodisperse case and 996 samples for the polydisperse case.

Parameter	Range of variation	
p	0.0836 - 0.50	Pa
d	100 - 200	μm
σ	0.1 - 0.3	(-)
ε	0.5 - 0.65	(-)
d_c	30 - 100	nm

Table 4.1 Range of variation of the geometrical parameters and operating conditions, selected as predictors, for the setup of the simulations.

4.1.3 Neural network setup

The data-driven models are trained with the dataset created as it is described in the previous paragraph. The porosity of the porous medium, the mean grain diameter and the standard deviation of the grain distribution (in the polydisperse case), are the predictors for the evaluation of the permeability of the porous media, calculated by Darcy's law, Eq.(2.7). The same geometric parameters, the inlet pressure of the fluid, and the colloid dimension (i.e. the diffusion coefficient of the species) are the predictors for the evaluation of the filtration rate, Eq. 2.27. Every sample fed to the neural network training algorithm is the result of a CFD simulation performed with a different set of these predictors.

The fully connected neural networks are trained using the Adam optimizer [107] until the maximum number of epochs is reached (20000) or when the MSE on the test set remains unchanged for a number of epochs equal to 1500. Different learning rates were employed in order to evaluate the value corresponding to the best performance of the network, in particular the following values of λ have been tested: $10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}$. Different architectures of the networks were also tested, starting from a shallow network composed by a single hidden layer with 32 neurons, then hidden layers are added with an increasing number of neurons: 64, 128, 256.

The performance of the model is evaluated on the test set of the dataset, for each sample of this set the relative error between the CFD result and the model prediction is calculated, Eq. 4.1. Then, the average error of the neural network is calculated as the average of the error on each sample, the different architectures and learning rate tested are compared on the basis of the average error, the maximum error and the standard deviation of the error on the test set.

$$\text{Error} = \frac{\hat{y} - y(\text{CFD})}{y(\text{CFD})}. \tag{4.1}$$

The hyperparameters evaluated in this study are the learning rate and the neural network architecture, the best combination of the two for each result (permeability of the porous media and filtration rate of the colloid) and for each system (monodispersed and polydisperse grain diameter distribution) is researched resulting in four different networks. The choice is made based on the lowest average relative error on the test set. In Fig. 4.3 the average relative error as a function of the learning rate for the four different architectures is reported. For the prediction of the permeability of the porous media in the monodisperse case the learning rate is chosen equal to 10^{-1} and the architecture of the network is $\{32\ 64\}$, i.e. two hidden layers with respectively 32 and 64 neurons. For the prediction of the permeability of the porous media in the polydisperse case the learning rate is chosen equal to 10^{-3} and the architecture of the network is $\{32\}$, i.e. one hidden layer with 32 neurons. For the prediction of the filtration rate of the colloid in the monodisperse case the learning rate is chosen equal to 10^{-2} and the architecture of the network is $\{32\ 64\ 128\}$, i.e. three hidden layers with respectively 32, 64 and 128 neurons. For the prediction of the filtration rate of the colloid in the polydisperse case the learning rate is chosen equal to 10^{-2} and the architecture of the network is $\{32\ 64\}$, i.e. two hidden layers with respectively 32 and 64 neurons.

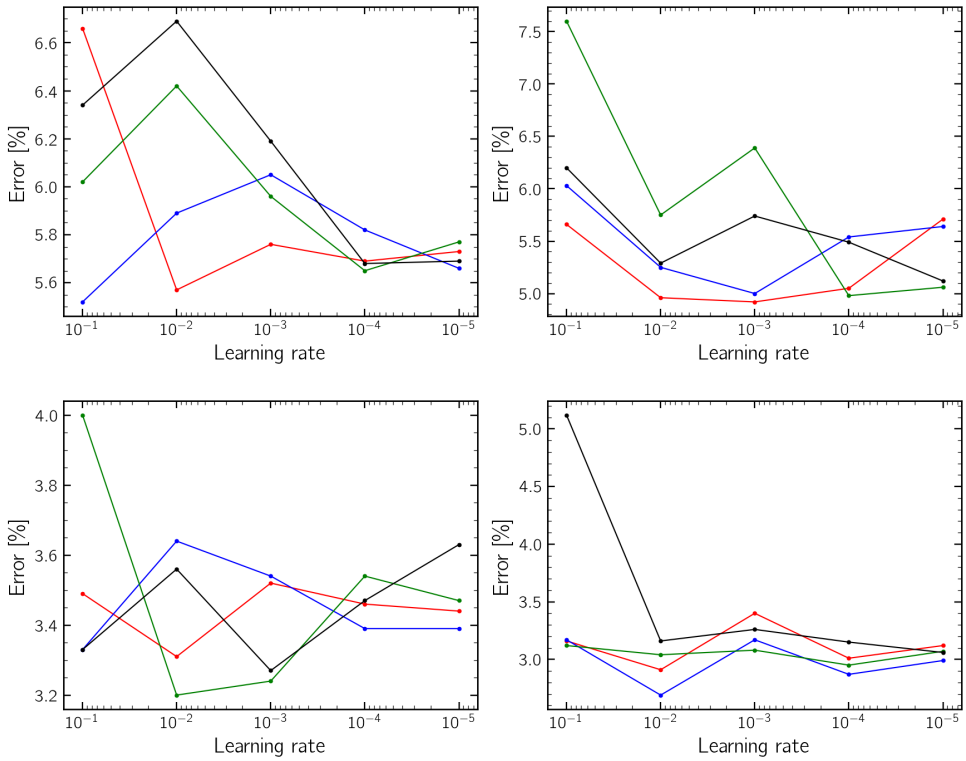

Fig. 4.3 Hyperparameter tuning of the fully conncted neural network: mean relative error on the test set as a function of the learning rate (10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}) and of the network architecture ({32} in red, {32, 64} in blue, {32, 64, 128} in green, {32, 64, 128, 256} in black, refer to the end of Section 2 for the bracket nomenclature). Top: error on the prediction of permeability; bottom: error on the prediction of filtration rate; left: monodisperse grains; right: polydisperse grains.

4.2 Results and discussion

In this section the results of the CFD simulations are presented and both analytical and data-driven models are proposed for the prediction of the upscaled parameters described before, namely permeability k and filtration rate K_f.

4.2.1 Flow field results

The results of the CFD simulations are at first compared with the Ergun equation 5.3. The inertial term of the Ergun equation in the cases of this work is negligible since

all the samples fall in the laminar regime. The results obtained from the simulations are not completely in agreement with the equation because the geometries are bi-dimensional whereas Ergun's equation is valid for three-dimensional packed beds. Similar results were obtained in a past work [51] for a smaller number of simulations performed. The polydisperse geometries, referring to the polydisperse grains, deviate less from the theoretical equation compared to monodisperse ones, moreover the higher is the standard deviation of the grain diameter distribution the lower is this deviation, Fig. 5.9. This behavior can be explained considering that the most likely bi-dimensional description of a monodisperse 3D porous medium is a 2D polydisperse geometry. In fact, if we imagine to cut with a surface a three-dimensional sphere arrangement in the flow direction we obtain circles with different diameter and most of them will not be in contact. Two contour plots for a qualitative description of the flow field are presented in Fig. 4.5, where the fluid flows from left to right and it is possible to appreciate the presence of the periodic boundary conditions on the upper and lower boundaries.

The permeability is the first upscaled parameter we have approached for testing the workflow proposed in the previous section. It is a well known descriptor of a porous medium from the geometrical point of view, and in our case it is calculated by comparing the results of the CFD simulations with Darcy's law, Eq. 2.7. The Kozeny equation:

$$k = C_k \frac{\varepsilon^3}{(1-\varepsilon)^2 M_s^2}, \tag{4.2}$$

where M_s is the specific surface of the porous medium (m^2/m^3), instead is an expression for the permeability that depends just on geometrical parameters: it is exploited as an analytical model to compare with the results of the data-driven model. The Kozeny constant is evaluated by fitting of the CFD results, Fig. 4.6, resulting in C_k equal to 0.084 for both monodisperse and polydisperse geometries. The average error associated to the prediction of the permeability by the Kozeny equation, calculated according to Eq. 4.1, is equal to 10.1% for monodisperse geometries and 8.2% for polydisperse geometries. The error linked to the polydisperse geometries is lower than in the monodisperse case, even in this case this behavior is amenable to the most likely 3D approximation of the first ones. The specific surface has been evaluated in an analytical way starting from the radii of the grains and the dimensions of the geometries, we proved that the quality of the predictions of the Kozeny law does not change if the specific surface is evaluated from the meshed geometry.

	monodisperse geometries		polydisperse geometries	
	k	K_f	k	K_f
Average error	5.52%	3.24%	4.92%	2.69%
Maximum error	25.85%	10.35%	20.03%	11.99%
Standard deviation	4.43%	2.44%	3.48%	2.49%

Table 4.2 Performance of the fully conncted neural networks.

The performance of the optimized fully connected neural networks is shown in the parity diagrams of Fig. 4.7, where the results of the CFD simulations and the result of the model are compared. In Table 4.2 the average error, the maximum error and the standard deviation of the error are reported. The fully connected neural networks are able to predict the permeability with an average error of 5.5% and 4.92% respectively for monodisperse and polydisperse geometries, instead the Kozeny equation results in errors of 10.1% and 8.2% respectively, so the neural networks perform better with average errors about 40% lower than the traditional correlation for the geometries proposed and for the range of operating conditions explored in this work.

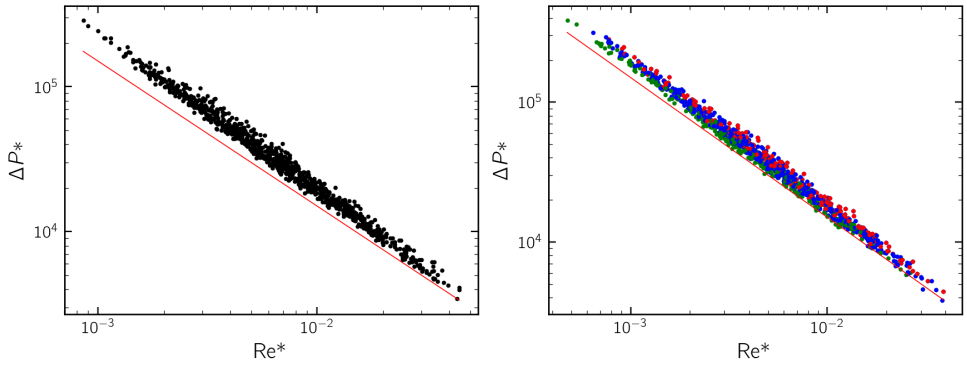

Fig. 4.4 Comparison of CFD results (black points) with Ergun's equation (red line). Left: monodisperse geometry; right: polydisperse geometry with σ (i.e. standard deviation in diameter distribution) between 0.1 and 0.15 in green, between 0.15 and 0.25 in blue, between 0.25 and 0.3 in red.

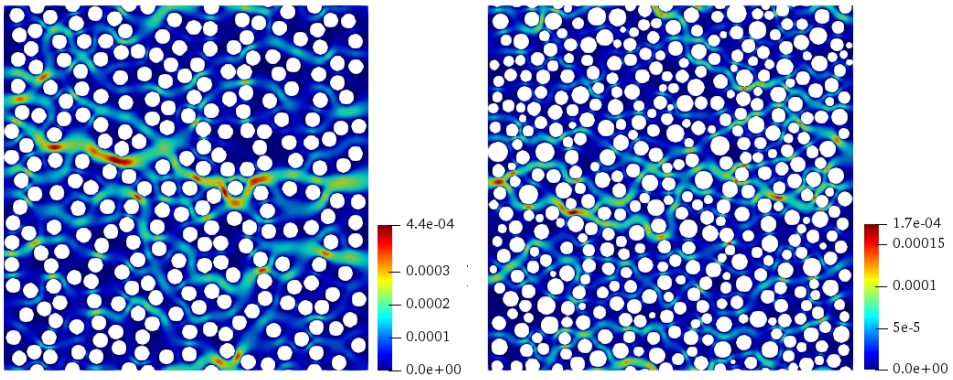

Fig. 4.5 Contour plot of the fluid velocity (ms^{-1}), from left to right. Left: monodisperse grain distribution geometry (Re = 0.0071); right: polydisperse grain distribution geometry (Re = 0.0021).

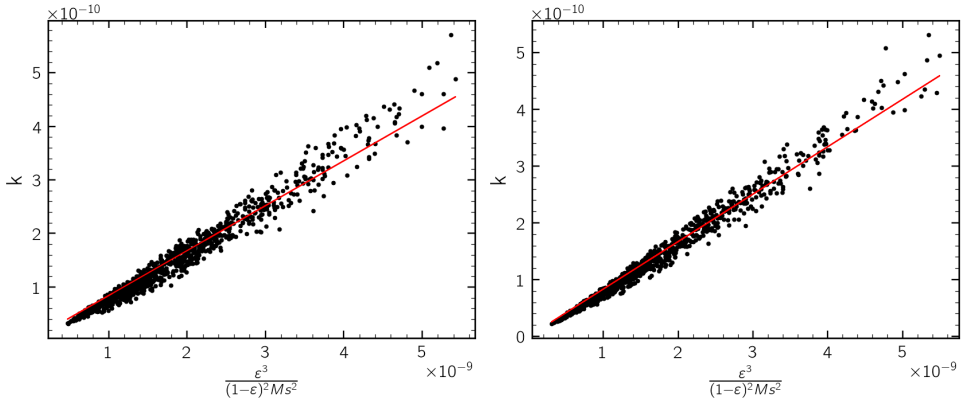

Fig. 4.6 Evaluation of the Kozeny coefficient through linear regression. Left: monodisperse geometry; right: polydisperse geometry.

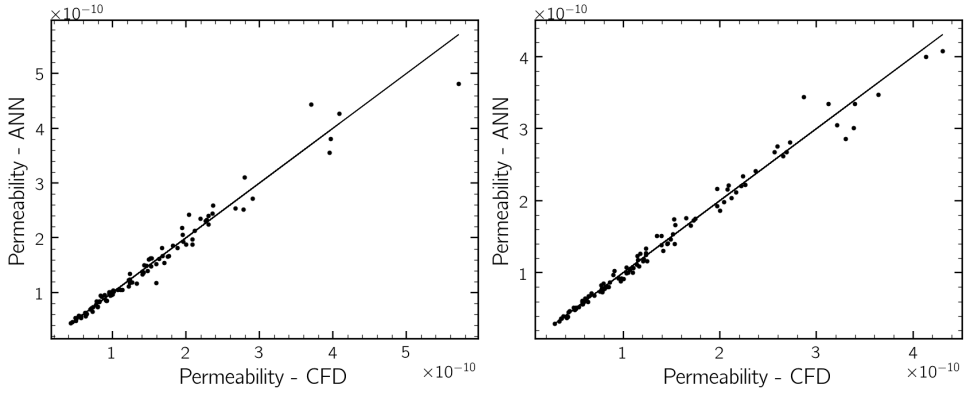

Fig. 4.7 Parity diagrams of the permeability prediction on the test set for the optimal fully conncted neural networks trained with the best hyperparameters. Left: monodisperse geometry; right: polydisperse geometry.

4.2.2 Transport field results

The particle deposition simulations are performed as described before on top of the flow field results. In Fig. 4.8 two contour plots of the concentration at the steady state are reported. The two cases proposed differ for the geometry and for the Pèclet number, Eq. 2.26. So at high Pèclet numbers the convective term prevails on the diffusive one and as a consequence the filtering performance of the porous medium decreases. The time required for the resolution of each CFD simulation, considering both `simpleFoam` and `scalarTransportFoam` solvers, is 1.2 hours in the case of monodisperse grains geometries and 4 hours in the case of polydisperse grains geometries, which have an higher number of cells.

The data-driven model we designed is aimed at the prediction of the filtration rate, K_f, as defined in the previous chapter Eq. 2.27. From the literature [83] we know that this upscaled parameter is correlated to the Pèclet number through the Damköhler number, which in this work we define as follows:

$$\mathrm{Da} = \frac{K_f L}{q}. \tag{4.3}$$

The classic upscaled parameter for the evaluation of the filtration performance is the deposition efficiency η, whose best adimensional descriptor[1] is the Pèclet number

[1] if one considers only Brownian diffusion, as it is the case in this work

[27, 108, 109]. In this work we have chosen to employ the more reliable macroscale parameter K_f (as explained in [83]), but the same functional dependence holds. Thus we consider a relationship of the form:

$$\mathrm{Da} = APe^B, \tag{4.4}$$

The values of parameters A and B obtained from the best fit of the results dataset, Fig. 4.9, the following values are obtained: A = 6.394×10^9, B = -0.928 for the monodisperse geometry, A = 7.413×10^9, B = -0.973 for the polydisperse geometry. The average error of the prediction is equal to 12.2% for the first case and 11.9% for the second case.

The performance of the optimized neural networks is evaluated in the parity diagrams of Fig. 4.10 where all the predictions on the test set samples are reported: the average error is equal to 3.24% for the monodisperse case and 2.69% for the polydisperse case. In Table 4.2 more details on the data-driven models accuracy are reported. Even in this case the data-driven model is able to predict better the deposition efficiency with respect to the classical relationship structured on dimensionless number dependency. In the limit of the cases explored in the current dataset, the error associated to the predictions is one order of magnitude lower than the one correlated to the analytical correlation, Eq. 5.7. The neural networks training process took about 4 minutes and the trained model is able to predict almost instantly its output.

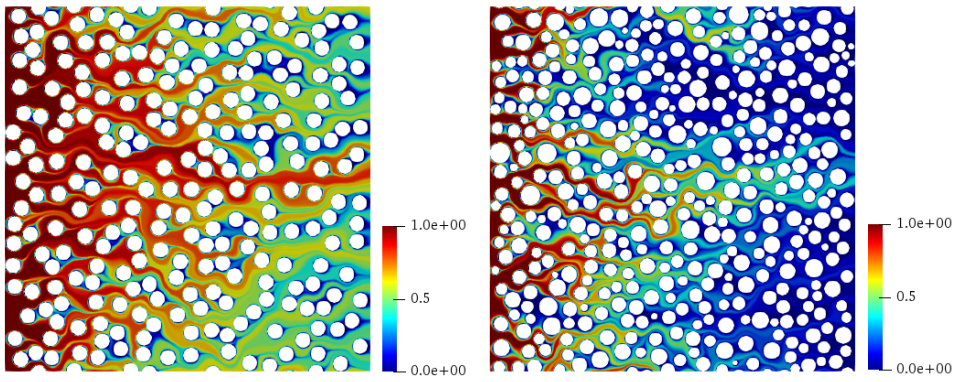

Fig. 4.8 Contour plot of the normalized concentration. Left: Monodisperse grain distribution geometry (Pe = 990.0); right: Polydisperse grain distribution geometry (Pe = 252.5).

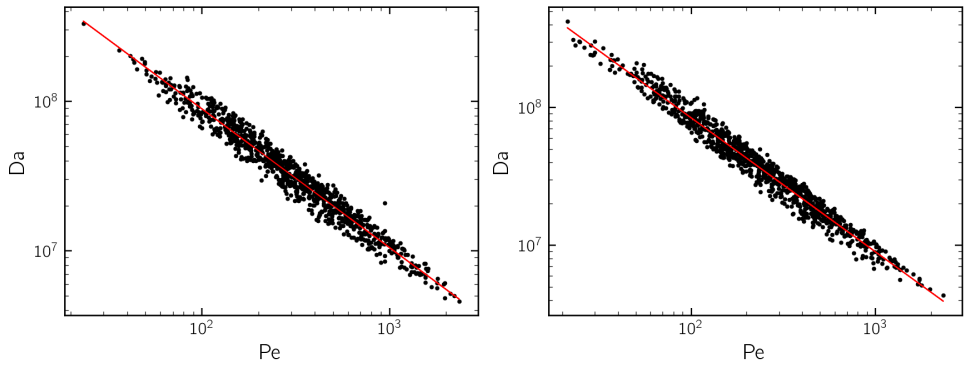

Fig. 4.9 Evaluation of the correlation between the Damköhler number and the Pèclet number. Left: monodisperse geometry; right: polydisperse geometry.

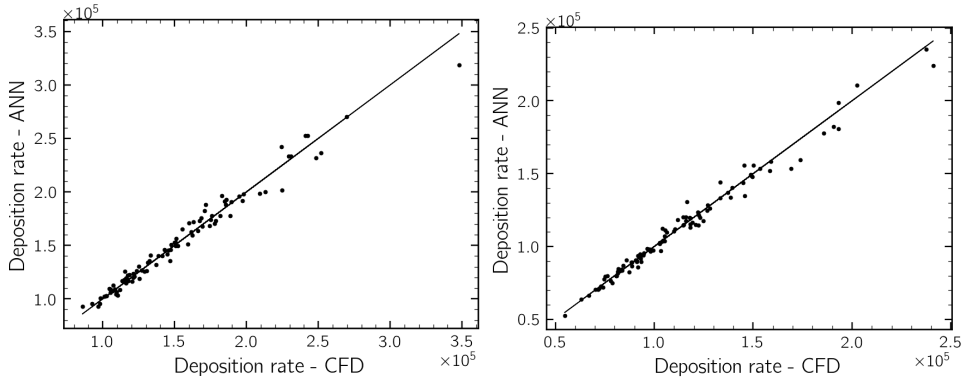

Fig. 4.10 Parity diagrams of the filtration rate prediction on the test set for the optimal fully conncted neural networks trained with the best hyperparameters. Left: monodisperse geometry; right: polydisperse geometry.

4.3 Conclusions

In this chapter we described a successful open-source workflow for the realization of a dataset starting from a campaign of CFD simulations aimed at the training of neural networks for the prediction of fluid dynamics quantities. Fully connected neural networks have been employed to predict the permeability and the filtration rate for the investigated cases. In conclusion:

- The average error on the test set for the permeability it is lower than 6% and for the filtration rate, which has never been predicted with machine learning tools before, is lower than 3.5% for both geometrical models;
- The networks allow more accurate predictions of the permeability and the filtration rate compared to traditional analytical expressions, with average errors lower than 4% in the range of operating conditions explored in this work;
- The data-driven models, once trained, can instantaneously give a satisfactorily accurate output, while a CFD simulation requires a certain amount of computational time. In our case each CFD simulation requires from one to four hours to be solved and the training takes four minutes. The increase in the predictive velocity can be exploited in multiscale modeling, in-line control and optimization problems;

ML algorithms, in particular neural networks, are promising tools in modeling advancement of intrinsically random problems as porous media are. To enhance the predictive accuracy new elaborated input features could be employed, such as the tortuosity and the pore size distribution, paying attention to the choice of parameters that can be experimentally evaluated. To avoid the arduous selection of parameters representing the randomness of the media, another way to solve the problem is to submit the entire geometry as training input for the neural network: in this way the algorithm itself could select the most important features in the image. In this case more sophisticated techniques must be used, such as convolutional neural networks, which will the focus of the next chapter.

Chapter 5

Convolutional neural networks for the prediction of integral parameters

In this chapter fully connected neural networks (FCNN) and convolutional neural networks (CNN) for the prediction of the permeability and the filtration rate are compared for the prediction of properties in three-dimensional porous media. This work applies (and expands) the workflow that we proposed in the previous chapter [16] and that is summarized in Fig. 5.1 for three-dimensional geometries. At first porous media geometries are created *in silico*, then the computational grids are built for the CFD simulations, that are in turn solved so as to prepare a dataset for the training. While for the FCNN case the neural network is provided with integral descriptors of the porous media geometries, in the CNN case the entire geometry together with the operating conditions input parameters are fed to the network model. Both these techniques give good predictions of the permeability and the filtration rate in porous media, with average errors lower than 5%.

Beyond this similarity in performance, this proven effectiveness of CNN in building a predictive model for non-trivial transport/reaction phenomena, with no need for arbitrary and potentially unsuccessful parameter hand-picking, will have important consequences in the construction of a general model, as it will be expanded upon in the remainder of this chapter. In closing, a numerical exploration of the coupling between these deep-learning models and CFD is presented: namely, we conducted a sensitivity study on the size of the dataset used in training the CNN in

The content of this chapter, in a modified form, has been published in Marcato et. al (2022) [14]

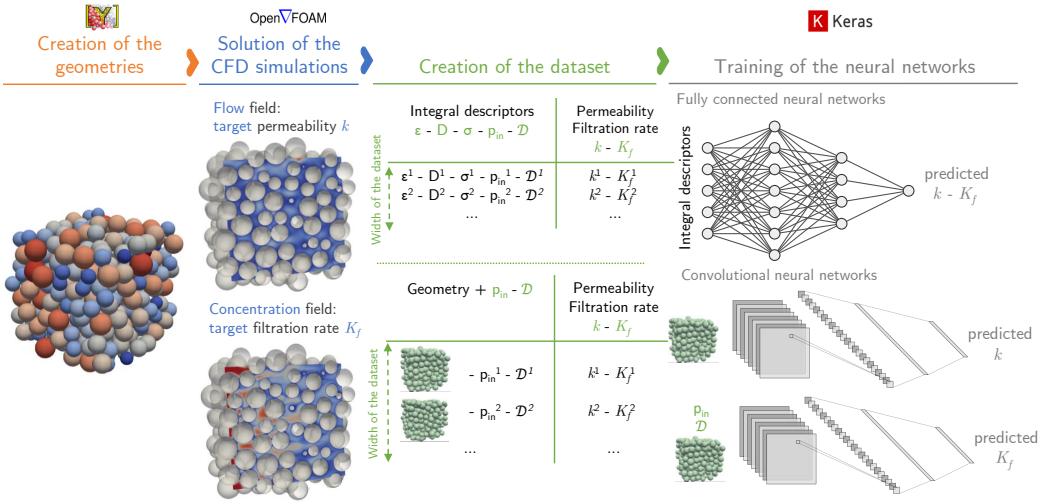

Fig. 5.1 Workflow for the construction of neural networks models for porous media applications: 1) *in silico* creation of the geometries, 2) setup and solution of the CFD simulations, 3) preparation of the dataset for the FCNN - integral descriptors as input and output- and for the CNN - entire geometry and integral descriptors as input and integral descriptors as output.

order to determine the minimum number of CFD simulations to be solved for the sake of obtaining an accurate data driven model, as well as a preliminary tuning of the relevant hyperparameters for the deep-learning model.

