Department of Computer Science

Machine Learning-based Generalized Multiscale Finite Element Method and its Application in Reservoir Simulation

Thesis submitted in accordance with the requirements of the University of Liverpool for the degree of Doctor in Philosophy

by

Abouzar Choubineh

November 2023

Dedication

This thesis is dedicated to the unfolding of my fate during my PhD study, encompassing numerous amazing experiences along with a few unanticipated challenges that have fostered within me happiness, wisdom, adaptability, and resilience.

Acknowledgements

As I look back on my PhD journey, especially the quarantine period which caused me a kind of stress and depression, I recall many people who have helped, supported, and guided me.

First, I am immensely grateful to my primary supervisor in XJTLU, Dr. Jie Chen, for his support, guidance, and advice throughout this research journey. His expertise in numerical modelling and porous media has been instrumental in shaping this thesis and my growth as a researcher. I would also like to thank Prof. Frans Coenen, my Liverpool supervisor, for arranging regular meetings and helping me in the field of machine learning/deep learning. Furthermore, I thank Prof. Fei Ma, my second supervisor in XJTLU for his feedback and suggestions.

I am very thankful to XJTLU and UoL for supporting me, as a fully-funded student, both financially and by providing me with resources and facilities.

I would like to extend my appreciation to the internal and external examiners of the viva, the IPAP members, and anonymous reviewers of my manuscripts. Their thoughtful feedback, suggestions, and constructive criticism have enhanced the quality of this work.

Furthermore, I am grateful to Dr. Roslyn Irving, Dr. David Wood, Dr. Trevor Mahy, and Mr. Zongyi Li for providing helpful comments and suggestions.

Specifically, I want to express my deepest appreciation to my family for their unconditional love, encouragement, understanding, and belief in me.

Lastly, I would like to thank other individuals who have supported me directly or indirectly throughout this research endeavor. I deeply appreciate their contributions.

Declaration

I hereby declare that this thesis, which contains 109 pages with 9 tables and 26 figures, has been composed by myself, and the results presented herein correspond to my original work, to fulfill the requirements for the PhD degree from the University of Liverpool. The thesis has not been previously published, is not currently submitted elsewhere, and will not be submitted elsewhere for a degree, diploma, or any other qualifications.

Abouzar Choubineh
November 2023

Abstract

In multiscale modeling of subsurface fluid flow in heterogeneous porous media, standard polynomial basis functions are replaced by multiscale basis functions, which are used to predict pressure distribution. To produce such functions in the mixed Generalized Multiscale Finite Element Method (GMsFEM), a number of Partial Differential Equations (PDEs) must be solved, leading to significant computational overhead. The main objective of the work presented in this thesis was to investigate the efficiency of Machine Learning (ML)/Deep Learning (DL) models in reconstructing the multiscale basis functions (Basis 2, 3, 4, and 5) of the mixed GMsFEM. To achieve this, four standard models named SkiplessCNN models were first developed to predict four different multiscale basis functions. These predictions were based on two distinct datasets (initial and extended) generated, with the permeability field being the sole input. Subsequently, focusing on the extended dataset, three distinct skip connection schemes (FirstSkip, MidSkip, and DualSkip) were incorporated into the SkiplessCNN architecture. Following this, the four developed models - SkiplessCNN, FirstSkipCNN, MidSkipCNN, and DualSkipCNN - were separately combined using linear regression and ridge regression within the framework of Deep Ensemble Learning (DEL). Furthermore, the reliability of the DualSkipCNN model was examined using Monte Carlo (MC) dropout. Ultimately, two Fourier Neural Operator (FNO) models, operating on infinite-dimensional spaces, were developed based on a new dataset for directly predicting pressure distribution. Based on the results, sufficient data for the validation and testing subsets could help decrease overfitting. Additionally, all three skip connections were found to be effective in enhancing the performance of SkiplessCNN, with DualSkip being the most effective among them. As evaluated on the testing subset, the combined models using linear regression and ridge regression significantly outperformed the individual models for all basis functions. The results also confirmed the robustness of MC dropout for DualSkipCNN in terms of epistemic uncertainty. Regarding the FNO models, it was discovered that the inclusion of a MultiLayer Perceptron (MLP) in the original Fourier layers significantly improved the prediction performance on the testing subset. Looking at this work as an image (matrix)-to-image (matrix) problem, the developed data-driven models through various techniques could find applications beyond reservoir engineering.

Keywords: subsurface fluid flow, finite element method, applied machine learning/deep learning, big data, reliability, neural operator.

Contents

List of Tables

List of Figures

Acronyms and Nomenclature

1D	one-dimensional
2D	two-dimensional
3D	three-dimensional
AdaBoost	Adaptive Boosting
AE	AutoEncoder
AI	Artificial Intelligence
ARD	Automatic Relevance Determination
BC	Boundary Condition
BN	Batch Normalization
BNN	Bayesian Neural Network
CNN	Convolutional Neural Network
DCCNN	Densely Connected Convolutional Neural Network
DEIM	Discrete Empirical Interpolation Method
DEL	Deep Ensemble Learning
DFM	Discrete Fracture Model
DFN	Discrete Fracture Network
DL	Deep Learning
DNN	Deep Neural Network
DT	Decision Tree
FC	Fully Connected
FEM	Finite Element Method
FFT	Fast Fourier Transform
FNO	Fourier Neural Operator
GBRT	Gradient Boosted Regression Tree
GMsFEM	Generalized Multiscale Finite Element Method

GNAT	Gauss–Newton with Approximated Tensor
GP	Gaussian Process
GP-LVM	Gaussian Process Latent Variable Model
IC	Initial Condition
K_f	fracture permeability
K_m	matrix permeability
KL divergence	Kullback-Leibler divergence
KL expansion	Karhunen-Loeve expansion
LightGBM	Light Gradient Boosting Machine
LSS	Land Subsidence Susceptibility
MC dropout	Monte Carlo dropout
MC-UNet	Martian Crater UNet
MCMC	Monte Carlo Markov Chain
mD	milliDarcies
ML	Machine Learning
MLP	MultiLayer Perceptron
MMP	Minimum Miscibility Pressure
MSE	Mean Squared Error
N_f	number of fractures
NLL	Negative Log Likelihood
NN	Neural Network
ODE	Ordinary Differential Equation
PDE	Partial Differential Equation
PINN	Physics-Informed Neural Network
POD	Proper Orthogonal Decomposition
R^2	coefficient of determination
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
RF	Random Forest
RL	Reinforcement Learning
RNN	Recurrent Neural Network

ROM	Reduced Order Model
SD	Standard Deviation
SEL	Shallow Ensemble Learning
SOD	Smooth Orthogonal Decomposition
SVM	Support Vector Machine
TE	Total Error
TgNN	Theory-guided Neural Network
TPwL	Trajectory Piecewise Linearization
UQ	Uncertainty Quantification

Chapter 1

Introduction

1.1 Overview

Modeling fluid flow in the Earth's subsurface is of interest to both engineers and scientists with respect to, for example, the seepage of waste water through soil, the flow of oil and gas to wells, and land subsidence as a consequence of groundwater extraction. Despite taking serious steps toward renewable energy, the oil/gas industry still provides a high proportion of the world's energy. In heterogeneous petroleum reservoirs, numerous fractures with varying lengths and widths may exist. These fractures can significantly impact the flow and transport processes of fluids within the reservoir, as they create high-permeability paths for fluid movement, which can lead to altered flow patterns, faster flow rates, changes in fluid compositions, and other alterations to the behavior of the reservoir. As a result, it is crucial to take into account the presence and properties of fractures when numerically modeling the flow and transport processes in the reservoir.

For numerical simulations of problems involving small-scale fractures, adequately fine grids are generally used to capture the details of these fractures. However, this can lead to the discrete formulation of these problems producing a large system of equations, which in turn increases the number of unknown parameters that must be solved for. The process of computing a solution, therefore, becomes expensive. To mitigate this, techniques such as reduced order modeling, upscaling, and multiscaling can considerably decrease computational cost while maintaining the necessary level of accuracy.

In the context of multiscale methods, a new framework named the mixed Generalized Multiscale Finite Element Method (GMsFEM) has been recently developed [1]. The model approximates reservoir pressure in multiscale space by applying several multiscale basis functions to a single coarse grid of the reservoir volume. The fluid velocity is directly estimated across a fine grid space. The focus of this thesis is mainly on multiscale basis functions.

The rest of this chapter is structured as follows. In Section 1.2, the motivation for the work contained in this thesis is stated. Section 1.3 presents the main research question along with subsidiary questions that the thesis aims to investigate. Section 1.4 elaborates on the research methodology employed to answer the questions mentioned in the previous section.

Section 1.5 itemise the main contributions that arise from the research conducted in this thesis. The publications related to the research are listed in Section 1.6. The structure of the forthcoming chapters is described in Section 1.7, followed by a summary in Section 1.8.

1.2 Motivation

In multiscale modeling of subsurface fluid flow in heterogeneous porous media, standard polynomial basis functions are replaced by multiscale basis functions. For instance, to produce such functions in the mixed GMsFEM, a number of Partial Differential Equations (PDEs) must be solved, leading to significant computational overhead. In recent years, data-driven methods have shown great potential in addressing complex problems involving multiple nonlinear relationships in various fields. Thus, there is an opportunity to replace PDE solvers with efficient data-driven methods. This involves developing Deep Learning (DL) models to provide reliable approximations for multiscale basis functions. These models can then be used to predict pressure distribution.

When conducting research, it is common for researchers to use existing datasets to apply new techniques or methods. Even when supplemented with newly generated or additional datasets, they may still face limitations such as restricted sample size or low dimensionality. For the research conducted in this thesis, a large number of new data samples were generated to train the developed DL models to ensure statistical confidence in the datasets evaluated and improve prediction accuracy.

While DL algorithms can be configured to effectively provide supervised learning of classification and regression problems, previous research has focused more on developing DL models to solve classification issues. The research conducted for this thesis addresses a regression-type problem and has offered the opportunity to provide innovative DL configurations that are tailored to be effective with regression datasets (i.e., those with continuous dependent variable distributions).

The motivation for this research is also driven by the need to enhance the reliability analysis of DL models. This is because DL models are highly complex, involving multiple control variables, and are prone to biases and overfitting, which can limit their reliability when deployed in real-world applications. Attention has been paid to the reliability of the DL models developed for this thesis concerning subsurface fluid flow modeling, leading to trained and tested models that have demonstrated reliable and consistent prediction performance and generalizability when deployed in practical settings.

Classical Neural Networks (NNs) focus on learning mappings between finite-dimensional spaces, which restricts the applicability of models trained in that way to a particular discretization. To overcome this limitation, a mesh-independent method that operates on infinite-dimensional spaces is developed for this thesis to predict pressure distributions in subsurface reservoirs without dimensional constraints.

1.3 Research Questions

The main research question that this thesis seeks to address is:

To what extent can standard DL models effectively reconstruct the multiscale basis functions of the mixed GMsFEM, and what steps can be taken to improve their performance and reliability? Moreover, is the mesh-independent approach of Fourier Neural Operator (FNO) able to accurately predict pressure distribution?

In order to comprehensively explore the above research question, it was broken down into five subsidiary questions, each of which was thoroughly examined and answered in a distinct chapter:

1. Is DL able to accurately reconstruct four distinct multiscale basis functions in the mixed GMsFEM in terms of statistical-graphical investigation, given its impressive performance with respect to datasets involving nonlinear relationships in recently published research in a range of scientific and engineering fields?

2. Will skip connections significantly affect the performance of Deep Neural Networks (DNNs) of low complexity or whether their inclusion has little or no effect?

3. Does combining multiple deep learners into an ensemble improve the accuracy of DL algorithms?

4. How does incorporating Uncertainty Quantification (UQ) methods improve the reliability of the CNN models in predicting new data points?

5. Can FNO models accurately perform on small-shape data problems to predict pressure distribution?

It should be noted that subsidiary research questions 1 to 4 pertain to multiscale basis functions, while subsidiary research question 5 is concerned with pressure distribution.

1.4 Research Methodology

This section describes the research methodology used in this thesis, which includes the generation and preparation of data, the development of DL models, and the validation of results.

To create a trustworthy DL-based model, it is important to take into account a broad range of input (here permeability field)/output (here multiscale basis functions and pressure) variables. A heterogeneous permeability field can be viewed as a realization of a random field that follows a particular distribution with the corresponding covariance. In this study, the Karhunen-Loeve (KL) expansion [2] was employed to parameterize the heterogeneous model. By assigning different values to the permeability of the formation matrix (K_{m}) and the

permeability of the fractures (K_f), a large number of datasets were separately generated for multiscale basis functions and pressure. After data generation, some tasks such as removing duplicates, scaling input/output, and changing the initial dimension of input/output were performed.

Based on the way an algorithm processes data, there are four types of ML/DL algorithms: supervised, unsupervised, semi-supervised, and reinforcement learning. Supervised learning was used with respect to the work presented in this thesis. More specifically, a Convolutional Neural Network (CNN) was used to predict multiscale basis functions. CNNs are specifically designed for problems with 2D arrays like our regression case. CNNs enable us to use 2D convolutional filters while developing a model. Furthermore, there is a reasonable and robust mathematical procedure behind convolutional filters. A CNN also automatically and adaptively learns the spatial hierarchies of features. Finally, the use of a CNN can decrease the number of parameters without reducing the quality of models. After developing a standard CNN model, the next stage was to attempt to improve its accuracy (through skip connections and Deep Ensemble Learning (DEL)) and its reliability (through Monte Carlo (MC) dropout). Furthermore, FNO was used to predict pressure directly using the permeability field as the input.

The results were validated statistically and graphically. Analysis using widely used prediction error accuracy parameters provides great insight into the performance of different models. The statistical error metrics considered were: coefficient of determination (R^2) and Mean Squared Error (MSE) for accuracy, and Standard Deviation (SD) for reliability. In terms of the graphical investigation, several examples are visualized to show the multiscale basis functions changes over a coarse grid, and pressure changes across the whole computational domain.

1.5 Contributions

The main contributions of this thesis can be summarized as follows:

1. **Generation of three separate large datasets.** Two datasets are for multiscale basis functions and one is for pressure. The only input for both cases is the permeability field. The generated datasets can be useful for researchers and practitioners working in the field of (regression) supervised learning to apply their methods, as publicly available datasets are mostly of the classification type.

2. **Investigation of DL models for multiscale basis functions reconstruction.** The thesis investigates the effectiveness of standard DL models for reconstructing Basis 2 to 5 in the mixed GMsFEM. It resolves the issue of the high computational cost associated with the mixed GMsFEM for obtaining multiscale basis functions.

3. **Study of skip connections in DL models for multiscale basis functions reconstruction.** The thesis investigates the effect of skip connections on the performance

of DNNs of low complexity. Three skip connection schemes were separately added to the standard structure.

4. **DEL for multiscale basis functions reconstruction.** The thesis introduces a DEL approach to improve the accuracy of DL models using linear regression and ridge regression, separately.

5. **Incorporation of UQ in DL models for multiscale basis functions reconstruction.** The thesis investigates the use of MC dropout to improve the reliability of DL models.

6. **Development of FNO models for pressure prediction in small-shape data problems.** These models act on infinite-dimensional spaces, unlike classical NNs that learn relationships between Euclidean spaces with finite dimensions.

1.6 Published Work

The findings of this thesis have resulted in five published journal papers and one book chapter. Here are the details:

- ◇ Choubineh, A., Chen, J., Coenen, F. and Ma, F., 2022. An Innovative Application of Deep Learning in Multiscale Modeling of Subsurface Fluid Flow: Reconstructing the Basis Functions of the Mixed GMsFEM. Journal of Petroleum Science and Engineering, 216, 110751.

- ◇ Choubineh, A., Chen, J., Coenen, F. and Ma, F., 2023. A Quantitative Insight into the Role of Skip Connections in Deep Neural Networks of Low Complexity: A Case Study Directed at Fluid Flow Modeling. Journal of Computing and Information Science in Engineering, 23(1), 014502.

- ◇ Choubineh, A., Chen, J., Coenen, F. and Ma, F., 2023. Applying Monte Carlo Dropout to Quantify the Uncertainty of Skip Connection-based Convolutional Neural Networks Optimized by Big Data. Electronics, 12(6), 1453.

- ◇ Choubineh, A., Chen, J., Wood, D.A., Coenen, F. and Ma, F., 2023. Fourier Neural Operator for Fluid Flow in Small-shape 2D Simulated Porous Media Dataset. Algorithms, 16(1), 24.

- ◇ Choubineh, A., Chen, J., Wood, D.A., Coenen, F. and Ma, F., 2023. Deep Ensemble Learning for High-dimensional Subsurface Fluid Flow Modeling. Engineering Applications of Artificial Intelligence, 126, 106968

- ⋆ Choubineh, A., Chen, J., Coenen, F., Ma, F. and Wood, D.A., 2022. Machine Learning to Improve Natural Gas Reservoir Simulations. In Sustainable Natural Gas Reservoir and Production Engineering (55-82). Gulf Professional Publishing, Elsevier.

1.7 Thesis Outline

The structure of the rest of this thesis is as follows:

Chapter 2: Background and Related Work. This chapter begins with an overview of fluid flow in petroleum reservoirs, focusing on heterogeneous porous media and the challenges associated with numerical modeling of fine grid systems. To address these challenges, three methods of reduced order modeling, upscaling, and multiscaling are introduced, which can meaningfully reduce computational cost while maintaining the necessary level of accuracy. Next, the chapter explains in detail the mixed GMsFEM as a multiscaling method. It includes the mathematical idea behind the mixed GMsFEM, the fluid flow conditions considered, and the general solution framework of this method. Then, a review of relevant published research regarding the application of data-driven methods in the reduced order modeling, upscaling, and multiscaling domains is provided. Overall, this chapter serves as an important background and foundation for the research presented later in this thesis.

Chapter 3: Generation and Preparation of Data. For each randomly generated porous medium in a Cartesian coordinate system over a unit square, pressure can be predicted in mixed GMsFEM, either directly or indirectly through multiscale basis functions. Therefore, this chapter provides detail of three separate datasets: two for multiscale basis functions and one for pressure. The only input for both cases is the permeability field. Basis 1 does not require training for DL modeling, while Basis 2 to 5 require training. The initial data generated for multiscale basis functions included 249,375 samples, which increased to 376,250 samples for the extended dataset. However, generating data for pressure can be challenging, resulting in 1700 samples, which is considerably lower than the number of samples for the basis functions. The chapter also outlines the necessary preparation involved in generating the data, such as removing duplicates, scaling input/output, and changing the initial dimension of input/output.

Chapter 4: SkiplessCNN for Basis 2, 3, 4, and 5. This chapter focuses on the design and training of distinct CNN models based on initial and extended datasets for Basis 2 to 5, separately. It begins by considering the differences between Artificial Intelligence (AI), Machine Learning (ML), and DL, as well as how a CNN architecture is designed. Then, the characteristics of the developed DNN of low complexity named SkiplessCNN, along with a parameter analysis are discussed. The SkiplessCNN architecture is then compared with two advanced CNN architectures, namely AlexNet and VGGNet. The chapter also provides information on the optimizers and loss functions used during the compilation and training process.

Chapter 5: Role of Skip Connections in Deep Neural Networks of Low Complexity. The focus of this chapter is on the problem of vanishing gradient in DNNs of low complexity, and it explores how skip connections can help mitigate this issue. In fact, skip connections enable the cost function gradient to be directly backpropagated to layers close to the input layer of a CNN, as mathematically explained. The chapter highlights several examples that illustrate the importance of skip connections and presents three skip connection schemes (FirstSkip, MidSkip, and DualSkip) that are added to the base CNN architecture.

Ultimately, this provides insight into the effectiveness of skip connections in improving the performance of DNNs of low complexity.

Chapter 6: Deep Ensemble Learning. The chapter starts by explaining the reducible and irreducible errors of ML/DL algorithms. Then, three categories of ensemble systems - boosting, bagging, and stacking - are presented, with examples to illustrate their application in the geoscience domain. The main objective of this chapter is to explore the potential prediction improvements that can be achieved by using DEL. This technique combines multiple deep base learners into a single ensemble model, eliminating the need for continuous adjustments to the architecture of individual networks or the nature of the propagation. The standard CNN (i.e., SkiplessCNN) and three skip connection-based CNNs (i.e., FirstSkipCNN, MidSkipCNN, and DualSkipCNN) are used as base learners. They are combined using linear regression and ridge regression, separately, as part of the stacking technique.

Chapter 7: Reliability of the DualSkipCNN Model. In this chapter, the significance of reliability analysis for DL models is explored, along with an investigation into how incorporating UQ methods can improve the reliability of CNN models in predicting new data points. It first provides an in-depth mathematical explanation demonstrating how MC dropout can be considered as a Bayesian approximation of the probabilistic deep Gaussian Process (GP), accompanied by several examples to show its application in subsurface fluid flow modeling. Then, MC dropout is applied to quantify the epistemic uncertainty of the DualSkipCNN model. The analysis is performed separately for each of the multiscale basis functions.

Chapter 8: Fourier Neural Operator for Pressure. Unlike classical NNs that learn relationships between Euclidean spaces with finite dimensions, the FNO algorithm aims to learn mappings between functions, making it mesh-independent. This means that FNO can be trained on one mesh and applied to another, providing greater flexibility and efficiency in modeling. In this chapter, FNO is applied to predict pressure distribution on a small-shape dataset using the permeability field as input. A CNN model is also developed based on the same dataset for comparison purposes.

Chapter 9: Comparative Evaluation. This chapter provides a comparative evaluation of standard CNN(SkiplessCNN) and skip connection-based CNNs (FirstSkipCNN, MidSkipCNN, and DualSkipCNN) for predicting multiscale basis function. The evaluation is conducted using statistical measures and graphical trends in each coarse block. Additionally, the effectiveness of DEL in improving the accuracy of the developed models for testing data is examined. The reliability of the DualSkipCNN model is also explored using MC dropout based on SD. Finally, the chapter presents the results of FNO and a novel CNN model for predicting pressure distribution.

Chapter 10: Conclusion. This chapter concludes the thesis by presenting a summary of the key findings in the context of the main research question and the relevant subsidiary questions. Furthermore, it provides an outlook on potential future work and opportunities for further exploration of the topic addressed in this thesis.

1.8 Summary

This introductory chapter has provided an overview of the work presented in this thesis together with the motivation behind it, the main research question and five subsidiary research questions that will be comprehensively examined and answered, and the adopted research methodology whereby answers to these questions could be arrived at. Additionally, the main research contributions of the work and publications arising from it were listed. The chapter concluded with an overview of the structure of the remainder of the thesis. In the next chapter, a literature review is presented to provide readers with an understanding of the background and previous research efforts that underpin the work presented in this thesis.

Chapter 2

Background and Related Work

2.1 Introduction

A wide range of phenomena/processes in science and engineering can be described using measurable/estimable quantities that rely on independent variables. As an example, in subsurface fluid flow, pressure and temperature are typically measured based on the time and location variables. Given the available fundamental laws, it is feasible to determine the relationships among the rates of change of these physical quantities. The mathematical correlations typically used to do this are Ordinary and/or Partial Differential Equations (ODEs/PDEs). In ODEs, the derivatives of the dependent variable(s) are taken with respect to only one independent variable. The hydrostatic equation is an example of an ODE that is written as follows:

$$\frac{dP}{dx} = -\frac{\rho g}{\beta}, \tag{2.1}$$

where P is the pressure, x is the depth below the surface, ρ is the density of the fluid, g is the gravity acceleration, and β is the fluid compressibility.

On the other hand, partial derivatives are required in PDEs when there are two or more independent variables involved. The heat equation is an example of a PDE:

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2}, \tag{2.2}$$

where T is the temperature at the position x and time t, and α is the thermal diffusivity.

To solve the mathematical models described, it is necessary to specify Boundary Conditions (BCs) and Initial Conditions (ICs). BCs exert a set of extra constraints to the problem on prescribed boundaries. There are typically three types of BCs: Dirichlet (the first kind), Neumann (the second kind), and Robin or Dankwerts (the mixed or third kind). In the first type, a value is assigned to the dependent parameter(s) (for example, pressure) while the derivative of the dependent variable(s) is known in Neumann's condition. Robin's BC is a weighted combination of the first two BCs. An IC refers to a value of a parameter at $t = 0$ in the dynamic simulation models.

Theories, methods, and tools available in scientific computing (also called computational science) make it possible to solve mathematical models of physical phenomena described in terms of ODEs and/or PDEs [3]. The collection of theories and methods used for this purpose is called numerical analysis/numerical mathematics, while tools refer to computer systems on which the codes are run. The more complex the mathematical models, the more advanced the computational hardware requirements are to solve them. There are various numerical methods available that can provide approximate solutions to such problems. These include the finite difference method [4], Finite Element Method (FEM) [5], finite volume method [6], spectral method [7], and meshless method [8]. It is worth mentioning that other techniques such as analytical and semi-analytical methods [9, 10, 11, 12, 13] can also be applied, which are usually not applicable in practice.

As mentioned earlier in this thesis, the new framework of the mixed GMsFEM is applied to a fluid flow problem in petroleum engineering. This method places emphasis on local mass conversation. Furthermore, it has been found to be effective in terms of computational cost, without significantly sacrificing accuracy in approximating pressure and/or velocity.

This chapter continues with Section 2.2, which provides information about fluid flow in petroleum reservoirs. To be more specific, it covers what a heterogeneous porous medium is, how the permeability in the matrix and fracture differs, what problem is with numerical modeling of fine grid systems, and eventually three methods of reduced order modeling, upscaling, and multiscaling are introduced to more efficiently solve subsurface flow problems. A detailed overview of the mixed GMsFEM numerical method is presented in Section 2.3. It first explains the idea behind the mixed GMsFEM. Then, the fluid flow conditions are defined. In the end, the general solution framework of the mixed GMsFEM is given. Section 2.4 gives relevant published research, with five examples related to the reduced order modeling domain (2.4.1), five examples to the upscaling domain (2.4.2), and ten examples to the multiscaling domain (2.4.3). The chapter concludes with a summary in Section 2.5.

2.2 Fluid Flow in Heterogeneous Porous Media

Modeling subsurface fluid flow in heterogeneous porous media has always been challenging, especially with real-world applications, such as the development of oil/gas fields and groundwater resources management. A heterogeneous porous medium indicates that it is not homogeneous; thus, formation-related properties can have multiple scales. For example, in petroleum reservoirs, there may be numerous fractures (connected or disconnected) with different lengths, whose width is much smaller than the domain size. Permeability is defined as the ability of a rock to permit fluids to pass through it. The permeability of fractures can be lower than that of the matrix in dissolved carbonate reservoirs. There are some examples in the pre-salt carbonate reservoirs in Brazil [14]. However, the permeability in fractures is generally much higher than that of the matrix. This is why the effect of fractures must be considered when modeling flow and transport processes.

Mesh generation plays a significant role in numerical simulation. It involves dividing the reservoir model into a grid of cells/elements that can be used to approximate the fluid

flow equations at discrete locations within or on the boundaries of the reservoir. The size of cells/elements must be chosen carefully to ensure that the solution obtained from the numerical simulation is accurate and efficient. Generally, adequately fine grids are used to resolve small-scale fractures. By doing so, the discrete formulation of such problems produces a large system of equations, and consequently, the number of unknown parameters increases. The computation of the solution, therefore, becomes expensive. Some techniques such as reduced order modeling [15, 16], upscaling (homogenization) [17, 18], and multiscaling [19, 20, 21, 22, 23, 24] are necessary to decrease the degrees of freedom. These methods can considerably decrease computational cost while maintaining the necessary level of accuracy.