5.1 Computational details

5.1.1 CFD model setup

The porous media geometries for the CFD simulations, Fig. 5.2, were created *in silico* by means of the open-source toolbox Yade which implements a discrete element method (DEM). This tool was selected for the purpose of obtaining periodic packings of spheres with a low computational cost, in fact, the creation of a representation of hundreds of spheres takes only a few minutes. Since a dataset of hundreds of CFD simulations is required for the training of the neural network model, the fast creation of geometries is an advantage to reduce computational costs. The DEM model integrates the motion equations for each particle of the packing, so the position of a single particle is updated at each time step using the acceleration (a) computed from

the total force acting on the particle (F) and its mass (m) (5.1):

$$ma = \sum_i F_i. \tag{5.1}$$

The forces taken into account are the interparticle normal and shear forces. A periodic boundary condition is set up for the sake of studying the behaviour of a material in bulk, and as such avoiding the wall effects on the packing. A detailed explanation of the method is provided in Appendix A.

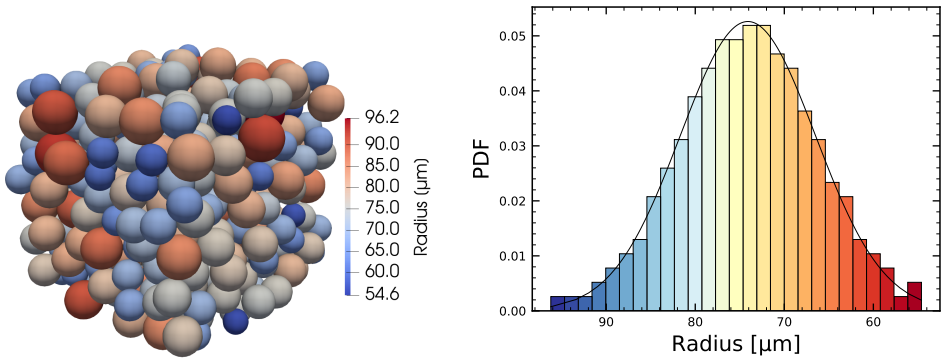

Fig. 5.2 The porous media geometries were created *in silico* by means of YADE with a Gaussian particle size distribution.

The smallest representative elementary volume (REV) was identified in order to fix the minimum number of grains necessary for a macroscopic characterization independent from the size of the porous medium. Since the grain size distribution of the geometries follows a Gaussian distribution, the REV study was performed considering the highest standard deviation. The geometries for the CFD simulations are made by 250 grains, this dimension is selected on the basis of a representative elementary volume study. The study was conducted on Gaussian particle size distributions having the highest standard deviation (Tab. 6.1) in this way the study can be extended to all the cases explored in the dataset. In Fig. 5.3 it is possible to visualize how the porosity changes with the number of grains. Given the number of grains (50-1450), 100 representations of the same geometry were made by means of Yade, on the chart the average of the porosity on the 100 representations is reported, with the error bar representing the standard deviation. In Fig. 5.4 the relative error between the average porosity at a certain dimension and the average porosity of the larger geometries we created (1450 grains) is reported. For porous media having

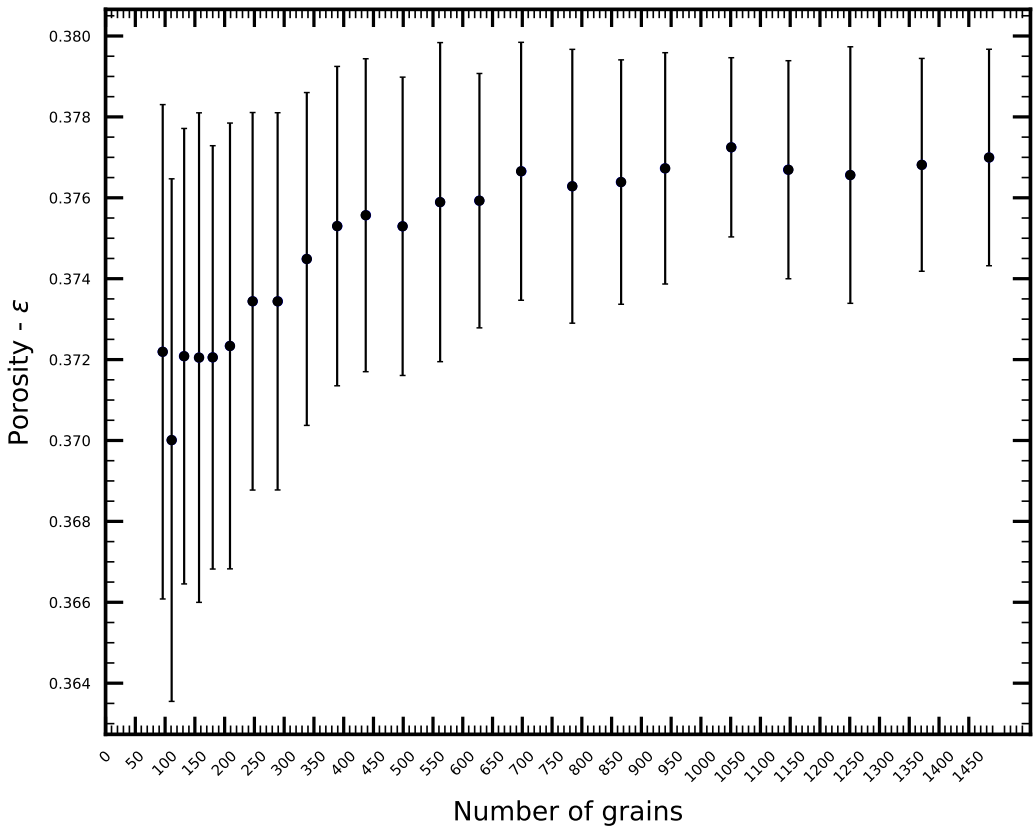

Fig. 5.3 Representative elementary volume study: the mean and the standard deviation of the porosity are reported for 100 representations of each packing composed by a certain number of grains. The study is performed with the highest standard deviation of the particle size distribution

more than 250 grains, the corresponding error on the porosity is lower than 1%, this is the basis on which we selected the dimension of our porous media geometries.

The geometries were used for the CFD simulations, that were solved by using the finite volume method implemented in the open-source toolbox OpenFOAM v7. We refer in this paragraph to the solvers and tools of this software, which are essentially the same employed for the construction of the dataset detailed in Chapter 4, we briefly recall them in this paragraph. The system is considered isothermal at 298K and the fluid phase has the properties of pure water, therefore it is a Newtonian fluid with density equal to 997 kg m^{-3} and kinematic viscosity equal to 0.89 $\times 10^{-6}$ m^2 s^{-1}.

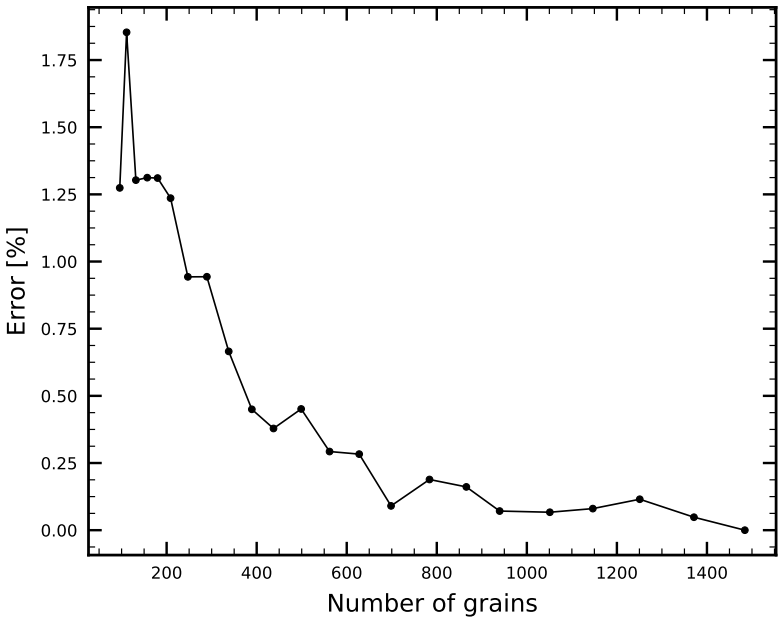

Fig. 5.4 Representative elementary volume study: the relative error on the average porosity is reported for 100 representations of each packing composed by a certain number of grains.

The fluid flows in the porous media in laminar conditions, so it is not needed to model the turbulence, and the effect of gravity is not considered. The solver `simpleFoam` is employed for the resolution of Eq. (2.6). The boundary conditions applied for the numerical solution of these equations are a pressure drop between the inlet and outlet boundaries, a no-slip condition on the grains surface and symmetry conditions on the other faces of the simulation box.

The transport of the colloid in the fluid is modelled as a scalar transport of the normalized concentration C, due to the hypothesis mentioned in the previous paragraph. The `scalarTransportFoam` solver is employed for the resolution of Eq. (2.20) in steady state conditions. At the inlet boundary a normalized concentration equal to one is imposed, instead at the grains surface a null concentration is set. The null normalized concentration can be physically interpreted as a clean bed filtration [51] or as an instantaneous reaction on the grains surface.

The computational grid, i.e. the mesh, was constructed by means of the OpenFOAM tools `blockMesh` and `snappyHexMesh` and a grid independence study is performed. The operating conditions of the simulation for the grid independence study were selected so that the highest Péclet number is reached, in fact, all the CFD simulations that were solved for the creation of the dataset have a Péclet number

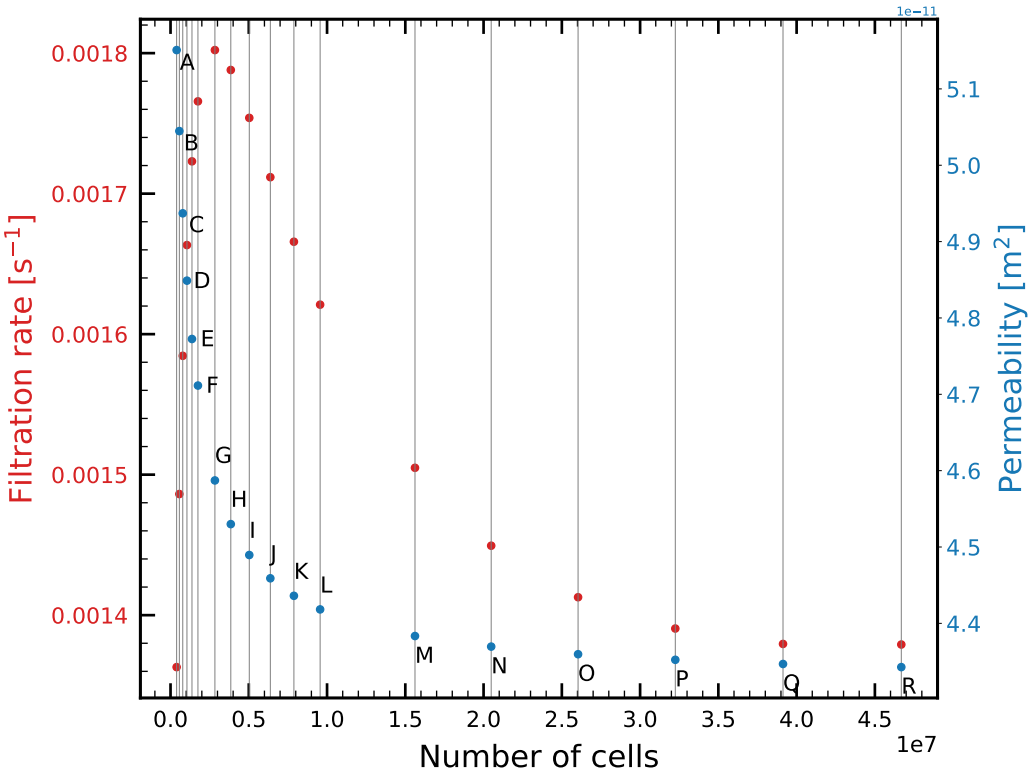

Fig. 5.5 Grid independence study: effect of the number of cells on the permeability and on the filtration rate, refer to Tab. 5.1 for the details on the meshing strategies.

lower than 1500. Figure 5.5 reports the effect of the number of cells on the permeability (on the right axis) and the filtration rate (on the left axis). In Table 5.1 the meshing strategy and the errors are reported, the errors are calculated as the relative error between the current strategy and the last strategy at which correspond the highest number of cells, so the most accurate result. The meshing strategy adopted for all the CFD simulations solved in this work is the `N`, Fig. 5.6, which represents an acceptable trade-off between the accuracy and the computational cost of the simulations since the relative error on the permeability is 0.62%, the error on the filtration rate is 5.1% and the CFD simulation requires 19 hours on a single core to be solved.

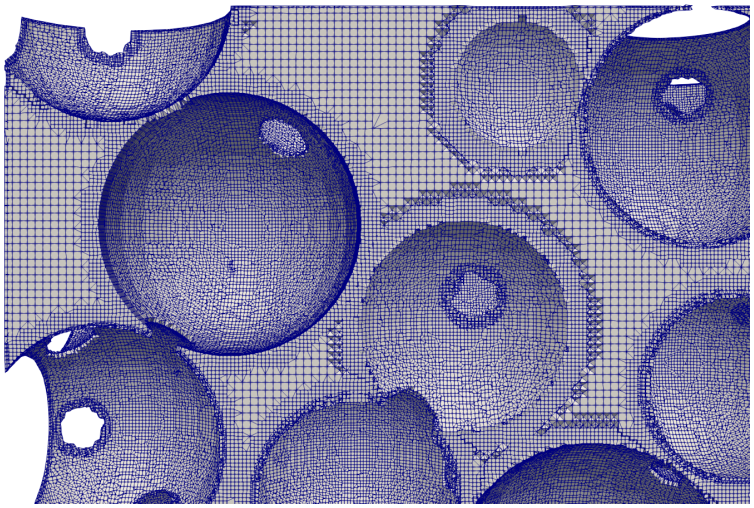

Fig. 5.6 Zoom on the computational grid of a sample representation, the meshing strategy is the `N` of Tab. 5.1.

Table 5.1 Grid independence study: relative error between each strategy (`A`-`Q`) and the `R` strategy on the permeability and the filtration rate. C.P.D.: number of Cells Per mean Diameter, R.L.: Refinement Level in `snappyHexMesh`.

Meshing strategy	C.P.D.	R.L.	Computational time [h]	Error on permeability [%]	Error on filtration rate [%]
A	16	0	0.2	18.61	1.17
B	18	0	0.3	16.17	7.77
C	20	0	0.5	13.69	14.90
D	22	0	0.6	11.66	20.61
E	24	0	0.8	9.90	24.94
F	26	0	1.1	8.49	28.03
G	16	1	1.7	5.63	30.68
H	18	1	2.7	4.31	29.65
I	20	1	3.5	3.38	27.18
J	22	1	4.5	2.68	24.12
K	24	1	6.0	2.15	20.79
L	26	1	6.8	1.74	17.54
M	16	2	12.8	0.94	9.12
N	18	2	19.0	0.62	5.10
O	20	2	21.0	0.39	2.44
P	22	2	28.5	0.22	0.83
Q	24	2	40.5	0.09	0.03
R	26	2	38.6	0.00	0.00

5.1.2 Neural networks setup

The dataset is employed for the training of neural networks models. In this work FCNNs and CNNs are trained and tested.

Two FCNNs were trained, one for the prediction of the permeability and one for the prediction of the filtration rate. The permeability is predicted starting from the geometrical parameters of the porous media: porosity, mean diameter and standard deviation of the grains diameter distribution. The filtration rate is predicted starting from the same geometrical parameters, together with the operating conditions of the filtration simulation: the inlet pressure of the fluid and the diffusion coefficient of the colloid. Different architectures and learning rates were investigated in order to determine an optimal neural network model, in the following sections the outcomes are presented and discussed.

Even for the CNN models two networks had to be trained. The one for the prediction of the permeability receives exclusively as input the entire geometry of the porous media. The geometry was analytically defined by means of Yade as a list of centres and radii, for the purpose of providing the CNN with a compatible input a voxelization representation is created. Starting from a STL file the Python's library `Trimesh` was used for the voxelization of the geometry. The resulting array contains 0 in the solid phase and 1 in the fluid phase, afterwards the Euclidean distance transform was applied to the resulting array so as to provide a more descriptive input to the CNN [62], resulting in each voxel, i.e. component, in the solid phase being marked with 0 and each voxel in the fluid phase with the distance from the closest solid voxel, a detailed description of this procedure can be found in Appendix B. The CNN for the prediction of the filtration rate takes as input the same geometry together with the inlet pressure of the fluid and the diffusion coefficient, which are concatenated to the flattened output of the CNN portion. The architecture of the CNN is summarized in Fig. 5.7. The hyperparameter tuning in this case was done on the choice of the learning rate, on the scaling of the input data and on the use of the batch normalization layers, in the following section these results are discussed.

The numerical inputs and outputs for both types of neural networks were scaled between -1 and 1 in order to optimize the activation function effectiveness. The optimization algorithm that was used for the training in this work is Adam [107] which is considered the best overall choice among the gradient descent optimization

algorithms [110]. The loss function minimized by the algorithm is the mean squared error, Eq. (5.2), which is defined as follows:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2, \tag{5.2}$$

where n is the number of samples used during training, $\hat{y}_i$ is the true output (namely, the CFD simulation results, representing the ground truth in our model) and y_i is the output of the neural network prediction. Given the entire dataset, 70% of it was used as training set, and the remaining part as validation and test set, respectively 20% and 10%.

The training of the neural networks was performed on graphics processing units nVidia Tesla V100 SXM2, the Python libraries `Keras` and `Tensorflow` were used for the setup, the training and the testing. The computational time for the training of the FCNN is about 30 minutes, instead the training of the CNN takes around 15 hours using the entire dataset of 280 samples.

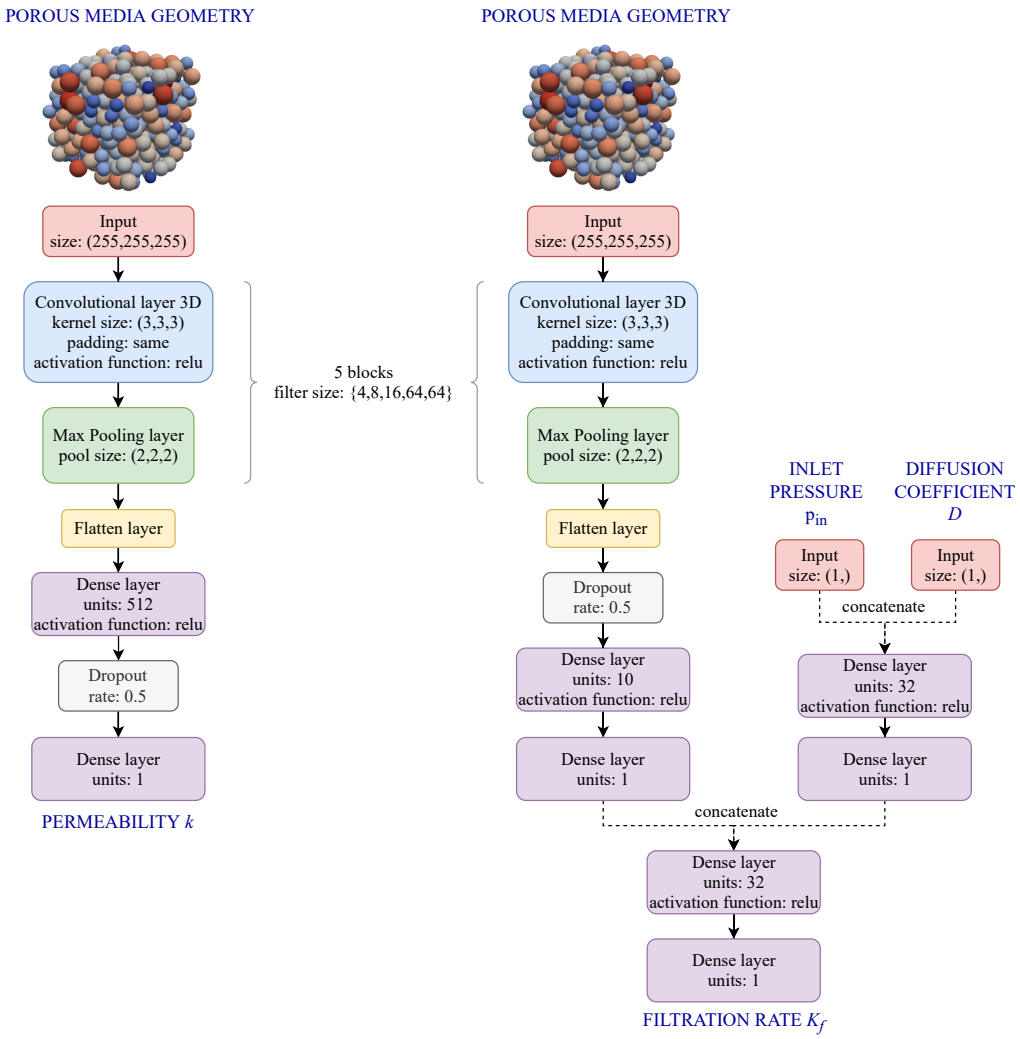

Fig. 5.7 Summary of the convolutional neural networks architectures. The CNN for the prediction of the permeability receives as input the porous media geometry - in this image the input is represented as the packing of spheres for the sake of clarity of presentation, whereas the actual input to the CNN is the Euclidean distance field from the closest solid. The CNN for the prediction of the filtration rate receives both the geometry, the inlet pressure and the diffusion coefficient. The parameters of each layer are reported in the blocks, all unspecified parameters are set as the `Keras` defaults.

Feature	Range of variation
Mean diameter [μm]	100 - 200
Standard deviation	0.0 - 0.3
Inlet Pressure [Pa]	8.2×10^{-5} - 9.74×10^{-4}
Colloid diameter [nm]	30 - 100

Table 5.2 Range of variation of the features in the dataset. The mean diameter and the normalized standard deviation (i.e.: standard deviation normalized by mean diameter) of the particle size distribution are the input features for the creation of the geometries. The inlet pressure and the colloid diameter are the input features for the CFD simulations.

5.2 Dataset

280 simulations were solved for the creation of the dataset employed for the training of the neural networks. The samples in the dataset differ in some operating conditions and in some geometrical parameters: the inlet pressure of the fluid in the porous media, the colloid dimension (thus the colloid diffusion coefficient), the mean diameter and the standard deviation of the grains size distribution. These features variation ranges as detailed in Tab. 6.1.

The CFD simulations were performed in single core on an HPC cluster equipped with 29 nodes with CPU 2x Intel Xeon E5-2680 v3 2.50 GHz 12 cores RAM of 384 GB. The computational time for each simulation is about 19 hours, Tab. 5.1.

5.3 Results and discussion

5.3.1 CFD results

In Fig. 5.8 two contour plots of the velocity and the normalized concentration for a sample simulation are reported. The pressure drop through porous media can be related to the Reynolds number through the Ergun equation (5.3):

$$\frac{\Delta P \rho}{G_0^2}\frac{d_g}{L}\frac{\varepsilon^3}{(1-\varepsilon)} = 150\frac{1-\varepsilon}{(d_g G_0)/\mu} + 1.75, \tag{5.3}$$

where G_0 is the mass flux, d_g is the surface averaged mean diameter, ε is the porosity, L is the porous medium length in the flux direction. The dimensionless numbers Re^*

and ΔP^* can be introduced so that the Ergun equation results in:

$$\Delta P^* = \frac{150}{Re^*} + 1.75, \tag{5.4}$$

For the creation of the dataset 280 simulations were performed, in Figure 5.9 each point represents a CFD simulation, instead the straight line represents the Ergun equation. As it is possible to notice there is a good agreement between the microscale CFD results and the analytical correlation.

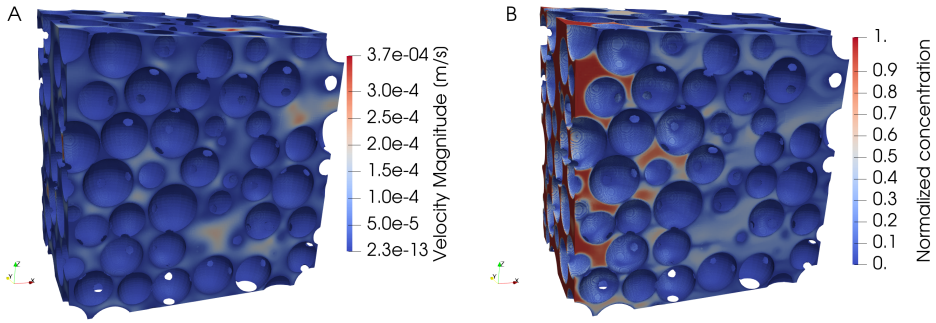

Fig. 5.8 Contour plot of the velocity (A) and of the normalized concentration (B) for a sample of the dataset.

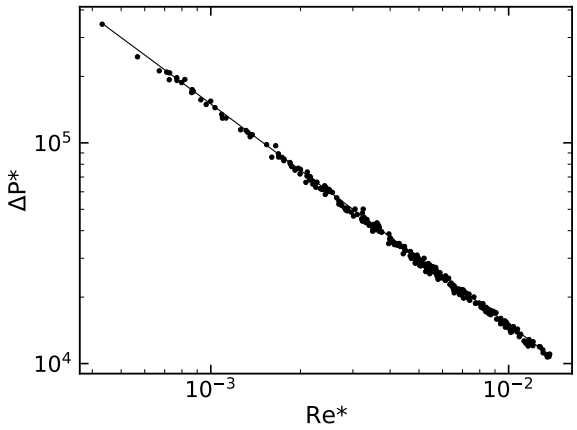

Fig. 5.9 Comparison between the CFD results (points) and the Ergun equation (line) Eq.5.3.

Concerning the transport simulations results, the upscaled filtration rate K_f, Eq. (2.27), can be related to the Péclet number via an advective Damköhler number, Fig.

5.10. Those are defined as follows:

$$\mathrm{Da} = \frac{K_f L}{q}, \tag{5.5}$$

$$\mathrm{Pe} = \frac{q d_g}{\mathscr{D}}. \tag{5.6}$$

The filtration rate is usually interpreted via a power-law relationship with the Péclet number [27, 109, 78], resulting in the following exponential relationship [83]:

$$\mathrm{Da} = A Pe^B, \tag{5.7}$$

where A and B are the coefficients that best fit the results, in our case $A = 1.20$ and $B = -0.65$. The fitting was performed by linear regression on all the data points of the dataset.

It has to be noted that while in this specific case it was possible to find a constitutive equation relating Péclet (i.e. operating conditions) and Damköhler (i.e. filtration efficiency) with a clear power-law functional form, it has been shown that for arbitrary and realistic geometries this is not the case, as it can be seen in *Boccardo et. al. (2014)* [51], resulting in the need of a model able to interpret these features - this will be explored in the following section.

This effect of complex geometries on fluid flow-reaction structures presents itself more clearly in the diffusion-limited low-Péclet range in Fig. 5.10 where a correct estimation of dispersive fluxes is complicated by the complex random geometry by the arising of terms depending on the correlation between solute concentration and flow velocity spatial fluctuations [38]. The reader is referred to this literature for more details.

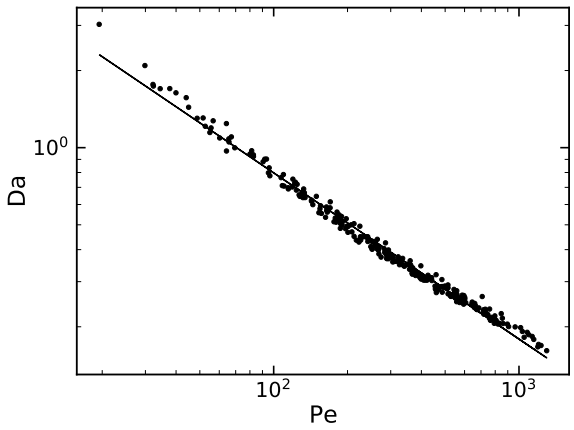

Fig. 5.10 Comparison of the CFD results (points) and the Damköhler Péclet correlation (line) Eq.5.7.

5.3.2 Neural networks

In this section the results of the neural networks modelling are presented. For the case of the FCNN, different kinds of architectures and different learning rates were explored in order to tune the model. In Table 5.3 the results are summarized. Four different learning rates, in a range commonly used, are proposed: 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}; four different architectures were tested with an increasing number of hidden layers - from one to four hidden layers - the number of components in the square brackets of Table 5.3 is the number of hidden layers and the value of each component is the number of neurons per layer. The different learning rates and architectures do not strongly affect the results. The best set of results for the prediction of the permeability can be found with a learning rate equal to 10^{-3} and architecture type: [128, 64, 32]; these parameters result in an average error of 2.46% and a maximum error of 7.62%. The best set of results for the prediction of the filtration rate is a learning rate equal to 10^{-4} and architecture type: [128, 64, 32]; these parameters result in an average error of 2.20% and a maximum error of 7.66%. The parity diagrams for these cases are displayed in Fig. 5.11.

The CNN architectures were set up as described in the previous section. In order to find the optimal architecture and training strategy, a limited number of alternatives between the many available were explored, in particular the learning rate, as for the FCNN, the scaling of the input data, and the batch normalization layers presence.

For the permeability neural network the learning rates tested were 10^{-4}, 10^{-5} and 10^{-6}, while for the filtration rate neural network 10^{-3}, 10^{-4}, 10^{-5}. The scaling proposed in this work is a min-max strategy (min equal to 0 and max equal to the maximum Euclidean distance in the dataset). In Table 5.4 the results are summarized. As it is possible to catch, the presence of the normalization layers have a powerful effect on the results. For both the prediction of the permeability and the filtration rate the best architecture is found if the batch normalization layers are absent and if the scaling is applied. The best learning rate is equal to 10^{-5} for the prediction of the permeability, which leads to an average error of 2.33% and a maximum error of 12.24%. The best learning rate is equal to 10^{-4} for the prediction of the filtration rate, leading to an average error of 4.90% and a maximum error of 12.75%. The parity diagrams for these cases are displayed in Fig. 5.12.

Results in Tab. 5.3 and Tab. 5.4 show that both the FCNN and the CNN accurately predict the permeability for the case study approached in this work. This means that the input features fed to the FCNN precisely describe the geometry, and the convolutional layers of the CNN grasp the most effective features autonomously from the image fed to the network. The FCNN predicts better the filtration rate if compared to the CNN, even though the latter still has good accuracy in absolute terms. The difference in the prediction of the filtration rate can be addressed to the architecture of the CNN, in fact, this particular architecture may need further hyperparameter tuning. However from these results it is possible to deduce that the present methodology is promising for other applications and porous media geometries. In particular the use of CNN will be more and more useful for geometries whose featurization is more complex.

Regarding the dataset, it was created from 280 simulations in order to train the neural networks, then some parameters and training strategies were explored in order to train an optimal model. The hyperparameter tuning is performed on the largest dataset because it is necessary to have results independent from the dataset size. After having chosen the best set of hyperparameters/strategies, it is useful to understand how many samples are actually necessary for the sake of obtaining a certain prediction accuracy. For this purpose we trained the same neural network with an increasing number of samples and we compared the accuracy on the same test set, for both the prediction of the permeability and the filtration rate and for both FCNNs and CNNs. In Fig. 5.13 the average prediction error, in red, and the maximum error, in blue, on the test set is reported versus the dimension of the dataset (training plus

validation set). At least 50 samples are required for reaching a good accuracy, for both types of neural networks (FCNN in solid line and CNN in dashed line). In Fig. 5.14 the same study is proposed for the filtration rate prediction, in this case more samples are required, around 100 for FCNN, and 200 for CNN. Generally this is due to the highest number of parameters that characterize this problem compared to the permeability prediction. The need for more samples in the case of CNN for the filtration rate prediction is probably due to potential improvements on the structure of the treatment of multiple topological/integral inputs.

Table 5.3 Hyperparameter tuning of the FCNN. The average error and the maximum error on the test set (in brackets) are reported for the different architectures and learning rates. Underlined, the chosen best architecture/hyperparameter coupling.

	Permeability			
	10^{-3}	10^{-4}	10^{-5}	10^{-6}
[32]	2.45% (8.18%)	2.55% (8.03%)	2.52% (8.52%)	2.51% (9.0%)
[64,32]	2.49% (8.31%)	2.46% (8.53%)	2.51% (8.88%)	2.41% (9.09%)
[128,64,32]	<u>2.46% (7.62%)</u>	2.46% (8.49%)	2.42% (8.30%)	2.43% (8.67%)
[256,128, 64,32]	2.47% (8.44%)	2.49% (8.73%)	2.51% (8.04%)	2.49% (8.51%)
	Filtration rate			
	10^{-3}	10^{-4}	10^{-5}	10^{-6}
[32]	2.23% (10.57%)	2.27% (9.34%)	2.26% (8.24%)	2.81% (9.21%)
[64,32]	2.05% (10.56%)	2.21% (9.74%)	2.54% (8.11%)	2.67% (11.74%)
[128,64,32]	2.27% (11.7%)	<u>2.20% (7.66%)</u>	2.41% (9.64%)	2.69% (12.0%)
[256,128, 64,32]	2.17% (10.12%)	2.34% (9.13%)	2.28% (9.09%)	2.44% (8.99%)

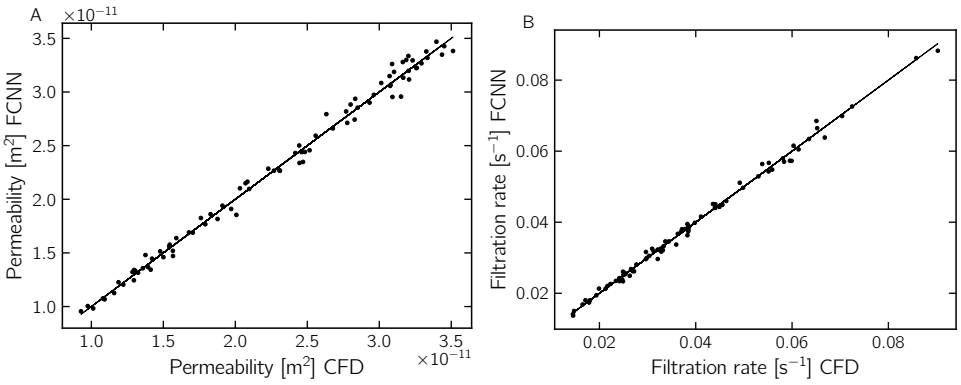

Fig. 5.11 Parity diagrams for the predictions on the test set of the permeability (A) and the filtration rate (B) by the use of FCNN.

Permeability				
Norm.	Scaling	10^{-4}	10^{-5}	10^{-6}
✓	✗	5.57% (22.50%)	15.00% (48.40%)	27.90% (95.07%)
✗	✓	3.14% (15.93%)	2.33% (12.24%)	2.39% (12.24%)
✓	✓	7.27% (21.57%)	17.44% (55.34%)	24.65% (58.34%)
✗	✗	39.74% (100.44%)	3.03% (13.66%)	2.52% (12.53%)
Filtration rate				
Norm.	Scaling	10^{-3}	10^{-4}	10^{-5}
✓	✗	5.90% (15.58%)	6.62% (19.00%)	8.55% (27.90%)
✗	✓	5.02% (14.99%)	4.90% (12.75%)	6.44% (23.33%)
✓	✓	6.68% (24.91%)	7.85% (29.09%)	8.69% (32.49%)
✗	✗	25.01% (65.00%)	25.83% (66.67%)	6.40% (18.20%)

Table 5.4 Hyperparameter tuning of the CNN. The average error and the maximum error on the test set (in brackets) are reported for the different learning rates, for the presence of batch normalization layers and for the scaling of the input data. Underlined, the chosen best architecture/hyperparameter coupling.

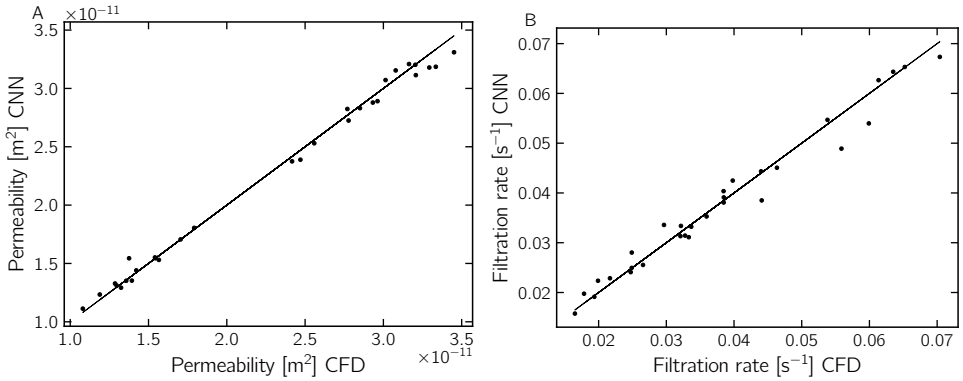

Fig. 5.12 Parity diagrams for the predictions on the test set of the permeability (A) and the filtration rate (B) by the use of CNN.

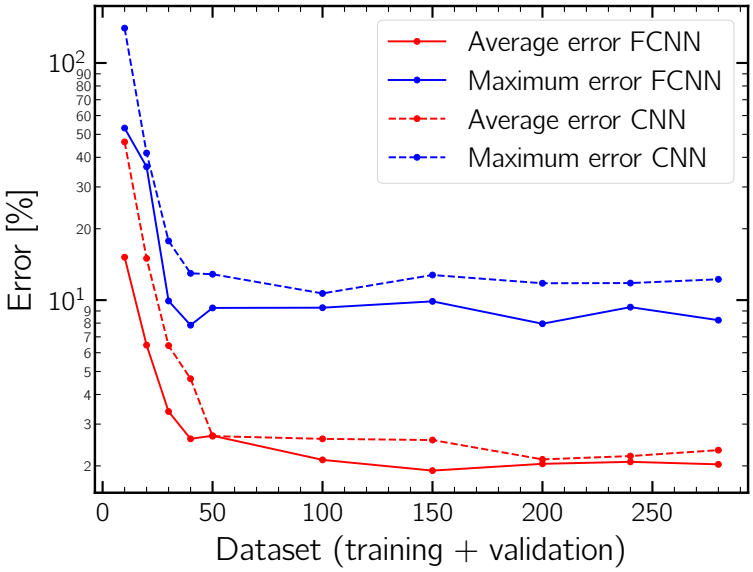

Fig. 5.13 Effect of the number of samples in the training and validation set on the prediction of the permeability for the FCNN and the CNN - the networks were tested on the same test set.

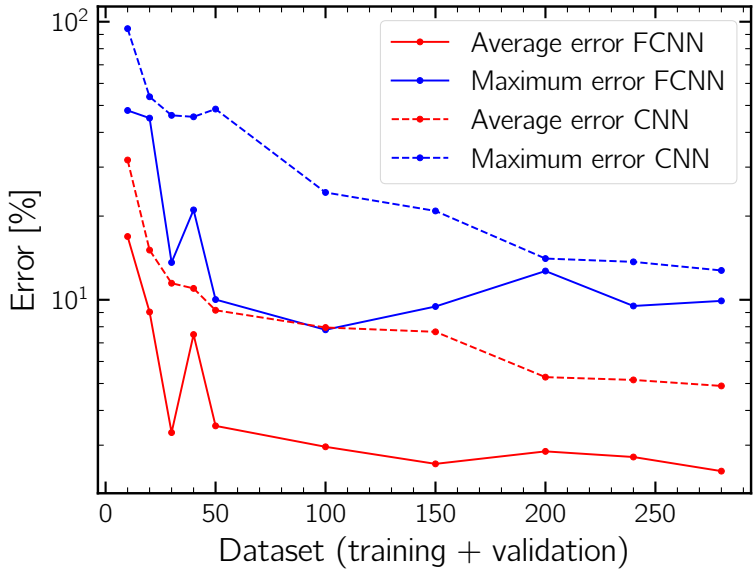

Fig. 5.14 Effect of the number of samples in the training and validation set on the prediction of the filtration rate for FCNN and CNN - the networks were tested on the same test set.

5.4 Conclusions

Prior work has documented the efficacy of training neural networks for the prediction of the permeability in various types of porous media systems [104, 111, 60, 59, 112]. However, the majority of these works has focused just on models that deal with geometrical descriptors. Since the applications of porous media systems in chemical engineering are usually related to catalytic chemical reactions and filtration (more in general, *fast surface reactions*), we developed a methodology for the prediction of the filtration rate in porous media. Both fully connected and convolutional neural networks were tested in our workflow [16]. The main difference between these two approaches is the input to the model, in the first case hand selected features are the input to the FCNN, in the second case the entire image is fed to the networks, which is able to autonomously detect the features in the image for the prediction of its target output.

While the two models exhibit similar (and satisfactory) accuracy, it is notable that a comparable number of CFD simulations between the two cases were sufficient to reach such accuracy, notwithstanding the clearly higher complexity of the convolutional model with respect to the classical fully connected network. Even more

so, in the light of the much higher flexibility of use of the model based on CNN, which (at the cost of higher computational cost for its training) will prove to be much more useful by means of opening to this kind of analysis even systems featuring complicated geometries - as are frequently found in chemical engineering both in purification/filtration apparatuses, and in packed bed catalytic reactors. Beside its usefulness in dealing with complicated *structures*, these convolutional models have already proven in this work to be able to treat complicated *phenomena*, by providing a way to robustly construct accurate models even in cases for which it may not be immediate (or feasible) to choose integral input features with which to build a predictive analytical model - case in point the problem of colloidal filtration, as explored in this work. The long term application of this methodology will be the use of neural networks for the prediction of other microscale properties that lack analytical models that correlate them to microscale properties.

Chapter 6

Multiscale convolutional neural networks for fields prediction

What seems a natural evolution of the research presented so far is to build CNNs that can be employed to surrogate the microscale local solution of the flow and transport equations. CNNs with encoding and decoding architectures have been used to train surrogate models able to predict the flow field in different microscale porous media systems [61–63]. Multiscale neural networks (MSNet) [69] came out to be a preferred alternative to the previous ones, from both the computational point of view and, notably, their generalization capability. In fact, it is possible to train the network with larger geometric samples than what more classical approaches allow, which is fundamental when dealing with representative elementary volumes of heterogeneous geometries and/or complicated transport phenomena.

Nonetheless, as the main effort in the last years was addressed to the development of architectures for the microscale prediction of flow fields, little was done to expand these methodologies to more complex physical systems, which are of common experience in the chemical engineering field. The complexity does not arise only from the different prediction objectives, but also from the kind of input necessary for the network. When the main objective is the prediction of the permeability, as in the above mentioned studies, it can be evaluated from the prediction of the flow fields. In laminar flow regime the permeability is just related to the geometry of the porous media [2, 113], so a dataset at constant pressure drop is sufficient. Instead,

The content of this chapter, in a modified form, has been published in Marcato et al. (2023) [18]

in coupled flow and solute transport problems, different pressure drops (or other operating conditions) impact the ultimate solute concentration field even if the flow regime does not change.

In this chapter we provide proof of the capability of these CNN models to surrogate CFD simulations of transport and reaction under a wide range of different transport conditions that are common in chemical engineering problems. It is evident that these models, in the limit of their accuracy, can integrate the knowledge that can be extracted from CFD simulations and used to construct workflows needing a large number of “calls" to the CFD model, like in multiscale modelling. In general, deep learning based surrogate models can help when it is hard or impossible to build appropriate constitutive equations for upscaled parameter estimation, due to difficulty in choosing the relevant microscale parameters (or dimensionless numbers) in order to construct the right functional form for the needed predictive model. Some other times still, and this is very relevant for porous media, it may be impossible to find or evaluate these parameters, like in the case of evaluating the effect of a complex geometry on flow, transport, reaction, and their interplay. A model able to “see” and interpret geometries and shapes as they are (as convolutional neural networks provably do in many different contexts) instead of through the lens of integral parameters (porosity, pore shape, surface area, et c.) would be more successful.

Thus, the objective we set for this work was to obtain a network able to predict concentration fields in porous media in the case of a heterogeneous surface reaction (or equivalently filtration) when dealing with a wide range of Reynolds and Péclet number. To this end, we modified the original MSNet and trained it with a “ground truth” dataset of CFD simulations of flow and reactive transport over many geometrical realizations and different operating conditions. As a result of this study, beyond only having a working neural network for the solution of this problem, we also propose a wide-ranging study of the different possible input features employable, resulting in the best set of features that allows the network to satisfactorily generalize its predictions to new geometries and operating conditions.

6.1 Dataset

The training of the neural network requires a dataset that encompasses a wide variety of geometries and operating conditions, and in this work we carried out 800 CFD

simulations of flow and transport in bi-dimensional porous media, all with different operating conditions and domain geometries. We've chosen to perform this quite extensive number of simulations precisely for the purpose of exploring the effect of the size of the training dataset on the neural network performance.

The samples of the dataset are chosen so to explore a wide range of diameter of the circles, porosity of the 2D packing, pressure drop across the porous medium, i.e. the Reynolds number, and diffusion coefficient, i.e. the Péclet number. These features range of variation is listed in Tab. 6.1.

The mesh created for the CFD simulations is stair-stepped (*castellated* in OpenFOAM parlance) in order to ease the use of the resulting concentration fields for the training of the neural networks without needing an interpolation from a body-fitted mesh (with non-Cartesian structure) to the matrix-like data structure needed by the neural network training calculations.

Then, a grid independence study was performed in order to choose the size of these computational grid elements, by monitoring the medium permeability and the average concentration in the domain with varying mesh cell size. Since in the dataset of CFD simulations a wide range of operating conditions was explored, the grid independence study was performed on the sample subjected to the highest Péclet number, the most critical condition from the computational point of view, due to the smaller boundary layer on the surface of the grains caused by the null concentration boundary condition. In Fig. 6.1 the study is reported, with the green line highlighting the chosen mesh strategy, corresponding to a linear discretization of the square domain of 1536 cells. The relative error between the average concentration and the permeability calculated from the simulations performed with the chosen meshing strategy and the most refined grid tested is, respectively, 0.5% and 0.1%. The results of the chosen strategy can thus be considered fully grid independent.

Parameter	Range of variation	
p	$0.30 - 0.50$	Pa
d_g	$100 - 200$	μm
ε	$0.5 - 0.65$	(-)
$\mathscr{D}$	$3.13 \times 10^{-11} - 5.71 \times 10^{-10}$	$m^2 s^{-1}$

Table 6.1 Range of variation of the features chosen for the creation of the geometries and the solution of the CFD simulations.

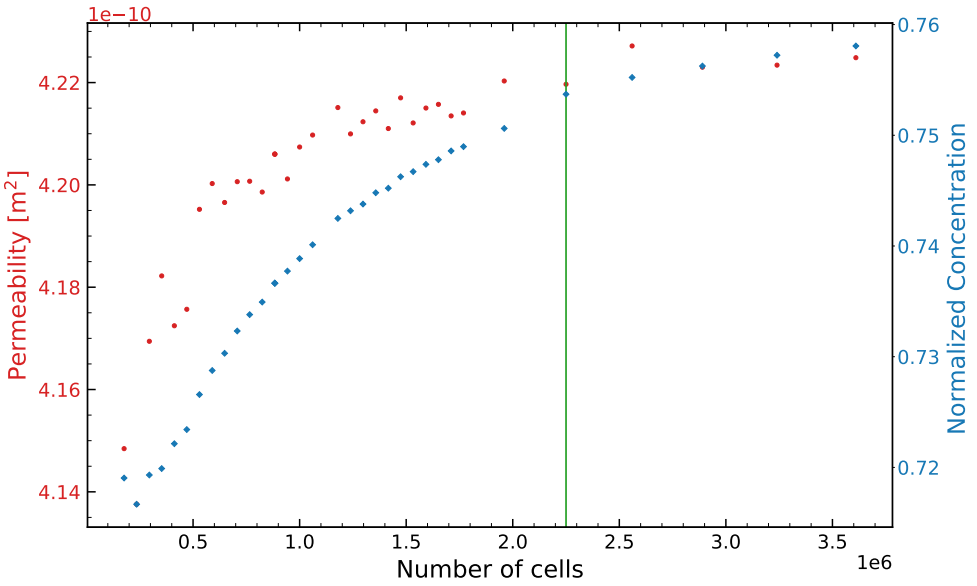

Fig. 6.1 Grid independence study performed for the CFD simulations. The average concentration in the domain (blue diamond markers) and the permeability (red bullet markers) were monitored. The green line indicates the grid independent result whose strategy was chosen.

6.2 Multiscale convolutional neural networks

CNNs are a class of neural networks suited to deal with grid-like objects as input features [6], like images. The CNN layers implement the convolution operation:

$$y = f\left(\sum_{i=1}^{F} x * k_i + b_i\right), \tag{6.1}$$

where $*$ denotes the convolution operation, x is the input, y is the output of the operation, f is the activation function, k_i is the kernel, F is the number of kernels, and b_i is the bias term. The kernel is a trainable array of floating point numbers of a certain size that is applied on the image. In this work the input features are images so the filter is two-dimensional and of size 3-by-3 which is the most computationally efficient size for GPU computations [101].

CNNs have exhibited excellent performance in deep learning tasks compared to classical fully connected neural networks both in terms of generalization capability and computational cost of the training [102]. This is partly due to the fact that convolutional kernels share their parameters, because the same filter slides on the image and is applied in different regions, thus every output is connected just on

a small portion of the input, which is referred to as sparse connectivity. As a consequence, CNN are characterized by equivariance to translation, so the layers learn the influence of the images features independently from their location.

The training of neural networks aims to optimize the trainable parameters of the network to minimize the loss function, which is done with the back-propagation algorithm. In this work we used the `Pytorch` [114] implementation of Adam [107], which is one of the most employed and stable gradient descent optimizers [110].

As with other deep learning algorithms, CNNs may suffer of gradient vanishing issues [100], thus the choice of an appropriate activation function is crucial to assure the contribution of each neuron to the final prediction. The selection of the activation function is one of the many choices to be made for the design of the neural network model.

The most used activation functions nowadays are the Exponential Linear Units (ELUs), nevertheless activation functions like the Gaussian Error Linear Unit (GELU) seems to be a promising alternative for convolutional networks [115]. For this work we tested both the use of a ELU, the continuously differentiable exponential linear unit (CELU), and GELU, as it will be discussed in the following section of the chapter. Since there was no apparent impact of the activation function on the accuracy of the predictions for our trainings, we decide to use CELU, the original and widely tested activation function of MSNet. The continuously differentiable exponential linear unit (CELU) reads as follows [116]:

$$\mathrm{CELU}(x) = \max(0,x) + \min\left(0, \alpha \cdot \left(e^{\frac{x}{\alpha}} - 1\right)\right) \tag{6.2}$$

where α is a parameter controlling saturation for negative inputs [117]. CELU avoids saturation of the output unlike other activation functions traditionally employed in neural networks, such as the sigmoid function, and avoids the *dead ReLU* problems being an ELU activation function [118].

Now the main characteristics of MSNet are summarized. This architecture was originally presented by Santos et al. [69], so the reader may refer to this work for a more detailed description.