Reduced order modeling involves projecting the high-dimensional system of equations onto a low-dimensional subspace using a set of basis functions (e.g., via Proper Orthogonal Decomposition (POD)) that capture the dominant modes of the solution. For instance, the researchers in [25] suggested a Trajectory Piecewise Linearization (TPwL) method for two-phase flow problems. This Reduced Order Model (ROM) involved the representation of new states based on linear expansions around states already simulated and saved during pre-processing training runs. The computations were carried out in a low-dimensional space obtained by POD of the pressure and saturation states encountered during the training simulations. Two examples consisting of 24,000 and 79,200 grid blocks were selected as case studies. The results confirmed the high accuracy of the TPwL model, especially in the range of the pressures used in the training process. Furthermore, the runtime was meaningfully decreased.

Upscaling is an averaging process in which the static (e.g., porosity) and/or dynamic (e.g., fluid saturation) properties of a fine grid model are scaled up to equivalent characteristics defined at a coarse grid level. This procedure should be performed in such a way that the results of the coarse grid are broadly consistent with those generated by the fine grid. For example, kernel bandwidth and wavelet transformation techniques were used to simultaneously scale up the porosity and permeability of a synthetic reservoir model by the researchers in [26]. Under the same circumstances, the simulation runs demonstrated that the upscaling error of the bandwidth method was much smaller than that of the wavelet method.

Multiscaling techniques, such as multiscale FEMs [19, 24], multiscale finite volume methods [20, 22] and mortar multiscale methods [21, 23] solve flow problems on coarse grids through pre-calculated multiscale basis functions. These are developed locally on fine grids to capture the local multiscale information of a medium. Researchers have always been interested in enhancing the accuracy of multiscale solutions. For instance, a framework of GMsFEM was proposed by three researchers in [27]. This model generalizes the multiscale FEM [28] by including further basis functions that can provide more local multiscale details to enrich the multiscale space.

2.3 Mixed GMsFEM

Local mass conservation is of great importance for subsurface flow problems. The mixed multiscale FEM is regarded as one of the most commonly used mass conservative multiscale techniques. In this technique, the multiscale process is used on two coarse elements with a common edge for the velocity, and piecewise constant basis functions are employed on a single coarse block for the pressure.

Following the basic computational procedure of GMsFEM [27], a framework for the mixed GMsFEM has recently been proposed to solve Darcy flow conditions (linear pressure gradient versus velocity) considering single-phase fluids in a porous medium characterized by heterogeneities in two dimensions (i.e., matrix composition and fracture distribution) [1]. The model approximates reservoir pressure in multiscale space. It does so by applying several multiscale basis functions to a single coarse grid of the reservoir volume. The fluid velocity is directly estimated across a fine grid space.

The flow of fluids through porous media can be effectively described in terms of (i) the Darcy (or momentum) law, (ii) mass conservation, (iii) energy drive, and(iv) case-specific rock-fluid correlations including compressibility and saturation equations, where more than one fluid is involved (e.g., gas, oil, and water) [29]. Energy conservation can in many cases be disregarded if isothermal conditions are considered. However, for reservoir systems in which temperature changes over time, such as for surface water injected into subsurface reservoirs, energy conservation cannot be disregarded. For a single, incompressible fluid phase with constant viscosity in a 2D linear and isothermal system, Darcy law, assuming steady-state flow and ignoring gravitational effects, can be expressed as [29]:

$$k^{-1}u + \nabla p = 0 \quad \text{in} \quad \Omega, \tag{2.3}$$

where k = permeability, u = Darcy velocity, ∇p = gradient pressure $(\frac{\partial p}{\partial x}, \quad \frac{\partial p}{\partial y})$, and Ω is the computational domain.

The formulation for the mass conservation law (also known as the continuity equation) is [29]:

$$\nabla . u = f \quad \text{in} \quad \Omega, \tag{2.4}$$

here, $\nabla . u$ = divergence velocity $(\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y})$, and f = source term (here it is zero).

Heterogeneous BCs are included:

$$u.n = g \quad \text{on} \quad \partial\Omega, \tag{2.5}$$

in which n is the outward unit norm vector on the boundary, and g is normal to the Darcy velocity prevailing at the reservoir boundary.

To illustrate the general solution framework of the mixed GMsFEM, τ^H is considered a confirming partition of Ω into finite elements with a coarse block size H, and τ^h is the fine grid partition with mesh size h. Assuming $V = H(\text{div}, \Omega)$ and $W = L^2(\Omega)$, the mixed finite element spaces become:

$$V_h = \left\{ v_h \in V : v_h\left(t\right) = \left(b_t x_1 + a_t, d_t x_2 + c_t\right), \ a_t, \ b_t, \ c_t, \ d_t \ \in \ \mathbb{R}, t \in \ \tau^h \right\},$$

$$W_h = \left\{ w_h \ \in W : \ w_h \text{ is a constant on each element in } \tau^h \right\}.$$

Considering that $\{\Psi_j\}$ represents a set of multiscale base functions related to the coarse element, the multiscale space relating to pressure (p) can then be expressed as the linear extent of the local basis functions. This relationship is expressed as:

$$W_H = \oplus \{\Psi_i\} \quad \text{in} \quad \tau^H.$$

In that form, the mixed GMsFEM is configured to find $(u_H, p_H) \in (V_h, W_H)$ constrained by:

$$\int k^{-1} u_H . v_H - \int \operatorname{div}(v_H)\, p_H = 0 \quad \forall\ v_H \ \in \ V_h^0, \tag{2.6}$$

$$\int \operatorname{div}(u_H) w_H = \int f w_H \quad \forall\ w_H \ \in \ W_H, \tag{2.7}$$

in which $u_H.n = g_H$ on $\partial\Omega$ is relating to the coarse edges at the boundaries, whereas g_H is the average of the function g at those coarse edges.

It is necessary to establish a multiscale space, W_H, to approximate p. This is achieved by solving local cell conditions for each coarse grid element by applying Dirichlet's BCs. If $T_i \in \tau^H$ represents the coarse grid elements relating to Ω, the purpose is to find $(u_j^{(i)}, p_j^{(i)}) \ \in (V_h,\ W_h)|_{T_i}$ by solving the following problem on T_i:

$$k^{-1} u_j^{(i)} + \nabla p_j^{(i)} = 0 \quad \text{in } T_i, \tag{2.8}$$

$$\operatorname{div}(u_j^{(i)}) = 0 \quad \text{in } T_i, \tag{2.9}$$

where $(V_h,\ W_h)|_{T_i}$ is the restriction of $(V_h,\ W_h)$ on T_i.

The coarse grid boundary element represents the junction of fine grid edges, i.e., $\partial T_i = \bigcup_{j=1}^{J_i} e_j$ in which J_i is the total number of fine grid edges at boundary T_i. $\delta_j^{(i)}$ represents a piecewise constant related to ∂T_i and the fine grid and equals "one" for e_j and "zero" for the remaining fine grid edges. Therefore, the BC on the boundary of T_i is taken as the Dirichlet's BC:

$$p_j^{(i)} = \delta_j^{(i)} \quad \text{on } \partial T_i. \tag{2.10}$$

By combining the local problem solutions, a snapshot of spatial conditions is derived. Assuming $\Psi_j^{i,\text{snap}} := p_j^{(i)}$ defines the snapshot fields, then the snapshot space can be expressed:

$$W_{\text{snap}} = \operatorname{span}\left\{ \Psi_j^{i,\text{snap}} : 1 \le j \le \ J_i \ , \ 1 \le i \le \ N_t \ \right\}. \tag{2.11}$$

In the case of using the single-index notation:

$$W_{\text{snap}} = \operatorname{span}\left\{ \Psi_i^{\text{snap}} : 1 \le i \le \ M_{\text{snap}} \right\}, \tag{2.12}$$

where $M_{\text{snap}} = \sum_{i=1}^{N_t} J_i$ represents the total number of snapshot fields.

The snapshot space can then be further reduced by solving local grid problems. The local problem solutions are referred to as the offline space. The snapshot space corresponding to T_i becomes:

$$W_{\text{snap}}^{(i)} = \text{span}\left\{\Psi_j^{i,\text{snap}} : 1 \leq j \leq \ J_i\right\}.$$

In a local grid problem, the real number $\lambda \geq 0$ and the function $p \in W_{\text{snap}}^{(i)}$ need to be derived

$$a_i(p, w) = s_i(p, w) \quad \forall w \in W_{\text{snap}}^{(i)}. \tag{2.13}$$

For each T_i:

$$a_i(p, w) = \sum_e k[p][w] \quad \text{and} \quad s_i(p, w) = \int kpw \ \text{ in } T_i, \tag{2.14}$$

in which $[p]$ and $[w]$ are the jump of functions p and w, respectively. Also, e represents the fine edge interior of T_i.

The eigenvalues of Equation 2.13 are arranged in increasing order:

$$\lambda_1^{(i)} < \lambda_2^{(i)} < \cdots < \lambda_{J_i}^{(i)}, \tag{2.15}$$

where $\lambda_k^{(i)}$ denotes the kth eigenvalue for T_i. The corresponding eigenvectors are $Z_k^{(i)} = (Z_{kj}^{(i)})_{j=1}^{J_i}$ with $Z_{kj}^{(i)}$ being the jth component of the vector $Z_k^{(i)}$. Initial l_i eigenfunctions are selected to represent the offline space. Offline basis functions are then defined as:

$$\Psi_k^{i,\text{off}} = \sum_{j=1}^{J_i} Z_{kj}^{(i)} \Psi_j^{i,\text{snap}} \quad k = 1,\ 2,\ \ldots,\ l_i.$$

Then, global offline space becomes:

$$W_{\text{off}} = \text{span}\left\{\Psi_k^{i,\text{off}} : 1 \leq k \leq \ l_i\ ,\ 1 \leq i \leq \ N_t\ \right\}.$$

Applying single-index notation, the global offline space can be defined as:

$$W_{\text{off}} = \text{span}\left\{\Psi_k^{\text{off}} : 1 \leq k \leq \ M_{off}\right\},$$

where $M_{\text{off}} = \sum_{i=1}^{N_t} l_i$ is the total number of offline basis functions.

Each Ψ_k^{off} can be expressed by a vector ψ_k^{off} which contains coefficients from Ψ_k^{off} relating to the fine grid basis functions. Thus:

$$R_{\text{off}} = \left[\psi_1^{\text{off}}, \ldots, \psi_{M_{\text{off}}}^{\text{off}}\right].$$

The offline space is mapped using these functions to the fine grid space. The mixed GMsFEM system (Equations 2.6 and 2.7) is expressed in matrix terms as:

$$M_{\text{fine}} U_H + B_{\text{fine}}^T R_{\text{off}} P_H = 0, \tag{2.16}$$

$$R_{\text{off}}^T B_{\text{fine}} U_H = R_{\text{off}}^T F_H, \tag{2.17}$$

M_{fine} constitutes a symmetric, positive definite, and sparse matrix. U_H and P_H are the unknown fluid velocity and pressure vectors that describe grid spaces V_h and W_H, respectively. Execution of the mixed GMsFEM, therefore, requires two fine grid matrices to be constructed (M_{fine}, B_{fine}) accompanied by one offline matrix (R_{off}).

Fluid velocity can be solved directly from the fine grid matrix combination. Considering k as a diagonal tensor, M_{fine} is readily estimated with the diagonal matrix $\widehat{M_{\text{fine}}}$, applying the trapezoidal quadrature rule. The convergence rate of that easier-to-execute system is essentially the same as the unmodified matrix system. M_{fine} can therefore be replaced by that diagonal matrix $\widehat{M_{\text{fine}}}$ without compromising prediction accuracy. As $\widehat{M_{\text{fine}}}$ is easier to invert, the system described by Equations 2.16 and 2.17 is solved as follows:

$$-R_{\text{off}}^T B_{\text{fine}} \widehat{M_{\text{fine}}}^{-1} B_{\text{fine}}^T R_{\text{off}} P_H = R_{\text{off}}^T F_H.$$

Taking this approach, an original mixed formulation is expressed approximately by a positive-definite, sparse linear system. In that linear system, fewer pressure unknowns are involved for each coarse-grid element.

Generally, the number of PDEs requiring solutions to enable multiscale basis functions to be derived is dependent on the number of local cells and local eigenvalue problems involved. The local cell problem relating to the coarse grid relates to the original system definition but excludes the source function in Equation 2.4. A BC (delta) relates to the coarse grid boundary; delta = 1 for fine grid edges and delta = 0 for coarse grid edges. Local cell problems are therefore determined by the fine grid edges impacting the coarse grid boundary. In the model configured for this study, the number of fine grid edges/coarse grid boundary is 12.

2.4 Data-driven Methods for Model Reduction-based Numerical Simulation

This section discusses the application of data-driven methods, including ML, DL, and statistical analysis in the context of model reduction-based numerical simulation. It consists of three discrete subsections: 2.4.1, 2.4.2, and 2.4.3.

2.4.1 Reduced Order Modeling Domain

This subsection presents five cases that demonstrate the application of data-driven methods in the reduced order modeling domain. The purpose of the work presented in [30] was to combine deep residual recurrent neural network [31] with POD and Discrete Empirical Interpolation Method (DEIM) for UQ in subsurface porous media. The uncertainty parameter was the permeability field, which followed a log-normal distribution. The simulations were based on 2000 random permeability realizations. The results confirmed the accuracy of the

developed framework with a much lower computational cost compared to the POD–Galerkin reduced model joined with DEIM.

The authors in [32] implemented a POD-based Gauss–Newton with Approximated Tensors (GNAT) framework for two-phase reservoir simulation. The dimension reduction of state variables was performed using POD. The gappy POD technique was employed to approximate the nonlinear residuals using their values at a fraction of cells. Furthermore, the least-squares Petrov–Galerkin projection was applied for constraint reduction, rather than the Galerkin projection used in the POD-DEIM procedure. The study considered 576 various GNAT parameter combinations to assess the effect of parameter values on error. The results showed that this method achieved high accuracy.

A model involving Smooth Orthogonal Decomposition (SOD) and DEIM was developed in [33] to simulate fluid flow in two-phase reservoirs for both 2D and 3D cases. The main reason for using SOD rather than POD was to build higher efficient basis vectors for state parameters. The SOD-DEIM approach showed better performance than POD-DEIM in reproducing the full order models.

Two non-intrusive ROMs were proposed in [34] to investigate fluid flow in gas reservoirs. The first model, called POD-RBF, involved POD and Radial Basis Function (RBF), while in the second framework, called POD-AE, an AutoEncoder (AE) was employed instead of RBFs. Both frameworks were trained using data obtained from simulations of a real heterogeneous gas reservoir with time-varying production. The results showed that both models were computationally much faster (between 0.22 and 300 times) than classical simulation. Considering the simulation outputs as the reference, the POD-RBF performed better than the other model.

The researchers in [35] aimed to create data driven ROMs for production forecasting as an alternative to standard simulations. To do so, a black-oil reservoir simulation model based on the production history of the Volve oil field was used to produce training data. A long short-term memory topology was chosen for its ability to learn long-term dependencies. Monthly oil production was predicted at the individual wells and full-field levels and then validated against real-field data for production history to compare its predictive accuracy against the simulation results. The results indicated that a univariate model with a single time lag as input and a stateless configuration was the most accurate for predicting monthly oil production. Using a walk-forward validation strategy, the single-well ROM reduced prediction error by an average of 95

2.4.2 Upscaling Domain

Five examples are given to illustrate the application of data-driven techniques in the upscaling domain. The researchers in [36] investigated the performance of ordinary least squares and Kernel Ridge regression algorithms for the computation of upscaled permeability. For this, 100 samples were generated, with a permeability field produced using the Gaussian probability distribution and an upscaled permeability which was calculated using a finite volume method. The results demonstrated that Kernel Ridge regression performed with a

lower error to capture the upscaled behavior of the flow (i.e., the underlying physics of the problem) compared to the other technique.

The authors in [37] automated permeability upscaling from the detailed geological characterization of fractured reservoirs expressed by a Discrete-Fracture Model (DFM). A DL-based design was developed to find a suitable relationship between DFM images (input) and the equivalent continuum model (output), which contained the estimated equivalent permeability of every grid block. The suggested upscaling workflow had three stages: (i) data generation, (ii) model development, and (iii) validation. The 10,000 samples were generated to train a model with 18 hidden layers. The performance of the model was tested in four cases, and it was found to perform better than static and flow-based upscaling methodologies.

The Darcy Brinkman–Stokes model was combined with Decision Tree (DT) multivariate regression, to upscale a microporous carbonate from the pore scale to the Darcy scale in [38]. Using a limited number of training images, the developed model performed as well as numerical upscaling to predict permeability. The main advantage was that the regression model was around 80 times less computationally prohibitive than the numerical methods.

The purpose of the work presented in [39] was to obtain fine scale flow behavior in terms of the reservoir saturation map for an upscaled model using the generative adversarial network. This was done on a system of the two-phase, deal-oil, and heterogeneous oil reservoir. The authors also showed how the network can be trained using dynamic coarse geological properties at different resolutions. The results implied that even when coarse geological features and with limited resolution, the super-resolution reconstructions can recreate missing information that is close to the ground facts.

Given that existing equivalent permeability prediction models were only applicable to reservoirs with a simple Discrete Fracture Network (DFN), the authors in [40] attempted to propose CNN models to forecast the equivalent permeability of a complex multiscale fracture network. The images and features of the DFN were considered as the input. The developed models of ResNet-18, VGG-16, GoogLeNet, and MsNet-8-4 were validated with the simulation results of the Lattice Boltzmann method. Then, these models were compared with an existing model called CNN-4. The results confirmed the superiority of these models, especially MsNet-8-4 to CNN-4.

2.4.3 Multiscaling Domain

These ten cases are related to assisting the data analysis in multiscaling methods on coarse grids. The researchers in [41] concentrated on the multiscale finite volume method presented in [42]. Generally, a series of local problems over dual-grid cells have to be solved to extract coarse scale basis functions. The researchers used shallow NNs adjusted by a series of solution samples for the computation of basis functions. The results of this data-driven method over a computational domain $[0, 1]^2$ were promising for elliptic problems.

When the permeability field is fixed, the main quantities of GMsFEM can be precomputed in an offline stage by solving local problems. Given several choices of permeability fields, however, repetitively formulating and solving such local problems might be compu-

tationally expensive. A DL-based approach was presented in [43] for the rapid prediction of the GMsFEM ingredients for any online permeability fields. Deep structures were used to illustrate non-linear mapping from the fine scale permeability field coefficients to the key factors of the multiscale basis functions and the coarse scale variables. The results showed that if samples of the GMsFEM discretizations were adequate, the developed models would be able to provide exact approximations.

The researchers in [44] investigated a new deep network to a model reduction technique of non-local multi-continuum upscaling developed in [45] for multiscale simulations. The input-output maps were developed on a coarse block and trained by applying multilayer neural network approaches. Soft thresholding operators were selected as an activation function. By relating a soft-thresholding neural network and minimization of PDE solutions, multi-scale features of coarse element solutions were extracted, establishing a ROM for solution approximation. The numerical results confirmed the good performance of this method.

Various deep networks were used to approximate flow and transport formulae in [46]. Implementing the sparsity structures of the underlying discrete systems, these networks had far fewer learnable parameters compared with Fully Connected (FC) models. DL was used to approximate the map from the source terms to the velocity solution. The networks were developed with convolutional and locally (not fully) connected layers to execute model reductions. Furthermore, a custom loss function was defined to apply the local mass conservation constraints. The achieved velocity fields were then fed into the saturation equation, and a residual network served to approximate the dynamics. Numerical results of both the single-phase and two-phase cases highlighted the huge potential of novel models to precisely forecast the underlying physical system and make computational efficiency improvements.

A multi-agent Reinforcement Learning (RL) methodology was presented to hasten the multi-level Monte Carlo Markov Chain (MCMC) sampling algorithms in [47]. The authors confirmed their approach by solving an inverse, multiscale problem, which classical MCMC techniques struggle with. The first issue was computing the posterior distribution, which required a lot of time for heterogeneous media. To solve this, a GMsFEM was used as the forward solver. Moreover, finding a function able to generate meaningful sampling was not simple. For this, an RL policy was learned as a proposal generator. Experimentation revealed that this approach was capable of considerably improving the sampling process.

A DL-based method within GMsFEM was presented in [48] to cluster (coarsen) the uncertainty space. By doing so, the number of multiscale basis functions per coarse element could decrease over the uncertainty space. CNNs were joined with some methods in adversary NNs. Simulation runs were carried out based on almost 240,000 local spatial fields. The numerical results confirmed an increase in the number of clusters from 5 to 11 could decrease relative error.

A combination of DL techniques and local multiscale model reduction methods was used in [49] to predict flow dynamics considering observed data and physics-based modeling concepts. Flow dynamics can be viewed as a multilayer network. This means that the solution (e.g. saturation) at the time instant "n+1" relies on the solution of "n" as well as input parameters. Each layer is considered a nonlinear forward map, and the number of layers re-

lates to the internal time steps. Multilayer NNs find a nonlinear mapping between the time steps. ROMs provide important coarse grid parameters and some other information. Three examples of numerical simulations demonstrated the efficacy of this hybrid methodology.

A Theory-guided Neural Network (TgNN) which incorporated scientific knowledge into traditional NNs was developed for the simulation of subsurface flow in [50]. The KL expansion was used to parameterize heterogeneous porous media. The potency of the suggested framework was evaluated by multiple cases, such as predicting future responses, training from noisy data, and transfer learning. In comparison with ordinary models, TgNN produced more precise outputs. For example, the R^2 values were 0.484 for the artificial neural network and 0.996 for the TgNN to predict future responses with changed BCs.

The researchers in [51] proposed a multi-stage DL-based method for multiscale problems. Each stage shared a similar structure and estimated the same ROM of the problem with multiscale features. The prediction of the first stage was quite imprecise. It was demonstrated numerically that performance improvements could be achieved using several ROMs as inputs at each stage. The proposed strategy yielded good outputs on two time dependent linear and non-linear PDEs with the steady state condition based on 1600 samples.

An ensemble variable-separated multiscale method was presented in [52] for elasticity problems in random media. This method was combined with the GMsFEM to achieve a reduced model. In this regard, several local problems were solved for snapshots to obtain the multiscale basis functions of GMsFEM. An ensemble method was applied to construct stochastic basis functions shared by a series of elasticity equations, and a residual decomposition was adopted to calculate the deterministic physical components for all ensemble members. The ensemble variable-separation was used to solve the local problems of the multiscale basis functions and attained effective online computation for the basis. Then, under the Galerkin projection of the generalized FEM, the solutions of stochastic elasticity problems were projected onto the low-dimensional space spanned by the online multiscale basis functions. Multiple case studies confirmed the effectiveness of the suggested methodology.

2.5 Summary

This chapter provided a review of the background and previous work related to the research presented in this thesis. The mixed GMsFEM is an effective numerical method capable of approximating reservoir pressure in heterogeneous porous media. This is achieved by applying several multiscale basis functions to a single coarse grid of the reservoir volume. The related work was divided into three categories: (i) reduced order modeling domain (five examples), (ii) upscaling domain (five examples), and (iii) multiscaling domain (ten examples). A description of the generated data using the MatLab software and their preparation for developing models are given in the next chapter.

Chapter 3

Generation and Preparation of Data

3.1 Introduction

To create a trustworthy ML/DL-based model, it is important to take into account a wide range of the input (here permeability field)/output (here multiscale basis functions and pressure) variables. A heterogeneous permeability field can be viewed as a realization of a random field that follows a particular distribution with the corresponding covariance. In this study, the KL expansion [2] was employed to parameterize the heterogeneous model. This Gaussian random field generation method decomposes a random process into the eigenvalue and eigenfunction of its covariance kernel. The general form of the KL expansion is given by:

$$Z(x,y) = m(x,y) + \sum_{i=1}^{\infty} \sqrt{\lambda_i}\phi_i(x,y)\xi_i, \tag{3.1}$$

where $Z(x,y)$ is the random field, $m(x,y)$ is the mean value of the random field, λ_i is the corresponding eigenvalue, $\phi_i(x,y)$ is the eigenfunction of the covariance function of $Z(x,y)$, and ξ_i is the independent standard normal variable.

In practice, the expansion is truncated after a finite number of terms, resulting in an approximation of the original random field. The truncated expansion takes the form:

$$Z(x,y) = m(x,y) + \sum_{i=1}^{n} \sqrt{\lambda_i}\phi_i(x,y)\xi_i, \tag{3.2}$$

where n is the number of terms used in the expansion.

To apply the KL expansion to permeability fields in petroleum engineering, it is first needed to estimate the covariance function of the field. Once the covariance function is known, the eigenvalues and eigenfunctions can be computed by solving some integral equations, and the expansion can be used to generate random permeability fields.

For every randomly produced porous medium in a Cartesian coordinate system over a unit square, there are five basis functions (numbered Basis 1 to 5). Basis 1 is a piecewise

constant, with binary values of -1 and 1. Basis 1 is defined as part of FEM, it, therefore, requires no training for ML/DL modeling. On the other hand, Basis 2 to 5 take values distributed across the range $(-1, 1)$, and therefore require training for ML/DL modeling.

With regards to the other output i.e., pressure, Dirichlet's condition was applied on two sides: $p = 100$ (left-side boundary) and $p = 0$ (right-side boundary) for each porous medium. Neumann's condition was applied to the other two sides: $\partial p = 0$ (top and bottom sides).

Following on from this introduction, Section 3.2 provides an explanation of the computational domain, the fine grid system, and the coarse grid system for the multiscale basis functions. Subsection 3.2.1 describes the initial data generated (249,375 samples) using the MatLab software. The details of the extended data (376,250 samples) are given in the next Subsection 3.2.2. With regards to the pressure distribution, there is only one dataset, 1400 samples for the training part and 300 samples for testing the model (3.3). This chapter is finished with a summary in Section 3.4.

3.2 Multiscale Basis Functions

For the system analyzed here, the computational domain was defined as $\Omega = [0, 1]^2$, representing a square 2D domain. The fine grid system adopted involves a uniform 30×30 mesh. On the other hand, a sparser, uniform 10×10 mesh was applied to represent the coarse grid network. This means that each coarse grid contains nine fine grids. A coarse grid may contain fractures or not. If so, it can include partial or complete fractures. Also, a fracture can pass another.

The ranges of permeability values applied to the formation matrix (K_{m}) were 1, 2, 3, 4, and 5 milliDarcies (mD), and to the fractures (K_{f}) were 500, 750, 1000, 1250, 1500, 1750, and 2000 mD. The number of fractures (N_{f}) available in a porous medium was varied between 1 and 25. The length of fractures was randomly distributed.

The format defined for the permeability field was as a vector (900×1), subsequently adjusted to be expressed as a 2D tensor (100×9), in which, coarse grid units $= 100$ and each coarse grid contains 9 fine grids. Each row in the array, therefore, represents a coarse grid. Such a configuration enables the use of 2D CNN kernels. However, basis functions remained in the format of a 900×1 vector.

Figure 3.1 shows a permeability field of the fractured porous media with K_{m} of 4 mD, K_{f} of 2000 mD, and the number of fractures is 15. Although multiscale basis functions are defined on the coarse grids, their coordinates are also important. For example, coarse grids no. 10 and no. 91 have the same permeability, but their corresponding basis functions may have been different.

Multiscale basis functions are defined in a single coarse grid element. Here, the pattern available in a coarse block is tracked for the graphical investigation, with an unfractured case (Figure 3.2) and a fractured case (Figure 3.3). In the top section of these two figures, fine grids in blue refer to the matrix and yellow to the fracture.

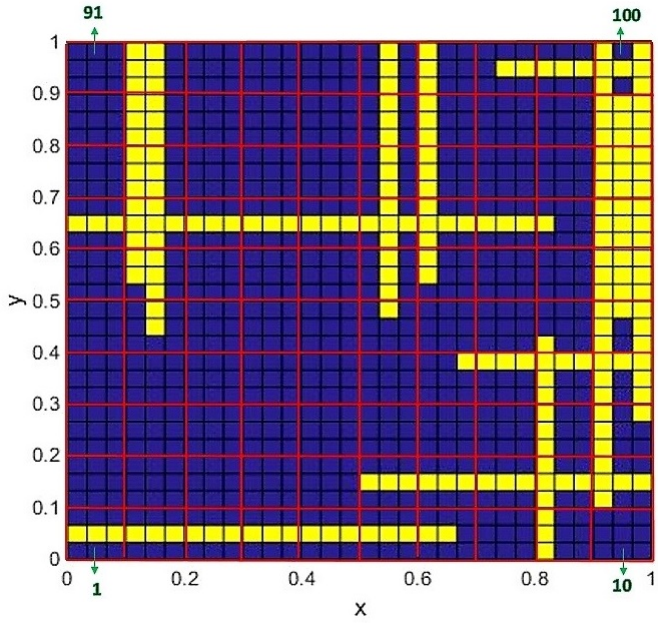


Figure 3.1: Schematic description of the permeability field of a simulated fractured porous reservoir formation. K_{m} is assumed to be 4 mD. K_{f} is assumed to be 2000 mD. Fine grid squares represent the formation matrix (blue) in some cases and fractures (yellow) in other cases (selected randomly). The red lines define the coarse grid. Each coarse grid square contains nine fine grid squares. There are fifteen fractures assigned to this porous medium.

3.2.1 Initial Dataset

CNNs are developed using training, validation, and testing data all drawn from the same data distribution. The training subset is utilized for training the model. To develop robust CNN models, so that they are fit for purpose in the context of real-life settings, the data used to train the model must be large. The model is evaluated during the training process using the validation data. The testing data (unseen data) is only used to appraise the model's performance once the training process has been completed.

For each of the 875 (equals to $5 \times 7 \times 25$) cases, the MatLab code was run as many as 280 times for the training data, two times for validation, and three times for testing. So, 249,375 samples were generated with 245,000 examples for training, 1750 for validation, and 2625 for testing. The random generation of permeability fields involves the possibility that some duplicate fields could be generated. Consequently, the generated dataset was filtered to remove any duplicate data records. This is necessary to remove the risk of introducing bias towards specific model configurations in the DL analysis. Overall, 6653 training, 8 validation, and 13 testing samples were excluded. This reduced the training, validation, and testing samples to 238,347, 1742, and 2612, respectively.

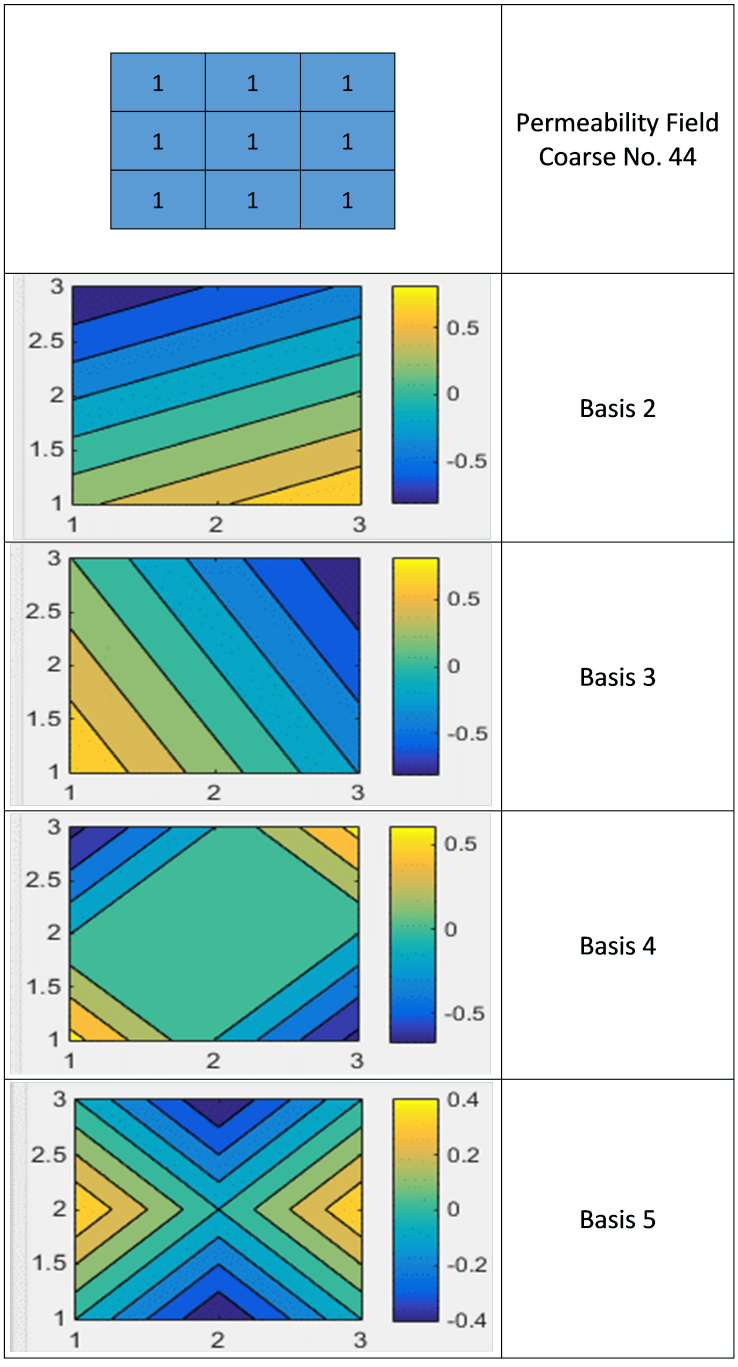


Figure 3.2: Pattern available in the coarse block no. 44 as an unfractured sample.

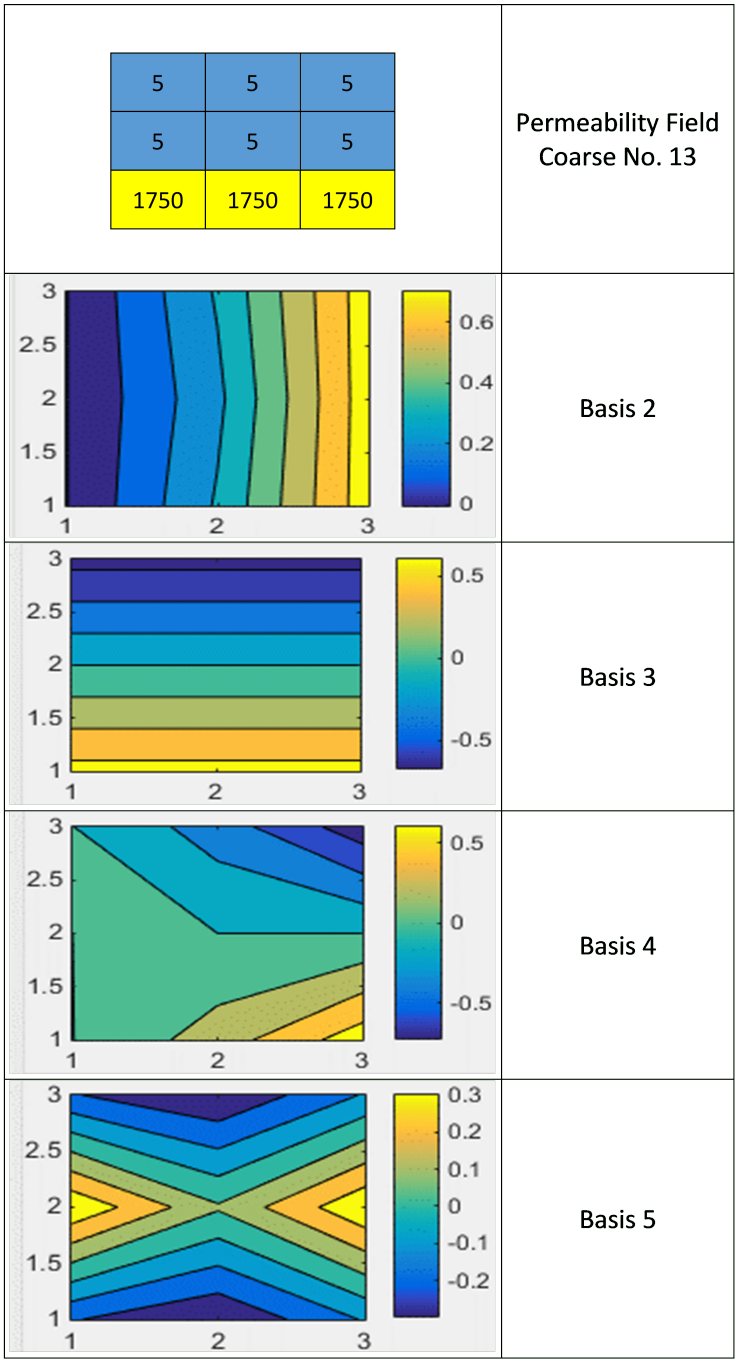


Figure 3.3: Pattern available in the coarse block no. 13 as a fractured sample.

3.2.2 Extended Dataset

With more data, DL models can learn a more diverse and robust set of features, which can help them generalize better to new data. In the initial dataset, the number of validation and

testing samples was very low. To investigate the effect of the number of data samples, each of the 875 cases was evaluated 350 times as part of model training. Additionally, 40 validation runs and 40 independent testing runs were executed for each case. In other words, 376,250 data records were generated in total. 306,250 records were used for DL model training, 35,000 records for DL model validation, and 35,000 for independently testing the trained and validated models. The data filtering step removed 1739 duplicate training data records, 579 duplicate validation data records, and 6121 duplicate testing data records. This pre-processing step reduced the training subset to 304,511 data records, the validation subset to 34,421 data records, and the independent testing subset to 28,879 data records.

3.3 Pressure Distribution

To produce data for pressure distribution, K_{m} and K_{f} were assigned fixed values of 1 and 1000 mD, respectively. N_{f} available in a porous medium was set to 5, and fractures are allowed to intersect with each other. The length of individual fractures was randomly distributed. A total of 1400 sample grids were generated in the MatLab software to constitute the training dataset, and a further 300 sample grids were generated to constitute the testing dataset. The testing data, therefore, made up 17.65% of the generated grids and the training data 82.35%. Duplicate fields were not allowed to exist in the training and/or testing datasets. Only two of the generated grids were removed during pre-processing to avoid intruding bias to specific permeability fields. In order to visualize the pressure changes occurring over the defined shapes, a representative sample is illustrated in Figure 3.4.

3.4 Summary

Pressure can be predicted in the mixed GMsFEM, either directly or indirectly through multiscale basis functions. The purpose of this chapter was to explain that data generation and preparation were separately performed for each of the four different multiscale basis functions and pressure using the MatLab software. The only input for both cases was the permeability field, and preparation involved tasks such as removing duplicates, scaling input/output, and changing the initial dimension of input/output. Generating data for pressure was challenging, resulting in a lower number of data samples for pressure compared to the basis functions. Chapters 4, 5, 6, and 7 focus on developing models based on the basis functions, while Chapter 8 is solely dedicated to directly predicting pressure distribution.

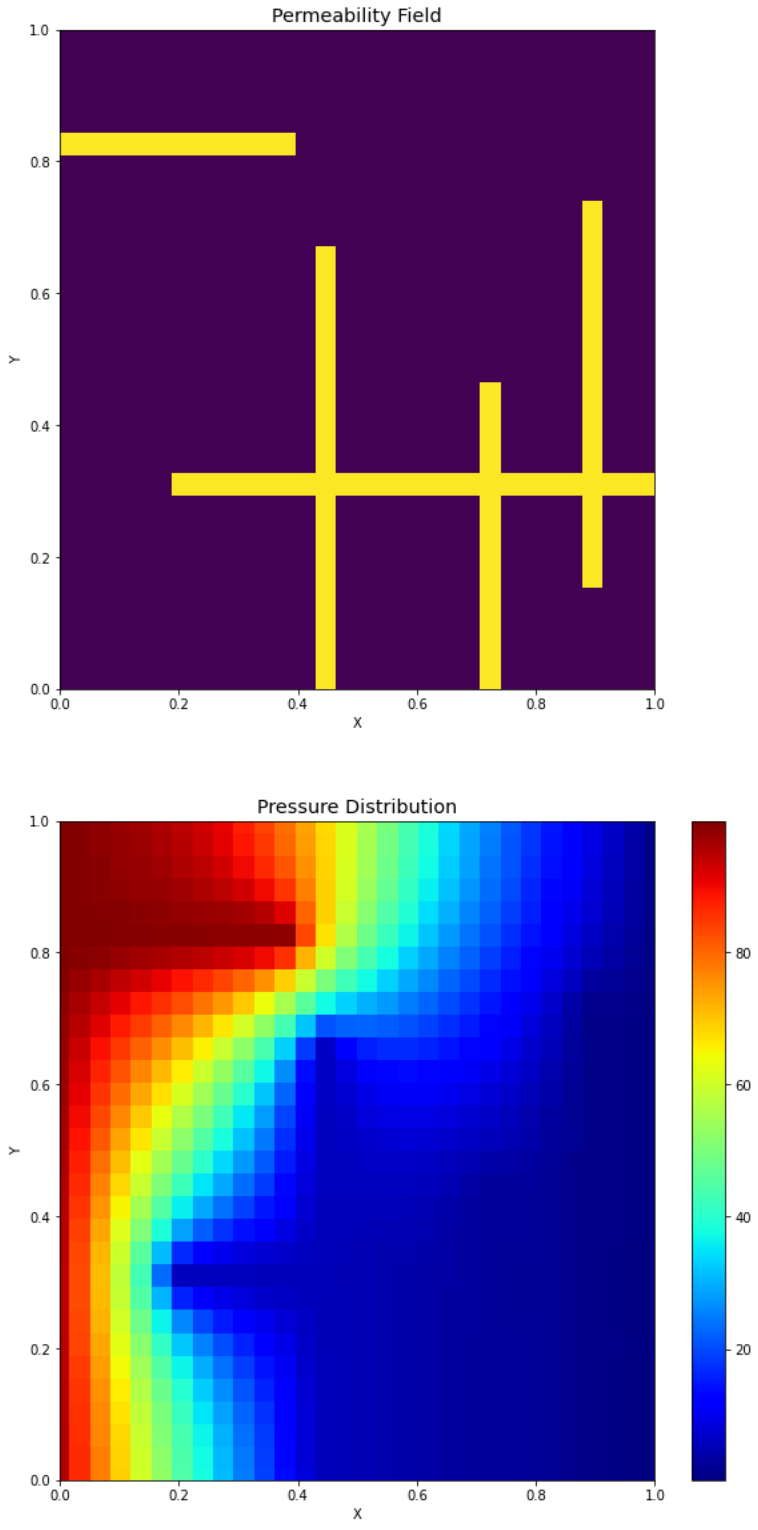


Figure 3.4: Pressure distribution for a corresponding random permeability over a unit square domain.

Chapter 4

SkiplessCNN for Basis 2, 3, 4, and 5

4.1 Introduction

In Chapter 2, the mixed GMsFEM framework was presented in the context of numerical approximation of subsurface fluid flow, emphasizing that numerous PDEs must be solved to construct multiscale basis functions. Chapter 3 was devoted to describing the data generated and prepared for a specific configuration. The work given in this chapter is directed at deriving an answer to the subsidiary question one from Chapter 1:

1. Is DL able to accurately reconstruct four distinct multiscale basis functions in the mixed GMsFEM in terms of statistical-graphical investigation, given its impressive performance with respect to datasets involving nonlinear relationships in recently published research in a range of scientific and engineering fields?

As the problem being studied is of a supervised type, there are several methods in DL to address it, such as CNNs and Recurrent Neural Networks (RNNs). RNNs are usually used while dealing with video, sound, or text data. On the other hand, CNNs are specifically designed for problems with 2D arrays like my regression case, mapping an input of 100×9 to an output of 900×1. CNNs enable us to use 2D convolutional filters while developing a model. Furthermore, there is a reasonable and robust mathematical procedure behind convolutional filters. A CNN also automatically and adaptively learns the spatial hierarchies of features. Finally, it can decrease the number of parameters without reducing the quality of models.

The remainder of this chapter is structured as follows. Section 4.2 explains the terms AI, ML, and DL, describes how a CNN architecture is designed, and what characteristics the best-performing network has. This network (i.e., SkiplessCNN) is then compared with two advanced CNN architectures in Section 4.3. Section 4.4 presents a detailed parameter analysis for the convolutional, Batch Normalization (BN), and FC layers. The network optimization is explained in Section 4.5 to find out how CNN models are developed. The chapter is concluded with a summary in Section 4.6.

4.2 SkiplessCNN Architecture

The terms AI, ML, and DL are wrongly used interchangeably, causing some confusion about the nuances between them. From a holistic perspective, DL is a subcategory of ML which, in turn, is a subdivision of AI (Figure 4.1). AI is a far-reaching branch of computer science in which a range of tools and techniques are used to make machines (namely computers and robots) more intelligent and consequently more effective and efficient.

ML methods are widely employed to find and predict relevant patterns within datasets. ML is now used to great advantage in various fields [53, 54, 55]. ML systems can be categorized in different ways [56, 57]. For instance, based on the way an algorithm processes data, there are four types of ML algorithms: supervised, unsupervised, semi-supervised, and reinforcement learning. The training process is performed using labeled data in supervised learning, while unlabeled data is used in unsupervised learning. Semi-supervised learning uses a limited amount of labeled data from which further labeled data is produced. The last category allows an agent to learn in a model through trial and error using feedback received from its own experience.

DL methods represent more complex extensions of classical ML methods, particularly NNs, and have demonstrated improved performance [58, 59, 60]. There are various DL algorithms, including CNN, deep AE, deep-belief network, RNN, and generative adversarial network. There are some differences between ML and DL. For instance, manual feature engineering tends to be performed with ML algorithms, sometimes requiring domain knowledge about a given problem. To make the point clear, consider "filling missing values". A dataset can include missing values due to the difficulty of collecting complete data. Missing values can be manually filled in based on expert knowledge, which is sometimes a tedious process. However, with DL algorithms this is more often not required, being performed automatically. In addition, because of the large amounts of data used and the complexity of computations, more powerful computer systems are needed in the DL process compared to the hardware requirements of traditional ML. For the same reasons, the training process often takes a significant amount of time in DL.

A typical CNN architecture may contain three types of layers: (i) convolutional, (ii) pooling, and (iii) FC. In mathematical sciences, convolution is a specialized linear operation on two functions that gives a third modified function. In the context of CNN, the fundamental idea is to consider an input (an array of numbers) as the first function and a convolutional filter (kernel) as the second. A kernel is a relatively small array of randomly generated numbers. The kernel moves over the whole input. The dot product of the kernel and input is calculated at each sub-region (with the same size as the kernel) of the input, obtaining an output value in the corresponding location of the convolved input. This process produces a feature map and is performed using different kernels. The outputs of the convolution process are passed through an activation (transfer) function. Such functions typically transform a linear operation into a nonlinear system [56, 57].

The key difference between a parameter and a hyperparameter is that a model's parameters are automatically updated during the training process, whereas hyperparameters are

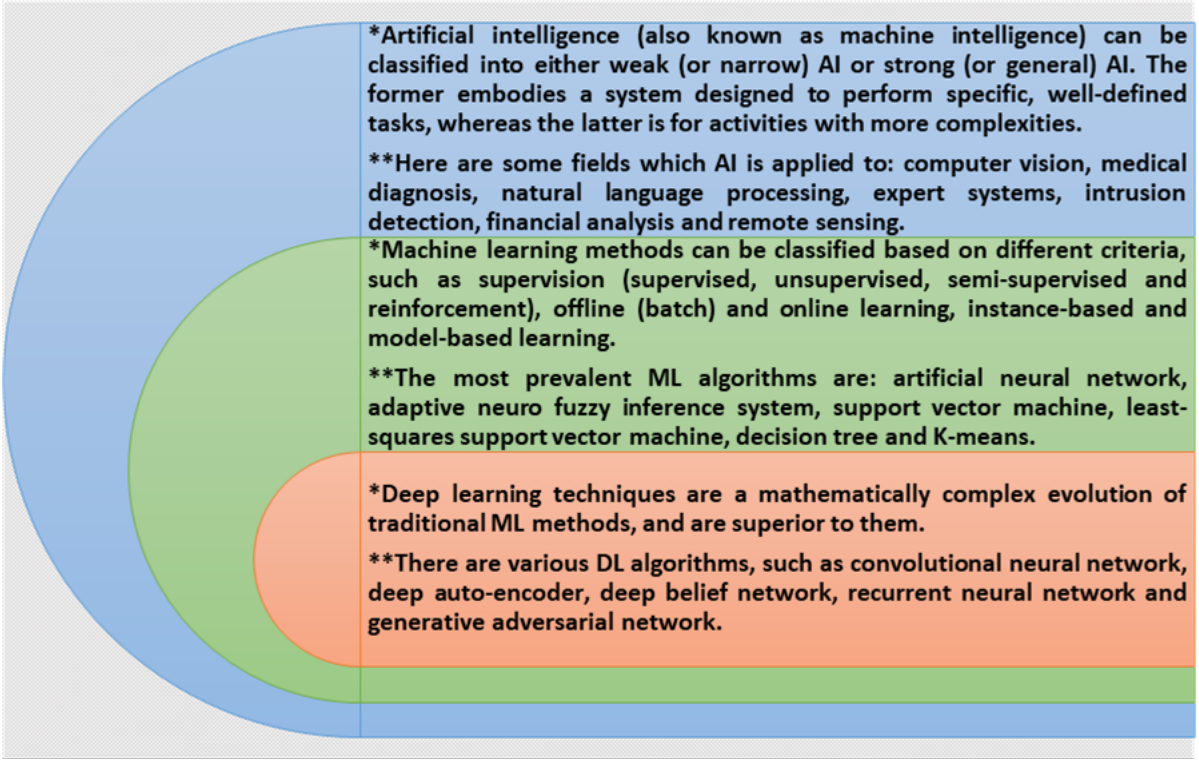


Figure 4.1: Disambiguating the terms: Deep Learning (DL) $\subset$ Machine Learning (ML) $\subset$ Artificial Intelligence (AI).

set manually before the model begins training like the size and number of kernels. Including more convolutional layers in a CNN model increases the number of parameters. The more parameters there are in a model, the more computationally expensive the learning process is. This is where a subsampling operation can be useful. In DL, pooling layers use statistical functions (maximum pooling and average pooling) to decrease the number of trainable parameters. This can decrease the computational complexity of mathematical operations and sometimes improve the robustness of feature maps. Pooling layers come after convolutional layers [56, 57].

When the output of the network is in the format of a vector, feature maps in the final convolutional or pooling layer are first flattened to a 1D array, and then connected to FC layers. In FC layers (dense layers), each neuron of a layer is connected to whole neurons in the previous layer and the next layer. It is common to put a dropout layer after each FC layer (except the output layer) at the end of a CNN model. Dropout omits a percentage of neurons in the previous FC layer. This percentage, as a hyperparameter, is defined when constructing a network. During the training process, some neurons may dominate, producing errors. Dropout balances a network, checking that all neurons work equally to minimize the cost function as much as possible [56, 57].

Distinct CNN model configurations, involving various combinations of convolutional,

pooling, FC, BN, regularization, and dropout filtering were tested separately for each basis function requiring training (i.e., Basis 2 to 5). In the regularization method, an extra element is added to the loss function. This regularization term penalizes a model for using higher values than needed in the weight matrix. A similar standard SkiplessCNN configuration was obtained as the best model for each of these four basis functions (Figure 4.2). That initial CNN architecture consists of five convolutional layers, and two FC layers but does not include any pooling layers. Convolutional layers 1 to 5 (CONV1 to CONV5) consist of 5, 10, 15, 20, and 25 kernels, respectively. To determine the size of a convolution output for an input with the size of I_h(height) $\times$ I_w(width) and a kernel with the size of $K_h \times K_w$, it is possible to use Equation 4.1 if the padding is set to "valid":

$$\begin{aligned} \text{Output height} = O_h &= (I_h - K_h)/S_h \ + 1 \\ \text{Output width} = O_w &= (I_w - K_w)/S_w \ + 1, \end{aligned} \tag{4.1}$$

where S_h and S_w are the vertical and horizontal strides.

When padding is set to "same", the size does not change. The kernel size for all convolutional layers is 3×3, and $S_h = S_w = 1$. The padding was set to "valid" only for CONV5. This means there was no padding for the first four convolutional layers. Therefore, CONV1, CONV2, CONV3, CONV4, and CONV5 have the size of 98×7, 96×5, 94×3, 92×1, 92×1, respectively.

Each convolutional layer is followed by a single BN layer of the same dimension. Typically, neural network models are able to apply higher learning rates and converge more quickly when the input to each layer is normalized; hence the value of adding the BN layers. In other words, normalization helps in stabilizing and accelerating the training process. Each FC layer contains 2000 neurons. For a given neuron or kernel, the inputs are multiplied by weights, and the resulting products are summed together. A bias term is then applied to that sum. Such rigid computations mean that only linear transformations are performed on the layer inputs using the weights and biases to generate the layer outputs. Although this operation makes the neural network simpler, it is less powerful and unable to learn complex patterns in a dataset. This is where the activation function is beneficial. Mathematically, this can be represented as shown in Equation 4.2 where w_{i} represents the weight value, z_{i} is the input value, b is the bias, f refers to the activation function applied, and y is the dependent variable prediction output. The developed models in this study used the "Rectified Linear Unit (ReLU)" activation function for the convolutional layers, the "sigmoid" activation function for the FC layers, and the "linear" activation function for the output.

$$y = f(\sum_{i=1}^{n}(w_{\mathrm{i}} z_{\mathrm{i}}) + b). \tag{4.2}$$

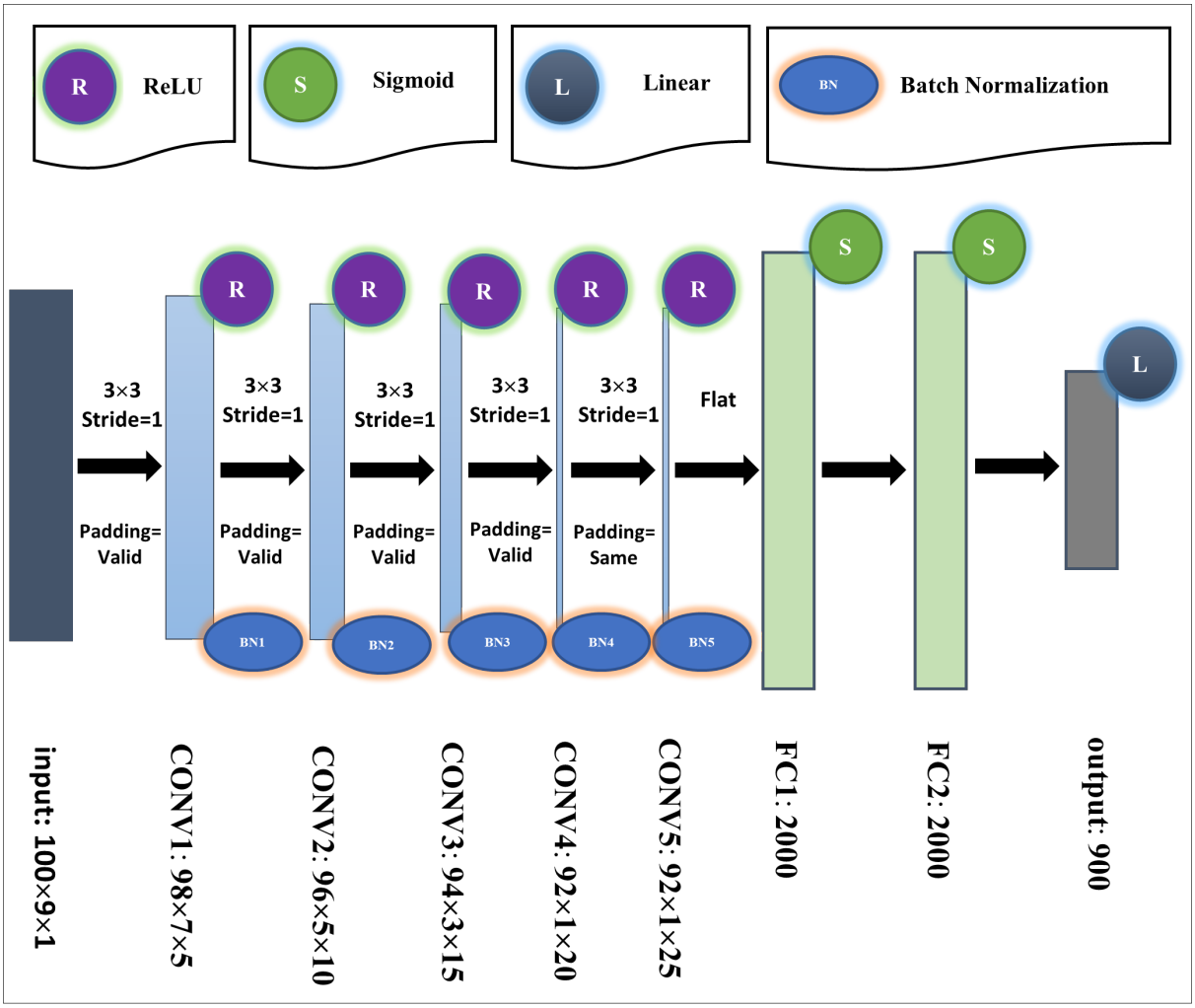


Figure 4.2: Standard CNN architecture for constructing the basis functions of the mixed GMsFEM.