MSNet is a convolutional neural network structured in scales, where each scale aims to predict the property field structure at a different level: a sketch of the mechanism is presented in Fig. 6.2. The architecture has two purposes: 1) letting

each scale focus on different length scales of the field (i.e. the scale with the largest field of vision can capture the global trend of the field) and 2) allowing to train computationally large images in a single-GPU (which is not feasible with a model such as U-Net [119] or Res-Net [120]). The scale that deals with the full domain size is denoted as *scale 0*, subsequent scales (1 through N) receive the same input as the previous scale but coarsened by a factor of two. The last scale N receives the coarsest representation of the input feature, consequently, its output is the coarsest representation of the predicted field. This output is refined and fed to the previous scale together with the refined input feature. The output of the $N-1$ scale is summed with the refined output of scale N. This procedure is replicated for all the intermediate scales, and can be summarized as follows:

$$\hat{y}_{N-1} = \mathrm{CNN}_{N-1}(X_{N-1}, \mathbb{R}(\hat{y}_N)) + \mathbb{R}(\hat{y}_N), \tag{6.3}$$

where $\hat{y}_N$ is the predicted field at scale N, CNN_N is the fully convolutional neural network of scale N, X_N is the set of input features coarsened by a factor 2^N, $\mathbb{R}()$ is the refinement operation.

The coarsening of the input features and the output fields is performed by a nearest neighbor averaging, meaning that the value of every 2^2 group of pixels is averaged into a single pixel. Instead, the refinement procedure consists in a masked nearest-neighbors re-scaling that maintains the shape of the geometry of the solid portion. For a more detailed explanation of the operation the reader may refer to Section 2 of the original MSNet work [69]. The coarsening and refinement operations conserve the average over space, and the coarsening of a refined image gives back the original image. It is important to notice that the coarsening operation is performed just once before the training in order to calculate the coarse fields of the input features and of the true concentration fields, that will be employed during the training as ground truth at each scale. The time required to perform the refinements is not rate determinant during the training, as just 0.8% of the time of the training is spent in the refinement calculations.

The loss function employed in the training of MSNet takes into account the prediction error of each scale. The contribution of each scale is given by the mean squared error between the predicted field $\hat{y}$ and the true field y (coarsened at the corresponding level) divided by the variance of the true field, $\sigma^2_{y_i}$, so the global loss

function L is:

$$L = \sum_{s=0}^{N} \sum_{i=0}^{N_S} \frac{\left\langle \left(y_{i,s} - \hat{y}_{i,s}\right)^2 \right\rangle}{\sigma_{y_i}^2}, \tag{6.4}$$

where the index s refers to the scale, i refers to the sample, N_S is the number of samples in the training set, the operator $< . >$ refers to the spatial average on the domain.

The architecture of the neural network is the same for every scale and is summarized in Fig. 6.2. The networks are fully convolutional and the size of the images is maintained along the layers of the scale. The first four blocks are made by a convolutional layer whose kernel size is 3^2, followed by a normalization layer, after that the CELU activation function is applied. A fifth block without activation function is then added in order to not constrain the output.

The number of kernels of each convolutional layer depends on the scale: 2^{2s+1}. Finally, the network ends with a convolutional layer with a single kernel size of 1^2 in order to reduce the dimensionality of the output to a single image, which is the predicted field.

Conceptually MSNet can grasp both short range and long range correlations in the field thanks to using the same features at different resolutions, resulting in good generalization capability for the flow field prediction, and the related permeability prediction. From the computational point of view, using the same input features at different resolutions allows for a higher number of kernels at the coarsest scales and a lower number of them at the finest scales: as a consequence, the training is faster and the memory requirements are decreased.

MSNet for the prediction of concentration fields

The MSNet architecture was originally conceived for the prediction of flow fields. Since the main objective was the prediction of the permeability, a dataset of simulations in laminar conditions at constant pressure drop was employed [69]. As mentioned, while permeability is strictly related to fluid flow, it is determined only by the geometric features of the porous medium [2, 113]: as a consequence, this geometrical description was the only input feature needed by the network for the prediction of the field. In this work the architecture was employed for the prediction of concentration fields, and the main effort was headed in the choice of the most

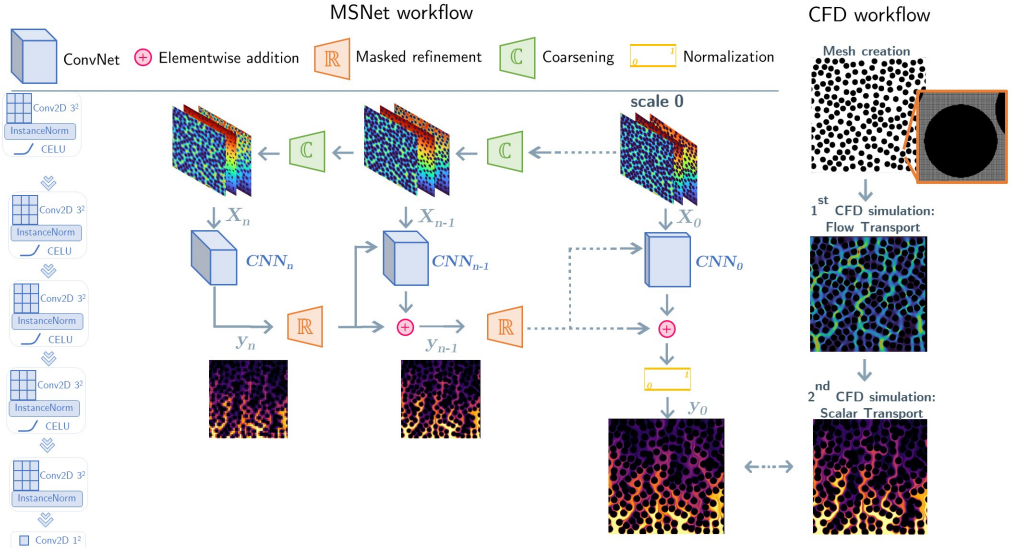

Fig. 6.2 MSNet workflow (left) and CFD workflow (right). A set of input features is chosen for the prediction of the steady-state concentration field (presented in rainbow color scale in the figure). The input features tested are the Euclidean distance transform, the time of flight, the local thickness, the operating pressure drop, and the diffusion coefficient. Each scale is made by a convolutional network block that is detailed on the left. The coarsening and masked refinement operations are employed to transfer information between scales. The CFD workflow is summarized on the right. The output of the simulations is the ground truth for the training of neural network.

appropriate input features. In this section the input features tested are described and information about their extraction is provided, while their effectiveness on the prediction of the concentration field is discussed in the following section. In the proposed dataset both the porous media geometries and the operating conditions of transport affect the concentration fields, so the neural network must be provided with both features.

Concerning the geometrical description of the porous media the features tested are: the Euclidean distance transform, the linear variation distance from a boundary, the time of flight, and the local thickness, Fig. 6.3.

The Euclidean distance transform is applied to the binary images of the porous media, where 0 labels the solid phase and 1 the fluid phase. As a result, for each pixel of the fluid phase the Euclidean distance is calculated from the closest solid pixel, i.e. the closest solid grain, Fig. 6.3A.

As stated in the previous section, CNNs are invariant to translation, but the concentration values in the pore space are linked to their position with respect to the inlet boundary. Even though MSNet is able to grasp the spatial correlations at different scales, being the transport of species concentration a strictly oriented phenomenon, it is necessary to provide the neural network with this information. It can be conveyed by a simple linear variation of the coordinate in the flow direction, Fig. 6.3B, or with more informative features that can account for the tortuosity of the porous medium.

The time of flight describes the tortuosity in the porous medium as the shortest distance of a point from a chosen boundary, Fig. 6.3C. In order to calculate it a boundary value problem of the Eikonal equation is solved:

$$F(x)|\nabla(t(x))| = 1, \tag{6.5}$$

where $F(x)$ is the speed at which the boundary x evolves in time t. The fast marching method implemented in scikit-fmm [121] was employed in this work to compute the time of flight.

The boundary selected is the inlet boundary of the transport simulations, and the speed field is the Euclidean distance field, Fig. 6.3A. Commonly the speed field is set with a constant value in the pore space and zero in the solid phase: however, the resulting time of flight field does not highlight the preferential paths and

a mostly linear variation of the coordinate is returned, since the circular obstacles have a negligible influence on the boundary path. If a non constant speed field as the Euclidean distance is employed, it is possible to extract a more informative feature describing the preferential paths in the porous media that are of the utmost importance in the shape of the concentration fields.

Other geometric features have been taken into consideration to aid the neural network for the non-trivial prediction of the concentration field, such as the local thickness, i.e. size of the maximum inscribed sphere, Fig. 6.3D. To obtain this feature, we used the approach implemented in PoreSpy [122] which consists in finding which group of pixels can accommodate a sphere of a given radius. Given the number of sizes and the bins in the size distribution, the algorithm detects the largest pore applying the Euclidean distance transform to the image. Then the algorithm searches for all the smaller spheres between zero and the largest one by using the Fast Fourier Transform convolution.

All the above mentioned features are not computationally expensive, the time required for their calculation is in the order of a few seconds per sample. As a consequence the choice of these features can easily scale to a three-dimensional dataset.

The operating conditions characterizing the dataset are the pressure drop across the porous media, and the diffusion coefficient of the chemical species transported. The latter is provided to MSNet as an image with the value of the diffusion in each pixel of the pore space. The pressure drop feature can also be supplied to the network in this way, like the diffusion coefficient, or it can be merged with the information about the distance from the inlet boundary (Fig. 6.3B) since the pressure decreases along the flow direction. Scaling the features of this kind (linear variation and time of flight) by the value of inlet pressure allows for a more compact set of features to pass to the CNN. Lastly, a unique feature for the operating conditions such as the ratio between the pressure drop and the diffusion coefficient was tested in this work too.

Concerning the scaling of the input features, those are scaled by their mean value over the whole dataset. In the case of the output concentration field instead, no further scaling is needed, as it is already normalized between 0 and 1 from the CFD simulations.

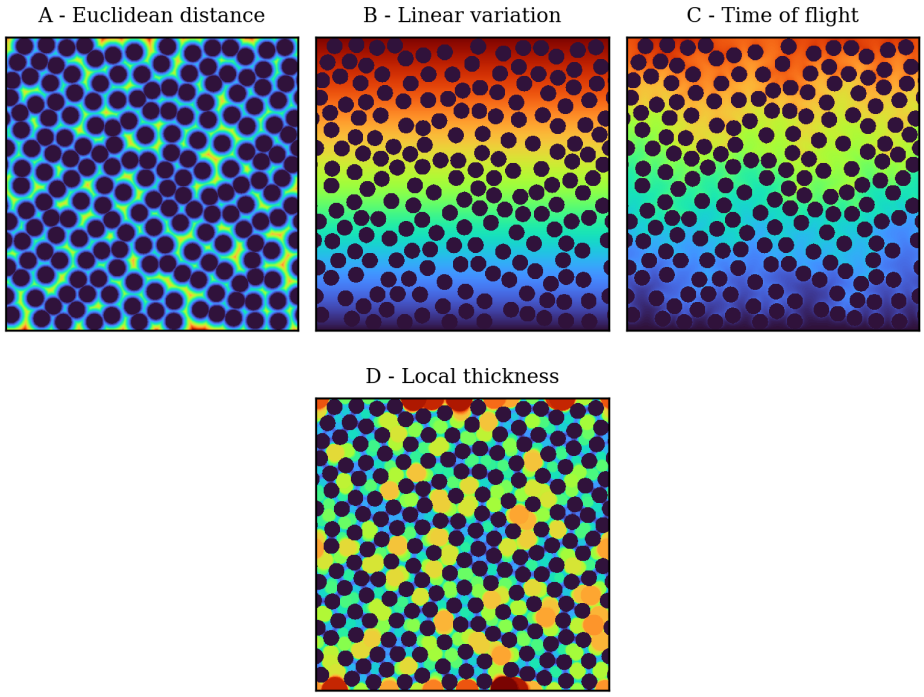

Fig. 6.3 Geometrical input features tested as input to MSNet for the prediction of the concentration fields. A - Euclidean distance transform of the binary image; B - Linear variation of the coordinate in the main flow direction; C - Time of flight in the main flow direction; D - Local thickness. Colors represent values normalized within each metric's range, transitioning from blue (minimum) to red (maximum).

Since the output concentration is always between 1, at the inlet boundary, and 0, at the grain surface, a normalizing layer was added at the output layer of MSNet to prevent the prediction of nonphysical results.

The number of scales chosen for the MSNets trained in this work is 6, which represents a good trade off between limiting the number of trainable parameters and obtaining a well coarsened representation at the last scale. In fact, the field of vision (FoV), i.e. the number of pixels of the input affecting each pixel of the output, is defined as follows for MSNet:

$$\mathrm{FoV}_{\mathrm{MSNet}} = (L(k_{size} - 1) + 1)2^n, \tag{6.6}$$

where L is the number of convolutional layers of the network (5 in this MSNet), k_{size} is the size of the kernels (3 in this MSNet), and n is the number of scales. Given the size of the CFD simulations fields (1536×1536), the FoV of the finest scale ($n = 0$)

is equal to 11 voxels and the one of the coarser scale ($n = 5$) is equal to 352. We performed the trainings on NVIDIA Volta V100 GPUs, Nvlink 2.0, 16GB.

6.3 Results and discussion

6.3.1 CFD simulations for the creation of the dataset

The neural network for the prediction of the concentration field is trained on a dataset made by CFD simulations. Two CFD simulations are solved to obtain the (normalized) concentration fields, as summarized in the workflow of Fig. 6.2. At first the velocity field is obtained from the coupled solution of the continuity and Navier-Stokes equations, after that the scalar transport is simulated by the solution of the advection diffusion equation.

Each complete simulation constitutes a sample point of the dataset. The total number of simulations solved is larger than the minimum set size required for the training of an MSNet for concentration prediction: this was done to perform a sensitivity analysis on the number of actual samples required to obtain a satisfactory accurate prediction.

An HPC cluster was employed to solve many (single core) simulations at the same time. In this way, 800 simulations have been solved. The time required to create the castellated mesh, and to solve the two CFD simulations is about 20 hours on an HPC cluster equipped with 29 nodes with CPU 2x Intel Xeon E5-2680 v3 2.50 GHz 12 cores RAM of 384 GB.

A random combination of the input features in the range displayed in Tab. 6.1 results in a range of Reynolds numbers of 3.96×10^{-4} - 1.58×10^{-2}, and in a range of variation of the Péclet number of 23 - 1500. Thus a wide range of transport conditions in laminar flow was explored in the dataset resulting in a challenging dataset for MSNet.

In Fig. 6.4 the flow field and the concentration fields for three samples of the dataset are shown. The lower is the Péclet number, the higher is the contribution of diffusion with respect to the convective term (which regulates the residence time of the chemical species in this reactive system). As a result, the average concentration

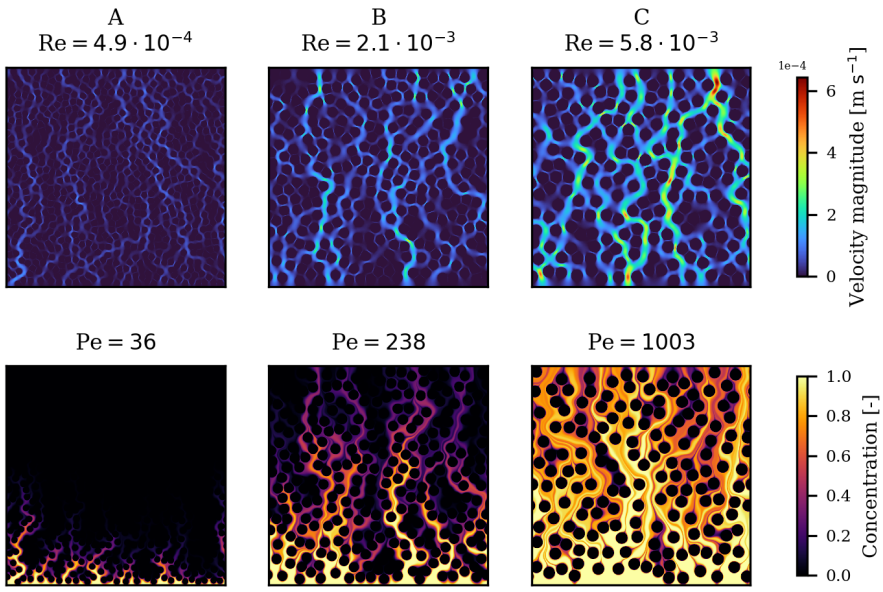

Fig. 6.4 Contour plots of the velocity and concentration fields for three samples of the dataset. A lowest Péclet number - the diffusive term prevails. B intermediate Péclet number. C highest Péclet number - advection term prevails, the chemical species flows easily through the porous medium.

in A is lower than the average concentration in B, which is lower than the average concentration in C.

6.3.2 Neural networks

Choice of the input features

The choice of the input features for the prediction of the concentration field is of the utmost importance for the generalization capability of the neural network.

The prediction accuracy of the networks is visually depicted with the fields of local errors between the predicted concentration field and the CFD result. In addition, three metrics are employed in order to easily compare the prediction accuracy on the test set:

- The percentage error on the average concentration of the field;
- The root mean squared error (RMSE) of the concentration profiles in the flow direction;

- The RMSE of the concentration profiles in the direction perpendicular to the flow.

The result of these metrics for each sample of the test set is averaged in order to obtain a metric for the entire test set.

	A	B	C	D	**E**	F	G
Linear variation normalized by the pressure drop		✓	✓				
Pressure drop	✓			✓		✓	
Pressure drop/Diffusion coefficient					✓		✓
Diffusion coefficient	✓	✓	✓	✓		✓	
Euclidean distance	✓	✓	✓	✓	✓	✓	
Time of flight			✓	✓	✓	✓	✓
Local thickness						✓	
$< e >_{test}$ on average concentration	13.5%	5.7%	5.3%	3.3%	**3.3**%	5.3%	5.5%
$RMSE_{test}$ on concentration profile parallel to flow	0.104	0.034	0.027	0.023	**0.024**	0.032	0.027
$RMSE_{test}$ on concentration profile perp to flow	0.074	0.054	0.042	0.042	**0.042**	0.044	0.044

Table 6.2 Input features tested for the training of MSNet. The different combinations are compared on the error on the prediction of the average concentration, on the RMSE of the concentration profiles in the flow direction and in the perpendicular direction to flow.

In Tab. 6.2 the different combinations of input features tested are summarized, and the metrics previously described are reported for each set of features. In case A, just the basic features are provided, namely the operating conditions (pressure drop and diffusion coefficient) and the geometrical conditions, by means of the Euclidean distance, Fig. 6.3 A. The three metrics show the highest error, and the predicted field is not physical since there are zones in the fluid where the concentration increases moving towards the outlet boundary; in a filtration problem, or a depleting reaction, this obviously should not happen, Fig. 6.5. This is due to the lack of information about the distance from the inlet boundaries: in fact, precisely because the convolutional layers are invariant to translation as mentioned earlier, it is necessary to provide the network with the position with respect to the inlet boundary.

If the distance from the inlet boundary is given as a coordinate linear variation, Fig. 6.3B, normalized by the pressure drop, case B, the accuracy remarkably improves. The error on the average concentration in the fields decreases from 13.5% to 5.7%, the RMSE for the concentration profile in the flow direction decreases from 0.104 to 0.034, and the RMSE for the concentration profile in the perpendicular direction to flow decreases from 0.074 to 0.054.

The time of flight, Fig. 6.3 is an alternative option for giving the information about the distance from the input boundary to the network. Compared to the linear variation of the coordinate, the time of flight embeds the description of the tortuosity in the geometry and underlines the presence of preferential paths. If both features (time of flight and linear variation of the coordinate) are fed to the network, case C, the improvement of the generalization capability does not increase in a relevant manner.

On the other hand, if just the time of flight is employed, together with the Euclidean distance, the diffusion coefficient and the pressure drop, case D, the accuracy remarkably increases. This results in an error on the average concentration of 3.3%. The redundant information about the distance is not useful for the network, so it is preferable to just provide the network with the time of flight.

Then, since it is desirable to reduce the number of input features, in order to decrease the memory load on the GPU and the number of trainable parameters, and as a consequence the computational cost of the trainings, the compression of the two operating conditions features into a single one was tested. Employing the ratio of pressure drop and diffusion coefficient, case E, the prediction accuracy remains unchanged, so this solution is preferable to the previous one.

Following the same approach, a training without the Euclidean distance feature was performed in order to reduce the number of input features to the minimum, case G. In this case the generalization capability is not preserved, in fact, the error on the average concentration increases again from 3.3% to 5.5%. Thus, the Euclidean distance can not be missed in the set of features for the prediction of the concentration fields.

The last set of features tested is the same of case D but with the additional feature of the local thickness, case F. The new geometrical feature does not improve the accuracy, so it's not worth it to add local thickness to the set of input features.

Given the presented results, the best set of features was found to be the one from case E, where the operating conditions are provided in a single feature in the form of the ratio of the pressure drop and the diffusion coefficient, and the geometric features into two features, the Euclidean distance and the time of flight. This last feature is particularly effective since the preferential paths information is clearly correlated to the concentration field patterns.

The results of the training of MSNet with the best set of features are reported in Fig. 6.6, where the predictions for four samples of the test set are shown. The percentage error on the average concentration between the result of the CFD and the prediction of MSNet is in the label of each sample. One point of note is that, beyond these quantitative error measures, the network is able to predict qualitative transport features with satisfactory precision, for example the low-concentration tails in the wake of the grains.

Since the samples in the test set differ from those in the training set for all the input features, including circles diameter and most importantly their placement, it is significant to note that these results describe the performance of this model on a generalized form of this transport and reaction problem. If this model were employed on a similar system (one never seen in the training of the network), equivalent performances to those reported would be expected.

In Fig. 6.7 the profiles corresponding to the fields of Fig. 6.6 represent an alternative way to evaluate the prediction error of the network. Also in this case, it can be appreciated that the average profiles predicted by MSNet quite closely follow the shape of the profiles calculated from the results of the CFD simulations.

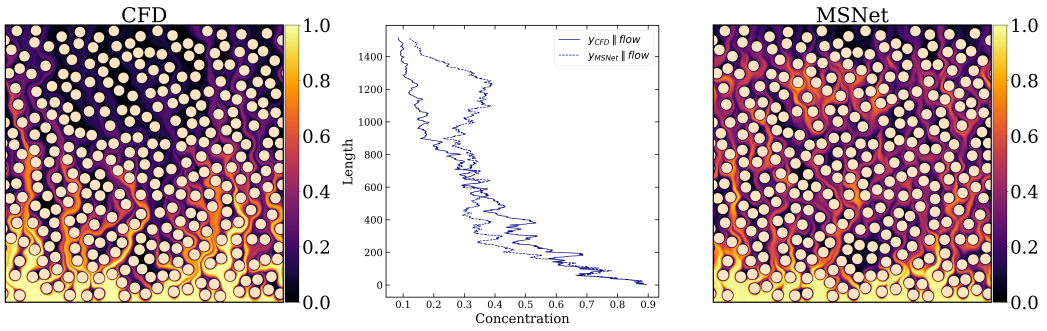

Fig. 6.5 Prediction of MSNet $\hat{y}$ for case A of Tab. 2. If the distance from the inlet boundary is not provided to the network the resulting concentration field shows non physical behaviors, like the increase of the concentration along the flow direction. In the profile plot the increase of the concentration is evident (CFD: solid blue line, MSNet prediction: dashed orange line).

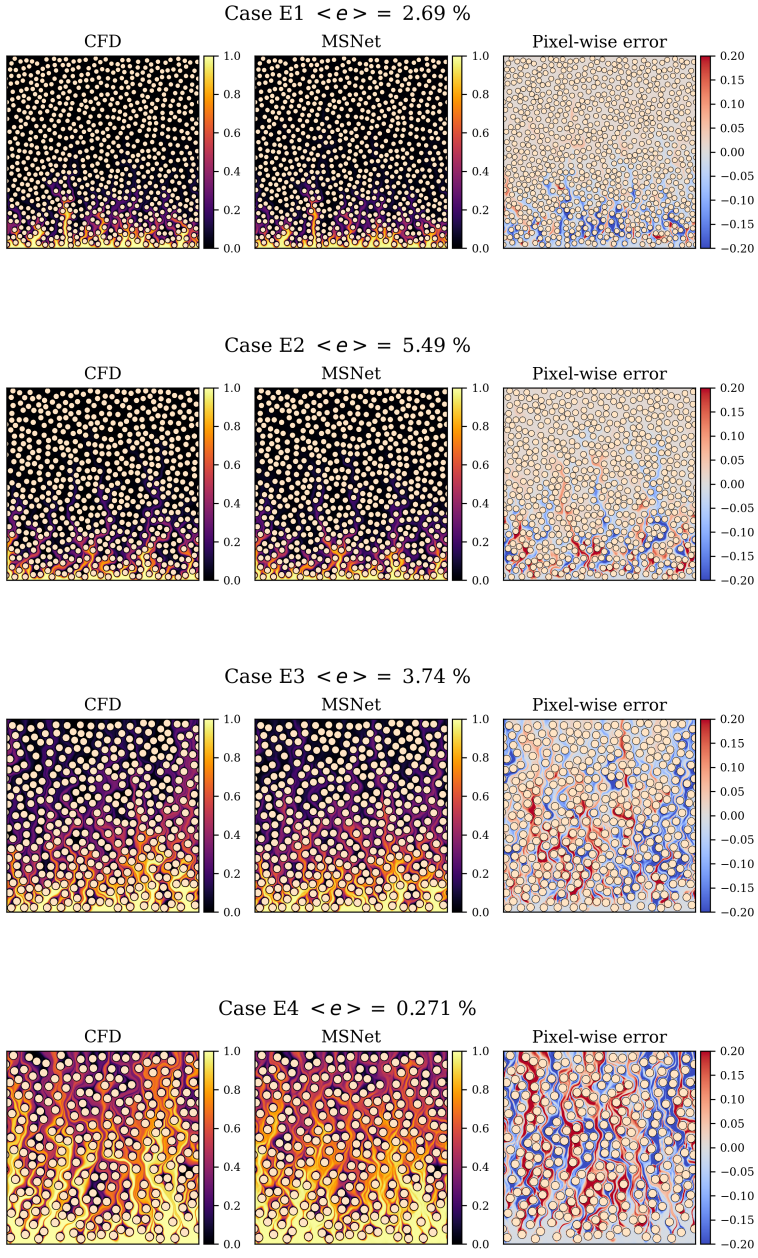

Fig. 6.6 Prediction of MSNet for four samples of the test set. From left to right: concentration field resulting from the CFD simulations, concentration field predicted by MSNet with the input features of case E Tab. 6.2, field of local error computed as the pixel-wise difference between the CFD result and the MSNet prediction.