4.3 Comparison of the SkiplessCNN Model with AlexNet and VGGNet

In order to better understand the SkiplessCNN architecture developed in this study, it is compared to structurally similar CNN architectures AlexNet [61] and VGGNet, also known as VGG16 [62]. AlexNet has five convolutional layers, three of which are followed by maximum pooling layers to decrease the computational cost. The number of kernels in each convolutional layer is 96, 256, 384, 384, and 256. There are two FC layers of 4096 neurons and a 1000-neuron output layer at the end of the network. VGGNet contains thirteen convolutional layers, five maximum pooling layers, two FC layers of 4096 neurons, and an output layer with 1000 neurons. The number of kernels used in the convolutional sections is 64,

128, 256, and 512. Each image has two basic elements: depth and size (height × width). The depth of an input image refers to the color channel, where one and three are used for grayscale and color images, respectively. For the later layers, the resulting feature maps indicate how deep a convolutional layer is. Similar to common CNN architectures, going deeper through the structure of developed models, the number of feature maps increases and their size decreases. However, the number of feature maps (equals the filters number) defined in this research is significantly less than that of common CNN models. In DL, pooling layers are primarily used to decrease the number of trainable parameters, mostly when the input shape is high, e.g., in AlexNet whose input shape is 224×224. However, the input dimension in this research is 100×9. This is why there is no pooling layer in SkiplessCNN. BN, similar to AlexNet, has helped to prevent over-fitting. As with AlexNet and VGGNet, the number of neurons (units) remained constant in FC layers, but no drop out layer was used in the proposed structure because it had a negative effect on the performance. The base structure of this work is for a regression-type problem, while AlexNet and VGGNet were essentially designed for a classification intent. Therefore, a linear activation function is used in our model for the output layer, but a softmax in AlexNet and VGGNet. Considering the fact that AlexNet and VGGNet are very complex, with many convolutional layers and filters, it is reasonable to call the presented architecture here a "DNN of low complexity."

4.4 Number of Parameters

There is no parameter for the input and flatten layers. The number of learnable parameters in a convolutional layer is determined through (Equation 4.3):

$$\text{Number of parameters in a convolutional layer} = ((K_h \times K_w \times M) + 1) \times N, \tag{4.3}$$

where M and N are the filters number (depth) of the previous layer and the current layer. The value of $K_h \times K_w$ is the size of the kernels used in the current layer. The added "1" alludes to the bias term for each kernel.

Since pooling layers only calculate a specific number by applying statistical functions, learnable parameters are not used.

Equation 4.4 is used to calculate the number of parameters in the first FC layer:

$$\text{Number of parameters in the first FC layer} = ((O_h \times O_w \times M) + 1) \times N_c, \tag{4.4}$$

where O_h(height) and O_w(width) and K (kernels number) are related to the size of the convolutional layer placed before the first FC layer. N_c is the number of neurons in the current FC layer. A bias is assigned to each neuron, indicated by the "1" included in the above formula.

If the last layer before the FC part is a pooling layer, size and kernels number are replaced.

When there are two FC layers successively, then Equation 4.5 can be used:

$$\text{Number of parameters in an FC layer (except the first FC)} = (N_p + 1) \times N_c, \tag{4.5}$$

where N_p and N_c are the number of neurons in the previous and the current layers, respectively. Again, "1" is added because of the bias assigned to each neuron.

The detailed information of each layer is given in Table 4.1. There are as many as 10,414,170 trainable parameters, which are mostly related to the FC layers. BN layers contain 300 (is equal to 4 × the filters number of the previous convolutional layer) parameters. Half of these parameters are non-trainable, updated with the mean and variance, but not trained with backpropagation. Furthermore, it is possible to keep some trainable parameters constant when training a model.

Table 4.1: Detailed parameter analysis for the SkiplessCNN architecture.

Type of layer	Output shape	Number of parameters	Type of parameters
input	$100 \times 9 \times 1$	0	-
CONV1	$98 \times 7 \times 5$	$((3 \times 3 \times 1) + 1) \times 5 = 50$	trainable
BN1	$98 \times 7 \times 5$	$4 \times 5 = 20$	trainable (10) non-trainable (10)
CONV2	$96 \times 5 \times 10$	$((3 \times 3 \times 5) + 1) \times 10 = 460$	trainable
BN2	$96 \times 5 \times 10$	$4 \times 10 = 40$	trainable (20) non-trainable (20)
CONV3	$94 \times 3 \times 15$	$((3 \times 3 \times 10) + 1) \times 15 = 1365$	trainable
BN3	$94 \times 3 \times 15$	$4 \times 15 = 60$	trainable (30) non-trainable(30)
CONV4	$92 \times 1 \times 20$	$((3 \times 3 \times 15) + 1) \times 20 = 2720$	trainable
BN4	$92 \times 1 \times 20$	$4 \times 20 = 80$	trainable (40) non-trainable (40)
CONV5	$92 \times 1 \times 25$	$((3 \times 3 \times 20) + 1) \times 25 = 4525$	trainable
BN5	$92 \times 1 \times 25$	$4 \times 25 = 100$	trainable (50) non-trainable (50)
flatten	2300	0	–
FC1	2000	$((92 \times 1 \times 25) + 1) \times 2000 = 4602000$	trainable
FC2	2000	$(2000 + 1) \times 2000 = 4002000$	trainable
output	900	$(2000 + 1) \times 900 = 1800900$	trainable
trainable parameters $= 10414170$ non-trainable parameters $= 150$ total parameters $= 10414170 + 150 = 10414320$			

4.5 Network Optimization

The CNN training process seeks to find optimum values for weights and biases applied to kernels (convolutional layers) and neurons (FC layers). Such values generate the lowest collective errors for all data records evaluated between actual and predicted dependent variable values. The back-propagation algorithms are commonly applied to train many types of neu-

ral network. In configuring back-propagation, three types of algorithms can be considered. These are:

1. **First-order iterative optimization algorithms**: Minimization (or maximization) of a loss function ($F(x)$) is performed using its gradient values based on a model's parameters. The first-order derivative indicates whether the loss function is increasing or decreasing at a certain point. This type was used in this research.

2. **Second-order iterative optimization algorithms**: These algorithms use the second-order derivative to minimize (or maximize) a loss function. The second-order derivative shows whether the first derivative is increasing or decreasing. Since this method is computationally expensive, it is rarely used.

3. **Derivative-free optimization algorithms**: At times, it is difficult to calculate the derivative of the loss function, or it might not exist. Such challenges can be overcome by applying derivative-free algorithms, of which there are many, such as genetic algorithm and particle swarm optimization. These are commonly referred to as evolutionary algorithms to distinguish them from gradient-descent algorithms.

The first-order iterative (gradient descent) is the most common optimization method used in DL (and also ML). When the loss function $F(x)$ is differentiable near a point a, then $F(x)$ decreases fastest if a is moved in the direction of the negative gradient of the loss function at a, $\nabla F(a_{\mathrm{n}})$. This can be expressed using Equation 4.6 where δ is the learning rate, which is changeable in each iteration:

$$\text{if } a_{\mathrm{n}+1} = a_{\mathrm{n}} - \delta\, \nabla F(a_{\mathrm{n}}) \text{ then } F(a_{\mathrm{n}}) \geq F(a_{\mathrm{n}+1}). \tag{4.6}$$

The gradient must decrease to move towards a local (or global) minimum point of the loss function. That is why $\delta\, \nabla F(a_{\mathrm{n}})$ is subtracted from a_{n}. In the gradient descent method, an initial guess x_0 is randomly considered for a local minimum of $F(x)$, and a sequence $x_1, x_2, \ldots$ is generated such that $x_{\mathrm{n}+1} = x_{\mathrm{n}} - \delta\, \nabla F(x_{\mathrm{n}})$ optimistically satisfies $F(x_0) \geq F(x_1) \geq F(x_2) \geq ...$ and gradually converges to the optimal value of the loss function.

The different optimizers available all strive to achieve a minimum loss or cost value. Multilayer NNs focus on the feed-forward sequence through its layered structure on to which weights and biases are initialized. However, in training the backward pathway is used to modify the layer weights and biases in each iteration. In that way back propagation acts to improve a model's performance.

The learning rate is a key DL hyper-parameter. It states how quickly a model learns in each epoch that parameters are updated. When it is too small, the training process takes a long time. If too large, it results in sub-optimal CNN learning, locking into sets of weights and biases too quickly, which can lead to a less stable training process that tends to converge prematurely. Hence, setting the right value for the learning rate is crucial. Adaptive methods such as Adam can be used to automatically resolve this issue. Adam applies distinctive learning rates to each scalar variable. It progressively adapts those rates throughout the

training iterations, with those adaptions being influenced by partial-derivative trends of rates applied to each variable in previous model iterations. Adam is gradient based in its calculations and benefits from a combination of its AdaGrad component to cope with sparse gradients and an RMSProp function in its application. It is suitable for DL applications to large datasets with many data records and/or multiple variables. In this study, default values were used i.e., the initial global learning rate = 0.001, $\text{beta}_1 = 0.9$, $\text{beta}_2 = 0.999$, and epsilon = 1e-7. The Adam learning rate can adjust at a finer scale as the optimum values are approached, although in some cases such fine tuning can result in overfitting. AMSGrad extends the performance of the Adam optimizer by converging in a more effective and smoother manner, avoiding step changes. By storing the highest values of second-momentum vectors generated in all previous model iterations, AMSGrad is able to normalize the moving average gradient in each iteration.

The CNN models in this study were constructed using Keras with TensorFlow as a backend on Python. The libraries of numpy, pandas, sklearn, and glob were also used in the Spyder module of Anaconda Distribution. The models were compiled using MSE as the loss (objective) function and Adam and AMSGrad separately as the optimizer and trained with a batch size of 32 samples.

The flow diagram in Figure 4.3 illustrates the CNN analysis methodology applied to the generated datasets. This step-by-step approach was separately done to develop a standard model for each individual basis function. In this regard, a low number of epochs was initially used to ensure that the model was still improving based on the obtained MSE. Hence, the number of epochs increased up to 100 as a stopping criterion that makes an optimizer terminate the process. If a predictive model fails to correctly capture the underlying trend of the training (seen) dataset, it is considered to "under-fit" the data. This is usually the result of designing an oversimplified model. To avoid under-fitting, including further layers and increasing the number of input parameters may be helpful. Once the model performs favorably on seen data, it can be applied to test (unseen) data to evaluate its generalizability. Over-fitting is a fundamental problem in ML, especially in DL due to a large number of trainable parameters. It happens when a model performs well with respect to the training data but poorly on the test dataset. This may occur when the model attempts to learn the noise patterns available in the unseen data.

4.6 Summary

The focus of this chapter was on designing and training distinct CNN models for Basis 2, 3, 4, and 5. A similar SkiplessCNN configuration was achieved as the best architecture for each of the four basis functions, consisting of five convolutional and two FC layers. The number of kernels in convolutional layers 1 to 5 was 5, 10, 15, 20, and 25, respectively. Each FC layer contained 2000 neurons. With regards to compiling and training, Adam and AMSGrad were separately used as the optimizer with MSE as the loss function. It is generally accepted that adjusting the architecture can obtain better ML/DL accuracy. This is why three skip connection schemes are added to the base structure in the next chapter.

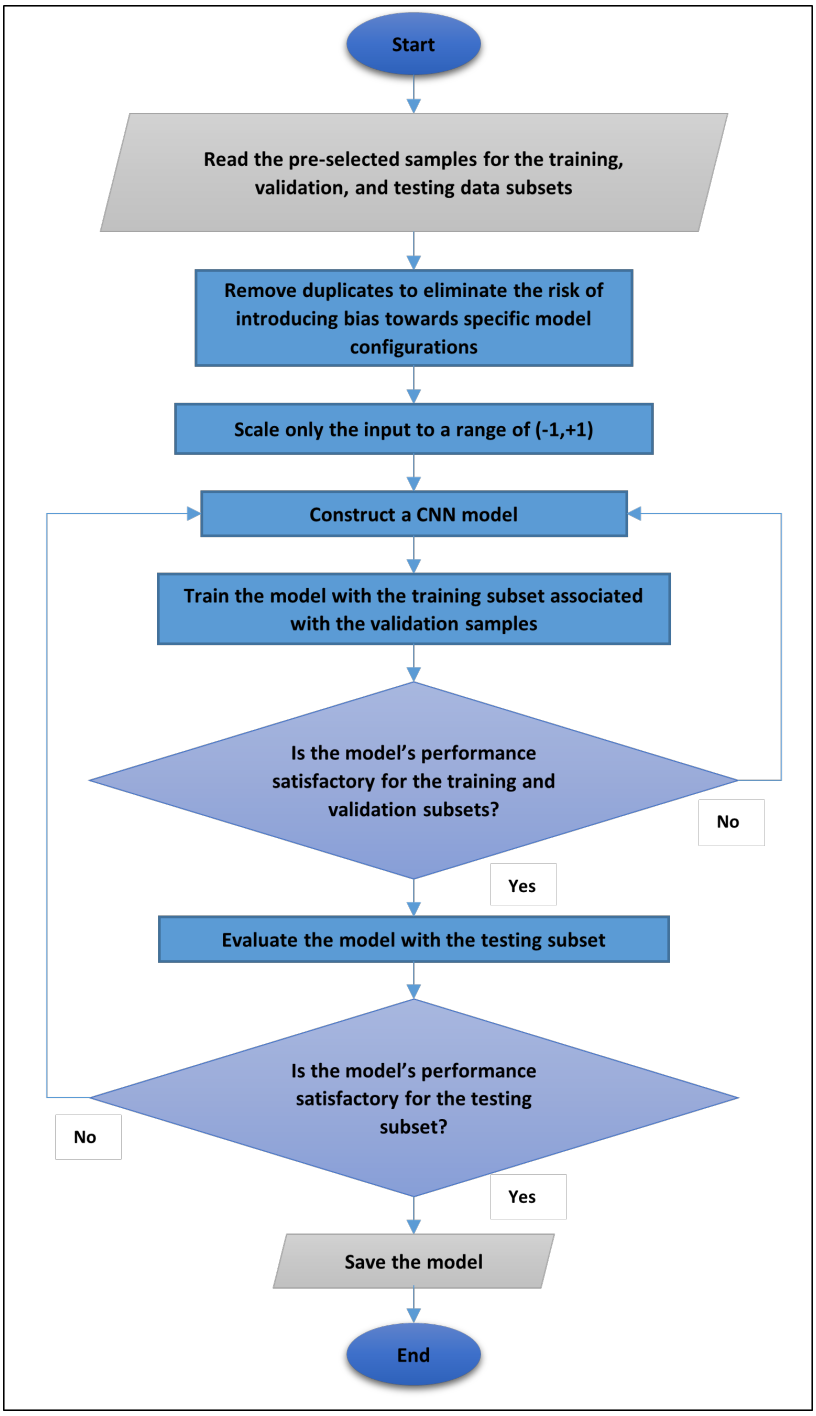


Figure 4.3: Graphical representation of different steps for reconstructing basis functions using a CNN model.

Chapter 5

Role of Skip Connections in Deep Neural Networks of Low Complexity

5.1 Introduction

Deep feed-forward networks, with high complexity, backpropagate the gradient of the loss function from the final layers to earlier layers. As a consequence, the gradient may descend rapidly toward zero. This is known as the "vanishing gradient phenomenon" that prevents earlier layers from benefiting from further training. One of the most efficient techniques to solve this problem is using skip connection (shortcut) schemes that enable the gradient to be directly backpropagated to earlier layers.

Despite much recent research directed at adopting skip connections in highly complex deep networks, there is a lack of critical analysis of the nature and causes of the vanishing gradient problem, and the comparative advantages gained when using skip connections in DNNs of low complexity, such as the one developed in Chapter 4. Additionally, although classification and regression algorithms are both categorized as supervised learning algorithms, reported work on the utilization of skip connections has mostly been directed at classification; there is much less reported work directed at regression. Hence, the work presented in this chapter is designed to address the second subsidiary research question from Chapter 1:

2. Will skip connections significantly affect the performance of Deep Neural Networks (DNNs) of low complexity or whether their inclusion has little or no effect?

There are two important motivations for doing so. First, the gradient in such networks might descend so rapidly to zero, possibly causing early convergence. Second, it might be the case that attempting to improve the performance of a CNN by simply adding more convolution blocks into the network has zero effect because of the vanishing gradient problem.

The rest of this chapter is organized as follows. Section 5.2 first explores different approaches to resolving the vanishing gradient problem. Next, the idea behind the concept of skip connection is given, with a particular focus on their mathematical underpinnings. Eventually, several examples are provided to illustrate the importance of skip connection.

Section 5.3 presents three skip connection schemes added to the SkiplessCNN architecture. The chapter is finished with a summary in Section 5.4.

5.2 Skip Connection Idea: Intuitive and Mathematical Explanation

Deep feed-forward networks are typically generated using some form of gradient-based training, such as backpropagation, whereby the gradient of the loss function (cost function) is calculated using the values assigned to weights and biases. The loss function measures how well the network is operating by considering the similarity between real and predicted outputs. There are various parameter optimizers that can be adopted to arrive at the minimum loss value. Multilayer NNs typically include a feed-forward pathway in which different layers are arranged, and parameters are initialized, as well as a backward pathway that progressively modifies parameters, thus gradually improving a model's performance.

DNNs contain several hidden layers as opposed to shallow networks, which have only one hidden layer. A significant advantage of deep networks with high complexity is that they can represent complex functions and learning features at various levels of abstraction. However, the disadvantage is the vanishing gradient phenomenon which may occur during the training process. As the network backpropagates the error gradient from the final layers to layers closer to the input layer, the gradient can descend rapidly to zero. This issue causes those parameters associated with layers near the input layer not to change as much as they should. In the worst case, the updating of these layers may cease to happen. This in turn is likely to have an adverse effect on the operation of the network.

Several approaches have been suggested to mitigate the vanishing gradient problem, such as: (i) applying multilevel hierarchies [63], (ii) employing long short-term memory units [64], (iii) defining orthogonal constraints on the initialization of the parameters [65], (iv) using regularization terms [66], (v) applying gated recurrent units [67], (vi) introducing extra layers such as BN layers [68, 69], and (vii) modification of the activation functions [70, 71].

The concept of using a skip connection (shortcut) in a neural network was first proposed in [72] as a way of mitigating the vanishing gradient problem in deep CNN models. The shortcut enables the cost function gradient to be directly backpropagated to layers close to the input layer. In traditional CNN architectures, the layers come one after another. Using the skip connection idea, a shortcut is added to the main path in the network. Depending on the dimensions of the input/output blocks, two main kinds of shortcuts can be defined: (i) identity shortcuts (used in this study), and (ii) convolutional shortcuts. An identity shortcut is employed where the input layer has the same size as the output layer. A convolutional shortcut is used where the sizes of the input and output layers do not match. As the name suggests, the convolutional shortcut comprises a convolutional layer, sometimes along with a BN unit. Skip connections can also be categorized as either short or long connections. Short skip connections typically exist in networks with consecutive convolutional layers, such as the residual neural network. Long skip connections are used with symmetrical architectures,

such as encoder-decoder sketches, where the spatial dimensionality decreases in the encoder section and increases in the decoder section. A combination of the encoder-decoder and long skip connections is referred to as U-Net [73].

The mathematical explanation of the identity skip connection concept is presented below. Supposing the output of an earlier layer as x_1 and the activation function as ReLU (Figure 5.1), a standard neural network carries out a series of operations in the next layers as follows:

$$z_1 = w_1\ x_1 + b_1, \tag{5.1}$$

$$x_2 = ReLU(z_1), \tag{5.2}$$

$$z_2 = w_2\ x_2 + b_2, \tag{5.3}$$

$$x_3 \text{ (without skip connection) } = ReLU(z_2), \tag{5.4}$$

where w_1 and w_2 refer to weights, and b_1 and b_2 are bias.

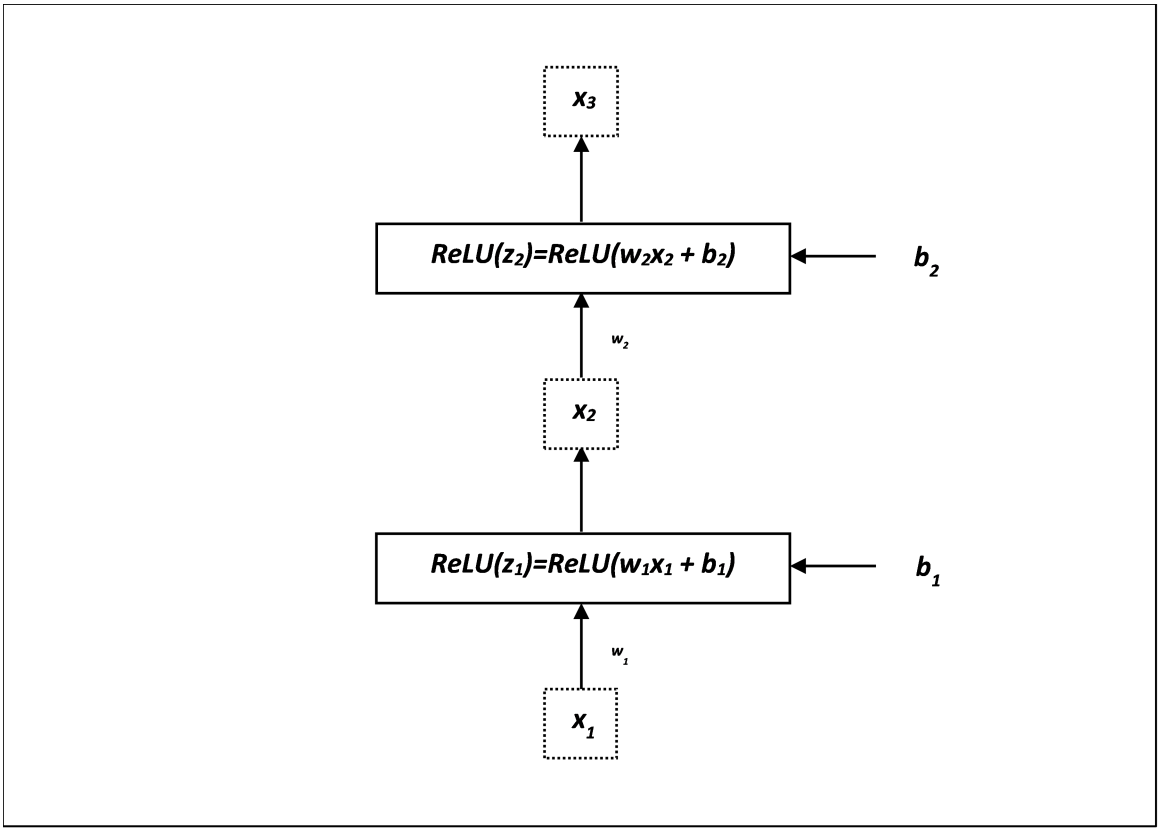


Figure 5.1: General structure of NNs without skip connection.

In addition to the mathematical operations above, a skip connection makes x_1 jump a layer and re-join the output before applying the second activation function (here ReLU):

$$x_3 \text{ (with skip connection)} = ReLU(z_2 + x_1). \tag{5.5}$$

All operations together form a residual block (Figure 5.2).

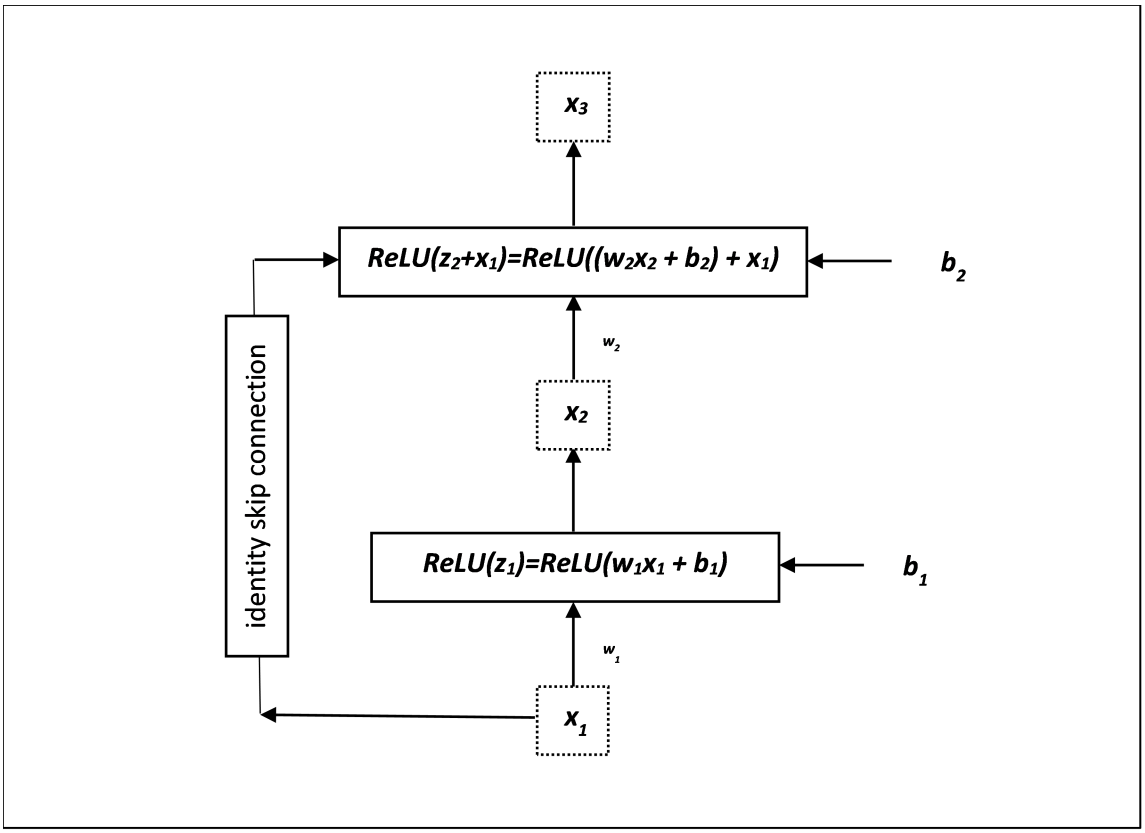


Figure 5.2: General structure of NNs with skip connection.

To better understand how skip connections address vanishing/degrading gradients, the calculations in the residual block are given as a function. With substitution of z_2 and x_1 by $\mathcal{F}(X)$ and X, it is necessary to define:

$$\mathcal{M}(X) = \mathcal{F}(X) + X. \tag{5.6}$$

The neural network may lose the ability to properly learn a complex mapping when $\mathcal{F}(X)$ becomes too small. With adding more layers to a network, this is more likely to happen.

By including X, the identity mapping of the input to $\mathcal{F}(X)$ is added. As a result, the network has a kind of baseline for the function it has to learn.

The cost (lost) function is differentiated with respect to weights during backpropagation. In order to reach the weights in the earlier layers, it is necessary to differentiate through all the intermediate functions in the previous layers. If the final output is produced by the last activation function in the residual block, then the cost function J is calculated immediately thereafter.

To propagate the gradient to the earlier layers, the gradient of the cost function is computed with respect to x, i.e., $\frac{\partial J}{\partial x}$. The chain rule makes the following for the full operation with all the intermediate steps to be conducted without the skip connection:

$$\frac{\partial J}{\partial x_1} = \frac{\partial J}{\partial x_3} \times \frac{\partial x_3}{\partial z_2} \times \frac{\partial z_2}{\partial x_2} \times \frac{\partial x_2}{\partial z_1} \times \frac{\partial z_1}{\partial x_1}. \tag{5.7}$$

NNs are prone to vanishing and exploding gradients as a result of this series of multiplications. In case of replacing the intermediate calculations with $\mathcal{F}(X)$, the gradient calculation changes to:

$$\frac{\partial J}{\partial X} = \frac{\partial J}{\partial \mathcal{F}(X)} \times \frac{\partial \mathcal{F}(X)}{\partial X}. \tag{5.8}$$

With including the skip connections, the rate of change of the cost function with respect to X in a neural network becomes:

$$\frac{\partial J}{\partial X} = \frac{\partial J}{\partial \mathcal{M}(X)} \times \frac{\partial \mathcal{M}(X)}{\partial X}. \tag{5.9}$$

Substitution of $\mathcal{M}(X)$ with $\mathcal{F}(X) + X$ results in the expression below:

$$\begin{aligned} \frac{\partial J}{\partial X} &= \frac{\partial J}{\partial \mathcal{M}(X)} \times \left(\frac{\partial (\mathcal{F}(X) + X)}{\partial X} \right) \\ &= \frac{\partial J}{\partial \mathcal{M}(X)} \times \left(\frac{\partial \mathcal{F}(X)}{\partial X} + 1 \right) \\ &= \left(\frac{\partial J}{\partial \mathcal{M}(X)} \times \frac{\partial \mathcal{F}(X)}{\partial X} \right) + \frac{\partial J}{\partial \mathcal{M}(X)}. \end{aligned} \tag{5.10}$$

The term $\frac{\partial J}{\partial \mathcal{M}(X)}$ i.e., the direct gradient of the cost function with respect to $\mathcal{M}(X)$ still exists even if $\frac{\partial \mathcal{F}(X)}{\partial X}$ becomes vanishingly small as a result of many multiplications applied during backpropagation through all the layers. This avoids the gradient from vanishing or exploding.

At the end of this section, six examples are presented to highlight the significance of utilizing skip connections in DNNs. The researchers in [74] investigated the impact of long and short skip connections on fully convolutional networks for biomedical image segmentation. To do so, the model was tested on electron microscopy data, with 25 images for training, and 5 for validation. The results revealed that only short skip connections could solve the vanishing gradient problem in their deep networks.

A DL model was presented to reconstruct super-resolution images in [75]. Using the proposed technique, a CNN was separated into multiple layer groups, and shortcuts with various multiplication factors were considered from the input data to these layer groups. In comparison with the alternative techniques, the suggested model reached a higher peak signal-to-noise ratio and better subjective quality given a similar amount of computation.

An attempt was made in [76] to develop a binary CNN with many skip connections for "Fog Computing" applications where many peripheral devices are connected using a cloud. The proposed binary CNN network featured decomposition convolution kernels and concatenated feature maps. The proposed model was tested on two datasets: ImageNet and CIFAR-10. Based on the results, this model was found to increase the classification accuracy with respect to both datasets.

A deep and high complex convolutional AE with several skip connections was developed in [77] to improve the colorization of grayscale images. Various numbers of kernels including 8, 32, 128, and 512 were used in this architecture. Additionally, there were two different kinds of skip connections: 3-layer-1-skip connection and 6-layer-1-skip connection. The stride was 2×2 in the former and 4×4 in the latter. Experiments on multiple datasets confirmed the effectiveness of the proposed model with the peak signal to noise ratio of 27.0595, root mean square error of 0.1311, structural similarity index measure of 0.561, and Pearson correlation coefficient of 0.9771.

The focus of the work present in [78] was on Densely Connected Convolutional Neural Network (DCCNN) to identify plant diseases. In this context, three architectures were developed: 6-block DCCNN (with 10 weight layers), 7-block DCCNN (11 weight layers), and 8-block DCCNN (12 weight layers). Different numbers of kernels were used to develop such models, such as 64, 128, 192, 256, 320, 384, 448, 512, and 576. The skip connections were put in modified dense blocks. According to the results, the 8-block DCCNN outperformed the other two models with an identification accuracy of 0.9823 – 0.9983 for different plants.

A novel mode namely Martian Crater UNet (MC-UNet) was developed in [79] to precisely recognize Martian craters at semantic and instance levels from thermal-emission-imaging-system daytime infrared images. This model had a depth of six, and the maximum number of convolutional channels was 256. In order to enhance the performance, average pooling was adopted and also channel attention was embedded into the skip-connection process between the encoder and decoder layers at the same network depth. The experimental results demonstrated that MC-UNet could recognize Martian craters with a maximum radius of 31.28 km (136 pixels) with a recall of 0.7916 and an F1-score of 0.8355.

5.3 Three Skip Connection-based CNN Architectures

To address the central objective of this section, comparisons are conducted using three different skip connection schemes as shown in Figure 5.3. How and where in the SkiplessCNN structure the shortcuts are located differs from scheme to scheme:

1. **FirstSkip**: a single skip connection from the first convolutional layer to the last one.
2. **MidSkip**: a single skip connection from the middle convolutional layer to the last layer.
3. **DualSkip**: two skip connections from the middle convolutional layer to the last and the second-to-last layers.

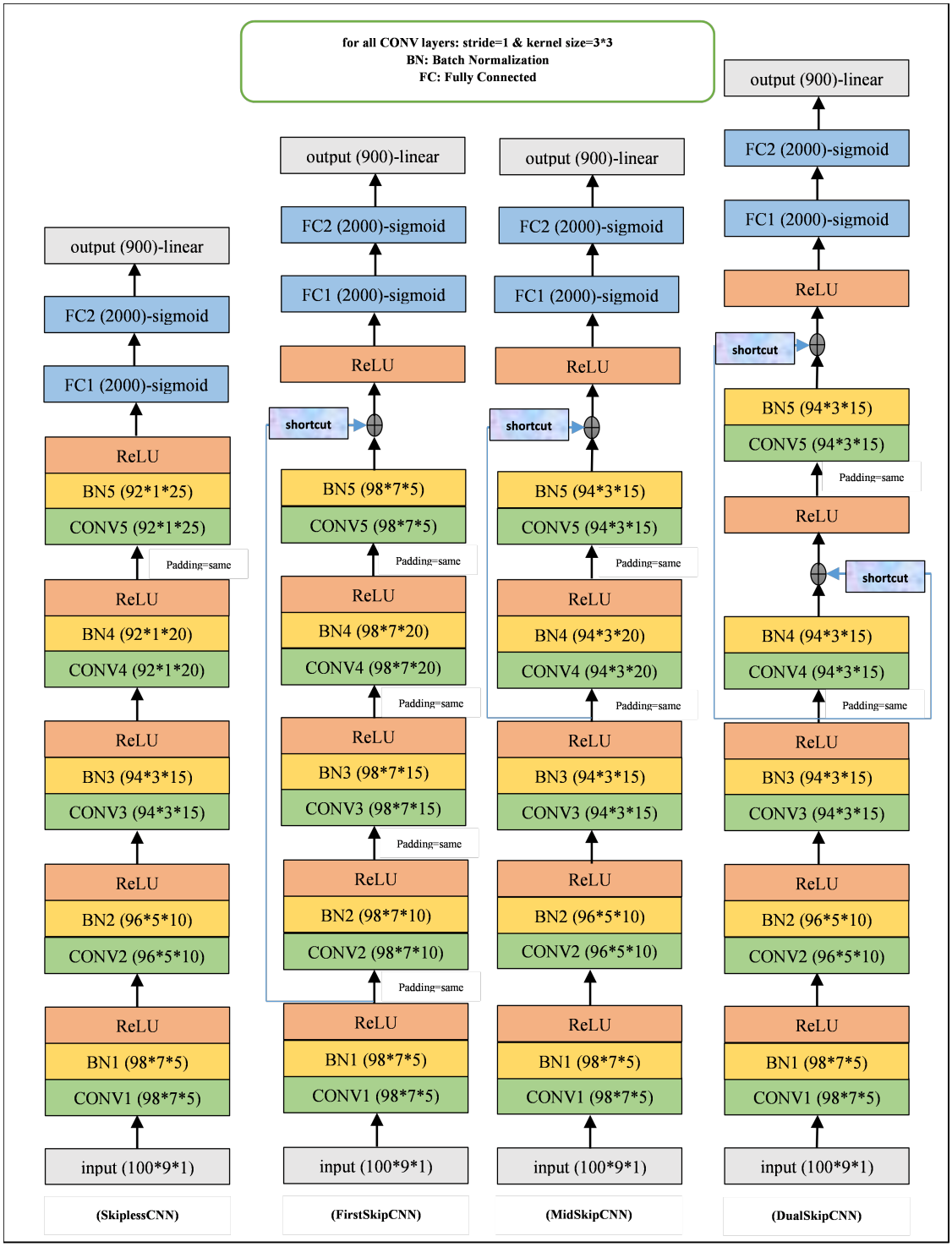


Figure 5.3: Structure of the CNN models with and without skip connections.

FirstSkip adds a single shortcut from the output of the first convolution layer to the last convolutional block. The input and output of this part have the same dimension of 98×7 because an identity type of shortcut is used. **MidSkip** is designed to discover how much a shortcut from the middle layer to the final layer can improve the performance of a model. Here, the input and output of this section with the shortcut have a dimension of 94×3. **DualSkip** was developed mainly to gain knowledge of the effect of involving the raw input features along with **MidSkip**. In all three cases, the main path and the shortcut meet each other before applying the activation function. For all three architectures, the FC layers remain unchanged.

Adding the skip connections increases the complexity of the standard structure (i.e., SkiplessCNN), in terms of the number of parameters. As explained in the previous section, there are as many as 10,414,170 trainable and 150 non-trainable parameters for the base structure, without shortcuts. By adding FirstSkip, the number of trainable and non-trainable parameters increases to 12,670,510 and decreases to 110, respectively. For MidSkipCNN, there are 14,272,340 trainable and 130 non-trainable parameters. For DualSkipCNN, the number of trainable and non-trainable parameters changes to 14,270,975 and 120, respectively. The number of parameters for all models is summarized in Table 5.1.

Table 5.1: Number of parameters for the developed CNN models.

Model	Number of trainable parameters	Number of non-trainable parameters
SkiplessCNN	10,414,170	150
FirstSkipCNN	12,670,510	110
MidSkipCNN	14,272,340	130
DualSkipCNN	14,270,975	120

5.4 Summary

Skip connection can mitigate the problem of the vanishing gradient phenomenon in deep feed-forward networks. The goal of this chapter was to investigate whether skip connections meaningfully affect the performance of DNNs of low complexity or whether their inclusion has little or no impact. In this sense, three different skip connection schemes were added to the SkiplessCNN structure. The next chapter is dedicated to using ensemble learning techniques to combine the models developed so far.

Chapter 6

Deep Ensemble Learning

6.1 Introduction

Ensemble learning is a technique in which several models or base learners are combined into an "ensemble". This produces a single model with better predictive or classification accuracy than the individual component base learners. Ensemble learning can be applied to various types of ML models, such as DTs, NNs, or Support Vector Machines (SVMs). There are several ways to combine the base learners in ensemble learning, including boosting, bagging, and stacking.

The majority of published research directed toward ensemble learning has been founded on traditional ML techniques. Such established mechanisms represent Shallow Ensemble Learning (SEL). The alternative is DEL, which combines a number of deep learners into an ensemble. Although DL algorithms tend to generate fewer prediction errors than traditional ML methods when applied to many datasets, there is scope to further improve their accuracy. Combining several deep learners into an ensemble is one way to potentially achieve this. Moreover, in a set of deep models, the different strengths of each DL model may complement one another, and weaknesses cancel each other out. Unlike SEL, DEL has received little attention to date. This chapter attempts to investigate the subsidiary question three, which was introduced in Chapter 1:

3. Does combining multiple deep learners into an ensemble improve the accuracy of DL algorithms?

Among popular ensemble learning methods, stacking is used for several reasons. Stacking improves predictive performance more effectively than boosting and bagging, particularly when dealing with complex and noisy datasets. It is more flexible than boosting and bagging since it allows for the use of different types of models, or even models with different hyperparameters, to create a diverse set of base models. Stacking can also help to address issues related to overfitting by providing a way to combine multiple models with different strengths and weaknesses.

The standard CNN model (SkiplessCNN) developed in Chapter 4 and skip connection-based CNN models (FirstSkipCNN, MidSkipCNN, and DualSkipCNN) developed in Chapter

5 are used as the base learners. These base learners are then combined using two regression models (linear regression and ridge regression), separately.

Subsequent to this introduction, three other sections are given. Section 6.2 begins with an explanation of reducible and irreducible errors of ML/DL algorithms, followed by the presentation of three important categories of ensemble systems: boosting, bagging, and stacking. Finally, a couple of examples are provided to demonstrate the application of ensemble learning in the geoscience field. Section 6.3 explains the stacking CNN ensemble model. The last section 6.4 mainly summarizes the key points expressed in this chapter.

6.2 Importance of Combining Models for Improved Predictive Performance

The wide range of ML/DL algorithms available all have to contend with reducible and irreducible errors. The latter is typically a consequence of noise within the datasets being evaluated and cannot be addressed by the ML/DL models themselves. On the other hand, bias and variance combine to generate reducible errors, which can be effectively reduced by the algorithmic actions of ML/DL. Bias errors are a consequence of the differences between predicted and actual dependent variable values generated with a training subset of samples. Variance errors result from small fluctuations in the training subsets actual values. Mathematically, it is supposed that there is an input vector $\mathbf{X}$ (here the permeability field) that influences an output vector $\mathbf{Y}$ (here the basis function). The function $f(\mathbf{X})$ denotes the correct relationship between the input and output, but it is accompanied by some noise that can be represented by σ_ϵ^2 that constitutes the irreducible error:

$$\mathbf{Y} = f(\mathbf{X}) + \sigma_\epsilon^2. \tag{6.1}$$

ML/DL models strive to determine the best function $\hat{f}(\mathbf{X})$ that can predict the true underlying function $f(\mathbf{X})$ as precisely as possible. Given the Total Error (TE) as $TE = E[(\mathbf{Y} - \hat{f}(\mathbf{X}))^2]$:

$$TE = [E\hat{f}(\mathbf{X}) - f(\mathbf{X})]^2 + E[\hat{f}(\mathbf{X}) - E\hat{f}(\mathbf{X})]^2 + \sigma_\epsilon^2, \tag{6.2}$$

$$TE = \text{bias}^2 + \text{variance} + \text{irreducible error}. \tag{6.3}$$

Simpler models tend to generate high bias accompanied by low variance. On the other hand, more elaborate models tend to generate lower bias accompanied by higher variance. Linear regression, for example, has a high bias since it tends to oversimplify and, therefore, cannot accurately capture the relationship between input variables and output data. In contrast, NNs involving multiple hidden layers and many nodes tend to generate a substantial variance, because they tend to overfit training datasets, making it difficult for the trained models to be generalized and accurately predict unseen data. High bias is typically a consequence of models underfitting a dataset, whereas high variance is typically a consequence of

models overfitting a dataset. As a modeling strategy, it makes sense, therefore, to attempt to trade off bias and variance errors to assist the trained models in being applied in a more generalized way and more accurately predict data not seen during the training/validation process.

Ensemble learning, whereby a number of base learners are combined into an "ensemble" to produce a single model whose predictive or classification accuracy is better than that of the individual component base learners, is a well-established technology [80, 81, 82, 83, 84, 85]. Ensemble systems can be categorised according to the method by which the ensemble learning is achieved:

1. **Boosting**: Boosting [86, 87] is a sequential method. The different base learners are dependent on each other. The aim is to fit models in steps such that model training at each step is influenced by the models constructed in the previous steps. Each step is focused on examples in the dataset that have been poorly predicted by the previous steps.

   Here is how boosting ensemble learning works:

   (i) The algorithm starts by training a weak learner on the entire dataset.

   (ii) The algorithm then looks at the instances that the first weak learner mispredicted, and assigns a higher weight to these instances.

   (iii) The algorithm trains a second weak learner on the same dataset but with the weights adjusted to give more importance to the previously mispredicted instances.

   (iv) The process is repeated, with each subsequent weak learner focusing on the instances that were mispredicted by the previous learners.

   (v) Finally, the predictions of all the weak learners are combined to make a final prediction. Typically, the final prediction is weighted by the accuracy of each weak learner, so that the more accurate weak learners have a higher weight in the final prediction.

   There are several types of boosting algorithms, including Adaptive Boosting (AdaBoost) and gradient boosting. Both algorithms work by iteratively adding weak learners to the ensemble, but they differ in the way the weights of the instances are adjusted and the way the weak learners are combined.

2. **Bagging**: Bagging (standing for bootstrap aggregating) [88, 89] is a parallel approach to ensemble learning where multiple models are generated using the same ML/DL algorithm but with different portions of the training data, which are then merged to produce a single more robust model than the individual base models. Multiple training subsets (bootstrap samples) are randomly selected from the initial training dataset with replacement (a single row of the initial data might be chosen zero, one, two, or even more times). Each model is developed from one subset, resulting in an ensemble of

several models. The final prediction is obtained by averaging (or simply ranking) all the predictions of the different learners.

Here are the steps involved in bagging:

(i) The training data is divided into multiple subsets (or bags) of equal size. These subsets are created by randomly selecting samples from the original dataset, with replacement (i.e., some samples may be selected more than once).

(ii) A model is trained on each of the bags. Typically, the same type of model is used for each bag, but with different subsets of the data.

(iii) Once all the models are trained, their predictions are combined using a simple average, weighted average, or majority voting method, depending on the type of problem being solved. The final prediction is made based on the combined predictions of all the models.

One of the most commonly used bagging algorithms is Random Forest (RF), which combines the bagging algorithm with DTs. In RF, multiple DTs are trained on different subsets of the data, and their predictions are combined using a weighted average or majority voting method.

3. **Stacking**: Unlike boosting and bagging, stacking [90, 91] uses base learners generated by different machine learners. The voting ensemble represents a simple stacking method in which a statistical mechanism is used to combine different types of ML/DL models, such as DTs and SVMs. No matter how well the individual ML models perform on the training dataset, they all contribute equally to the merged model. One can consider the simple average of the predictions from the underlying ML models. However, using a weighted average ensemble makes the results more sensitive to the prediction errors generated by each contributing ML/DL model. A further improvement can be made through stacked generalization, which applies an ML/DL model to learn how to best combine the predictions derived from the base learners. It does this by first developing base models using the training dataset inputs. It then feeds the underlying ML/DL models into a meta-learner, which attempts to make a new model using the predictions of the weak learners based on new data.

   Here is a step-by-step procedure of how stacking works:

   (i) The initial data is divided into two (in some cases more) subsets. One subset is used to train the individual base models, and the other subset is used to train the meta-model.

   (ii) Multiple base models are trained on one subset using different algorithms or model architectures. These models can be of different types, such as DTs, RFs, NNs, etc.

   (iii) Once all the base models are trained, they are used to make predictions on the other subset of the training data.

(iv) The predictions of the base models are then used as input features for the meta-model.

(v) A meta-model is trained on the second subset using the true labels of the validation set as the target variable.

(vi) Once the meta-model is trained, it can be used to make predictions on the testing data.

The meta-model can be any type of model, such as linear regression, logistic regression, or neural network.

Ensemble ML/DL techniques are now quite widely applied in various engineering and geoscience disciplines. For example, three models of Bayesian, functional, and meta-ensemble were applied to Land Subsidence Susceptibility (LSS) mapping in [92]. The models split the dataset 50:50 between training and testing subsets with errors measured in relation to the operating-characteristic curve. The ensemble including the logit boost model delivered the most accurate (91.44%) LSS maps.

Slope stability predictions were generated using a hybrid stacking ensemble method in [93]. An artificial bee colony optimizer was applied to identify the optimal combination of base classifiers (ensemble level 0). These were then used to develop an effective meta-classifier (ensemble level 1), considering eleven separate tuned ML models. Finite element analysis was employed to create a synthetic database (150 records) for training the models. The trained models were then applied to predict 107 naturally occurring slope cases to test model performances. The hybrid-stacking ensemble model generated less errors than each ML model used in isolation.

Ensemble RF, ensemble Gradient Boosted Regression Tree (GBRT), and MultiLayer Perceptron (MLP) neural network were applied to model the spatial extent of landslides in Norway in [94]. Eleven landslide-influencing factors were considered related to geomorphologic, geologic, geo-environmental, and anthropogenic effects. 3,399 positive landslide records and 6,798 non-landslide were considered. Seventy percent of the data records in each of these two categories were selected for training the models. The remaining thirty percent of the data records were used to test the trained models. The slope angle was confirmed by the models to be the most important influencing factor. The ensemble GBRT model outperformed the other ensemble models, achieving a 95% probability of detecting landslides in that region.

SVM, MLP, RF, AdaBoost, and extreme gradient boosting were used to develop synthetic geochemical logs for pre-salt reservoirs in Brazil in [95]. Seven petrophysical logs: natural gamma-ray, gamma-ray spectroscopy, density, photoelectric factor, neutron porosity, nuclear magnetic resonance, and sonic formed the input variables. The chemical element concentrations for Al, Ca, Fe, Mg, Na, Si, S, and Ti were the prediction objectives. In addition to showing the best results, AdaBoost was found to be the most practical tree-ensemble algorithm to apply as it involved simpler pre-processing and control variable optimization.

The researchers in [96] used stacking ensemble learning to filter and fuse multiple basic ML models to further improve the regression effect of Minimum Miscibility Pressure (MMP) data.

First, the correlation analysis and variance inflation factor of the MMP data in the dataset were performed, and the duplicate data were excluded due to correlation and collinearity problems. In total, 147 MMP data were then regressed using seven basic models, whose results were preliminarily screened and combined with empirical formula data to form a new dataset. Then, the final output result was achieved through a stacking model. Besides fitting curves, the results of the stacking model also exhibited improvements in the prediction of MMP, as evidenced by the reduction in mean absolute error, root mean square error, and increase in R^2.

A variety of models such as Light Gradient Boosting Machine [LightGBM], extreme gradient boosting, categorical boosting, RF, MLP, linear regression, and CNN were utilized to forecast porosity in a reservoir situated in Brazil in [97]. These models were trained using porosity and elastic parameters from the smoothed and standardized logging data. The trained models were applied to the inverted elastic properties to estimate the porosity profile from seismic data. In the crosswell blind tests, LightGBM had the highest accuracy, while also having the shortest runtime among the investigated models. Besides, the authors discovered that the incorporation of clay content had the potential to enhance the accuracy of porosity predictions, while oil saturation had less of an impact.

6.3 Stacking CNN Ensemble Model

A schematic of the proposed stacking CNN ensemble model is given in Figure 6.1. There are four base learners: SkiplessCNN, FirstSkipCNN, MidSkipCNN, and DualSkipCNN. All these models were constructed using 304,511 training data records, together with 34,421 validation data records. The SkiplessCNN architecture consists of seven weight/bias layers: five convolutional and two FC layers without any pooling layer. Convolutional layers 1 to 5 consist of 5, 10, 15, 20, and 25 kernels, respectively. Each convolutional layer is followed by a single BN layer of the same dimensions. Additionally, each FC layer contains 2000 neurons. In FirstSkipCNN, a single shortcut is added from the output of the first convolution process to the last convolutional block. In MidSkipCNN, there is a single skip connection from the middle convolutional layer to the last layer. The fourth model, i.e., DualSkipCNN includes two skip connections from the middle convolutional layer to the last and the second-to-last layers. In all CNN models with skip connections, the main path and the shortcut meet each other before applying the activation function. The structure of the FC layers remains unchanged for all three architectures.

A new training dataset for the meta learner was established by providing all data records from the validation subsets to each of the four sub-models and collecting the predictions they generated i.e., $P_{\text{SkiplessCNN}}$, $P_{\text{FirstSkipCNN}}$, $P_{\text{MidSkipCNN}}$, and $P_{\text{DualSkipCNN}}$. These resulted in four (referring to the number of base learners) arrays with the shape $[34421, 900]$, the first element referring to the number of validation data and the second to the output (basis function). Thus, a 3D array was developed with the shape $[34421, 4, 900]$, which was transformed into a $[34421, 3600]$ shaped array. This flattened input data, along with their output was used to train a meta learner.

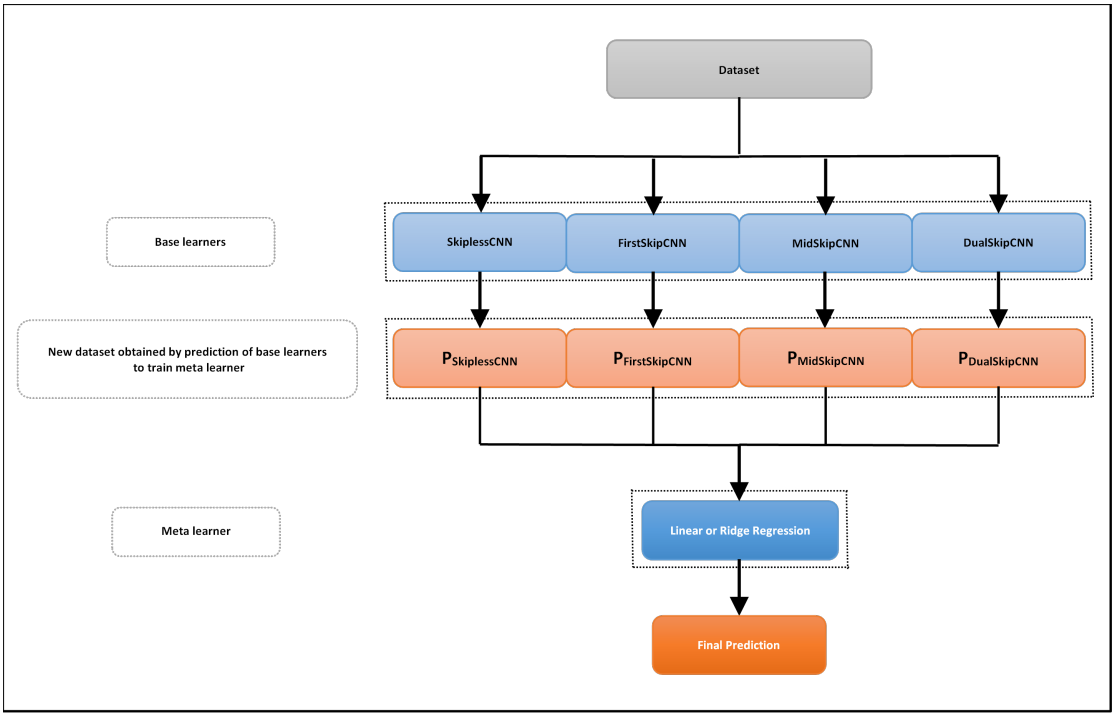


Figure 6.1: The workflow diagram of the stacking CNN ensemble model. Four base learners of SkiplessCNN, FirstSkipCNN, MidSkipCNN, and DualSkipCNN are developed using training/validation subsets. After being trained, they are used to make predictions on the validation data ($P_{SkiplessCNN}$, $P_{FirstSkipCNN}$, $P_{MidSkipCNN}$, and $P_{DualSkipCNN}$). Then, two meta models are separately developed using linear regression and ridge regression. Once the meta models are trained, they can be used to make predictions on the testing data (28,879 samples).