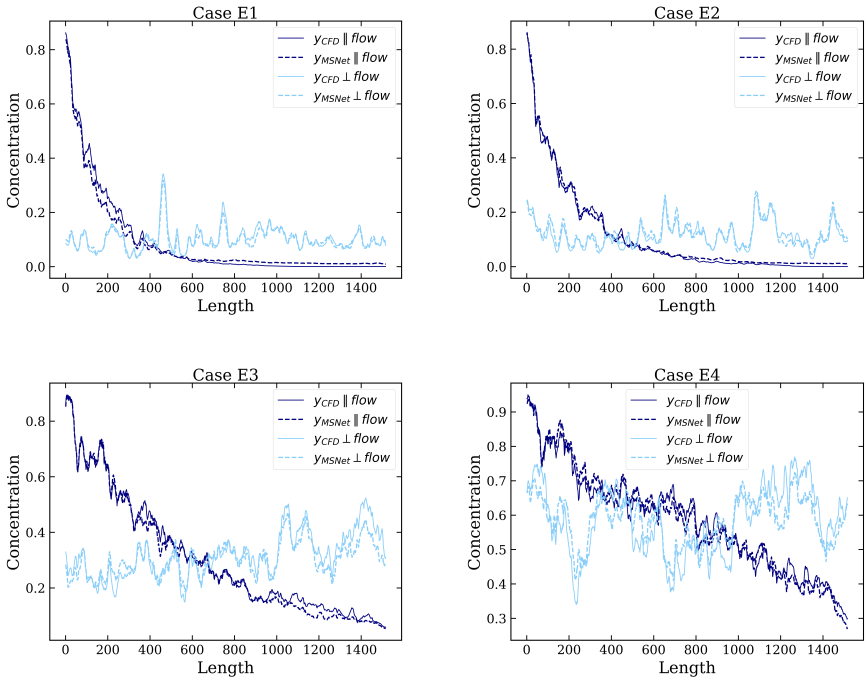

Fig. 6.7 Profiles of average concentration in the direction of flow and in the perpendicular direction to flow (CFD: solid, MSNet: dashed) for the four samples of Fig. 6.6.

Multiscale vs single scale

It is important to notice that the accuracy of the results obtained is deeply connected to the multiscale approach of the network. The generalization capability of MSNet relies both in the choice of the appropriate input features and in the intrinsic multiscale approach. This allows the network to extract information from the input features from different scales and, most of all, to employ a larger number of trainable parameters compared to classical CNN (considering a fixed amount of available RAM in the GPU).

To give a concrete example of the improvement afforded by this multi-scale approach, we tested the performance of a fully convolutional neural network, which is the case of a MSNet with a single scale.

The setup of the network was the same as the MSNet trained and presented in this work in terms of training strategy, dataset re-partition, and features choice. The architecture of the fully convolutional network was chosen maintaining the same

structure of the single scale (5 convolutional layers - on the left of Fig. 3). The number of filters of the convolutional layer was tuned in order to control the number of trainable parameters of the model. It would be necessary to match the number of trainable parameters (161 millions) of MSNet to have a consistent comparison between the model. Since the training was performed on the same GPU (NVIDIA Volta V100 GPUs, Nvlink 2.0, 16GB), it was not possible to match that amount for memory limitations of the GPU, so the number of filters of the tested network was tuned in order to obtain the highest number of trainable parameters allowed by the memory size of the GPU (1.3 millions).

The prediction accuracy on the test set of the trained fully convolutional neural network was evaluated in terms of the three metrics proposed in the manuscript, and is summarized in Tab. 6.3

The error on the prediction of the average concentration is higher than MSNet, but in particular the error on the profiles are one order of magnitude higher than MSNet, meaning that the local predictions are of poor quality.

Furthermore, the size of the samples that can be achieved with the setup of this work is higher that the one we employed (1536x1536). We tested that the maximum size achievable would be 5120x5120, so this MSNet can be trained on even bigger images of porous media.

Table 6.3 Comparison between MSNet and a fully convolutional neural network (single scale MSNet). The training was performed on the same GPU.

	MSNet	Fully Conv.
Number of parameters	161M	1.3M
$error_{testset}$ on average concentration	3.3%	4.9%
$RMSE_{testset}$ on concentration profile parallel to flow	0.023	0.18
$RMSE_{testset}$ on concentration profile perp to flow	0.042	0.17

Table 6.4 Prediction accuracy of MSNet using CELU or GELU as activation functions

	CELU	GELU
$error_{testset}$ on average concentration	3.3%	3.4%
$RMSE_{testset}$ on concentration profile parallel to flow	0.023	0.023
$RMSE_{testset}$ on concentration profile perp to flow	0.042	0.041

Choosing an activation functions: CELU vs. GELU

The choice of the activation function can have an impact on the accuracy of the predictions for deep neural networks, which is the case of our network. It is well known that the most classical sigmoid function should be avoided to prevent vanishing gradients - since its derivative is null for large negative and positive numbers, so the associated weights would not be updated during training [123].

The ReLU activation function was introduced as a computationally inexpensive alternative to prevent this issue, in fact, its derivative is 1 for positive numbers, and null for negative numbers. On the other hand the so-called *dead ReLU* appears when a large number of neurons are never updated [118].

To face this new issue Leaky ReLU or Exponential Linear Units (ELUs) have been proposed throughout the years, where the common strategy is to have non zero values as output for negative inputs.

Among the ELU family the most promising seem to be the functions CELU and GELU [115], which were then tested in MSNet. In Tab. 6.4 the accuracy of MSNet using GELU and CELU is summarized: the quality of the predictions is actually the same. Thus, we decided to employ the CELU activation function since it was the one already used in the original MSNet implementation.

Effect of the dataset dimension

A sensitivity analysis on the dimension of the dataset was performed in order to detect the minimum number of samples necessary to achieve the accuracy presented previously. An increasing number of samples are employed during training, from

200 to 800, the entire dataset. In Fig. 6.8 the results are reported, for each dataset size the training was reproduced three times. It is possible to conclude that at least 400 samples are required to achieve an accuracy on the average concentration of around 3.5%. The test set used to evaluate the generalization capability of the networks was the same. The use of a specialized neural network architecture and of appropriate input features, both conceived for the prediction objective, is essential to decrease the amount of samples necessary to obtain a satisfactory accuracy.

We tested these calculations on OpenFOAM on the test set of our dataset, and in Fig. 6.9 is reported a parity diagram for the filtration rate predicted by MSNet, and as a result of the CFD simulations. The Figure proves the capability of the neural network to predict the filtration rate, bringing a proof of concept of a possible application of this approach - letting the surrogate model calculate the upscaled parameter (for as many times and in as many different conditions) needed in a macroscopic transport and reaction model instead of building a complicated constitutive equation. In this respect the trained `Pytorch` models could be be easily embedded (for example) into the C++ code of the CFD software (`OpenFOAM`), taking care of computing the terms necessary to evaluate the filtration rate discussed above.

6.4 Conclusions

Neural networks, both fully connected and convolutional, have been widely employed to train data-driven models that can provide much faster predictions compared to the traditional alternative of performing computationally expensive physics-based simulations. In this work a multi-scale convolutional neural network was trained to reproduce the full concentration profile of different samples, which is commonly obtained via CFD simulations. This approach differs from other recently proposed machine learning workflows by being able to predict the entire concentration field of a large image, instead of just a scalar quantity. Our approach yields a a robust and flexible model that can be integrated in multiscale modelling workflows.

We studied the effect of different input features that inform the machine learning model about the boundary conditions, and that distill additional information about the geometry of the medium, for example, global and local paths available for flow. We showed that the Euclidean distance, the time of flight, the pressure drop, and the diffusion coefficient are sufficient to obtain very accurate pixel-wise predictions

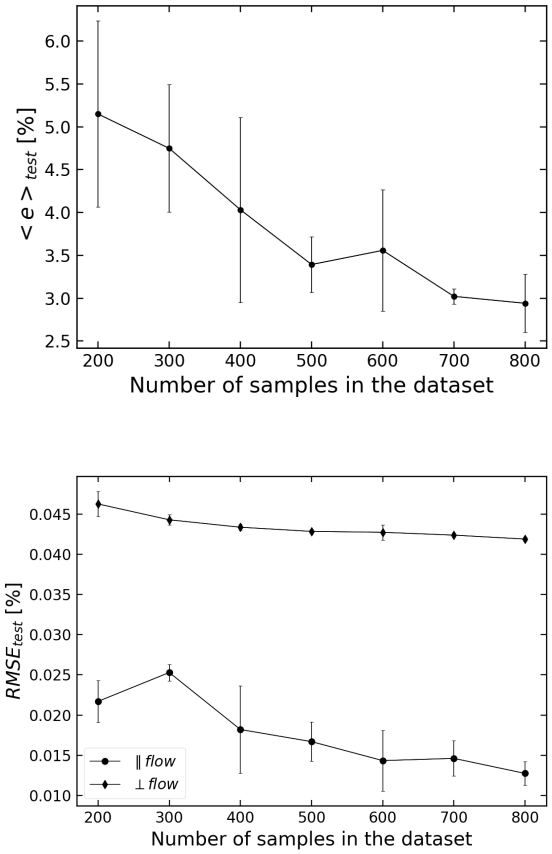

Fig. 6.8 Effect of the number of samples on the accuracy of the predicted fields. On the right: variation of the average error on the average concentration in the test set. On the left: variation of the root mean squared error in the parallel and perpendicular direction to flow on the average concentration in the test set. The test set is the same in all the cases.

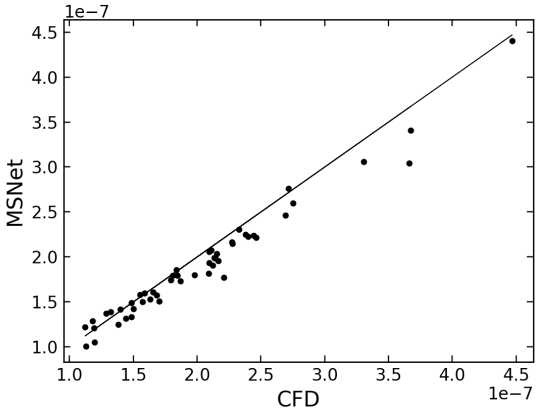

Fig. 6.9 Filtration rate predicted by MSNet.

for a wide range of sphere pack arrangements under varying operative conditions (a wide range of Péclet and Reynolds numbers).

This workflow is applicable to a wide range of systems that involve transport and reactions in porous media. The input features used in this Chapter uniquely describe the domain, hence these can be employed in new geometries and different physical phenomena. The dimension of the domain or the computation of the input features do not represent a problem for this approach, in fact, it is possible to train our model on three-dimensional datasets to create a model able to provide predictions with a similar prediction time.

These models, as it was shown, can be of use in aiding scale-bridging procedures for multi-scale simulations, or for the prediction of process-scale performance of reaction phenomena determined by complex micro-scale structures.

Thus, in conclusion, this methodology was demonstrated to be able to successfully provide predictions in a split-second of otherwise computationally intensive CFD simulations.

Chapter 7

Autoregressive neural networks for transient fields prediction

Electrodes at the microscale can be modelled as porous media. In Chapter 2 the electrodes microstructure is described: active material (AM), carbon binder domain (CBD), and electrolyte coexist, and transport equations of charge and mass balance can be solved at the microscale on these regions to simulate the charge/discharge behaviour of the battery. As discussed in the previous chapter for the filtration application, these microscale simulations are time consuming and computationally expensive, even more notably when a transient problem is faced, which is the case of charge/discharge simulations.

In this chapter the work done in collaboration with the research group of Prof. Alejandro Franco in the *Laboratoire de Réactivité et Chimie des Solides* (LRCS) in Amiens (France) is presented. An autoregressive multiscale neural network (MSNet) is trained to obtain a surrogate model for the discharge of the cathode side of a lithium ion battery. The starting point of the chapter is the description of the dataset, in Section 7.1: the simulations described therein were performed by the collaborators at LRCS, for the purposes of the construction of a suitable dataset for testing the construction of the new autoregressive MSNet. Then, the modifications to MSNet are detailed in Section 7.2, and finally the Results in Section 7.3, and the Conclusions of the work are summarized in Section 7.4.

The content of this chapter, in a modified form, has been published in Marcato et al. (2023) [124]

7.1 Dataset

The dataset employed for the construction of the data-driven model is made by half-cell microscale simulations focused on the cathode side, whose geometries are reproduced *in-silico*.

The workflow proposed for the creation of the geometries mirrors the experimental production steps of electrodes, as summarized in Fig. 5.1A. The key steps to be reproduced to obtain a full digital twin of the fabrication process are:

1. the preparation of the slurry with all the components;
2. the slurry casting on the current collector;
3. the solvent evaporation from the cast;
4. the calendering of the dried electrode;
5. the battery assembly.

For modelling purposes two solid phases are usually considered, as detailed in Chapter 2: the active material (AM) and the carbon binder domain (CBD), made by the binder and the conductive carbon, which actually constitute a single phase after the evaporation of the solvent.

The *in-silico* reproduction of the electrode geometry is preferred in this context, because, even though 3D images obtained from synchrotron X-ray tomographies are available [125], their segmentation in the three phases (AM, CBD, electrolyte) is not straightforward and still an open research problem to be addressed [126]. The capability of reproducing the electrodes geometries for modelling purposes is essential when a big campaign of simulations is needed, which is the case of this study.

To this end the processes at points 2, 3, 4, are modelled by using a molecular dynamics code, LAMMPS [127], as detailed in Rucci et al. [128]. At first, given the granulometry of the AM and of the CBD before the evaporation, a random configuration of particles is produced. Then, the slurry is equilibrated at constant pressure and temperature imposing Lennard-Jones and Granular Hertzian potentials between particles. After that, the evaporation of the solvent is modelled imposing the

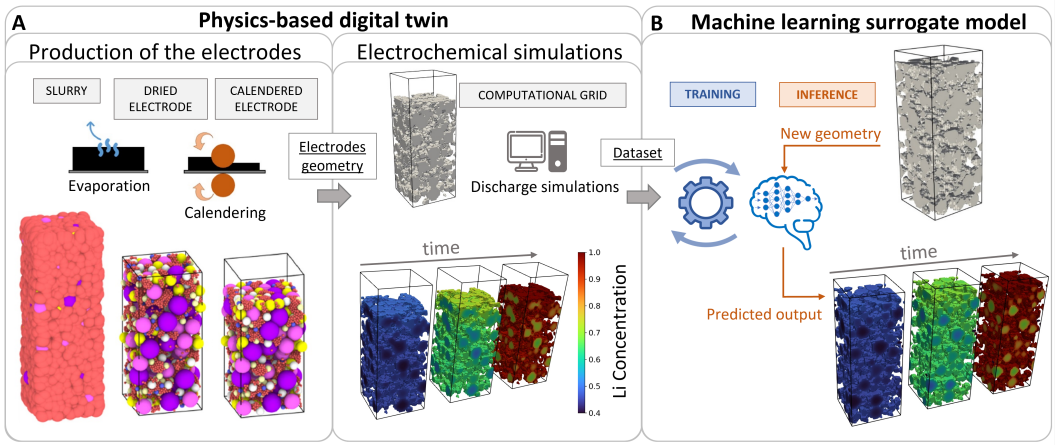

Fig. 7.1 A) The physics-based digital twin process replicates the experimental production of electrodes which are then used in the electrochemical simulations to create a dataset. B) The digital twin results are then used to train the machine learning surrogate model for the prediction of new 3D time-changing fields in new geometries.

shrinkage of the CBD particles until a set value is reached. Finally the calendering of the electrode is simulated imposing a compression on the dried particles. The computational details of this workflow can be found in Prof. Franco research group articles [129, 4, 95, 128].

The electrochemical simulations have been performed using the finite element method implemented in COMSOL on the cathodes geometries. In particular, the partial differential equations of charge and mass balance reported in Chapter 2 are solved in the AM, CBD, and electrolyte phases. The parameters present in the equations, Eq. 2.31-2.40, are set as reported in Tab. 7.1.

The starting point for the dataset used for the training of the neural networks is a set of different electrode geometries, varying in terms of AM/CBD proportions and calendering degree. Each geometry is then employed for simulations at different discharge rate, i.e. the C rate. These values are summarized in Tab. 7.2, all the possible combinations of those parameters have been simulated in order to create the dataset, thus a total of 45 transient simulations.

Since the dataset simulations were solved with COMSOL, a finite element method, the computational grid is unstructured. This is not compatible with the (matricial) data shape needed by a convolutional neural network, so the results have been interpolated into a structured grid, in order to be conveyed to the network as an image. The interpolation consists at first in the creation of the Cartesian grid by means

Parameter	Value/Reference
Porosity of CBD ε_{CBD}	0.27 [130]
Maximum Li concentration of NMC $C_{(s,max)}$ mol/m^3	48207
Initial DOL of NMC	0.45925
Reaction rate coefficient of NMC k_{NMC} (m/s)	4.38e-11 [131–133]
Ionic conductivity of electrolyte σ_l (S/m)	Ref [134] at 20°C
Electrical conductivity of NMC $\sigma_{s,\text{AM}}$ (S/m)	0.01
Electrical conductivity of CBD $\sigma_{s,\text{CBD}}$ (S/cm)	$-173.967\varepsilon + 0.1593$
Deformation of the CBD phase because of calendering ε	Ref [95]
Factor for effective transport correction in CBD f	0.05
Diffusion coefficient of Li$^+$ $\mathscr{D}_l$ (m^2/s)	Ref [134] at 20°C
Diffusion coefficient of Li in NMC $\mathscr{D}_s$ (m^2/s)	Ref [135]
Transport number of Li$^+$ t_+	Ref [134] at 20°C
Activity dependence of Li$^+$ $\left(1 + \frac{\partial(\ln f_\pm)}{\partial(\ln C_l)}\right)$	Ref [134] at 20°C
Anodic transfer coefficient α_a	0.5
Cathodic transfer coefficient α_a	0.5
SEI film resistance R_{SEI} (Ωm^2)	0.001

Table 7.1 Parameters employed in the equations of the electrochemical model of Chapter 2.

of a voxelization algorithm (as detailed in Appendix B), in the grid '1' is assigned to the phase to be interpolated and 0 elsewhere.Then the fields are interpolated from the unstructured grid to the structured one by means of the Gaussian kernel interpolator available in Paraview. The size of the Cartesian grid cells is 0.5 μm, which is the resolution of the elements in the original unstructured grid [95].

The dataset is made by samples, where a sample for a transient problem is defined as the collection of all the time-frames available from a simulation of a given electrode geometry and undergoing a certain discharge rate, so its dimensions are: (Number of time-frames, x, y, z). In a battery simulation the time required for a discharge process is linked to the C rate: if the C rate is doubled, the time required to completely discharge the battery is halved. In the dataset available the time-frames have been saved according to the C rate, thus resulting in samples with the same number of time-frames. It is important to underline that the time-frames are not saved at each time-step of the transient simulation because the memory load necessary to save them would have been prohibitory. A total number of 25 time-frames is available for each sample, for 2C discharge the time interval between two time-frames is 50 s, for 1C discharge rate 100s, for 0.5C discharge rate 200s. In this chapter we will refer

Parameter	Set
AM $\%_{wt}$	85-87-90-93-95
Calendering degree (%)	0-10-20
C rate	0.5C-1C-2C

Table 7.2 Parameters varied for the creation of the dataset.

to the normalized time-frames as Δt^*. The number of time-frames to extract from the simulations and employ in the training in order to describe a transient problem by means of a neural network is for sure an hyperparameter to be further investigated in order to evaluate its impact on the accuracy of the prediction.

7.2 Autoregressive MSNet

The multiscale convolutional neural network presented in Chapter 6 can be employed for the prediction of steady-state fields and can generalize on different geometries and operating conditions. For transient problems appropriate architectures have to be chosen. Recurrent neural networks, such as long-short term memory networks [136], have been widely employed in data science, but may not be the most appropriate structure for a problem of this kind. In this work MSNet has been modified in order to deal with a transient dataset, and to predict a sequence of fields starting from the initial conditions.

The most intuitive way to approach this problem is to consider the problem as a Markovian process, where each time-step result depends just on the previous one, as it is in the solution of a transient physics-based simulation. Therefore, the network should take as input the geometrical descriptors and the operating conditions (as it is done with how it will be called in this chapter: the “classic" approach), together with a temporal feature, which is: the initial condition for the prediction of the first time-frame, or the previous time-frame for the prediction of the following ones.

Given this premise the easiest solution to perform the training is to concatenate the input features with the previous time-frame field (the true field from the simulations) and train the network as the standard MSNet. In this way the training is carried on the shuffled time-frames, so the transient problem is decomposed as single frame samples. Nevertheless for testing purposes the first prediction is carried on concatenating the

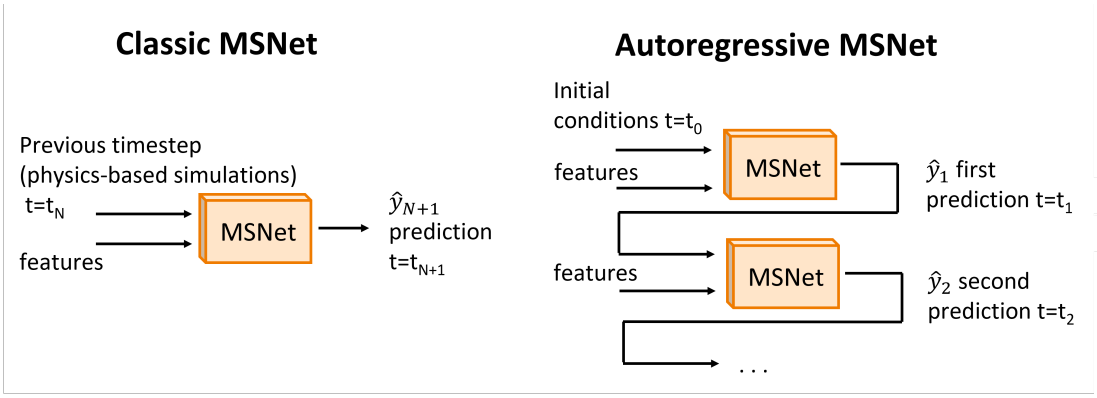

Fig. 7.2 Classic and autoregressive MSNet for the prediction of time-dependent fields.

input features with the initial condition, then the first output is concatenated with the other features for the prediction of the second time-frame, so on and so forth until the last prediction. This is obviously necessary since in the test mode the intermediate time-frames are not available, thus the network predictions have to be employed as input for the following time-frames.

Another training solution is to mimic the testing process, so that even during the training the input features are concatenated with the previous time-frame network prediction, and not the true field of the physics-based simulation. Using this approach the transient nature of the dataset is preserved, since MSNet is provided with the samples (each one with the time-frames from the initial condition to the final time-frame) at each epoch.

This autoregressive training approach, sketched in Fig. 7.2, can be summarized as follows [1]:

$$\hat{y}_{(0,1)} = \text{MSNet}(X_0, X_{initial}) \tag{7.1}$$

$$\hat{y}_{(0,2)} = \text{MSNet}(X_0, \hat{y}_{(0,1)}) \tag{7.2}$$

$$\ldots$$

$$\hat{y}_{(0,T)} = \text{MSNet}(X_0, \hat{y}_{(0,t-1)}) \tag{7.3}$$

for a generic time-frame t, MSNet performs the following operations:

$$\hat{y}_{(0,t)} = \text{NN}_0(X_0, \hat{y}_{(0,t-1)}, \mathbb{R}(\hat{y}_{(1,t)})) + \mathbb{R}(\hat{y}_{(1,t)}) \tag{7.4}$$

[1]The nomenclature of MSNet architecture is kept as in Chapter 6, a new index is introduced referring to the time-frame, so $y_{(n,t)}$ refers to the n^{th} scale and the t^{th} time-frame.

...

$$\hat{y}_{(N-1,t)} = \mathrm{NN}_{N-1}(X_{N-1}, \hat{y}_{(N-1,t-1)}, \mathbb{R}(\hat{y}_{(N,t)})) + \mathbb{R}(\hat{y}_{(N,t)}) \tag{7.5}$$

$$\hat{y}_{(N,t)} = \mathrm{NN}_{N}(X_{N}, \hat{y}_{(N,t-1)}) \tag{7.6}$$

Even in this case both geometrical descriptors and operating conditions have to be fed to MSNet. The Euclidean distance transform (Appendix B), the AM-CBD-electrolyte repartition, and the distance between current collector and separator have been employed as geometry features, respectively A, B, C of Fig. 7.3. The C rate value has been provided as operating condition input. It is important to notice that these features are unchanged for the predictions of the different time-frames of the same sample. Both input and output features have been scaled in order to range between 0 and 1, so MSNet predicts normalized concentration and potential fields.