As mentioned earlier, there are 875 cases that each need to be processed through the sequence of training, validation, and testing. Given that the input/output dimensions are so large, it did not make sense to apply boosting to focus on samples in the dataset that have been predicted incorrectly by the previous models in the sequence. Bagging is usually applied to relatively small datasets. Additionally, conducting bootstrap sampling incorporating all 875 cases was not feasible for addressing this large dataset. Therefore, stacking was selected to establish the ensemble model. A stacked generalization method was chosen, mainly because it is more flexible mathematically than voting or weighted average methods. To be more specific, the four base learners were combined into an ensemble model using linear and ridge regression, separately.

The linear regression is one of the most straightforward approaches to predicting output via a linear function of input features. In the context of ML/DL, it refers to the most usual least square linear regression method that attempts to minimize the cost function. A drawback, however, is that it does not penalize high magnitude weights in its error function and it assumes independence between its features. These characteristics can lead in some

cases to certain features being assigned very high weights during the training. The cost function for linear regression is typically expressed as:

$$\text{cost function}_{\text{linear}} = \sum_{i=1}^{m} (\mathbf{Y} - \hat{f}(\mathbf{X}))^2. \tag{6.4}$$

The ridge regression, as a modification of linear regression, involves a penalty (L2 regularization) to its error term, calculated as the sum of the squared value of the weights. Giving a penalty in such a way results in a set of more evenly distributed weights. The cost function for ridge regression becomes:

$$\text{cost function}_{\text{ridge}} = \sum_{i=1}^{m} (\mathbf{Y} - \hat{f}(\mathbf{X}))^2 + \alpha \sum_{j=1}^{p} (w_{\text{j}})^2, \tag{6.5}$$

here, α is included as a coefficient to penalize weights. It can take different values. A ridge model with $\alpha = 0$ is the same as a simple linear regression. As the α value nears infinity, an increasing number of coefficients of the model becomes zero until it is just a flat model with an intercept. In this study, the default value of "one" was used for all cases.

Once the meta model is trained, it can be used to make predictions on the test data. For each basis function, the 28,879 testing samples were separately used to assess the combined model and determine if it could perform better than any of the individual base learners.

6.4 Summary

The accuracy of DL algorithms can be improved by combining several deep learners into an ensemble. This eliminates the need for continuous adjustments to the architecture of individual networks or the nature of the propagation. This chapter investigated the potential prediction improvements achieved by using DEL to estimate four distinct multiscale basis functions in the mixed GMsFEM. A standard CNN and three skip connection-based CNNs were used for the base learners. For each basis function, these four CNNs were combined into an ensemble model using linear regression and ridge regression, separately. While DL models are effective prediction tools, they often neglect the issue of uncertainty. That is why MC dropout is applied in the next chapter to investigate the reliability of DualSkipCNN.

Chapter 7

Reliability of the DualSkipCNN Model

7.1 Introduction

ML/DL models have been introduced in various fields to make decisions using available data and domain knowledge. It is crucial to consider both accuracy and reliability when evaluating such models. These models are typically assessed based on their accuracy using statistical error metrics such as: (i) for regression: R^2, MSE, and relative error, and (ii) for classification: precision, F1 score, and confusion matrix.

In terms of reliability, ML/DL models deal with two main types of uncertainty: (i) aleatoric uncertainty (also called irreducible uncertainty/data uncertainty/inherent randomness) and (ii) epistemic uncertainty (also called knowledge uncertainty/subjective uncertainty) [98]. Aleatoric uncertainty arises from an inherent property of the data and cannot be reduced even with a higher volume of samples. The data used to develop a model can be sourced from experimental measurements, collected from other resources, or produced via simulation/programming (including our case). This data always contains noise, which refers to the data distribution and errors made while measuring, collecting, or generating data. A related problem is incomplete coverage of the domain. That is why most models are constructed based on a limited range of data and cannot be generalized. Epistemic uncertainty is a property of a model caused by factors such as the selection of very simple or complex structures, the stochastic nature of optimization algorithms, or the type of statistical error metrics. This uncertainty is reducible by feeding enough training samples into the model.

UQ techniques are beneficial to limit the effect of uncertainties on decision-making processes. According to [98], there are three main types of UQ: (i) Bayesian methods such as MC dropout, Markov Chain Monte Carlo, variational inference, Bayesian active learning, Bayes by backprop, variational AEs, Laplacian approximations, and UQ in RL like Bayes-adaptive Markov decision process, (ii) ensemble techniques such as deep ensemble, deep ensemble Bayesian/Bayesian deep ensemble, and uncertainty in Dirichlet deep networks like information-aware Dirichlet networks, and (iii) other methods such as deep GP and UQ in the traditional ML domain using ensemble techniques like SVM with Gaussian sample uncertainty.

Despite extensive research into the accuracy of DL models, their reliability analysis is still not adequate. This is because DL models are highly complex and have a large number of parameters, making it challenging to understand their decision-making process. Additionally, these models are susceptible to biases and overfitting. Therefore, improving the reliability analysis of DL models is necessary to ensure their trustworthy deployment in real-world applications. This chapter is going to investigate the subsidiary question four introduced in Chapter 1:

4. How does incorporating Uncertainty Quantification (UQ) methods improve the reliability of the CNN models in predicting new data points?

In this sense, MC dropout, a computationally efficient approach, is applied to investigate the reliability of DualSkipCNN model, with a focus on epistemic uncertainty. MC dropout is a regularization technique for NNs that is inspired by Bayesian inference. It involves randomly dropping out neurons at both training and test time.

The remainder of this chapter is organized in the following manner. Section 7.2 starts with a mathematical explanation of the standard GP, deep GP, and Bayesian approximation. Then, it is concisely demonstrated that how MC dropout can be interpreted as a Bayesian approximation of the probabilistic deep GP. Finally, four examples are provided to demonstrate the application of MC dropout in subsurface fluid flow modeling. Section 7.3 explains the structure of the DualSkipCNN model with dropout. The chapter is summarized in Section 7.4.

7.2 MC Dropout and its Related Work

Standard deterministic DNNs operate on a one-input-one-output basis. Unlike single-point predictions of such models, Bayesian methods such as Bayesian Neural Networks (BNNs) and (GPs) give predictive distributions. The weights of BNNs are incorporated with prior distributions, whereas GPs introduce priors over functions. A drawback of BNNs and GPs is the computational cost, which becomes prohibitive given a very large network, as in the case of deep networks. BNNs need to be fed the posterior distribution across the network's parameters, in which all possible events are obtained at the output. GPs require us to sample prior functions from multivariate Gaussian distribution, wherein the dimension of Gaussian distribution increases proportionally with the number of training points involving the whole dataset during predictions.

For a standard GP, a collection of training input-output pairs, stored in matrices $\mathbf{X} \in \mathcal{R}^{N\times Q}$ and $\mathbf{Y} \in \mathcal{R}^{N\times D}$ is given in the traditional probabilistic inference framework. The goal is to estimate the latent function $f = f(\mathbf{x})$, which is responsible for generating $\mathbf{Y}$ based on $\mathbf{X}$. GPs can be used as nonparametric prior distributions over f in this context [99]. Each datapoint $\mathbf{y}_n$ is generated from the corresponding $f(\mathbf{x}_n)$ by adding independent Gaussian noise. In other words:

$$\mathbf{y}_n = f(\mathbf{x}_n) + \epsilon_n, \quad \epsilon \sim \mathcal{N}\left(\mathbf{0}, \sigma_\epsilon^2 \mathbf{I}\right), \tag{7.1}$$

in which f is drawn from a GP, i.e., $f(\mathbf{x}) \sim \mathcal{GP}(\mathbf{0}, k(x, x'))$.

This (zero-mean) GP prior is solely dependent on the covariance function k operating on the inputs X. In order to create a flexible model, a covariance function is chosen that defines the characteristics of the mapping f, without imposing strong assumptions about its structure. For instance, an exponentiated quadratic covariance function, $k(\mathbf{x}_i, \mathbf{x}_j) = (\sigma_{se})^2 \exp\left(-\frac{(\mathbf{x}_i - \mathbf{x}_j)^2}{2l^2}\right)$, results in infinitely smooth latent functions. Any covariance function hyperparameters, such as (σ_{se}, l) of the aforementioned covariance function, are represented by $\boldsymbol{\theta}$. The collection of latent function instantiations, represented by $\mathbf{F} = \{\mathbf{f}_n\}_n^N$, follows a normal distribution. Then, the marginal likelihood can be analytically calculated:

$$\begin{aligned} p(\mathbf{Y} \mid \mathbf{X}) &= \int \prod_{n=1}^{N} p(\mathbf{y}_n \mid \mathbf{f}_n)\, p(\mathbf{f}_n \mid \mathbf{x}_n)\, \mathrm{d}\mathbf{F} \\ &= \mathcal{N}\left(\mathbf{Y} \mid \mathbf{0}, \mathbf{K}_{NN} + \sigma_\epsilon^2 \mathbf{I}\right), \mathbf{K}_{NN} = k(\mathbf{X}, \mathbf{X}). \end{aligned} \tag{7.2}$$

The successful use of GP can extend to unsupervised learning scenarios, where the input data $\mathbf{X}$ is not directly observed. In order to address this issue, the GP Latent Variable Model (GP-LVM) proposed in [100] is utilized, which treats the unobserved inputs $\mathbf{X}$ as latent variables. The model employs a product of D independent GPs as a prior for the latent mapping, resulting in an elegant solution. The generative procedure is assumed to be in the form: $y_{nd} = f_d(\mathbf{x}_n) + \epsilon_{nd}$, where $\boldsymbol{\epsilon}$ is again Gaussian with variance σ_ϵ^2 and $\mathbf{F} = \{\mathbf{f}_d\}_{d=1}^D$ with $f_{nd} = f_d(\mathbf{x}_n)$. Given a finite dataset, the GP priors can be expressed as:

$$p(\mathbf{F} \mid \mathbf{X}) = \prod_{d=1}^{D} \mathcal{N}(\mathbf{f}_d \mid \mathbf{0}, \mathbf{K}_{NN}). \tag{7.3}$$

This form allows for general non-linear mappings to be marginalised out analytically to obtain the likelihood $p(\mathbf{Y} \mid \mathbf{X}) = \prod_{d=1}^{D} \mathcal{N}(\mathbf{y}_d \mid \mathbf{0}, \mathbf{K}_{NN} + \sigma_\epsilon^2 \mathbf{I})$, analogously to Equation 7.2.

Now, it is time to explain deep GP. The architecture of a deep GP can be represented as a graphical model with three types of nodes: the leaf nodes $\mathbf{Y} \in \mathcal{R}^{N \times D}$ which are observed, the intermediate latent spaces $\mathbf{X}_h \in \mathcal{R}^{N \times Q_h}$, where h ranges from 1 to $H-1$ (the number of hidden layers), and the parent latent node $\mathbf{Z} = \mathbf{X}_H \in \mathcal{R}^{N \times Q_Z}$. The parent node may not be observed and can be subject to a prior of the available choices, such as a dynamical prior. Alternatively, it can serve as input for a supervised learning task. For simplicity, here the focus is on the unsupervised learning scenario. In this deep architecture, all intermediate nodes $\mathbf{X}_h$ act as inputs for the layer below (including the leaves) and as outputs for the layer above. To avoid complexity, a structure with only two hidden units is considered. The generative process can be expressed as follows:

$$\begin{aligned} y_{nd} &= f_d^Y(\mathbf{x}_n) + \epsilon_{nd}^Y, \quad d = 1, \ldots, D, \quad \mathbf{x}_n \in \mathcal{R}^Q \\ x_{nq} &= f_q^X(\mathbf{z}_n) + \epsilon_{nq}^X, \quad q = 1, \ldots, Q, \quad \mathbf{z}_n \in \mathcal{R}^{Q_Z}. \end{aligned} \tag{7.4}$$

The intermediate node is involved in two GPs, f^Y and f^X, playing the role of an input and an output respectively: $f^Y \sim \mathcal{GP}\left(\mathbf{0}, k^Y(\mathbf{X}, \mathbf{X})\right)$ and $f^X \sim \mathcal{GP}\left(\mathbf{0}, k^X(\mathbf{Z}, \mathbf{Z})\right)$. This structure

can be extended either vertically (i.e. deeper hierarchies) or horizontally (i.e. segmentation of each layer into different partitions of the output space). However, adding more layers increases the number of model parameters ($\mathbf{X}_h$) and causes a regularization challenge, since the size of each latent layer is crucial but has to be a priori defined. To address this, unlike the procedure suggested in [101], it is attempted to variationally marginalise out the whole latent space. By doing so, Occam's razor is automatically applied through Bayesian training and significantly reduces the number of model parameters, as the variational procedure only adds variational parameters. To achieve this, Automatic Relevance Determination (ARD) covariance functions are first defined for the GPs:

$$k\left(\mathbf{x}_i, \mathbf{x}_j\right) = \sigma_{ARD}^2 e^{-\frac{1}{2}\sum_{q=1}^{Q} w_q (x_{i,q} - x_{j,q})^2}. \tag{7.5}$$

This covariance function supposes a distinct weight w_q for each latent dimension, which can be utilized in Bayesian training to "switch off" unimportant dimensions by driving their corresponding weight to zero. This helps in discovering the structure of intricate models automatically. However, the nonlinearities introduced by this covariance function make it difficult to apply Bayesian techniques to this model. Nonetheless, recent non-standard variational inference methods have enabled the definition of an approximate Bayesian training procedure, as will be explained in the following.

A Bayesian training procedure requires optimisation of the model evidence:

$$\log p(\mathbf{Y}) = \log \int_{\mathbf{X},\mathbf{Z}} p(\mathbf{Y} \mid \mathbf{X}) p(\mathbf{X} \mid \mathbf{Z}) p(\mathbf{Z}). \tag{7.6}$$

If there is prior knowledge available about the observed data, such as their dynamic nature, the prior distribution on the parent latent node can be chosen to restrict the entire latent space by propagating the prior density through the cascade. Here, it may be feasible to take the general case where $p(\mathbf{Z}) = \mathcal{N}(\mathbf{Z} \mid \mathbf{0}, \mathbf{I})$. However, calculating the integral of Equation 7.6 is difficult because of the nonlinear treatment of $\mathbf{X}$ and $\mathbf{Z}$ through the GP priors f^Y and f^X. As a first step, Jensen's inequality is used to find a variational lower bound $\mathcal{F}_v \leq \log p(\mathbf{Y})$, considering:

$$\mathcal{F}_v = \int_{\mathbf{X},\mathbf{Z},\mathbf{F}^Y,\mathbf{F}^X} \mathcal{Q} \log \frac{p\left(\mathbf{Y}, \mathbf{F}^Y, \mathbf{F}^X, \mathbf{X}, \mathbf{Z}\right)}{\mathcal{Q}}. \tag{7.7}$$

Including variational distribution $\mathcal{Q}$, the joint distribution mentioned earlier can be expressed using the following expansion.

$$\begin{aligned} &p\left(\mathbf{Y}, \mathbf{F}^Y, \mathbf{F}^X, \mathbf{X}, \mathbf{Z}\right) = \\ &p\left(\mathbf{Y} \mid \mathbf{F}^Y\right) p\left(\mathbf{F}^Y \mid \mathbf{X}\right) p\left(\mathbf{X} \mid \mathbf{F}^X\right) p\left(\mathbf{F}^X \mid \mathbf{Z}\right) p(\mathbf{Z}). \end{aligned} \tag{7.8}$$

The integral of Equation 7.7 remains difficult to solve because $\mathbf{X}$ and $\mathbf{Z}$ still appear nonlinearly in the $p\left(\mathbf{F}^Y \mid \mathbf{X}\right)$ and $p\left(\mathbf{F}^X \mid \mathbf{Z}\right)$ terms respectively. A key result of the work presented in [102] is that expanding the probability space of the GP prior $p(\mathbf{F} \mid \mathbf{X})$ with

extra variables allows for priors on the latent space to be propagated through the nonlinear mapping f. More precisely, it is possible to augment the probability space of Equation 7.3 with K auxiliary pseudo-inputs $\tilde{\mathbf{X}} \in \mathcal{R}^{K \times Q}$ and $\tilde{\mathbf{Z}} \in \mathcal{R}^{K \times Q_Z}$ that correspond to a set of function values $\mathbf{U}^Y \in \mathcal{R}^{K \times D}$ and $\mathbf{U}^X \in \mathcal{R}^{K \times Q}$ respectively. Following this method, the augmented probability space is obtained: $p\left(\mathbf{Y}, \mathbf{F}^Y, \mathbf{F}^X, \mathbf{X}, \mathbf{Z}, \mathbf{U}^Y, \mathbf{U}^X, \tilde{\mathbf{X}}, \tilde{\mathbf{Z}}\right) =$

$$\begin{aligned} & p\left(\mathbf{Y} \mid \mathbf{F}^Y\right) p\left(\mathbf{F}^Y \mid \mathbf{U}^Y, \mathbf{X}\right) p\left(\mathbf{U}^Y \mid \tilde{\mathbf{X}}\right) \\ & \cdot p\left(\mathbf{X} \mid \mathbf{F}^X\right) p\left(\mathbf{F}^X \mid \mathbf{U}^X, \mathbf{Z}\right) p\left(\mathbf{U}^X \mid \tilde{\mathbf{X}}\right) p(\mathbf{Z}), \end{aligned} \tag{7.9}$$

where the pseudo-inputs $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Z}}$ are known as inducing points. For the sake of simplicity, they are dropped from now on. It is important to note that $\mathbf{F}^Y$ and $\mathbf{U}^Y$ are draws from the same GP so that $p\left(\mathbf{U}^Y\right)$ and $p\left(\mathbf{F}^Y \mid \mathbf{U}^Y, \mathbf{X}\right)$ are also Gaussian distributions. The same holds true for $p\left(\mathbf{U}^X\right), p\left(\mathbf{F}^X \mid \mathbf{U}^X, \mathbf{Z}\right)$.

With the help of the new expressions for the augmented GP priors, it is sensible to define a variational distribution $\mathcal{Q}$ that leads to a tractable variational bound. In other words:

$$\begin{aligned} \mathcal{Q} = & p\left(\mathbf{F}^Y \mid \mathbf{U}^Y, \mathbf{X}\right) q\left(\mathbf{U}^Y\right) q(\mathbf{X}) \\ & \cdot p\left(\mathbf{F}^X \mid \mathbf{U}^X, \mathbf{Z}\right) q\left(\mathbf{U}^X\right) q(\mathbf{Z}). \end{aligned} \tag{7.10}$$

The terms $q\left(\mathbf{U}^Y\right)$ and $q\left(\mathbf{U}^X\right)$ are selected to be free-form variational distributions. Additionally, $q(\mathbf{X})$ and $q(\mathbf{Z})$ are chosen to be Gaussian, factorised with respect to dimensions:

$$q(\mathbf{X}) = \prod_{q=1}^{Q} \mathcal{N}\left(\boldsymbol{\mu}_q^X, \mathbf{S}_q^X\right), q(\mathbf{Z}) = \prod_{q=1}^{Q_Z} \mathcal{N}\left(\boldsymbol{\mu}_q^Z, \mathbf{S}_q^Z\right). \tag{7.11}$$

When Equation 7.10 is substituted back into Equation 7.7 and the original joint distribution is replaced with its augmented version in Equation 7.9, it can be observed that the problematic terms $p\left(\mathbf{F}^Y \mid \mathbf{U}^Y, \mathbf{X}\right)$ and $p\left(\mathbf{F}^X \mid \mathbf{U}^X, \mathbf{Z}\right)$ are canceled out in the fraction, resulting in a quantity that can be analytically computed:

$$\mathcal{F}_v = \int \mathcal{Q} \log \frac{p\left(\mathbf{Y} \mid \mathbf{F}^Y\right) p\left(\mathbf{U}^Y\right) p\left(\mathbf{X} \mid \mathbf{F}^X\right) p\left(\mathbf{U}^X\right) p(\mathbf{Z})}{\mathcal{Q}'}, \tag{7.12}$$

where $\mathcal{Q}' = q\left(\mathbf{U}^Y\right) q(\mathbf{X}) q\left(\mathbf{U}^X\right) q(\mathbf{Z})$ and the above integration is with respect to $\mathbf{X}$, $\mathbf{Z}$, $\mathbf{F}^Y$, $\mathbf{F}^X$, $\mathbf{U}^Y$, and $\mathbf{U}^X$.

To be more specific, it is possible to break the logarithm in Equation 7.12 by grouping the variables of the fraction in such a way that the bound can be written as:

$$\mathcal{F}_v = \mathbf{g}_Y + \mathbf{r}_X + \mathcal{H}_{q(\mathbf{X})} - \mathrm{KL}(q(\mathbf{Z}) \| p(\mathbf{Z})), \tag{7.13}$$

where $\mathcal{H}$ refers to the entropy with respect to a distribution, KL represents the Kullback-Leibler divergence and, $\langle \cdot \rangle$ is used to denote expectations:

$$
\begin{aligned}
\mathbf{g}_Y &= g\left(\mathbf{Y}, \mathbf{F}^Y, \mathbf{U}^Y, \mathbf{X}\right) \\
&= \left\langle \log p\left(\mathbf{Y} \mid \mathbf{F}^Y\right) + \log \frac{p\left(\mathbf{U}^Y\right)}{q\left(\mathbf{U}^Y\right)} \right\rangle_{p(\mathbf{F}^Y|\mathbf{U}^Y,\mathbf{X})q(\mathbf{U}^Y)q(\mathbf{X})} \\
\mathbf{r}_X &= r\left(\mathbf{X}, \mathbf{F}^X, \mathbf{U}^X, \mathbf{Z}\right) \\
&= \left\langle \log p\left(\mathbf{X} \mid \mathbf{F}^X\right) + \log \frac{p\left(\mathbf{U}^X\right)}{q\left(\mathbf{U}^X\right)} \right\rangle_{p(\mathbf{F}^X|\mathbf{U}^X,\mathbf{Z})q(\mathbf{U}^X)q(\mathbf{X})q(\mathbf{Z})},
\end{aligned}
\tag{7.14}
$$

where the two terms $\mathbf{g}_Y$ and $\mathbf{r}_X$ involve known Gaussian densities and are, therefore, tractable. The $\mathbf{g}_Y$ term is only associated with the leaves and, thus, is the same as the bound found for the Bayesian GP-LVM [102]. Furthermore, $\mathbf{r}_X$ involve $\mathbf{X}$, which is the outputs of the top layer.

A computationally more efficient method called MC dropout has been recently developed [103]. A NN with any depth and non-linearities accompanying dropout before weight layers might be interpreted as a Bayesian approximation of the probabilistic deep GP. Additionally, the dropout objective minimizes KL divergence between an approximate distribution and the posterior of a deep GP.

Dropout basically serves as a regularization technique within the training process to reduce over-fitting in NNs. For the testing samples, the dropout is not applied, but weights are adjusted, e.g., multiplied by "1 − dropout ratio". With regards to MC dropout, the dropout is applied at both training and test time. So, the prediction is no longer deterministic at test time.

Given that $\hat{y}$ is an output of a NN model with hidden layers L. Also, $w = \{W_1, \ldots, W_L\}$ represents the NN's weight matrices, and y^* is the observed output corresponding to input x^*. By defining $X = \{x_1, \ldots, x_N\}$ and $Y = \{y_1, \ldots, y_N\}$ as the input and output sets, the predictive distribution is expressed as:

$$
p(y^* \mid x^*, X, Y) = \int p(y^* \mid x^*, w) p(w \mid X, Y) dw, \tag{7.15}
$$

here, $p(y^* \mid x^*, w)$ and $p(w \mid X, Y)$ are the NN model's likelihood and the posterior over the weights.

The predictive mean and variance are used in the predictive distribution to estimate uncertainty. The posterior distribution is, however, analytically intractable. As a replacement, an approximation of variational distribution $q(w)$ can be obtained from the GP such that it is as close to $p(w \mid X, Y)$ as possible, in which the optimization process happens through the minimization of the KL divergence between the preceded distributions as shown below:

$$
KL(q(w) \mid p(w \mid X, Y)). \tag{7.16}
$$

With variational inference, the predictive distribution can be described as follows:

$$
q(y^* \mid x^*) = \int p(y^* \mid x^*, w) q(w) dw. \tag{7.17}
$$

According to [103], $q(w)$ is selected to be the matrices distribution whose columns are randomly set to zero given a Bernoulli distribution, specified as:

$$W_i = M_i.\text{diag}([z_{i,j}]_{j=1}^{K_i}), \tag{7.18}$$

where $z_{i,j} \sim \text{Bernoulli}(p_i)$ for $i = 1, \ldots, L$ and $j = 1, \ldots, K_{i-1}$ with $K_i \times K_{i-1}$ as the dimension of matrix W_i. Also, p_i represents the probability of dropout and M_i is a matrix of variational parameters. Therefore, drawing T sets of vectors of samples from Bernoulli distribution gives $(W_1^t, \ldots, W_L^t)_{t=1}^T$, and consequently, the predictive mean will be:

$$E_{q(y^*|x^*)(y^*)} \approx \frac{1}{T} \sum_{t=1}^{T} \hat{y}^*(x^*, W_1^t, \ldots, W_L^t) = p_{MC}(y^* \mid x^*), \tag{7.19}$$

where $\hat{y}^*$ is the output obtained by the given NN for input x^*, and p_{MC} is the predictive mean of MC dropout, equivalent to doing T stochastic forward passes over the network during the testing process with dropout and then averaging the results.

It is useful to view this method as an ensemble of approximated functions with shared parameters, which approximates the probabilistic Bayesian method known as deep GP. In this method, there are several outputs (considered 30, 50, 100, and 200 in this research) for a given input. Subsequently, uncertainty could be examined in terms of factors such as variance, entropy, and mutual information.

In the following, four examples are given to show the application of MC dropout in modeling subsurface fluid flow. The researchers in [104] investigated the uncertainty involved in ML seismic image segmentation models. Salt body detection was considered as an example. They used MC dropout and concluded the developed models were reliable.

The researchers in [105] used the dropout method for a classification problem to quantify the fault model uncertainty of a reservoir in the Netherlands. The networks were trained with dropout ratios of 0.25 and 0.5. The researchers concluded that the model variance increased by increasing the dropout ratio. Also, they suggested training with more data is needed.

The MC dropout approach and a bootstrap aggregating method were used to quantify uncertainties of CO_2 saturation based on seismic data in [106]. The researchers carried out DL inversion experiments using noise-free and noisy data. The results showed that the model can estimate 2D distributions of CO_2 moderately well, and UQ can be done in real-time.

A semi-supervised learning workflow was used to effectively integrate seismic data and well logs and simultaneously predict subsurface characteristics in [107]. It had three distinct benefits: (i) using 3D seismic patterns for developing an optimal nonlinear mapping function with 1D logs, (ii) being capable of automatically filling the gap of vertical resolution between seismic and well logs, and (iii) having an MC dropout-based epistemic uncertainty analysis. The results of the two examples showed robust estimation of properties like density and porosity obtained by this procedure.