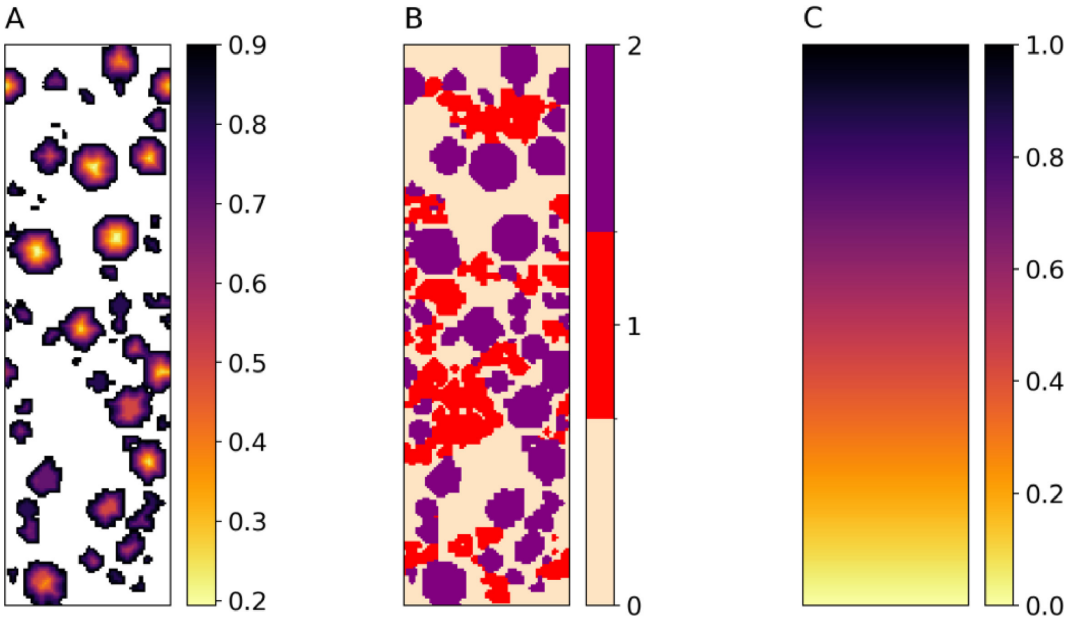

Fig. 7.3 Geometrical descriptors employed as input to MSNet (cathode: 85% AM, calendering: 0). Plot A: inverted Euclidean distance, plot B: AM/CBD/electrolyte repartition, C: current collector separator distance

The size of the samples is: (72,72,244) so three scales have been trained in the network in order to ensure a good field of vision. The number of filters has been improved compared to the original MSNet in order to optimize the local prediction in 3D. The number of filters is set to 14, which is the maximum size allowed by the

RAM limitations of the GPUs employed in this work. This three-scale configuration of MSNet is represented in Fig. 7.4.

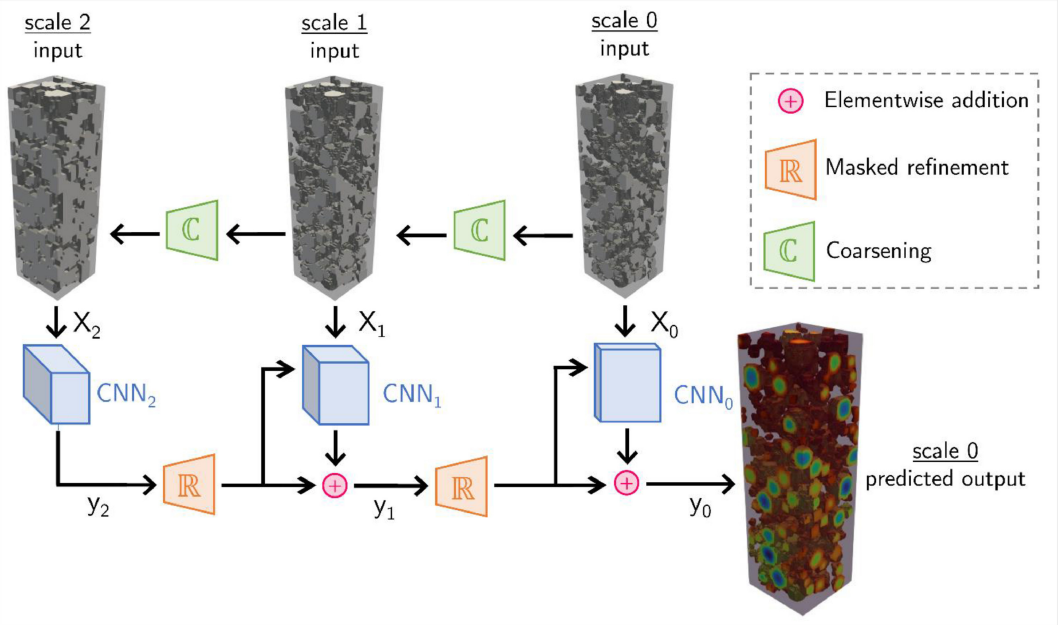

Fig. 7.4 Workflow of the MSNet architecture. Each scale takes as input the same set of features, at different resolutions, thanks to coarsening operations (only the binary geometry feature is shown). The convolutional layers (blue blocks) have an increasing number of filters for increasing scales to optimize the tradeoff between the total number of trainable parameters and the memory requirements during training.

7.3 Results

The main prediction objective of this study is the reconstruction of the discharge curves, which describe the potential decay as a function of the degree of lithiation of the cathode. So MSNet was employed for the prediction of the lithium concentration field in the AM phase, and the potential field in the solid phase (AM + CBD) of the electrode. From these predicted fields it is possible to integrate their quantities over the phases and obtained the desired discharge curves.

At first the two training approaches proposed in the previous section are compared for the prediction of the concentration field, then the best approach among the two is employed for the prediction of all the fields required for the reproduction of the discharge curves. In the first approach the training inputs are the geometrical

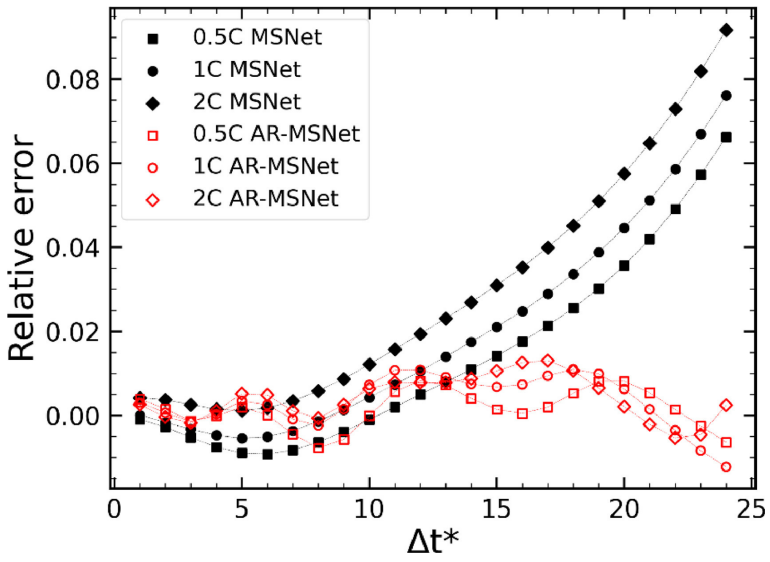

Fig. 7.5 Error on the prediction of the average concentration for the time-frames and the different C rates.

descriptors, the C rate, and the previous time-frame field (from the physics-based simulation). In the second approach the training inputs are the geometrical descriptors, the C rate, and the previous network-predicted time-frame field. The comparison is based on the prediction of the lithium concentration on three test samples: AM weight percentage: 90, calendering degree: 10, C rate: 0.5C, 1C, 2C.

In Fig. 7.5 the relative error between true and predicted average lithium concentration is reported for the different C rates along subsequent time-frames. It is evident that by using the first approach the error increases with time, while this does not happen with the second approach. In Fig. 7.6 three time-frames for the test sample at discharge 1C are compared for the two approaches. For each time-frame the true (from the finite elements simulations) and predicted local concentration fields are displayed for a slice of the 3D domain, and the pixel-wise error is reported on the right. It is possible to see that using a "classical" (not autoregressive) approach even the local prediction deteriorates in time, in fact, the radial profile of the lithium concentration is lost in time in favor of a random noisy concentration field.

The different behavior in generalization is due to the error propagation in time. When MSNet is trained with the true fields as input, the network is not trained to deal with slight (but cumulating) fluctuations of prediction errors as input fields, thus it is not trained to dampen these fluctuations, on the contrary it magnifies the errors in the predictions leading to a big propagation of the error. In the second approach, during

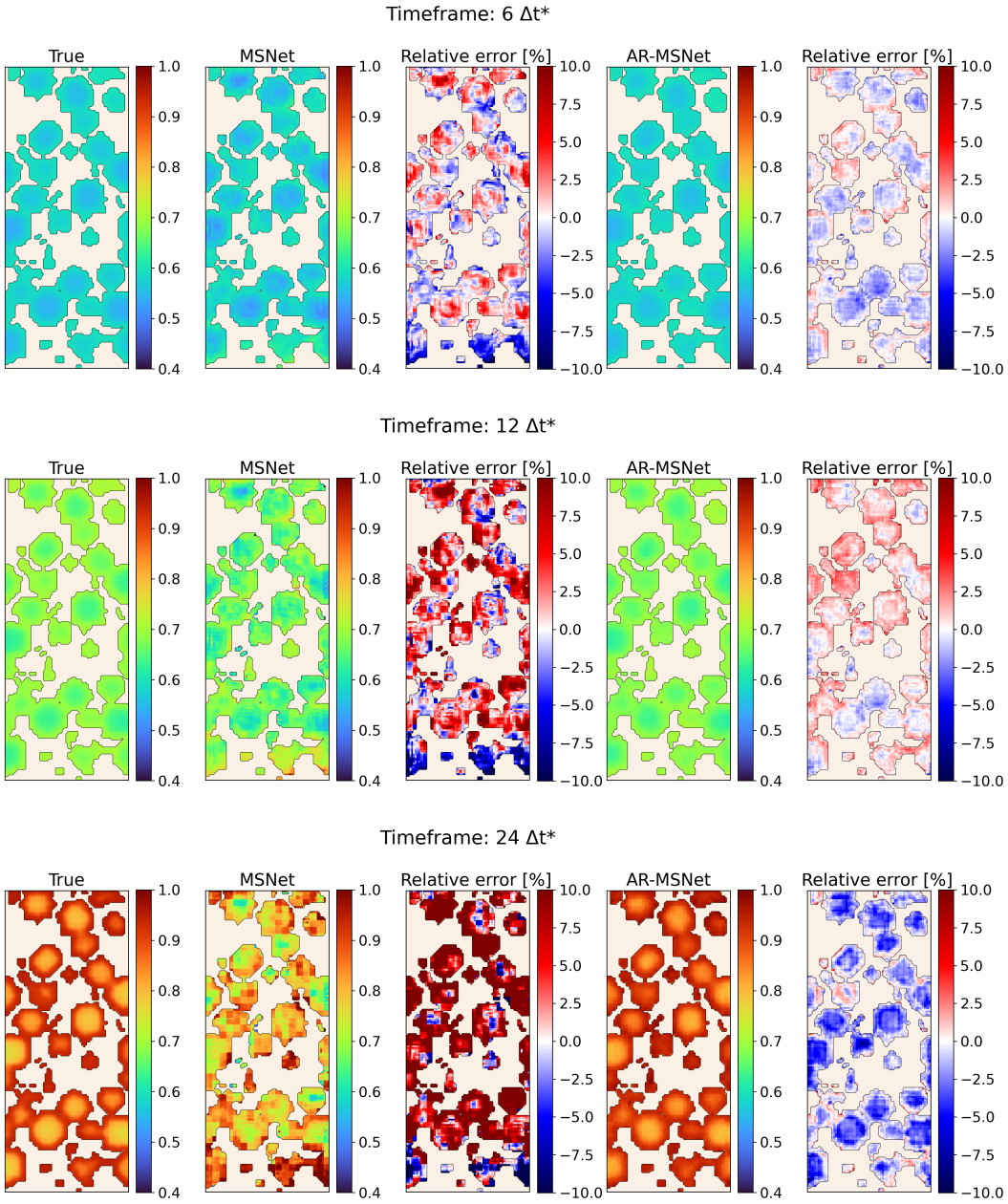

Fig. 7.6 Local true and predicted concentration fields for three time-frames of a slice of the 3D sample (1C). For each time-frame: concentration field from the physics-based simulation, prediction by the classical MSNet, relative error between true and classical approach, prediction by the autoregressive MSNet, relative error between true and autoregressive MSNet.

the training the network learns how to dampen fluctuations in the input resulting in much better generalization.

Given these results, the autoregressive MSNet was employed for the prediction of the transient concentration and potential fields. The predictions of the concentration and potential fields over time are summarized in Fig. 7.7, for each time-frame the true field from the COMSOL simulations is reported, then the predicted field, and the local relative error. The average quantities are needed to obtain the predicted discharge curves of Fig. 7.8. In these charts the potential is expressed as a function of the lithium concentration for the different C rates.

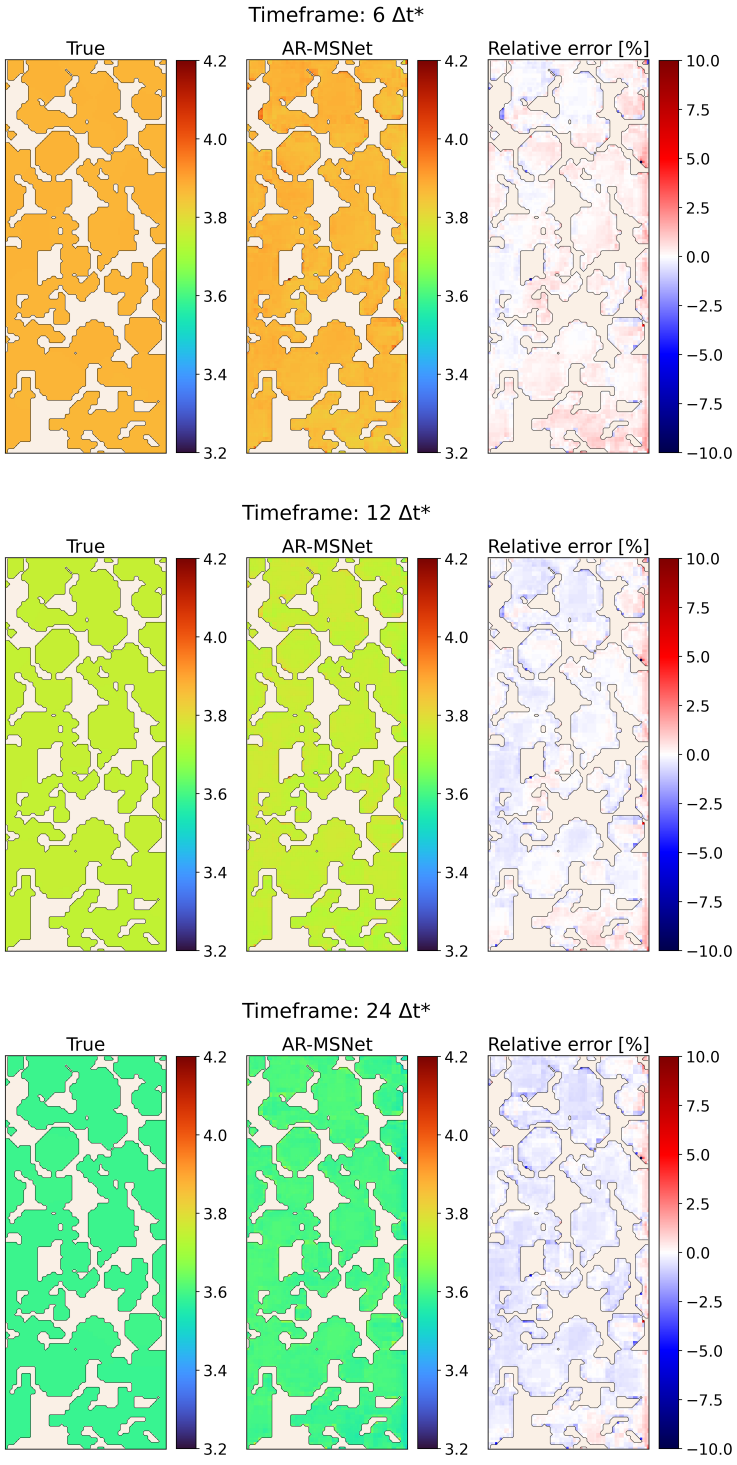

Fig. 7.7 Local true and predicted potential fields for three timeframes of a slice of the 3D sample (1C). For each timeframe: potential field from the physics-based simulation, prediction by the autoregressive MSNet, relative error between true and autoregressive MSNet.

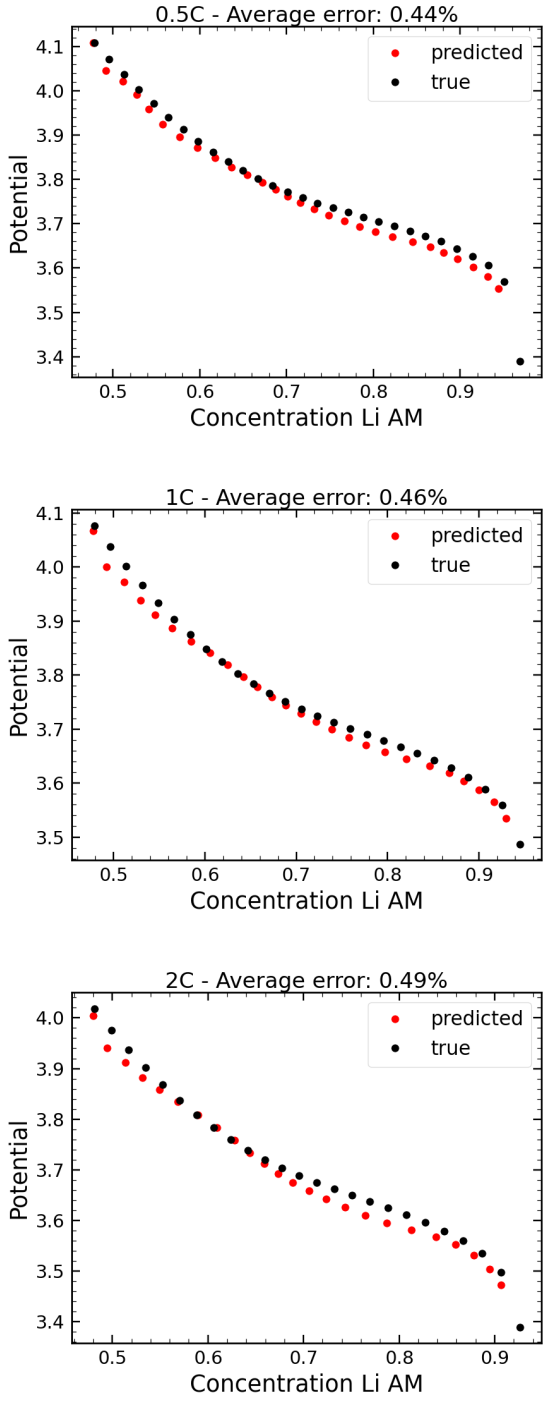

Fig. 7.8 Discharge curves showing potential versus concentration values for C rates equal to 2C, 1C, 0.5C. True values from simulations compared with predictions by AR-MSNet.

7.4 Conclusions

In this chapter, focused on an energy storage application, we have decided to tackle a single, but central, aspect of the field of computer-aided battery research: the problem of facilitating the optimization processes needed in the design of energy storage systems. In the fast-moving landscape of modern battery design, with its demand for ever growing performance in multiple areas, this means computational loads that workflows based on traditional modelling cannot bear. A wide variety of low-order, data-driven, and in general surrogate models are thus being studied and employed to aid in this prospect.

Our tools of choice for this effort are neural networks, and especially convolutional neural networks. The effectiveness of this kind of machine learning technique is apparent (and was again shown here) when dealing with the interpretation of transport phenomena which are greatly influenced by the geometric structure characterizing the system, as is the case for porous media in general and clearly for the random and heterogeneous structure of lithium-ion battery electrodes.

Starting from an innovative multi-scale convolutional network architecture, proven to accurately reproduce results of detailed steady-state physics-based simulations, we have developed an improvement in order to obtain a data-driven model able to also explore variations in time (and space) of properties of interest.

Two connected takeaways result from the work shown here. First, as mentioned, the results confirm the great flexibility of convolutional neural networks as algorithms for the treatment of physics-based simulation data, and that they are able (with appropriate modifications) to also be effectively used to treat time-varying sequences of three-dimensional data. This was not granted, as this kind of application lies somewhat outside of the applications for which convolutional neural networks are now solidly a standard, namely image analysis. Secondly, the process we presented shows how this extension is not trivial, as a "classic" sequential approach to the neural network training would fail by means of error accumulation in time. Thus, an autoregressive approach was needed, meaning that both a suitable architecture had to be developed and appropriate loss functions were formulated in order to properly teach the network how to interpret the data coming from physics-based simulations, and how to learn to make comparably good predictions, even when moving beyond the cases on which it was trained.

In conclusion, we consider this to be a successful proof of concept for fast, reliable, and full-order surrogation of accurate numerical simulations. Fast surrogate models are essential to deal with the wide parameter space intrinsic in contemporary battery design and optimization: the results of the work shown here aim to show how such surrogates can not only predict single key metrics but are also able to disclose the full dynamics of battery systems as they evolve in time.

Chapter 8

Conclusions

As it emerges from the former chapters, the main objective of the work presented in this dissertation is to show the performance of neural networks as data-driven models. The chosen research field for this methodology is the numerical study of flow, transport, and reaction in porous media, through the investigation of a series of applications: clean-bed filtration and the discharge dynamics of energy storage systems.

This choice was made, beyond the clear scientific and practical interest of investigating these various problems, also and perhaps even more strongly as they constitute the perfect test case for the wide range of data-driven models employed in this work. The accurate study of porous media stands upon a deep understanding of the impact of the microscale geometric structure on transport phenomena, which is scarcely obtainable by averaged integral descriptions, to be used in classical simplified input-output models. As such, the study of these models of increasing complexity (both in structure description and in the type of phenomena investigated) is well suited to avail itself of possibilities afforded by the recent advances in deep learning algorithms, now able to very effectively interpret images, three-dimensional structures, and even time-evolving sequences of data, and connect them to the desired outputs, even to the point of surrogating (i.e.: reproducing) data in the form of the results of full-order numerical simulations. By way of conclusion, the main results obtained throughout this work are here summarized, together with their significance and impact on research and practical applications, more directly on chemical and pro-

cess engineering, and by extension on the related fields of environmental engineering and subsurface transport studies.

In Chapter 4 we described a successful open-source workflow for the realization of a dataset starting from a campaign of CFD simulations aimed at the training of neural networks for the prediction of fluid dynamics quantities. Fully connected neural networks have been employed to predict the permeability and the filtration rate for the investigated cases. The average error on the test set for the permeability is lower than 6% and for the filtration rate is lower than 3.5% for both geometrical models. The networks provide more accurate predictions of the permeability and the filtration rate compared to traditional analytical expressions, with average errors lower than 40% in the range of operating conditions explored in this work. The data-driven models, once trained, can instantaneously give a satisfactorily accurate output, while a CFD simulation requires a certain amount of computational time. In our case each CFD simulation requires from one to four hours to be solved and the training takes four minutes. The increase in the predictive velocity can be exploited in multiscale modeling, in-line control and optimization problems.

In Chapter 5 both fully connected and convolutional neural networks were tested in our workflow. The main difference between these two approaches is the input to the model, in the first case hand selected features are the input to the FCNN, in the second case the entire image is fed to the networks, which is able to autonomously detect the features in the image for the prediction of its target output. While the two models exhibit similar (and satisfactory) accuracy, it is notable that a comparable number of CFD simulations between the two cases were sufficient to reach such accuracy, notwithstanding the clearly higher complexity of the convolutional model with respect to the classical fully connected network. Even more so, in the light of the much higher flexibility of use of the model based on CNN, which (at the cost of higher computational cost for its training) will prove to be much more useful by means of opening to this kind of analysis even systems featuring complicated geometries - as are frequently found in chemical engineering both in purification/filtration apparatuses, and in packed bed catalytic reactors. Beside its usefulness in dealing with complicated structures, these convolutional models have already proven in this work to be able to treat complicated phenomena, by providing a way to robustly construct accurate models even in cases for which it may not be immediate (or feasible) to choose integral input features with which to build a predictive analytical model - case in point the problem of colloidal filtration, as explored in this work. The

long term application of this methodology will be the use of neural networks for the prediction of other microscale properties that lack analytical models that correlate them to microscale properties.

In Chapter 6 a multi-scale convolutional neural network was trained to reproduce the full concentration profile of different samples, which is commonly obtained via CFD simulations. This approach differs from other recently proposed machine learning workflows by being able to predict the entire concentration field of a large image, instead of just a scalar quantity. Our approach yields a robust and flexible model that can be integrated in multiscale modelling workflows. We studied the effect of different input features that inform the machine learning model about the operating conditions, and that distill additional information about the geometry of the medium, for example, global and local paths available for flow. We showed that the Euclidean distance, the time of flight, the pressure drop, and the diffusion coefficient are sufficient to obtain very accurate pixel-wise predictions for a wide range of sphere pack arrangements under varying operative conditions (a wide range of Péclet and Reynolds numbers). This workflow is applicable to a wide range of systems that involve transport and reactions in porous media. The input features that we proposed in this work uniquely describe the domain, hence these can be employed in new geometries and different physical phenomena. The dimension of the domain or the computation of the input features do not represent a problem for this approach, in fact, it is possible to train our model on three-dimensional datasets to create a model able to provide predictions with a similar prediction time. We showed how these models can be of use in aiding scale-bridging procedures for multi-scale simulations, or for the prediction of process-scale performance of reaction phenomena determined by complex micro-scale structures. In conclusion, we demonstrated a methodology that can successfully provide predictions in a split-second of otherwise computationally intensive CFD simulations.