7.3 DualSkipCNN Model with Dropout

As mentioned earlier, DualSkipCNN model consists of five convolutional layers and two FC layers but does not include any pooling layers. Each convolutional layer is followed by a single BN layer of the same dimensions. Furthermore, there are two shortcut schemes in this structure: (i) from the middle to the last layer and (ii) from the middle to the second-to-last layer. The two FC layers contain 2000 neurons, to which a dropout rate of 0.05 is added to investigate the reliability. The model employs the activation function of ReLU for the convolutional layers, "sigmoid" for the FC layers, and "linear" for the output (Figure 7.1).

When treated as a Bayesian approach, the model produces a different output each time it is called with the same input. This is because each time a new set of weights is sampled from the distributions to develop the network and produce an output. After examining various cases, it was discovered that 30 outputs could be the ideal case for a given input, representing the most efficient and effective solution.

7.4 Summary

This chapter focused on Bayesian methods to take a positive step toward the reliability of DNNs of low complexity. BNNs are different from standard NNs in that they assign probability distributions to their weights, rather than a single value or point estimate. These probability distributions describe the uncertainty in weights and can be used to estimate uncertainty in predictions. Taking advantage of its computational efficiency, MC dropout was used to quantify the epistemic uncertainty of the DualSkipCNN model. A dropout ratio of 0.05 was used for the FC layers, and the analysis was performed separately for each of the multiscale basis functions (Basis 2, 3, 4, and 5). The suitability of FNO is investigated to directly predict pressure distribution using the permeability field as input in the subsequent chapter.

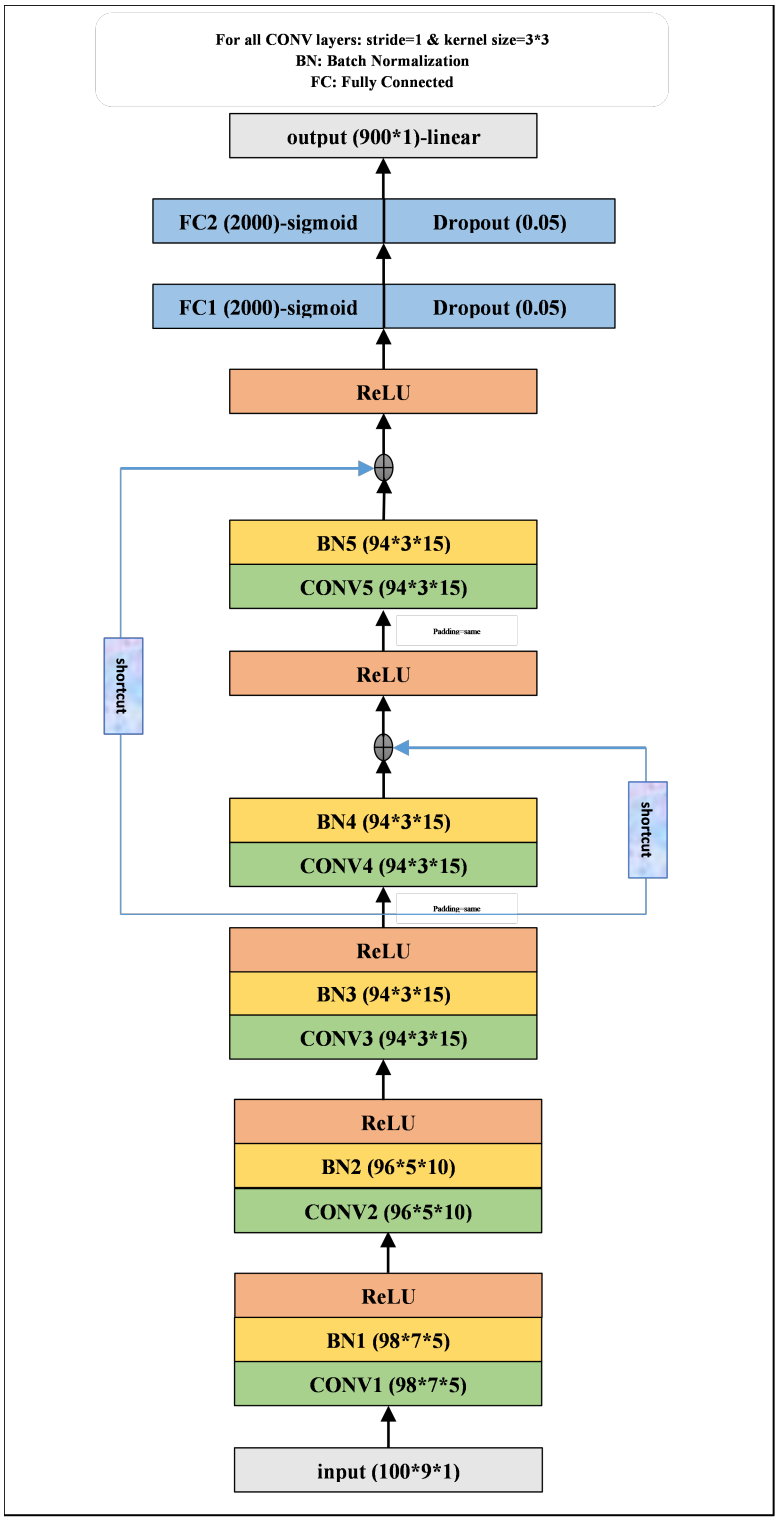


Figure 7.1: Structure of the DualSkipCNN model with dropout developed in this study.

Chapter 8

Fourier Neural Operator for Pressure

8.1 Introduction

There are different numerical methods such as FEM that can provide approximate solutions to problems related to fluid flow in porous media. Such methods are usually time-consuming to apply. One way to mitigate the problem of the high computational cost in numerical calculations is to apply ML/DL techniques including NNs.

Classical NNs focus on learning mappings between finite-dimensional spaces. This makes such networks, when configured, confined to a particular discretization (i.e., they are mesh-dependent). Mesh-independent networks have been developed to reduce such constraints. In this regard, FNO has recently been proposed to learn a continuous function via parameterizing the model in its function space [108]. This makes it possible for FNO to be trained on one mesh and subsequently evaluated on another. Unlike standard feed-forward networks that use activation functions (e.g., sigmoid and tanh), training an FNO model using the Fourier transform to find optimum weights and biases is performed by employing sines and cosines as activation functions [109, 110, 111].

Some research has been conducted applying FNO models to various existing datasets [112, 113, 114]. However, much of that research addresses problems involving big-shape data (e.g., 421×421 and 256×256). Thus, there is a critical lack of analysis regarding the performance of FNO models on small-shape data. Consequently, this chapter tries to investigate the fifth subsidiary question introduced in Chapter 1:

5. Can FNO models accurately perform on small-shape data problems to predict pressure distribution?

A 30×30 uniform mesh problem is considered to address this research question. In fact, FNO is applied to predict pressure distribution using 1700 FEM generated samples, from heterogeneous permeability fields as the input.

The remaining sections of this chapter are arranged as follows. Section 8.2 provides a mathematical explanation of FNO using different assumptions, definitions, and function approximations. Additionally, two FNO architectures developed for pressure prediction are presented. Section 8.3 describes the configurations of the CNN model based on the same

dataset. The chapter concludes with a summary in Section 8.4.

8.2 FNO: Mathematical Foundations and Architecture Development

The FNO methodology learns a mapping between two infinite dimensional spaces using a limited amount of input-output data. It is supposed that $D \subset \mathbb{R}^d$ is a bounded and open set, and $\mathcal{A} = \mathcal{A}\left(D; \mathbb{R}^{d_a}\right)$ and $\mathcal{U} = \mathcal{U}\left(D; \mathbb{R}^{d_u}\right)$ represent the separable Banach spaces of function that take values in $\mathbb{R}^{d_a}$ and $\mathbb{R}^{d_u}$ respectively. In addition, $G^\dagger : \mathcal{A} \to \mathcal{U}$ is supposed to be a non-linear map. Here, maps $G^\dagger$ are studied which arise as the solution operators of parametric PDEs. Supposing there are samples $\{a_j, u_j\}_{j=1}^N$ where $a_j \sim \mu$ is an i.i.d. sequence from the probability measure μ supported on $\mathcal{A}$ and $u_j = G^\dagger(a_j)$ may be corrupted with noise. The purpose is to approximate $G^\dagger$ by developing a parametric map for some finite-dimensional parameter space Θ by selecting $\theta^\dagger \in \Theta$ so that $G\left(\cdot, \theta^\dagger\right) = G_{\theta^\dagger} \approx G^\dagger$:

$$G : \mathcal{A} \times \Theta \to \mathcal{U} \quad \text{or equivalently,} \quad G_\theta : \mathcal{A} \to \mathcal{U}, \quad \theta \in \Theta. \tag{8.1}$$

This is a natural framework for learning in infinite dimensions in which a cost functional $C : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ can be defined. Then, it is possible to minimize the problem:

$$\min_{\theta \in \Theta} \mathbb{E}_{a \sim \mu}\left[C\left(G(a, \theta), G^\dagger(a)\right)\right],$$

which directly parallels the classical finite-dimensional setting [115]. However, finding minimizers in the infinite-dimensional context is still a difficult issue that needs to be addressed. A solution is to adopt a test-train setting, where a data-driven empirical approximation is used for the cost function that determines the optimal parameter θ. The accuracy of this approximation is then verified. This methodology is conceptualized in the infinite-dimensional setting, ensuring all finite-dimensional approximations have the same set of parameters that are consistent with infinite dimensions.

Approximating the operator $G^\dagger$ is generally more challenging than finding a solution $u \in \mathcal{U}$ of a PDE for a single instance of the parameter $a \in \mathcal{A}$. Traditional techniques, such as classical finite elements, finite differences, and finite volumes, as well as modern ML methods including Physics-Informed Neural Networks (PINNs) [116], primarily focus on the latter task, which can be computationally expensive. Therefore, these methods may not be practical for scenarios that involve finding a PDE solution for many different instances of the parameter. In contrast, FNO directly approximates the operator itself, making it much faster and more cost effective than traditional solvers, resulting in substantial computational savings.

In practice, when working with data a_j and u_j, which are typically functions, numerical evaluations are used. To enable numerical computations, it is assumed that only point-wise evaluations of these functions are available. $D_j = \{x_1, \ldots, x_n\} \subset D$ is defined as a n-point discretization of the domain D and $a_j|_{D_j} \in \mathbb{R}^{n \times d_a}$, $u_j|_{D_j} \in \mathbb{R}^{n \times d_v}$ are observations for a

finite set of input-output pairs indexed by j. It is crucial for the neural operator to be discretization-invariant, enabling it to produce an output $u(x)$ for any $x \in D$, even if x is not part of the specific discretization D_j. This is significant because it facilitates the transfer of solutions between different grid geometries and discretizations.

In the neural operator framework proposed in [117], the architecture is designed as an iterative process $v_0 \mapsto v_1 \mapsto \ldots \mapsto v_T$ where v_j for $j = 0, 1, \ldots, T-1$ is a sequence of functions, each taking values in $\mathbb{R}^{d_v}$. As illustrated in Figure 8.1 a, the process starts by lifting the input $a \in \mathcal{A}$ to a higher dimensional representation $v_0(x) = P(a(x))$ using a local transformation P, typically parameterized by a shallow FC neural network. After that, several iterations of updates $v_t \mapsto v_{t+1}$ (defined below) are performed. The output $u(x) = Q\left(v_T(x)\right)$ is obtained by projecting v_T by the local transformation $Q : \mathbb{R}^{d_v} \to \mathbb{R}^{d_u}$. In each iteration, the update $v_t \mapsto v_{t+1}$ is defined as the composition of a non-local integral operator $\mathcal{K}$ and a local, nonlinear activation function σ. The difference between the architectures in Figure 8.1 b and c is the existence of an MLP after inverse Fourier transform (F^{-1}). MLP can be implemented using a CNN with a 1×1 kernel size. When used this way, the CNN conducts pointwise multiplication on the channel dimension, functioning similarly to an MLP.

The update to the representation $v_t \mapsto v_{t+1}$ is defined by

$$v_{t+1}(x) := \sigma\left(W v_t(x) + \left(\mathcal{K}(a;\phi)v_t\right)(x)\right), \quad \forall x \in D, \tag{8.2}$$

where $\mathcal{K} : \mathcal{A} \times \Theta_{\mathcal{K}} \to \mathcal{L}\left(\mathcal{U}\left(D;\mathbb{R}^{d_v}\right), \mathcal{U}\left(D;\mathbb{R}^{d_v}\right)\right)$ maps to bounded linear operators on $\mathcal{U}\left(D;\mathbb{R}^{d_v}\right)$ and is parameterized by $\phi \in \Theta_{\mathcal{K}}, W : \mathbb{R}^{d_v} \to \mathbb{R}^{d_v}$ is a linear transformation, and $\sigma : \mathbb{R} \to \mathbb{R}$ is a non-linear activation function whose action is defined component-wise.

$\mathcal{K}(a;\phi)$ is chosen to be a kernel integral transformation parameterized by a neural network.

The kernel integral operator ($\mathcal{K}$) mapping in 8.2 is defined by

$$\left(\mathcal{K}(a;\phi)v_t\right)(x) := \int_D \kappa(x, y, a(x), a(y);\phi)v_t(y)\mathrm{d}y, \quad \forall x \in D, \tag{8.3}$$

where $\kappa_\phi : \mathbb{R}^{2(d+d_a)} \to \mathbb{R}^{d_v \times d_v}$ is a neural network parameterized by $\phi \in \Theta_{\mathcal{K}}$.

Indeed, κ_ϕ serves as a kernel function that is learned from data. The definitions of 8.2 and 8.3 represent a generalization of NNs to infinite-dimensional spaces, which was initially introduced in [117]. Despite the linearity of the integral operator, the neural operator can learn highly non-linear operators by composing linear integral operators with non-linear activation functions, similar to standard NNs.

By removing the dependency on the function a and imposing $\kappa_\phi(x, y) = \kappa_\phi(x - y)$, it becomes a convolution operator, which is a natural choice based on fundamental solutions. This characteristic is exploited by parameterizing κ_ϕ directly in Fourier space and utilizing the Fast Fourier Transform (FFT) to efficiently compute 8.3. This results in a fast architecture that produces superior results for PDE problems.

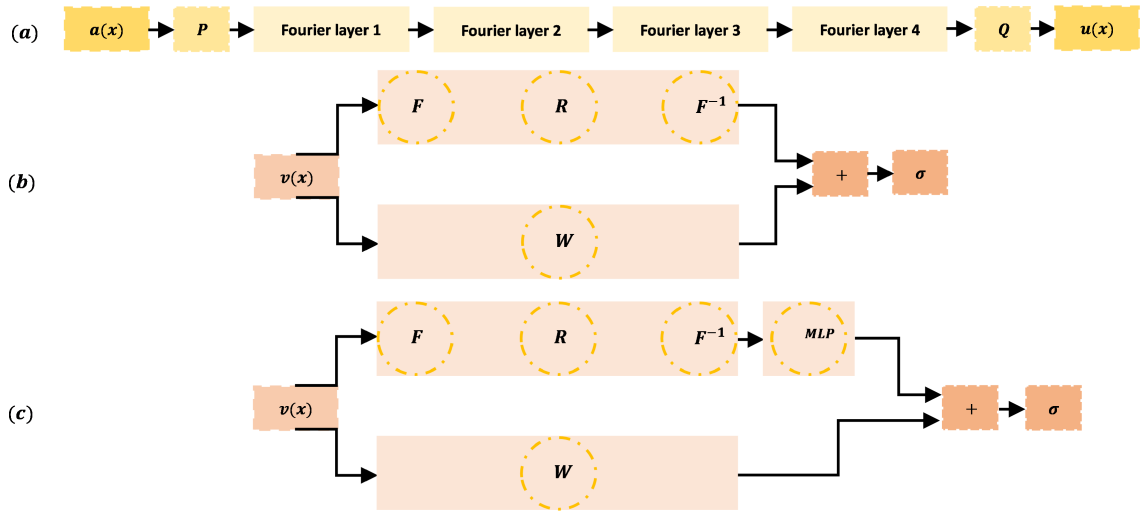


Figure 8.1: (a) Architecture of the neural operator: the input $a(x)$ is first lifted to a higher-dimension channel space $v_0(x)$, where $v_0(x) = P(a(x))$. It does this by locally applying the transform P. Then, four successive Fourier layers are applied to v_0. Subsequently, another transform is applied locally Q. This final transform projects $v_4(x)$ to the output by $u(x) = Q(v_4(x))$., (b) Architecture of a Fourier layer: $v(x)$ passes through two routes in the Fourier layers. In the top path, a Fourier transform F, a linear transform R on the lower Fourier modes, and an F^{-1} are applied. $v(x)$ undergoes only a local linear transform W in the bottom path. Outputs of each path are added together and then subjected to an activation function., and (c) Architecture of a Fourier layer with an MLP after the inverse FFT.

A proposed approach involves replacing the kernel integral operator in 8.3 with a convolution operator defined in Fourier space. The Fourier transform of a function $f : D \to \mathbb{R}^{d_v}$ is represented as $\mathcal{F}$, and its inverse as $\mathcal{F}^{-1}$. Then, the below is defined:

$$(\mathcal{F}f)_j(k) = \int_D f_j(x) e^{-2i\pi\langle x,k\rangle} \mathrm{d}x, \quad \left(\mathcal{F}^{-1}f\right)_j(x) = \int_D f_j(k) e^{2i\pi\langle x,k\rangle} \mathrm{d}k,$$

for $j = 1, \ldots, d_v$ where $i = \sqrt{-1}$ is the imaginary unit. By putting $\kappa_\phi(x, y, a(x), a(y)) = \kappa_\phi(x - y)$ in 8.3 and utilizing the convolution theorem, it can be inferred that:

$$\left(\mathcal{K}(a;\phi)v_t\right)(x) = \mathcal{F}^{-1}\left(\mathcal{F}\left(\kappa_\phi\right) \cdot \mathcal{F}\left(v_t\right)\right)(x), \quad \forall x \in D,$$

it would be preferable to directly parameterize κ_ϕ in Fourier space.

The Fourier integral operator $(\mathcal{K})$ is defined as:

$$\left(\mathcal{K}(\phi)v_t\right)(x) = \mathcal{F}^{-1}\left(R_\phi \cdot (\mathcal{F}v_t)\right)(x) \quad \forall x \in D, \tag{8.4}$$

where R_ϕ is the Fourier transform of a periodic function $\kappa : \bar{D} \to \mathbb{R}^{d_v \times d_v}$ parameterized by $\phi \in \Theta_{\mathcal{K}}$. A visual representation can be seen in Figure 8.1 (b).

For a given frequency mode $k \in D$, there are $(\mathcal{F}v_t)(k) \in \mathbb{C}^{d_v}$ and $R_\phi(k) \in \mathbb{C}^{d_v \times d_v}$. Given the periodicity of κ, it can be represented using a Fourier series expansion, working with the discrete modes $k \in \mathbb{Z}^d$. A finite-dimensional parameterization is picked by truncating the Fourier series at a maximal number of modes $k_{\max} = |Z_{k_{\max}}| =| \left\{k \in \mathbb{Z}^d : |k_j| \leq k_{\max,j}\right., \text{ for } \left. j = 1, \ldots, d\right\} |$. This allows directly parameterizing R_ϕ as a complex-valued $(k_{\max} \times d_v \times d_v)$-tensor comprising a set of truncated Fourier modes. Therefore, ϕ is removed from notations. Since κ is real-valued, conjugate symmetry can be imposed. It is worth noting that the set $Z_{k_{\max}}$ is not the canonical choice for the low frequency modes of v_t. Typically, the low frequency modes are defined by placing an upper-bound on the ℓ_1-norm of $k \in \mathbb{Z}^d$. Here, $Z_{k_{\max}}$ is selected as mentioned above because it facilitates an efficient implementation.

Assuming that the domain D is discretized with $n \in \mathbb{N}$ points, there are $v_t \in \mathbb{R}^{n \times d_v}$ and $\mathcal{F}(v_t) \in \mathbb{C}^{n \times d_v}$. Since v_t is convolved with a function that only has $k_{\max}$ Fourier modes, it is possible to simply truncate the higher modes and obtain $\mathcal{F}(v_t) \in \mathbb{C}^{k_{\max} \times d_v}$. Subsequently, the multiplication by the weight tensor $R \in \mathbb{C}^{k_{\max} \times d_v \times d_v}$ can be expressed as:

$$(R \cdot (\mathcal{F}v_t))_{k,l} = \sum_{j=1}^{d_v} R_{k,l,j} (\mathcal{F}v_t)_{k,j}, \quad k = 1, \ldots, k_{\max}, \quad j = 1, \ldots, d_v. \tag{8.5}$$

If the discretization is uniform with resolution $s_1 \times \cdots \times s_d = n$, FFT can replace $\mathcal{F}$. For $f \in \mathbb{R}^{n \times d_v}, k = (k_1, \ldots, k_d) \in \mathbb{Z}_{s_1} \times \cdots \times \mathbb{Z}_{s_d}$, and $x = (x_1, \ldots, x_d) \in D$, the FFT $\hat{\mathcal{F}}$ and its inverse $\hat{\mathcal{F}}^{-1}$ can be defined as:

$$(\hat{\mathcal{F}}f)_l(k) = \sum_{x_1=0}^{s_1-1} \cdots \sum_{x_d=0}^{s_d-1} f_l(x_1, \ldots, x_d) e^{-2i\pi \sum_{j=1}^{d} \frac{x_j k_j}{s_j}},$$

$$\left(\hat{\mathcal{F}}^{-1}f\right)_l(x) = \sum_{k_1=0}^{s_1-1} \cdots \sum_{k_d=0}^{s_d-1} f_l(k_1, \ldots, k_d) e^{2i\pi \sum_{j=1}^{d} \frac{x_j k_j}{s_j}},$$

for $l = 1, \ldots, d_v$. In this case, the set of truncated modes becomes

$$Z_{k_{\max}} = \{(k_1, \ldots, k_d) \in \mathbb{Z}_{s_1} \times \cdots \times \mathbb{Z}_{s_d} \mid k_j \leq k_{\max,j} \text{ or } s_j - k_j \leq k_{\max,j}, \text{ for } j = 1, \ldots, d\}.$$

During implementation, R is considered as a $(s_1 \times \cdots \times s_d \times d_v \times d_v)$-tensor. The above definition of $Z_{k_{\max}}$ corresponds to the corners of R, enabling convenient parallel implementation of 8.5 through matrix-vector multiplication. Empirically, it was determined that setting $k_{\max,j} = 12$ (resulting in $k_{\max} = 12^d$ parameters per channel) proved to be adequate for all the examined tasks.

Generally, R can be defined based on $(\mathcal{F}a)$ in a way that parallels 8.3. Specifically, $R_\phi : \mathbb{Z}^d \times \mathbb{R}^{d_v} \to \mathbb{R}^{d_v \times d_v}$ can be defined as a parametric function that maps $(k, (\mathcal{F}a)(k))$ to the corresponding values of the relevant Fourier modes. When exploring both linear and

neural network parameterizations of R_ϕ, it was observed that the linear parameterization yielded similar performance to the direct parameterization described earlier, whereas the NNs performed less effectively. This outcome can likely be attributed to the discrete structure of the space $\mathbb{Z}^d$.

The Fourier layers are discretization-invariant because they can effectively learn from and evaluate functions that are discretized in any manner. By learning parameters directly in Fourier space, the process of resolving functions in physical space is a projection onto the basis $e^{2\pi i\langle x,k\rangle}$ which remains well-defined throughout $\mathbb{R}^d$. This characteristic enables zero-shot super-resolution. Additionally, the architecture had a consistent error regardless of the resolution of the inputs and outputs.

The weight tensor R contains $k_{\max} < n$ modes, thus the inner multiplication has complexity $O(k_{\max})$. Therefore, the majority of the computational cost comes from computing the Fourier transform $\mathcal{F}(v_t)$ and its inverse. While general Fourier transforms have complexity $O(n^2)$, the truncation of the series reduces the complexity to $O(nk_{\max})$. The FFT provides an efficient alternative with a complexity $O(n \log n)$ using a uniform discretization.

8.3 CNN-based Model for Pressure Distribution

In preparing a CNN simulation involving a unit square, a 30×30 uniform mesh was selected. On the other hand, the input/output values were defined as a 900×1 1D tensor (vector). The input shape was then changed to $30 \times 30 \times 1$ for processing through 2D convolutional filters. Regarding to the output in CNN, there were two options: keeping the initial shape or reshaping to a 2D tensor. While reshaping to 30×30, the accuracy achieved by the CNN model became substantially impaired. On the other side, many fewer errors were generated by CNN models that retained the initial 900×1 shape. Therefore, the CNN model was developed and its computational layers were processed with the 900×1 shape and only reshaped to the 30×30 output size for final visualization purposes.

A standard CNN architecture was developed with five convolutional layers and two FC layers (Figure 8.2). The kernel numbers in the convolutional layers (referred to as CONV1 to CONV5) were 5, 45, 85, 125, and 165, respectively. Padding was set to "same" only in CONV5 to prevent the size from changing. A 3×3 kernel size and 1×1 stride were applied to all convolutional layers, providing those layers with sizes 28×28, 26×26, 24×24, 22×22, and 22×22, respectively. A BN layer (referred to as BN1 to BN5) followed CONV1 to CONV5, without changing size. Normalization of the input layer makes the CNN converge more quickly to outputs. The layers FC1 and FC2 contain 1500 neurons. The ReLU activation function was applied to CONV1 to CONV5, whereas the sigmoid activation function was applied to FC1 and FC2, with a linear transformation applied to generate the output layer. The CNN was trained to apply a batch size of 16 samples and run with 500 epochs.

8.4 Summary

Classical NNs attempt to learn mappings between finite-dimensional Euclidean spaces, which confines them to a particular discretization. On the other hand, FNO is a mesh-independent algorithm that tries to learn function-to-function mappings. This makes it possible for FNO to be trained on one mesh and subsequently assessed on another. The purpose of this chapter was to apply FNO to predict pressure distribution over a small, specified shape-data problem using 1700 FEM generated samples, from heterogeneous permeability fields as the input. Specifically, 1400 sample grids were generated to constitute the training dataset, and a further 300 sample grids to constitute the testing dataset. A CNN model, as a classical NN, was also developed based on the same dataset. The next chapter, the penultimate chapter of this thesis, presents the comparative evaluation of the models proposed using different techniques.