Finally, in Chapter 7 we have shown an improvement of the MSNet architecture, and its use in treating a new time-dependent problem, namely the simulation of charge/discharge processes in lithium-ion batteries. There are two main identifiable takeaways resulting from the work presented in this chapter. First, this confirms the great flexibility of convolutional neural networks as machine learning algorithms for the treatment of physics-based simulation data: they are able to be adapted to interpret and reliably reproduce full 3D simulation data, evolving in time, even when dealing with quite complicated systems, with multiple domains and boundaries regulated by

non-trivial governing equations. This of course when the starting neural network, which was further modified for this application, has the multi-scale structure and thus scale-probing capabilities that had to be developed to interpret the diverse results sets coming from accurate (steady-state) pore-scale simulations. The second thing of note is actually connected to this last point. As it was needed when originally developing the multi-scale architecture, or later adapting it to reactive transport problems, also in this case when dealing with a new problem (charge/discharge processes vs. reactive transport) which may be set down in a different modelling perspective (steady-state vs. transient transport), some fundamental modifications were needed to obtain a robust model capable of generalization to a reasonably large set of cases and conditions. Concluding, we consider this to be a successful proof of concept for fast, reliable, and full-order surrogation of accurate numerical simulations, tackling multi-domain species transport and reaction, with a time-dependent evolution. The choice for this last specific case-study, the case of the modelling of operation of energy storage systems, was motivated both by the similarity of electrode structures to the porous systems extensively explored in this dissertation, and by the ever increasing importance (and need for accurate modelling) of this kind of applications.

Lastly, the work shown in the last chapter, more in the sense of the type of network architecture presented than in the strictest sense of technical performance, is best seen at the top of an evolving continuum starting from the simple network models employed in the first chapter. Indeed, together with the results coming from the preceding chapters, it completes a path on a continuum of increasing complexity of the surrogate model. There are two, more immediate, meanings of this statement. Scientifically, the models are more complex as they move from a low-order surrogation, capable of predicting scalar metrics from a collection of integral features, to high-order surrogation when the capability to interpret 3D sets is learned, to full-order space and time model surrogation - where the networks are able to reproduce how multiple local fields vary in time when regulated by complex transport and reaction equations. This is made possible since technically, the capabilities of the networks are augmented by an equivalent increase of the complexity of the architecture of the network, afforded by (but by all means not exclusively) a huge increase in the parameters of the neural networks used, namely its size.

A third, and more significant, aspect would be that employing more complex data-driven models did not result just in a more accurate description of the system,

or lower errors when compared to an analytical or experimental ground truth. That would be the case in the classic choosing process employed by the model user who, informed by the principle of economy, looks along a line of different available models looking for the simplest, but still appropriate, tool for their use case. As it were, what we called earlier a "continuum" of complexity is nothing of the sort, and indeed there is a marked difference between specifically the earliest input-output models (classical neural networks) predicting scalar quantities starting from a few chosen parameters, and those (based on convolutional networks) able to first understand and eventually surrogate whole fields starting from three-dimensional geometries. This is conveyed first by the different capacity for expressivity in the results which are the output of these models: these can be simple numerical performance metrics, or they can give local details about the quantities of interest, even with the passing of time, if that is what is needed. But a probably even more important aspect, is the flexibility given to the practitioner in the choice of input parameters. Case in point, reframing this in statistical terms related to the study of uncertainty, which is even more central in the study of porous media, the most impactful break in descriptive capability comes perhaps from exactly this flexibility of the inputs when building these models.

This dissertation hopes to show that it is possible to perform this work and, beyond the shown examples of applications in the field of energy storage and transport and reaction problems in porous media in general, a methodological approach based on coupling accurate physics-based simulations with specially engineered data-driven models based on neural networks can find applications and can be employed to great success in the general area of accurate investigation of transport phenomena in chemical and process engineering.

References

[1] Leyla Amiri, Marco Antonio Rodrigues de Brito, Seyed Ali Ghoreishi-Madiseh, Navid Bahrani, Ferri P Hassani, and Agus P Sasmito. Numerical evaluation of the transient performance of rock-pile seasonal thermal energy storage systems coupled with exhaust heat recovery. *Applied Sciences*, 10(21):7771, 2020.

[2] Jacob Bear. *Dynamics of fluids in porous media*. Courier Corporation, 1988.

[3] Renat T Sibatov, Vyacheslav V Svetukhin, Evgeny P Kitsyuk, and Alexander A Pavlov. Fractional differential generalization of the single particle model of a lithium-ion cell. *Electronics*, 8(6):650, 2019.

[4] Mehdi Chouchane, Alexis Rucci, Teo Lombardo, Alain C Ngandjong, and Alejandro A Franco. Lithium ion battery electrodes predicted from manufacturing simulations: Assessing the impact of the carbon-binder spatial location on the electrochemical performance. *Journal of Power Sources*, 444:227285, 2019.

[5] B. D. Hammel. What learning rate should i use? http://www.bdhammel.com/learning-rates/. [Online; accessed 2023/1/12].

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[7] Agnese Marcato, Daniele Marchisio, and Gianluca Boccardo. Reconciling deep learning and first-principle modelling for the investigation of transport phenomena in chemical engineering. *The Canadian Journal of Chemical Engineering*, 101(6):3013–3018, 2023.

[8] NEURIPS. Machine learning and the physical sciences. https://ml4physicalsciences.github.io/2022/. [Online; accessed 2023/1/12].

[9] K.T. Schütt, H.E. Sauceda, P.-J. Kindermans, A. Tkatchenko, and K.-R. Müller. Schnet - a deep learning architecture for molecules and materials. *Journal of Chemical Physics*, 148(24), 2018. cited By 625.

[10] S. Chmiela, H.E. Sauceda, K.-R. Müller, and A. Tkatchenko. Towards exact molecular dynamics simulations with machine-learned force fields. *Nature Communications*, 9(1), 2018. cited By 286.

[11] B. Siddani, S. Balachandar, W.C. Moore, Y. Yang, and R. Fang. Machine learning for physics-informed generation of dispersed multiphase flow using generative adversarial networks. *Theoretical and Computational Fluid Dynamics*, 35(6):807–830, 2021. cited By 6.

[12] S. Balachandar, W.C. Moore, G. Akiki, and K. Liu. Toward particle-resolved accuracy in euler–lagrange simulations of multiphase flow using machine learning and pairwise interaction extended point-particle (piep) approximation. *Theoretical and Computational Fluid Dynamics*, 34(4):401–428, 2020. cited By 18.

[13] Lifei Zhao, Zhen Li, Bruce Caswell, Jie Ouyang, and George Em Karniadakis. Active learning of constitutive relation from mesoscopic dynamics for macroscopic modeling of non-newtonian flows. *Journal of Computational Physics*, 363:116–127, 2018.

[14] Agnese Marcato, Gianluca Boccardo, and Daniele Marchisio. From computational fluid dynamics to structure interpretation via neural networks: An application to flow and transport in porous media. *Industrial & Engineering Chemistry Research*, 2022.

[15] Allen K Ting, Javier E Santos, and Eric Guiltinan. Using machine learning to predict multiphase flow through complex fractures. *Energies*, 15(23):8871, 2022.

[16] Agnese Marcato, Gianluca Boccardo, and Daniele Marchisio. A computational workflow to study particle transport and filtration in porous media: Coupling cfd and deep learning. *Chemical Engineering Journal*, 417:128936, 2021.

[17] Graziano Frungieri, Gianluca Boccardo, Antonio Buffo, Hossein Ali Karimi-Varzaneh, and Marco Vanni. Cfd-dem characterization and population balance modelling of a dispersive mixing process. *Chemical Engineering Science*, 260:117859, 2022.

[18] Agnese Marcato, Javier E Santos, Gianluca Boccardo, Hari Viswanathan, Daniele Marchisio, and Maša Prodanović. Prediction of local concentration fields in porous media with chemical reaction using a multi scale convolutional neural network. *Chemical Engineering Journal*, page 140367, 2022.

[19] Donggeun Park, Jemyung Cha, Moonjeong Kim, and Jeung Sang Go. Multi-objective optimization and comparison of surrogate models for separation performances of cyclone separator based on cfd, rsm, gmdh-neural network, back propagation-ann and genetic algorithm. *Engineering Applications of Computational Fluid Mechanics*, 14(1):180–201, 2020.

[20] Marc Duquesnoy, Chaoyue Liu, Diana Zapata Dominguez, Vishank Kumar, Elixabete Ayerbe, and Alejandro A Franco. Machine learning-assisted multi-objective optimization of battery manufacturing from synthetic data generated by physics-based simulations. *arXiv preprint arXiv:2205.01621*, 2022.

[21] Teo Lombardo, Marc Duquesnoy, Hassna El-Bouysidy, Fabian Årén, Alfonso Gallo-Bueno, Peter Bjørn Jørgensen, Arghya Bhowmik, Arnaud Demortière, Elixabete Ayerbe, Francisco Alcaide, et al. Artificial intelligence applied to battery research: hype or reality? *Chemical Reviews*, 2021.

[22] Ian L Molnar, Erica Pensini, Md Abdullah Asad, Chven A Mitchell, Ludwig C Nitsche, Laura J Pyrak-Nolte, Gastón L Miño, and Magdalena M Krol. Colloid transport in porous media: a review of classical mechanisms and emerging topics. *Transport in Porous Media*, 130(1):129–156, 2019.

[23] Gianluca Boccardo, Frederic Augier, Yacine Haroun, Daniel Ferre, and Daniele L Marchisio. Validation of a novel open-source work-flow for the simulation of packed-bed reactors. *Chemical Engineering Journal*, 279:809–820, 2015.

[24] Kalyani Pangarkar, Tilman J Schildhauer, J Ruud van Ommen, John Nijenhuis, Freek Kapteijn, and Jacob A Moulijn. Structured packings for multiphase catalytic reactors. *Ind. Eng. Chem. Res.*, 47(10):3720–3751, 2008.

[25] Oleg Iliev, Ralf Kirsch, Zahra Lakdawala, Stefan Rief, and Konrad Steiner. Modeling and simulation of filtration processes. In *Currents in Industrial Mathematics*, pages 163–228. Springer, 2015.

[26] S. Bensaid, D. L. Marchisio, D. Fino, G. Saracco, and V. Specchia. Modelling of diesel particulate filtration in wall-flow traps. *Chemical Engineering Journal*, 154:211–218, 2009.

[27] Kuan-Mu Yao, Mohammad T Habibian, and Charles R O'Melia. Water and waste water filtration. concepts and applications. *Environmental science & technology*, 5(11):1105–1112, 1971.

[28] E Crevacore, G Boccardo, D Marchisio, and Rajandrea Sethi. Microscale colloidal transport simulations for groundwater remediation. *Chemical Engineering Transactions*, 47:271–276, 2016.

[29] Rouhollah Farajzadeh, Alexey Andrianov, Rumen Krastev, GJ Hirasaki, and William Richard Rossen. Foam–oil interaction in porous media: Implications for foam assisted enhanced oil recovery. *Adv. Colloid Interfac.*, 183:1–13, 2012.

[30] Paolo Gabrielli, Matteo Gazzani, and Marco Mazzotti. The role of carbon capture and utilization, carbon capture and storage, and biomass to enable a net-zero-co2 emissions chemical industry. *Ind. Eng. Chem. Res.*, 59(15):7033–7045, 2020.

[31] Renato Baciocchi, Giuseppe Storti, and Marco Mazzotti. Process design and energy requirements for the capture of carbon dioxide from air. *Chem. Eng. Process.*, 45(12):1047–1058, 2006.

[32] Alejandro A Franco. Multiscale modelling and numerical simulation of rechargeable lithium ion batteries: concepts, methods and challenges. *Rsc Adv.*, 3(32):13027–13058, 2013.

[33] Rui Fang, Christoph P Schmidt, and Wolfgang A Wall. A coupled finite element approach to spatially resolved lithium plating and stripping in three-dimensional anode microstructures of lithium-ion cells. *Journal of Computational Physics*, 461:111179, 2022.

[34] Ricardo Pinto Cunha, Teo Lombardo, Emiliano N Primo, and Alejandro A Franco. Artificial intelligence investigation of nmc cathode manufacturing parameters interdependencies. *Batteries & Supercaps*, 3(1):60–67, 2020.

[35] Matteo Icardi, Gianluca Boccardo, Daniele L Marchisio, Tiziana Tosco, and Rajandrea Sethi. Pore-scale simulation of fluid flow and solute dispersion in three-dimensional porous media. *Phys. Rev. E*, 90(1):013032, 2014.

[36] Gianluca Boccardo, Eleonora Crevacore, Alberto Passalacqua, and Matteo Icardi. Computational analysis of transport in three-dimensional heterogeneous materials. *Comput. Vis. Sci.*, 23(1):1–15, 2020.

[37] Kaj Pettersson, Dario Maggiolo, Srdjan Sasic, Pär Johansson, and Angela Sasic-Kalagasidis. On the impact of porous media microstructure on rainfall infiltration of thin homogeneous green roof growth substrates. *J. Hydrol.*, 582:124286, 2020.

[38] Dario Maggiolo, Francesco Picano, Filippo Zanini, Simone Carmignato, Massimo Guarnieri, Srdjan Sasic, and Henrik Ström. Solute transport and reaction in porous electrodes at high schmidt numbers. *J. Fluid Mech.*, 896, 2020.

[39] Federico Municchi, Nicodemo Di Pasquale, Marco Dentz, and Matteo Icardi. Heterogeneous multi-rate mass transfer models in openfoam®. *Comput. Phys. Commun.*, 261:107763, 2021.

[40] Lifei Zhao, Zhen Li, Zhicheng Wang, Bruce Caswell, Jie Ouyang, and George Em Karniadakis. Active-and transfer-learning applied to microscale-macroscale coupling to simulate viscoelastic flows. *Journal of Computational Physics*, 427:110069, 2021.

[41] Nicodemo Di Pasquale, Joshua D Elliott, Panagiotis Hadjidoukas, and Paola Carbone. Dynamically polarizable force fields for surface simulations via multi-output classification neural networks. *Journal of Chemical Theory and Computation*, 17(7):4477–4485, 2021.

[42] Davide Fissore, Antonello A Barresi, and Davide Manca. Modelling of methanol synthesis in a network of forced unsteady-state ring reactors by artificial neural networks for control purposes. *Chemical engineering science*, 59(19):4033–4041, 2004.

[43] Stephen Whitaker. *The method of volume averaging*, volume 13. Springer Science & Business Media, 2013.

[44] Federico Municchi and Matteo Icardi. Macroscopic models for filtration and heterogeneous reactions in porous media. *Advances in Water Resources*, 141:103605, 2020.

[45] Grégoire Allaire and Anne-Lise Raphael. Homogenization of a convection–diffusion model with reaction in a porous medium. *Comptes Rendus Mathematique*, 344(8):523–528, 2007.

[46] Ulrich Hornung. *Homogenization and porous media*, volume 6. Springer Science & Business Media, 1996.

[47] I Battiato and DM Tartakovsky. Applicability regimes for macroscopic models of reactive transport in porous media. *Journal of contaminant hydrology*, 120:18–26, 2011.

[48] Menachem Elimelech, John Gregory, and Xiadong Jia. *Particle deposition and aggregation: measurement, modelling and simulation*. Butterworth-Heinemann, 2013.

[49] Kirk E Nelson and Timothy R Ginn. New collector efficiency equation for colloid filtration in both natural and engineered flow conditions. *Water Resources Research*, 47(5), 2011.

[50] William P Johnson and Markus Hilpert. Upscaling colloid transport and retention under unfavorable conditions: Linking mass transfer to pore and grain topology. *Water Resources Research*, 49(9):5328–5341, 2013.

[51] Gianluca Boccardo, Daniele L Marchisio, and Rajandrea Sethi. Microscale simulation of particle deposition in porous media. *Journal of colloid and interface science*, 417:227–237, 2014.

[52] Ian L Molnar, William P Johnson, Jason I Gerhard, Clinton S Willson, and Denis M O'carroll. Predicting colloid transport through saturated porous media: A critical review. *Water Resources Research*, 51(9):6804–6845, 2015.

[53] Huilian Ma and William P Johnson. Colloid retention in porous media of various porosities: Predictions by the hemispheres-in-cell model. *Langmuir*, 26(3):1680–1687, 2010.

[54] Huilian Ma, Michal Hradisky, and William P Johnson. Extending applicability of correlation equations to predict colloidal retention in porous media at low fluid velocity. *Environmental science & technology*, 47(5):2272–2278, 2013.

[55] Francesca Messina, Tiziana Tosco, and Rajandrea Sethi. On the failure of upscaling the single-collector efficiency to the transport of colloids in an array of collectors. *Water Resources Research*, 52(7):5492–5505, 2016.

[56] Mahdi Saeedan, Ali Reza Solaimany Nazar, Yaser Abbasi, and Reza Karimi. Cfd investigation and neutral network modeling of heat transfer and pressure drop of nanofluids in double pipe helically baffled heat exchanger with a 3-d fined tube. *Applied Thermal Engineering*, 100:721–729, 2016.

[57] Yangyao Ding, Yichi Zhang, Yi Ming Ren, Gerassimos Orkoulas, and Panagiotis D Christofides. Machine learning-based modeling and operation for ald of sio2 thin-films using data from a multiscale cfd simulation. *Chemical Engineering Research and Design*, 151:131–145, 2019.

[58] Agnese Marcato, Gianluca Boccardo, and Daniele L Marchisio. A computational workflow to study particle transport in porous media: coupling cfd and deep learning. In *Computer Aided Chemical Engineering*, volume 48, pages 1759–1764. Elsevier, 2020.

[59] Jinlong Wu, Xiaolong Yin, and Heng Xiao. Seeing permeability from images: fast prediction with convolutional neural networks. *Science bulletin*, 63(18):1215–1222, 2018.

[60] Naif Alqahtani, Fatimah Alzubaidi, Ryan T Armstrong, Pawel Swietojanski, and Peyman Mostaghimi. Machine learning for predicting properties of porous media from 2d x-ray images. *Journal of Petroleum Science and Engineering*, 184:106514, 2020.

[61] Ying Da Wang, Traiwit Chung, Ryan T Armstrong, and Peyman Mostaghimi. Ml-lbm: predicting and accelerating steady state flow simulation in porous media with convolutional neural networks. *Transport in Porous Media*, 138(1):49–75, 2021.

[62] Javier E Santos, Duo Xu, Honggeun Jo, Christopher J Landry, Maša Prodanović, and Michael J Pyrcz. Poreflow-net: A 3d convolutional neural network to predict fluid flow through porous media. *Advances in Water Resources*, 138:103539, 2020.

[63] Oliver Hennigh. Lat-net: compressing lattice boltzmann flow simulations using deep neural networks. *arXiv preprint arXiv:1705.09036*, 2017.

[64] Nanzhe Wang, Haibin Chang, and Dongxiao Zhang. Efficient uncertainty quantification for dynamic subsurface flow with surrogate by theory-guided neural network. *Computer Methods in Applied Mechanics and Engineering*, 373:113492, 2021.

[65] Eric Laloy, Romain Hérault, John Lee, Diederik Jacques, and Niklas Linde. Inversion using a new low-dimensional representation of complex binary geological media based on a deep neural network. *Advances in water resources*, 110:387–405, 2017.

[66] Shaoxing Mo, Yinhao Zhu, Nicholas Zabaras, Xiaoqing Shi, and Jichun Wu. Deep convolutional encoder-decoder networks for uncertainty quantification of dynamic multiphase flow in heterogeneous media. *Water Resources Research*, 55(1):703–728, 2019.

[67] Meng Tang, Yimin Liu, and Louis J Durlofsky. A deep-learning-based surrogate model for data assimilation in dynamic subsurface flow problems. *Journal of Computational Physics*, 413:109456, 2020.

[68] Shaoxing Mo, Nicholas Zabaras, Xiaoqing Shi, and Jichun Wu. Deep autoregressive neural networks for high-dimensional inverse problems in groundwater contaminant source identification. *Water Resources Research*, 55(5):3856–3881, 2019.

[69] Javier E Santos, Ying Yin, Honggeun Jo, Wen Pan, Qinjun Kang, Hari S Viswanathan, Maša Prodanović, Michael J Pyrcz, and Nicholas Lubbers. Computationally efficient multiscale neural networks applied to fluid flow in complex 3d porous media. *Transport in porous media*, 140(1):241–272, 2021.

[70] R Byron Bird. Transport phenomena. *Appl. Mech. Rev.*, 55(1):R1–R4, 2002.

[71] Henry Darcy. *Les fontaines publiques de la ville de Dijon: Exposition et application des principes à suivre et des formules à employer dans les questions de distribution d'eau: Ouvrage terminé par un appendice relatif aux fournitures d'eau de plusieurs villes, au filtrage des eaux et à la fabrication des tuyaux de fonte, de plomb, de tôle et de bitume*, volume 2. V. Dalmont, 1856.

[72] Philipp Forchheimer. Wasserbewegung durch boden. *Z. Ver. Deutsch, Ing.*, 45:1782–1788, 1901.

[73] Albert Einstein. Über die von der molekularkinetischen theorie der wärme geforderte bewegung von in ruhenden flüssigkeiten suspendierten teilchen. *Annalen der physik*, 4, 1905.

[74] Simon L Goren and Michael E O'Neill. On the hydrodynamic resistance to a particle of a dilute suspension when in the neighbourhood of a large obstacle. *Chemical Engineering Science*, 26(3):325–338, 1971.

[75] Howard Brenner. The slow motion of a sphere through a viscous fluid towards a plane surface. *Chemical engineering science*, 16(3-4):242–251, 1961.

[76] Z Adamczyk, T Dabros, J Czarnecki, and Th GM Van De Ven. Particle transfer to solid surfaces. *Advances in Colloid and Interface Science*, 19(3):183–252, 1983.

[77] Rajamani Rajagopalan and Chi Tien. Trajectory analysis of deep-bed filtration with the sphere-in-cell porous media model. *AIChE Journal*, 22(3):523–533, 1976.

[78] Veniamin Grigorevich Levich. Physicochemical hydrodynamics. 1962.

[79] Dennis C Prieve and Eli Ruckenstein. Effect of london forces upon the rate of deposition of brownian particles. *AIChE Journal*, 20(6):1178–1187, 1974.

[80] Bruce E Logan, DG Jewett, RG Arnold, EJ Bouwer, and CR O'Melia. Clarification of clean-bed filtration models. *Journal of environmental engineering*, 121(12):869–873, 1995.

[81] William P Johnson and Markus Hilpert. Upscaling colloid transport and retention under unfavorable conditions: Linking mass transfer to pore and grain topology. *Water Resources Research*, 49(9):5328–5341, 2013.

[82] Kirk E Nelson and Timothy R Ginn. New collector efficiency equation for colloid filtration in both natural and engineered flow conditions. *Water Resources Research*, 47(5), 2011.

[83] Gianluca Boccardo, Eleonora Crevacore, Rajandrea Sethi, and Matteo Icardi. A robust upscaling of the effective particle deposition rate in porous media. *Journal of contaminant hydrology*, 212:3–13, 2018.

[84] Ghassan Zubi, Rodolfo Dufo-López, Monica Carvalho, and Guzay Pasaoglu. The lithium-ion battery: State of the art and future perspectives. *Renewable and Sustainable Energy Reviews*, 89:292–308, 2018.

[85] Abbas Fotouhi, Daniel J Auger, Karsten Propp, Stefano Longo, and Mark Wild. A review on electric vehicle battery modelling: From lithium-ion toward lithium–sulphur. *Renewable and Sustainable Energy Reviews*, 56:1008–1021, 2016.

[86] Zeyu Deng, Vipin Kumar, Felix T Bölle, Fernando Caro, Alejandro A Franco, Ivano E Castelli, Pieremanuele Canepa, and Zhi Wei Seh. Towards autonomous high-throughput multiscale modelling of battery interfaces. *Energy & Environmental Science*, 15(2):579–594, 2022.

[87] Ali Jokar, Barzin Rajabloo, Martin Désilets, and Marcel Lacroix. Review of simplified pseudo-two-dimensional models of lithium-ion batteries. *Journal of Power Sources*, 327:44–55, 2016.

[88] Valentin Sulzer, Scott G Marquis, Robert Timms, Martin Robinson, and S Jon Chapman. Python battery mathematical modelling (pybamm). *Journal of Open Research Software*, 9(1), 2021.

[89] Marc Doyle, Thomas F Fuller, and John Newman. Modeling of galvanostatic charge and discharge of the lithium/polymer/insertion cell. *Journal of the Electrochemical society*, 140(6):1526, 1993.

[90] Graham M Goldin, Andrew M Colclasure, Andreas H Wiedemann, and Robert J Kee. Three-dimensional particle-resolved models of li-ion batteries to assist the evaluation of empirical parameters in one-dimensional models. *Electrochimica Acta*, 64:118–129, 2012.

[91] Zohaib Atiq Khan, Mehrez Agnaou, Mohammad Amin Sadeghi, Ali Elkamel, and Jeff T Gostick. Pore network modelling of galvanostatic discharge behaviour of lithium-ion battery cathodes. *Journal of The Electrochemical Society*, 168(7):070534, 2021.

[92] Mehdi M Forouzan, Chien-Wei Chao, Danilo Bustamante, Brian A Mazzeo, and Dean R Wheeler. Experiment and simulation of the fabrication process of lithium-ion battery cathodes for determining microstructure and mechanical properties. *Journal of Power Sources*, 312:172–183, 2016.

[93] Andreas H Wiedemann, Graham M Goldin, Scott A Barnett, Huayang Zhu, and Robert J Kee. Effects of three-dimensional cathode microstructure on the performance of lithium-ion battery cathodes. *Electrochimica Acta*, 88:580–588, 2013.

[94] Alain C Ngandjong, Alexis Rucci, Mariem Maiza, Garima Shukla, Jorge Vazquez-Arenas, and Alejandro A Franco. Multiscale simulation platform linking lithium ion battery electrode fabrication process with performance at the cell level. *The journal of physical chemistry letters*, 8(23):5966–5972, 2017.

[95] Chaoyue Liu, Teo Lombardo, Jiahui Xu, Alain C Ngandjong, and Alejandro A Franco. An experimentally-validated 3d electrochemical model revealing electrode manufacturing parameters' effects on battery performance. *Energy Storage Materials*, 54:156–163, 2023.

[96] Giles W Richardson, Jamie M Foster, Rahifa Ranom, Colin P Please, and Angel M Ramos. Charge transport modelling of lithium-ion batteries. *European Journal of Applied Mathematics*, 33(6):983–1031, 2022.

[97] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. " O'Reilly Media, Inc.", 2022.

[98] Ke-Lin Du and Madisetti NS Swamy. *Neural networks in a softcomputing framework*, volume 501. Springer, 2006.

[99] Amirhossein Tavanaei, Masoud Ghodrati, Saeed Reza Kheradpisheh, Timothée Masquelier, and Anthony Maida. Deep learning in spiking neural networks. *Neural networks*, 111:47–63, 2019.

[100] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[101] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13733–13742, 2021.

[102] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[103] Jianwei Tian, Chongchong Qi, Yingfeng Sun, Zaher Mundher Yaseen, and Binh Thai Pham. Permeability prediction of porous media using a combination of computational fluid dynamics and hybrid machine learning methods. *Engineering with Computers*, pages 1–17, 2020.

[104] Arash Rabbani and Masoud Babaei. Hybrid pore-network and lattice-boltzmann permeability modelling accelerated by machine learning. *Adv. Water Resour.*, 126:116–128, 2019.

[105] Yehuda Bachmat and Jacob Bear. On the concept and size of a representative elementary volume (rev). In *Advances in transport phenomena in porous media*, pages 3–20. Springer, 1987.

[106] Gianluca Boccardo, Rajandrea Sethi, and Daniele L Marchisio. Fine and ultrafine particle deposition in packed-bed catalytic reactors. *Chem. Eng. Sci.*, 198:290–304, 2019.

[107] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[108] Menachem Elimelech. Kinetics of capture of colloidal particles in packed beds under attractive double layer interactions. *Journal of colloid and interface science*, 146(2):337–352, 1991.

[109] Nathalie Tufenkji and Menachem Elimelech. Correlation equation for predicting single-collector efficiency in physicochemical filtration in saturated porous media. *Env. Sci. Tech.*, 38(2):529–536, 2004.

[110] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[111] Siyan Liu, Arsalan Zolfaghari, Shariar Sattarin, Amirmasoud Kalantari Dahaghi, and Shahin Negahban. Application of neural networks in multiphase flow through porous media: Predicting capillary pressure and relative permeability curves. *J. Petrol. Sci. Eng.*, 180:445–455, 2019.

[112] Haiyi Wu, Wen-Zhen Fang, Qinjun Kang, Wen-Quan Tao, and Rui Qiao. Predicting effective diffusivity of porous media from images by deep learning. *Sci. Rep.*, 9(1):1–12, 2019.

[113] J Salles, J-F Thovert, Renaud Delannay, L Prevors, J-L Auriault, and PM Adler. Taylor dispersion in porous media. determination of the dispersion tensor. *Physics of Fluids A: Fluid Dynamics*, 5(10):2348–2376, 1993.

[114] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[115] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11976–11986, 2022.

[116] Jonathan T Barron. Continuously differentiable exponential linear units. *arXiv preprint arXiv:1704.07483*, 2017.

[117] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[118] Lu Lu, Yeonjong Shin, Yanhui Su, and George Em Karniadakis. Dying relu and initialization: Theory and numerical examples. *arXiv preprint arXiv:1903.06733*, 2019.

[119] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[120] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[121] Jason Furtney. scikit-fmm: the fast marching method for python. https://github.com/scikit-fmm/scikit-fmm, 2022. [Online; accessed 2-February-2022].

[122] Jeff T Gostick, Zohaib A Khan, Thomas G Tranter, Matthew DR Kok, Mehrez Agnaou, Mohammadamin Sadeghi, and Rhodri Jervis. Porespy: A python toolkit for quantitative analysis of porous media images. *Journal of Open Source Software*, 4(37):1296, 2019.

[123] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[124] Agnese Marcato, Javier E Santos, Chaoyue Liu, Gianluca Boccardo, Daniele Marchisio, and Alejandro A Franco. Modeling the 4d discharge of lithium-ion batteries with a multiscale time-dependent deep learning framework. *Energy Storage Materials*, page 102927, 2023.

[125] Zhe Deng, Xing Lin, Zhenyu Huang, Jintao Meng, Yun Zhong, Guangting Ma, Yu Zhou, Yue Shen, Han Ding, and Yunhui Huang. Recent progress on advanced imaging techniques for lithium-ion batteries. *Advanced Energy Materials*, 11(2):2000806, 2021.

[126] Francois LE Usseglio-Viretta, Andrew Colclasure, Aashutosh N Mistry, Koffi Pierre Yao Claver, Fezzeh Pouraghajan, Donal P Finegan, Thomas MM Heenan, Daniel Abraham, Partha P Mukherjee, Dean Wheeler, et al. Resolving the discrepancy in tortuosity factor estimation for li-ion battery electrodes through micro-macro modeling and experiment. *Journal of The Electrochemical Society*, 165(14):A3403, 2018.

[127] A. P. Thompson, H. M. Aktulga, R. Berger, D. S. Bolintineanu, W. M. Brown, P. S. Crozier, P. J. in 't Veld, A. Kohlmeyer, S. G. Moore, T. D. Nguyen, R. Shan, M. J. Stevens, J. Tranchida, C. Trott, and S. J. Plimpton. LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. *Comp. Phys. Comm.*, 271:108171, 2022.

[128] Alexis Rucci, Alain C Ngandjong, Emiliano N Primo, Mariem Maiza, and Alejandro A Franco. Tracking variabilities in the simulation of lithium ion battery electrode fabrication and its impact on electrochemical performance. *Electrochimica Acta*, 312:168–178, 2019.

[129] Alain C Ngandjong, Teo Lombardo, Emiliano N Primo, Mehdi Chouchane, Abbos Shodiev, Oier Arcelus, and Alejandro A Franco. Investigating electrode calendering and its impact on electrochemical performance by means of a new discrete element method model: Towards a digital twin of li-ion battery manufacturing. *Journal of Power Sources*, 485:229320, 2021.

[130] Sohrab R Daemi, Chun Tan, Tobias Volkenandt, Samuel J Cooper, Anna Palacios-Padros, James Cookson, Dan JL Brett, and Paul R Shearing. Visualizing the carbon binder phase of battery electrodes in three dimensions. *ACS Applied Energy Materials*, 1(8):3702–3710, 2018.

[131] Johannes Sturm, Alexander Rheinfeld, Ilya Zilberman, Franz B Spingler, Stephan Kosch, Fabian Frie, and Andreas Jossen. Modeling and simulation of inhomogeneities in a 18650 nickel-rich, silicon-graphite lithium-ion cell during fast charging. *Journal of Power Sources*, 412:204–223, 2019.

[132] Chaoyue Liu, Hang Li, Xiangbang Kong, and Jinbao Zhao. Modeling analysis of the effect of battery design on internal short circuit hazard in lini0. 8co0. 1mn0. 1o2/siox-graphite lithium ion batteries. *International Journal of Heat and Mass Transfer*, 153:119590, 2020.

[133] Jelle Smekens, J Paulsen, W Yang, N Omar, J Deconinck, A Hubin, and J Van Mierlo. A modified multiphysics model for lithium-ion batteries with a lixni1/3mn1/3co1/3o2 electrode. *Electrochimica Acta*, 174:615–624, 2015.

[134] Johannes Landesfeind and Hubert A Gasteiger. Temperature and concentration dependence of the ionic transport properties of lithium-ion battery electrolytes. *Journal of The Electrochemical Society*, 166(14):A3079, 2019.

[135] Shao-Ling Wu, Wei Zhang, Xiangyun Song, Alpesh K Shukla, Gao Liu, Vincent Battaglia, and Venkat Srinivasan. High rate capability of li (ni1/3mn1/3co1/3) o2 electrode for li-ion batteries. *Journal of The Electrochemical Society*, 159(4):A438, 2012.

[136] Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. Learning precise timing with lstm recurrent networks. *Journal of machine learning research*, 3(Aug):115–143, 2002.

[137] Vaclav Smilauer. *Yade documentation*. The Yade Project, November 2021.

[138] PA Cundall and ODL Strack. Discussion: a discrete numerical model for granular assemblies. *Geotechnique*, 30(3):331–336, 1980.

[139] Enrico Agostini, Gianluca Boccardo, and Daniele Marchisio. An open-source workflow for open-cell foams modelling: Geometry generation and cfd simulations for momentum and mass transport. *Chemical Engineering Science*, 255:117583, 2022.

[140] Lorenzo Stratta, Merve B Adali, Antonello A Barresi, Gianluca Boccardo, Agnese Marcato, Raffaele Tuccinardi, and Roberto Pisano. A diffused-interface model for the lyophilization of a packed bed of spray-frozen particles. *Chemical Engineering Science*, 275:118726, 2023.

[141] Jeff T Gostick. Versatile and efficient pore network extraction method using marker-based watershed segmentation. *Physical Review E*, 96(2):023307, 2017.

[142] Pauli Virtanen. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

Appendix A

Creation of sphere packings by discrete element method

The sphere packings, employed as geometries in the CFD simulations, were created by a discrete element method (DEM) code in order to physically reproduce a realistic particle packing. Two kind of simulations have been conceived to this end: sedimentation A.1 and periodic compression simulations A.2. In this work the open-source Yade code was employed [137], the basic theory of the DEM simulations and the numerical details are presented in this section in relation to this tool.

A.1 Sedimentation simulations

In the sedimentation simulations the initial configuration for the objects to be packed is a cloud of spheres in a box which will then be subjected to gravity deposition: as a result a realistic packing of spheres is obtained, Fig. A.1. Each particle in the computational box interacts with the neighboring particles, these interactions are modeled by a non-cohesive elastic-frictional contact model [138]. In this modelling approach the particles are allowed to overlap at the contact point depending on the Young's module of the materials. In particular, the contact force between two spheres is made by two contributions: the normal force and the shear force.

The normal force arises from a difference of linear velocity along the interaction axis, and reads as follows:

$$\mathbf{F}_N = K_N \mathbf{u}_N, \tag{A.1}$$

where $\mathbf{u}_N$ is the normal displacement and K_N is the normal interaction stiffness, that is defined as:

$$K_N = 2\frac{E_i r_i E_j r_j}{E_i r_i + E_j r_j}, \tag{A.2}$$

where E is the Young's modules of i and j particles materials and r is the radius of particle i and j. If $\mathbf{u}_N \geq 0$ the normal force is null, so the interactions between spheres that are not in contact are not considered in the model.

The shear force arises from the perpendicular component of the linear velocities difference and from the perpendicular component of rotational velocities summation. It is defined in the same way with the corresponding shear stiffness and displacement:

$$\mathbf{F}_T = K_T \mathbf{u}_T, \tag{A.3}$$

where K_T is a function of K_N according to the Poisson's law. A check is performed on the shear force in order to follow the Coulomb friction law, that imposes a maximum limit equal to $\mathbf{F}_N \, tan(\phi)$, where ϕ is the friction angle between the two particles.

Given the forces, the DEM code integrates the motion equation for each particle. Starting from the position of a certain sphere, $\mathbf{x}_P(t)$, the code calculates the position after the time step Δt, $\mathbf{x}_P(t+\Delta t)$, integrating the Newton's second law for translation and rotation:

$$m_P \frac{\partial^2 \mathbf{x}}{\partial t^2} = \sum_{i=0}^{n_C} (\mathbf{F}_{i,P}^N + \mathbf{F}_{i,P}^T) + m_P \mathbf{g}, \tag{A.4}$$

$$\mathbf{I}_P \frac{\partial \omega}{\partial t} = \sum_{i=0}^{n_C} (\mathbf{F}_{i,P}^T \times d_p(n)), \tag{A.5}$$

where m_P is the mass of the particle, $\mathbf{g}$ is the gravitational acceleration, n_C is the number of contact points, $\mathbf{F}_{i,P}^N$ and $\mathbf{F}_{i,P}^T$ are the normal and shear component of the contact force between the couple of particles, $\mathbf{I}_P$ is the moment of inertia, ω is the angular velocity, d_P is the distance between the contact point and the center of the particle, and n is the contact plane normal.

The numerical damping, which is applied on the forces acting on the particles in order to dissipate kinetic energy every time step, introduced in the Yade formulation

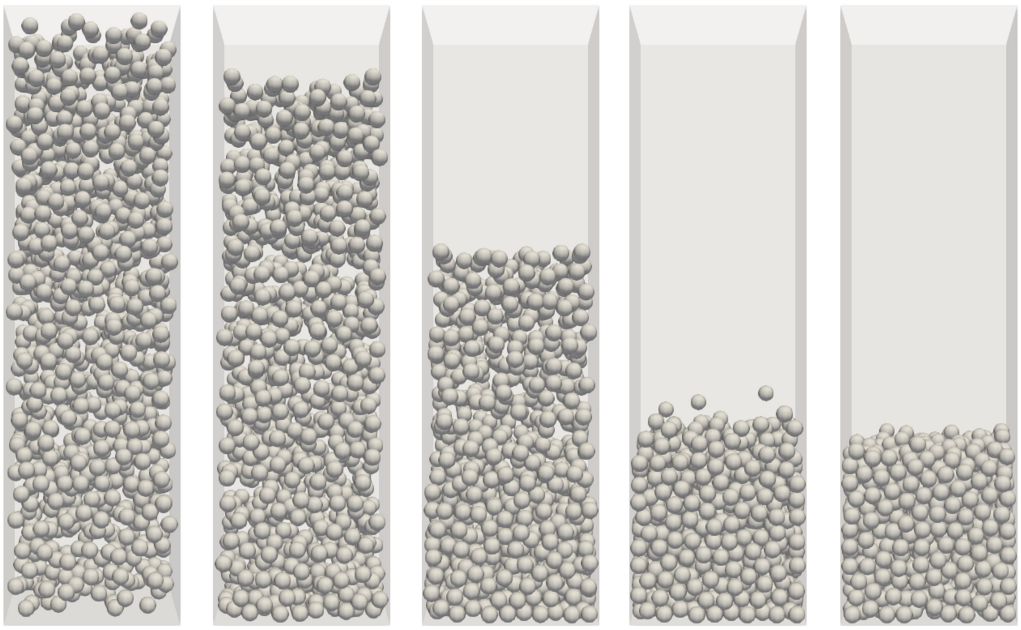

Fig. A.1 Creation of packing of spheres by gravity deposition: from a cloud of sphere (first time-frame) to the final packing (last time-frame).

is set equal to 0.4, as suggested by the developers. The time step of integration is set equal to half of the critical timestep in order to guarantee stable solution. The critical timestep is calculated as

$$\Delta t_{cr} = min_i \left(R_i \sqrt{\frac{\rho_i}{E_i}} \right), \tag{A.6}$$

where ρ is the density of the particle. This formulation is based on the propagation of the elastic waves in the solid at the speed of sound, in fact, it must be lower than the minimum distance of the integration points, the radius in this case. Finally the criterion on the unbalanced forces was set equal to $1 \cdot 10^{-3}$ Pa.

The creation of sphere packings by gravity deposition is advised when one is interested in the simulation of the real process of packing, such as the filling of a reactor by a catalyst [23]. In this way the porosity of the packing will be influenced by the wall effect (as in, the solid walls acting as a constraint to the position of the outermost spheres), but a realistic behaviour of the flow and transport can be studied based on a realistic geometry.

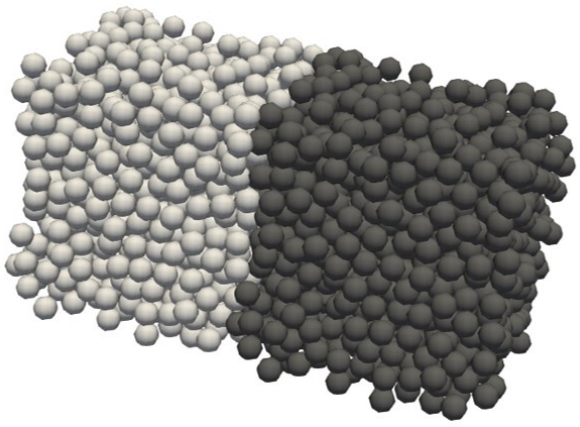

Fig. A.2 Periodic packing created via Yade, the lighter and the darker packings are the same, in fact, on the edge they enter in contact in continuity because of periodicity.

A.2 Periodic compression simulations

By periodic compression simulations the packing results completely periodic in all the directions, as displayed in Fig. A.2. The initial position of the spheres in the computational box is set in a cloud as in the case of the sedimentation simulations.

The cell is subjected to a deformation over time, an iso-compression, so a velocity gradient $\nabla \mathbf{v}$ is imposed on the bounding box in an homogeneous way. The transformation matrix $\mathbf{F}$ is updated as follow:

$$\mathbf{F}(t+\Delta t) = (\mathbf{I} + \mathbf{v}\Delta t)\, F(t), \tag{A.7}$$

where $\mathbf{I}$ is the identity matrix.

The compression ramp is set by a periodic boundary condition, in this work the default values suggested by the Yade developers have been used, so a range of stresses between $-100 \cdot 10^9$ and $-1 \cdot 10^8$ Pa. The Newton equations of motion, Eq. A.4 - A.5, are integrated imposing the stress at each timestep. The criterion set on the unbalanced forces is equal to $1 \cdot 10^{-2}$ Pa.

This kind of packings should be employed when one is interested in studying a bulk porous medium, as in this way no wall effect is embedded in the geometry.

Appendix B

Porous media characterization with Python

Porous media can be deeply heterogeneous and random systems, so their geometrical characterization is essential to extract macroscale quantities useful for labelling different samples (i.e. foams employed for filtration and as support for catalysts [139], rocks [62], or dried products [140]). The solid phase as well as the pore space should be subject to a careful statistical analysis in order to have a complete understanding of the system. The tools usually employed for the characterization of the porous structure can also be exploited to extract the descriptive features for convolutional neural networks.

The tools used in the work to this end are detailed in the following two sections: in B.1 the problem of the geometries voxelization is addressed, in B.2 the main characterization techniques are summarized.

B.1 Voxelization

Most of the tools employed for the characterization of porous media come from the image processing field, so the input is supposed to be a bi-dimensional or three-dimensional array of grey-scale values. Even very specific tools for porous media work with images as starting objects, since the wider pool of data comes

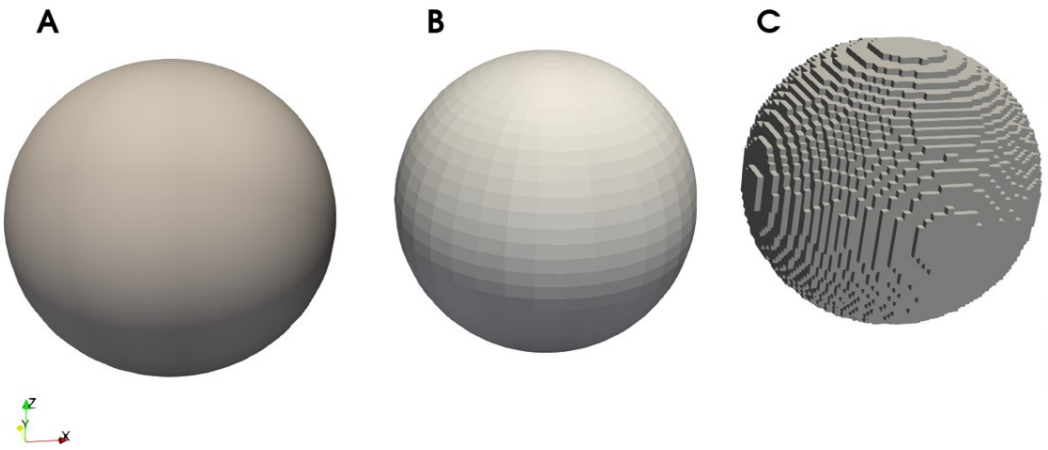

Fig. B.1 Spherical particle represented analytically (A), in `.stl` format (B), and as a result of the voxelization (C).

from segmentation of images from imaging techniques (i.e. electron microscopy, tomographies).

In this work the geometries are produced *in silico* by means of a discrete element method (as described in Appendix A). The resulting geometry is a packing of spheres described by the coordinates of their centers and their radii. In order to obtain a matrix representation of the geometries the steps to follow are:

1. extraction of the bounding surface between solid and pore phase. The most common format is the standard triangle language (`.stl`), the number of elements of the surface affect the size of the file and the accuracy of the object reproduction.

2. voxelization of the domain. The Python library `trimesh` was used. The surface must be watertight in order to fill the voxels inside it, instead the pitch (i.e. the size of the voxel) determines the resolution of the object.

B.2 Characterization tools

The euclidean distance transform of a porous media image gives an interesting insight in the local pore dimensions, and as such is a common step for other characterizing

features, like the pore size distribution, or the pore network extraction [141]. Moreover, it is a very fast algorithm which is cheap to apply even in three-dimensional images, for this reason it is widely used as input feature for convolutional neural network models.

The euclidean distance transform computes the minimum distance between each point in the pore phase (r^p) and the closest point of solid phase (r^s):

$$ed = \sqrt{\sum_{k=x,y,z} \left(r_k^p - r_k^s\right)^2} \tag{B.1}$$

In this work the function available in the Python library `scipy.ndimage` [142] was employed to compute the euclidean distance transform of three-dimensional porous media structures.

Besides this general-purpose libraries for multidimensional image processing, the library PoreSpy [122] can be a powerful tool since it was developed for specific porous media applications. In fact, it implements a function to evaluate the pore size distribution, a property of interest in this work. The pore size distribution is the probability density function of the dimensions of the voids in a porous medium. The approach implemented in PoreSpy consists in finding which voxel can accommodate a sphere of a given radius. Given the number of sizes and the bins in the size distribution, the algorithm detects the largest pore applying the Euclidean distance transform to the image. After that the algorithm looks for all the smaller spheres between 0 and the largest one by using the Fast Fourier Transform convolution. The result is a map of the spheres placed in the pore phase, from which the pore size distribution histogram can be easily computed, together with the mean dimension of the pores.

Here follows an example of a script where the steps of section B.1 are implemented and the features of section B.2 are computed, the output images are shown in Figure B.2.

```
import trimesh
import numpy
import matplotlib.pyplot as plt
import porespy
from scipy.ndimage import morphology

pitch = 0.003
```

```
mesh = trimesh.load_mesh('packing.stl')
volume = mesh.voxelized(pitch=pitch).fill(method='holes')
matrix = volume.matrix.astype(numpy.float64)
porespy.io.to_vtk(matrix,'matrix',voxel_size=0.01) #for
    Paraview visualization
im = numpy.logical_not(matrix)
ed = morphology.distance_transform_edt(im)
thk = porespy.filters.local_thickness(im, sizes= 100,mode='dt')
psd = porespy.metrics.pore_size_distribution(im=thk,bins=20)
plt.figure()
plt.xlabel('Pore Radius [voxels]')
plt.ylabel('Normalized Volume Fraction')
fig = plt.bar(x=psd.bin_centers, height=psd.pdf, width=psd.
    bin_widths)
plt.figure()
ed[im==0] = numpy.nan
plt.imshow(ed[:,int(ed.shape[1]/2.),:])
plt.colorbar()
plt.figure()
thk[im==0] = numpy.nan
plt.imshow(thk[:,int(ed.shape[1]/2.),:])
plt.colorbar()
plt.show()
```

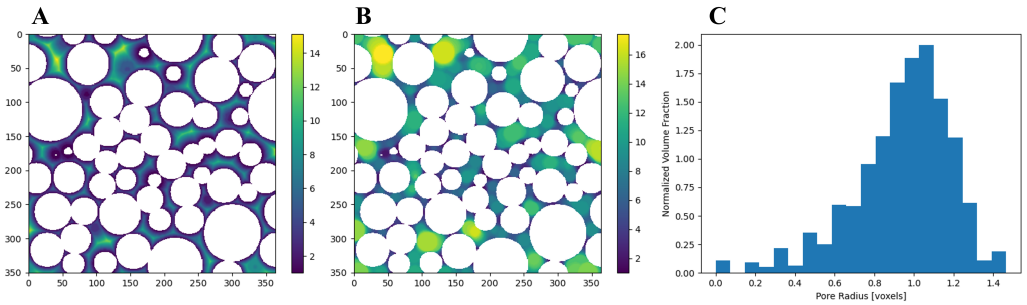

Fig. B.2 Output images of the Python code: the euclidean distance transform (A), the local thickness feature (B), and the pore size distribution histogram (C).

Appendix C

Numerical details of the CFD simulations on OpenFOAM

In this appendix the numerical setup of the CFD simulations is detailed in order to facilitate the reproducibility of the dataset employed in this work.

The OpenFOAM v6 and OpenFOAM v8 have been employed throughout the thesis work. The momentum and continuity equations have been solved using the `simpleFoam` solver, while the advection-diffusion equation has been solved using the `scalarTransportFoam` solver.

Concerning the coupled solution of the Navier-Stokes and continuity equations, the standard `OpenFoam` second order schemes proposed for `simpleFoam` were employed:

- Gradient scheme: `Gauss linear` - Gaussian second order integration linear interpolation;
- Divergence scheme: `bounded Gauss linearUpwind grad(U)` - bounded Gaussian second order integration linear upwind interpolation;
- Laplacian scheme: `Gauss linear corrected` - Gaussian second order integration linear interpolation;

The convergence criteria on the residuals are set: 10^{-6} for the pressure, 10^{-5} for the velocity.

For the solution of the advection-diffusion equation by `scalarTransportFoam`, the default schemes have been changed into limited second order schemes to force the normalized concentration to range between 0 and 1.

- Gradient scheme: `faceLimited Gauss linear 1` - limited Gaussian second order integration linear interpolation;
- Divergence scheme: `Gauss limitedLinear 1` - limited Gaussian second order integration linear interpolation;
- Laplacian scheme: `Gauss linear limited corrected 1` - limited Gaussian second order integration linear interpolation;

The equation is solved in steady state conditions. The convergence criteria on the concentration residual is 10^{-6}.