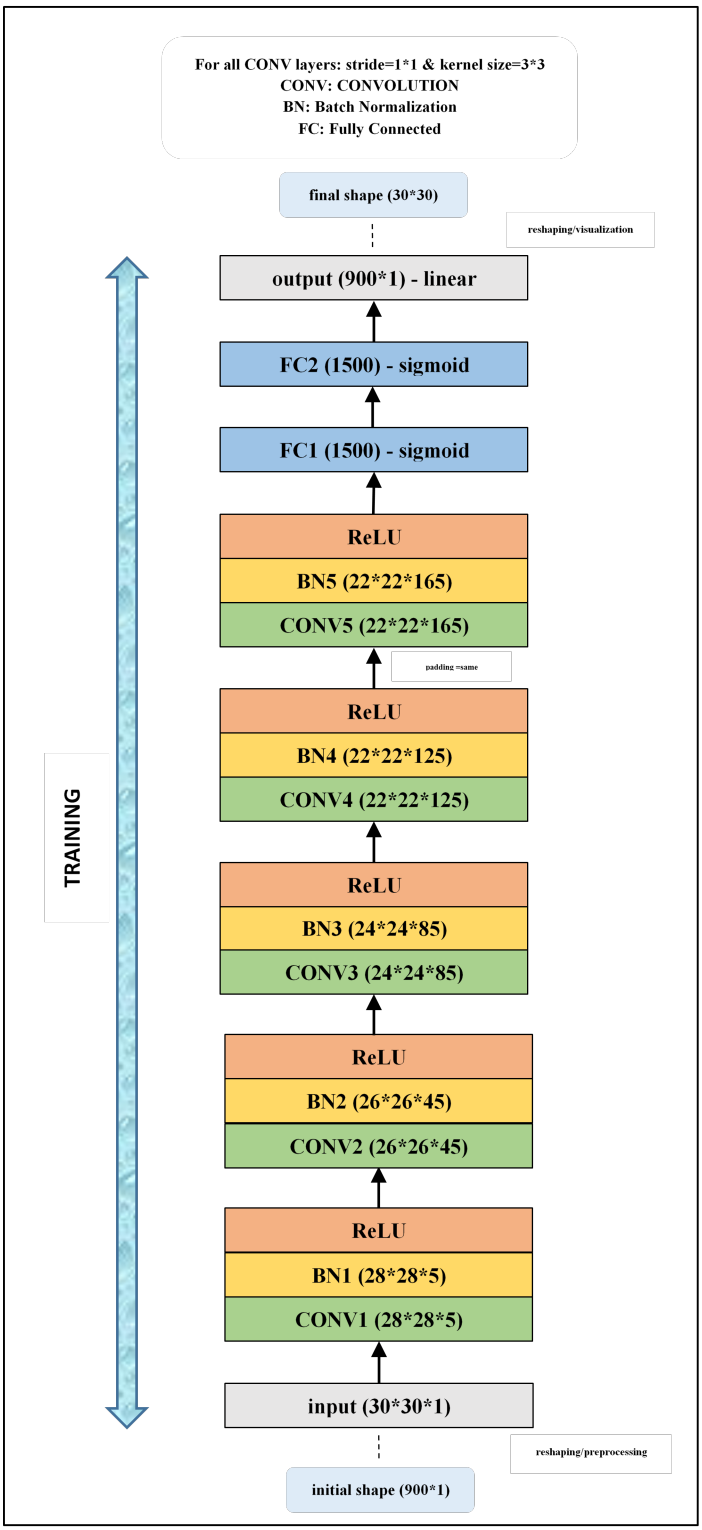


Figure 8.2: Structure of the CNN model developed for pressure prediction.

Chapter 9

Comparative Evaluation

9.1 Introduction

In Chapter 4, a DNN of low complexity called SkiplessCNN was developed to predict Basis 2 to 5 in the mixed GMsFEM. This network consisted of five convolutional layers and two FC layers. The number of kernels in CONV1 to CONV5 was 5, 10, 15, 20, and 25, respectively. Each FC layer contained 2000 neurons.

The purpose of Chapter 5 was to investigate the effect of skip connections on the performance of DNNs of low complexity. In this sense, three different skip connection schemes named FirstSkip, MidSkip, and DualSkip were added to SkiplessCNN, subsequently developing three skip connection-based models named FirstSkipCNN, MidSkipCNN, and DualSkipCNN.

The potential prediction improvements achieved by using DEL to estimate the multiscale basis functions were investigated in Chapter 6. Specifically, four models - SkiplessCNN, FirstSkipCNN, MidSkipCNN, and DualSkipCNN - were combined into an ensemble model using linear regression and ridge regression, separately.

The focus of Chapter 7 was on the reliability of the DualSkipCNN model. In this sense, MC dropout, a computationally efficient approach, was applied with a focus on epistemic uncertainty.

Two FNO-based models were developed to predict pressure distribution over a small, specified shape-data problem in Chapter 8. Furthermore, a new CNN model was developed.

To evaluate the accuracy of the developed models, two statistical error metrics were employed:

$$\mathrm{R}^2 = 1 - \frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{N}(\hat{y}_i - \bar{y})^2}, \tag{9.1}$$

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2, \tag{9.2}$$

where y_i, $\bar{y}$, and $\hat{y}_i$ are the actual basis function of the ith data point, the average of actual basis function for all samples, and the predicted basis function for the ith data point, respectively. Also, N is the number of data points.

As mentioned earlier, each basis function is in the form of a 900×1 1D tensor, and R^2 of all outputs are averaged, weighted by the variances of each individual output. The R^2 value lies between $-\infty$ and 1. The closer this value is to 1, the more precise predictions a model yields. Conversely, a negative or small positive R^2 value indicates that the wrong model is chosen. MSE measures the average of squares of errors (i.e., the difference between predicted and real values). It is always non-negative and values close to zero indicate a more accurate performance.

Depending on the input/output dimensions, type (classification/regression), and algorithm/methodology employed for addressing a problem, the magnitude of uncertainty can be analyzed statistically and graphically. SD measures the dispersion of a dataset relative to its average. It is the square root of the variance. The closer the value of SD is to zero, the values of the data are closer to the average. A high SD indicates that the values are spread out over a broad range. Basically, the variance and SD are defined for a single-point dataset (there is only one output). On the other hand, the output (basis functions) in this study is in the form of a 900×1 vector. While dealing with a vector, it is necessary to calculate the variance of each element of the vector separately. Then, the obtained variances are averaged to reach the total variance. Finally, the SD is obtained as the square root of the variance for each case. Standard CNNs produce a single output per input, and therefore, the SD is not defined for such models (it is always zero).

In terms of graphical investigation, the pattern available in a coarse block was tracked for the multiscale basis functions. Regarding uncertainty, the dispersion of outputs obtained by the MC dropout was illustrated over a coarse grid. The pressure changes were visualized over the entire porous medium for the FNO models.

Following the introduction, Section 9.2 presents a performance comparison of two optimizers, Adam and AMSGrad, for Basis 2 to 5 prediction based on the initial dataset. The impact of dataset size (extended dataset versus initial dataset) on AMSGrad-based models' performance is discussed in Section 9.3. Section 9.4 is dedicated to investigating the impact of three skip connection schemes (i.e., FirstSkip, MidSkip, and DualSkip) on the Skipless-CNN performance. Moving on to Section 9.5, the performance of DEL-based models and CNN models based on the testing subset is analyzed. The reliability of DualSkipCNN using MC dropout is investigated in Section 9.6. Section 9.7 presents the performance results of the models developed for pressure distribution. The chapter concludes with a summary in Section 9.8.

9.2 Performance Comparison of the Adam and AMS-Grad Optimizers for Basis Functions Prediction based on the Initial Dataset

The performance of the constructed SkiplessCNN models by Adam in terms of R^2 and MSE for the training, validation, and testing subsets, and the total dataset is separately shown in Figures 9.1 and 9.2. According to Figure 9.1, the models developed for Basis 2, 3, and 4 have an R^2 of nearly 0.9. To be more specific, R^2 is 0.8945, 0.9017, and 0.9054 for the models developed for Basis 2, 3, and 4. Therefore, they are able to predict training samples very well. The model designed for Basis 5 has an acceptable performance but is not as good as the others. Furthermore, it reveals the superior performance of the CNN model for Basis 4, in validation and testing subsets in terms of the R^2 error parameter. The performance of the developed models based on the total dataset is the same as the training subset. This is because a large proportion of the data generated in the MatLab software was designed to train the model. Figure 9.2 also demonstrates that the MSE performance of the models created for Basis 4 and 5 are better than those for Basis 2 and 3. In this regard, the CNN model of Basis 5 is the best with an MSE of 0.0108, 0.0158, and 0.0154 for the training, validation, and testing data subsets.

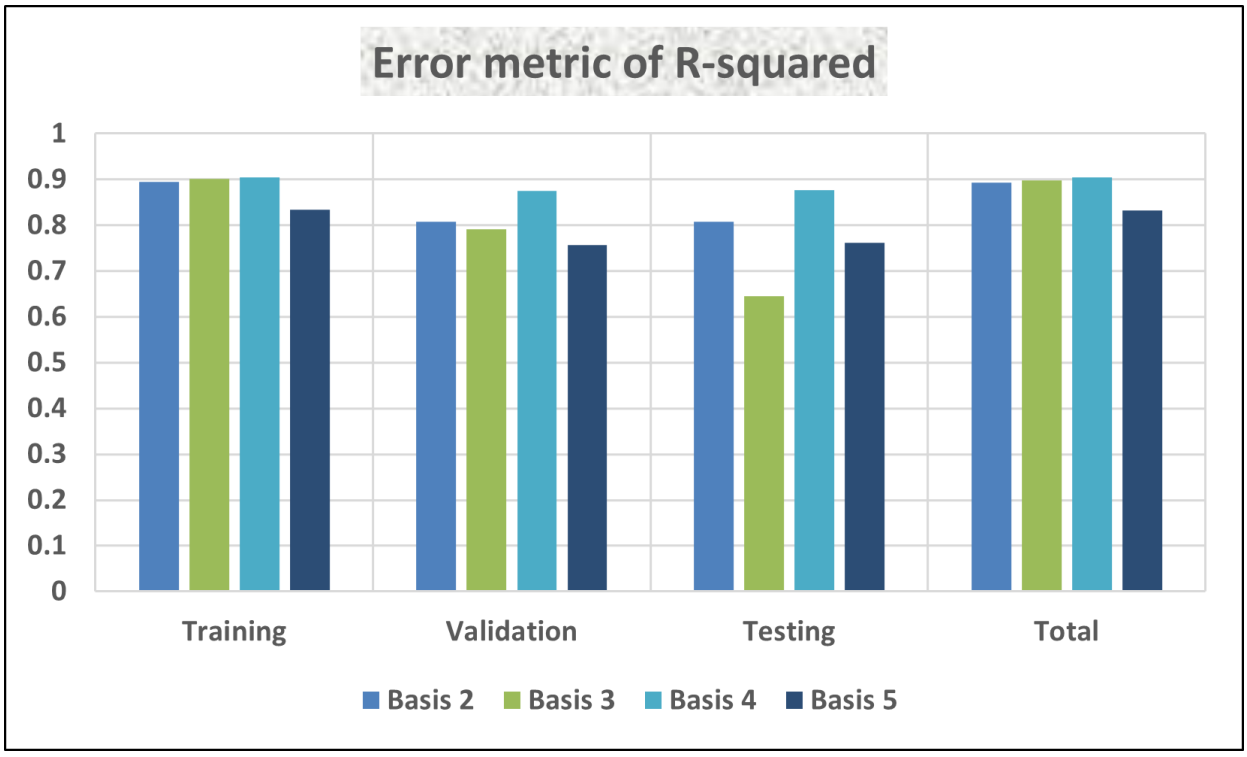


Figure 9.1: Performance of the models developed by the Adam algorithm on the initial dataset based on R^2.

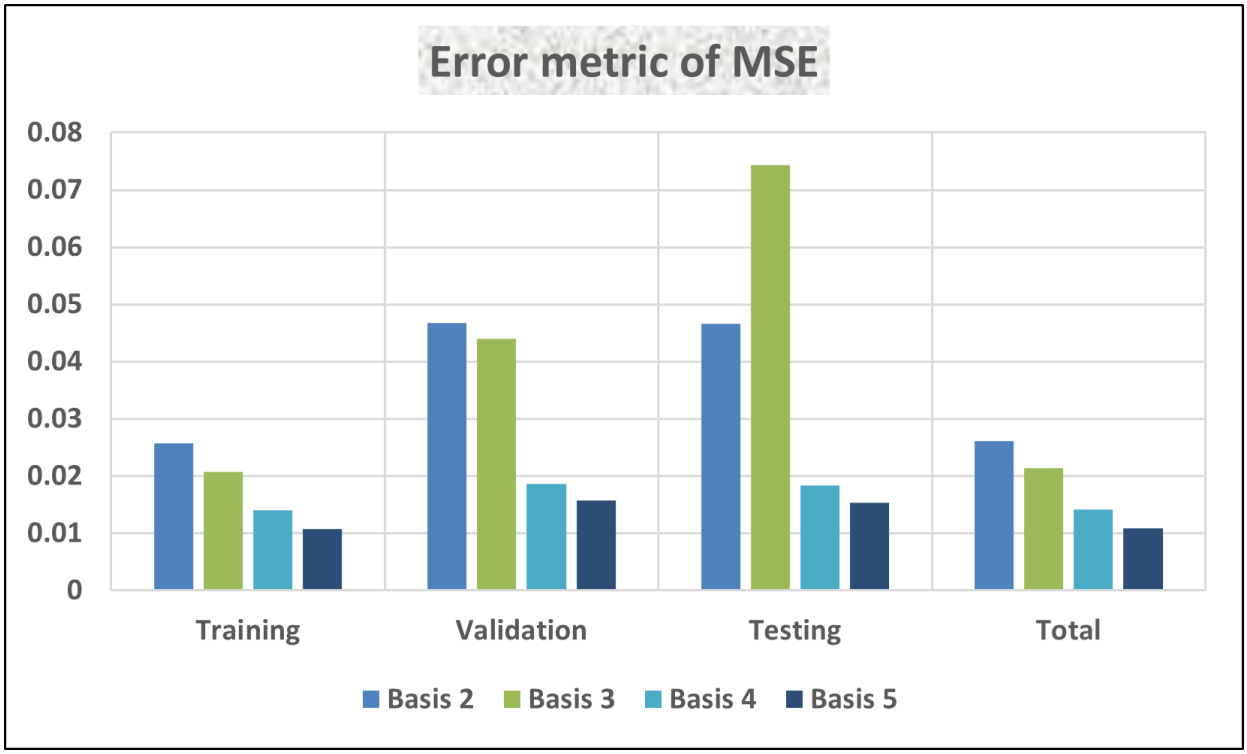


Figure 9.2: Performance of the models developed by the Adam algorithm on the initial dataset based on MSE.

Table 9.1 is given to investigate if replacing the Adam optimizer with AMSGrad affects the performance of the SkiplessCNN architecture. Comparing the R^2 parameter, it is clear that the performance of all models improves, especially for the training subset. For instance, R^2 increases from 0.8945 to 0.9079 for the Basis 2 model. The MSE error parameter decreases as expected when using the AMSGrad optimizer. Across the total data, MSE decreases from 0.0261, 0.0214, 0.0142, and 0.0109 to 0.0206, 0.0167, 0.0103, and 0.0078 for the Basis 2, 3, 4, and 5 models, respectively.

The permeability field consists of a 30×30 uniform mesh on a fine grid system. This is equivalent to a 10×10 uniform mesh on a coarse grid system, so each coarse grid contains nine fine grids. Multiscale basis functions, on the other hand, are defined in a single coarse grid element, as mentioned earlier. Consequently, the pattern available in a coarse block is tracked for the graphical investigation, with an unfractured case (Figure 9.3) and a fractured case (Figure 9.4), as representative samples. In the top section of the figures, fine grids in blue refer to the matrix and yellow to the fracture. In general, the patterns reconstructed by AMSGrad follow the observed trend in each coarse block slightly better than those of Adam.

Table 9.1: Performance comparison of Adam and AMSGrad in terms of R^2 and MSE based on the initial dataset.

Error statistic	CNN model	Training	Validation	Testing	Total
$\mathbf{R^2}$	Basis 2 (Adam)	0.8945	0.8076	0.8083	0.8929
	Basis 2 (AMSGrad)	0.9079	0.8141	0.8156	0.9062
	Basis 3 (Adam)	0.9017	0.7911	0.6445	0.8981
	Basis 3 (AMSGrad)	0.9142	0.7974	0.6503	0.9105
	Basis 4 (Adam)	0.9054	0.8751	0.8762	0.9049
	Basis 4 (AMSGrad)	0.9172	0.8777	0.8815	0.9165
	Basis 5 (Adam)	0.8341	0.7569	0.7625	0.8328
	Basis 5 (AMSGrad)	0.8449	0.7592	0.7671	0.8434
MSE	Basis 2 (Adam)	0.0257	0.0468	0.0466	0.0261
	Basis 2 (AMSGrad)	0.0202	0.0441	0.0445	0.0206
	Basis 3 (Adam)	0.0207	0.044	0.0743	0.0214
	Basis 3 (AMSGrad)	0.0159	0.0401	0.0697	0.0167
	Basis 4 (Adam)	0.0141	0.0186	0.0184	0.0142
	Basis 4 (AMSGrad)	0.0102	0.0158	0.0169	0.0103
	Basis 5 (Adam)	0.0108	0.0158	0.0154	0.0109
	Basis 5 (AMSGrad)	0.0077	0.0137	0.0141	0.0078

9.3 Impact of Dataset Size on the AMSGrad-based Models' Performance

The previous section revealed that the AMSGrad optimization algorithm performed slightly better than Adam. Its R^2 ranged from 0.8434 to 0.9165 and its MSE ranged from 0.0078 to 0.0206, whereas Adam had an R^2 of 0.8328–0.9049 and an MSE of 0.0109–0.0261. The results were obtained using an initial dataset of 238,347 examples for training, 1742 for validation, and 2612 for testing. To examine the impact of dataset size on the CNN models and increase the number of samples for validation and testing subsets, an extended dataset was generated consisting of 304,511 examples for training, 34,421 for validation, and 28,879 for testing, as explained earlier in Chapter 3.

According to Table 9.2, extending the dataset leads to a decrease in model performance for the training subset for Basis 2, 3, and 4. For example, the R^2 value for the Basis 3 model trained on the extended dataset (0.8952) is lower than that trained on the initial dataset (0.9142). However, except for Basis 2, the models trained on the extended dataset tend to perform better than those trained on the initial dataset for the validation subset. The most significant effect of extending the dataset is related to the testing subset of the Basis 3 model, where R^2 increases from 0.6503 to 0.8247, and MSE decreases from 0.0697 to 0.0369. Overall, these findings indicate the importance of having a sufficient amount of data for the validation and testing subsets, which can help decrease overfitting.

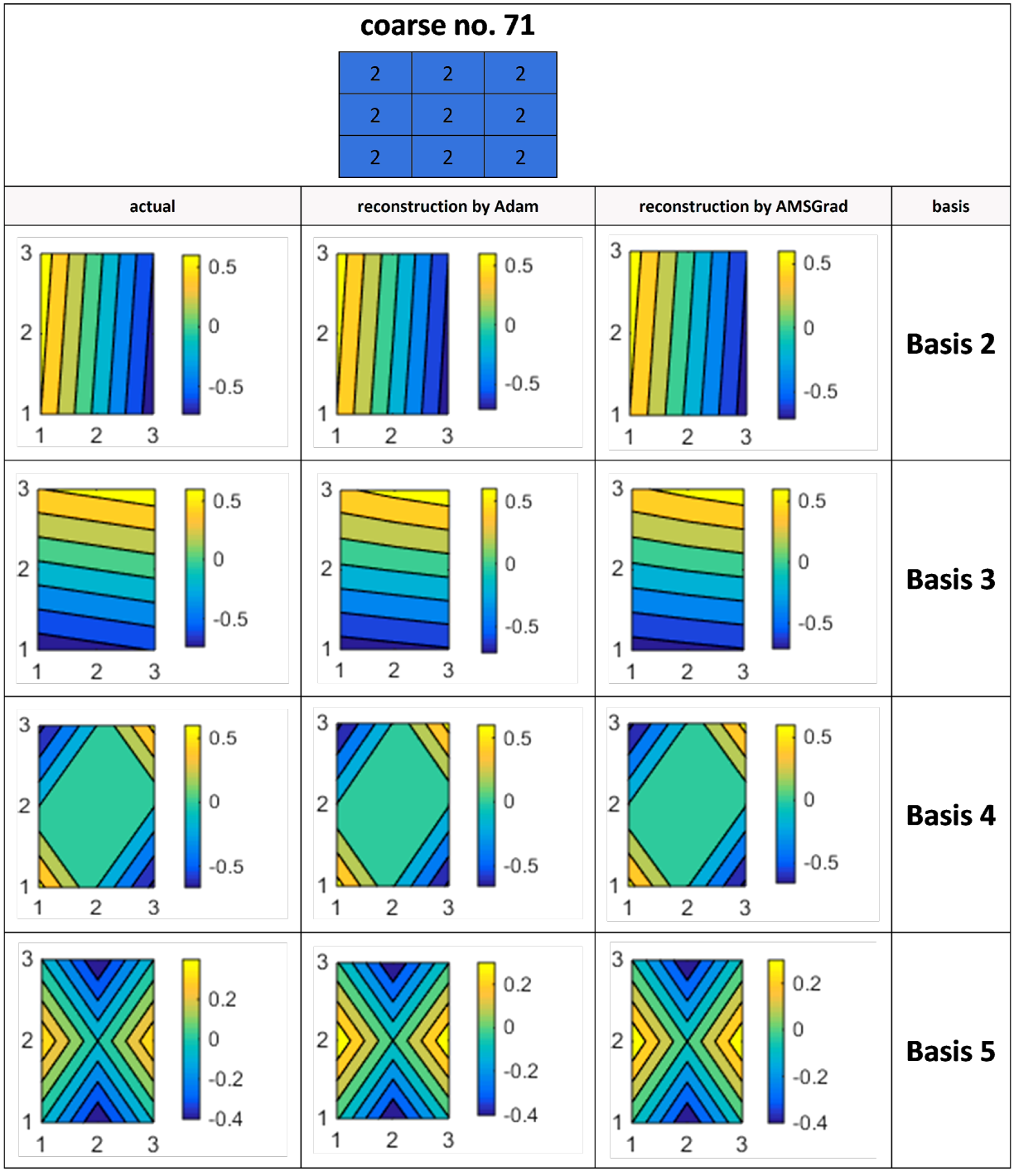


Figure 9.3: Comparing actual and reconstructed patterns in an unfractured case.

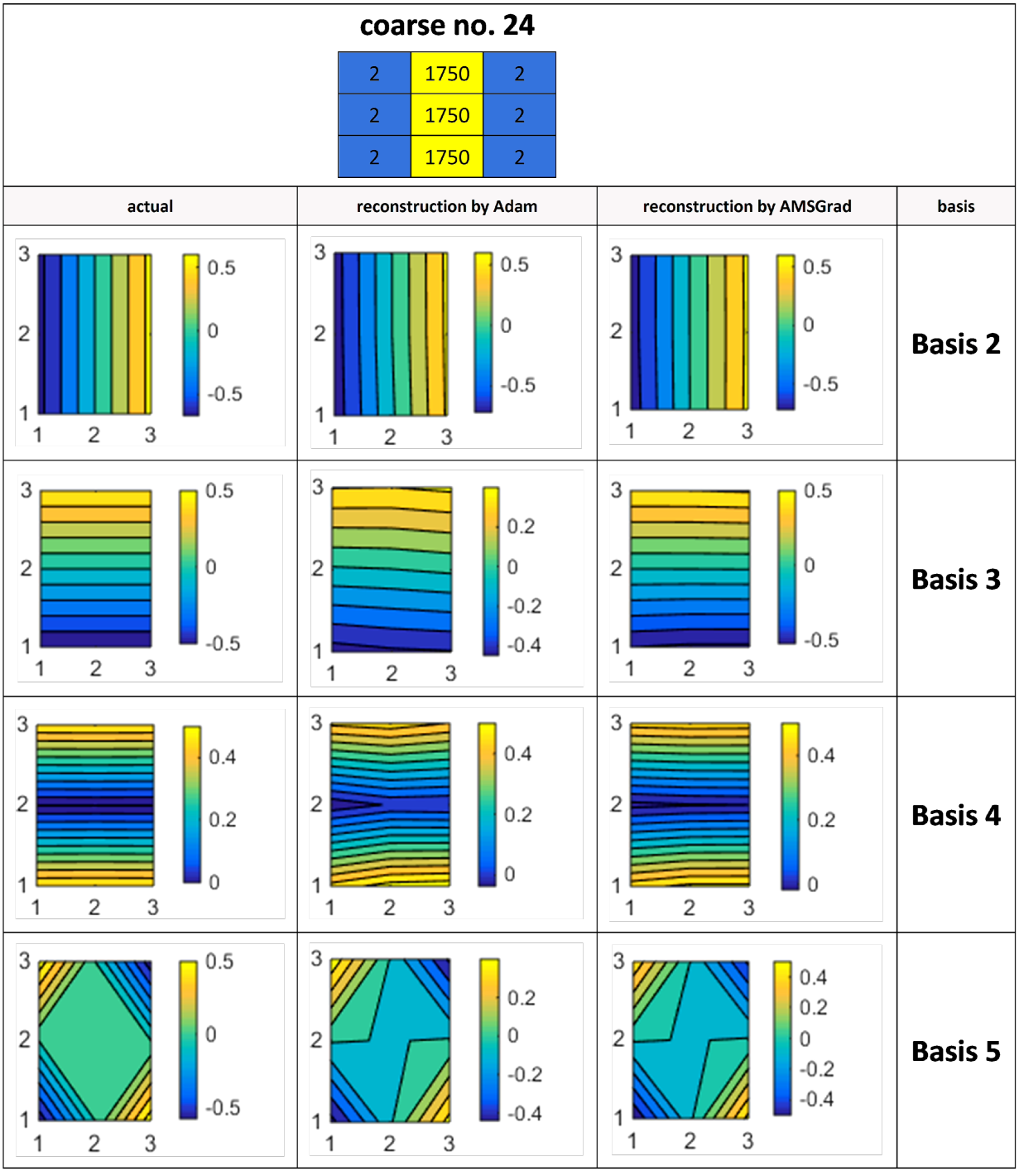


Figure 9.4: Comparing actual and reconstructed patterns in a fractured case.

Table 9.2: Comparison of the AMSGrad-based models trained on the initial and extended datasets.

Error statistic	CNN model	Training	Validation	Testing
$\mathbf{R^2}$	Basis 2 (initial dataset)	0.9079	0.8141	0.8156
	Basis 2 (extended dataset)	0.8657	0.7770	0.7750
	Basis 3 (initial dataset)	0.9142	0.7974	0.6503
	Basis 3 (extended dataset)	0.8952	0.8237	0.8247
	Basis 4 (initial dataset)	0.9172	0.8777	0.8815
	Basis 4 (extended dataset)	0.9156	0.8777	0.8769
	Basis 5 (initial dataset)	0.8449	0.7592	0.7671
	Basis 5 (extended dataset)	0.8466	0.7816	0.7800
MSE	Basis 2 (initial dataset)	0.0202	0.0441	0.0445
	Basis 2 (extended dataset)	0.0327	0.0544	0.0538
	Basis 3 (initial dataset)	0.0159	0.0401	0.0697
	Basis 3 (extended dataset)	0.0220	0.0371	0.0369
	Basis 4 (initial dataset)	0.0102	0.0158	0.0169
	Basis 4 (extended dataset)	0.0126	0.0182	0.0184
	Basis 5 (initial dataset)	0.0077	0.0137	0.0141
	Basis 5 (extended dataset)	0.0100	0.0142	0.0142

9.4 Impact of Skip Connection Schemes on the performance of SkiplessCNN

The analysis presented here is based on the models developed using the extended dataset and AMSGrad optimization algorithm as the optimizer. Figure 9.5 presents three bar charts giving the R^2 results of the developed models for the training, validation, and testing subsets, respectively. Figure 9.5 a demonstrates that all three skip connection schemes produce an improved performance over SkiplessCNN for all multiscale basis functions with respect to the training subset. For example, MidSkip increases it from 0.8657 to 0.9083 for the Basis 2 model. Both MidSkip and DualSkip perform marginally better than FirstSkip.

From Figures 9.5 b and c, it is shown that FirstSkip has a marginally positive effect on the validation and testing subsets with respect to the Basis 2, 4, and 5 models. For instance, regarding Basis 5, the R^2 value increases from 0.7816 and 0.7800 to 0.7974 and 0.7954 for validation and testing, respectively. However, it has an adverse effect on the Basis 3 model. Specifically, R^2 decreases from 0.8237 and 0.8347 to 0.8160 and 0.8164 for validation and testing, respectively. Compared to FirstSkip, MidSkip has a more positive effect on the Basis 2, 4, and 5 models. Furthermore, it has a negative effect on the Basis 3 model. DualSkip is beneficial in all cases related to validation and testing, especially for Basis 3 and 5. For example, for Basis 5, R^2 increases from 0.7816 and 0.7800 to 0.8038 and 0.8044 for validation and testing, respectively.

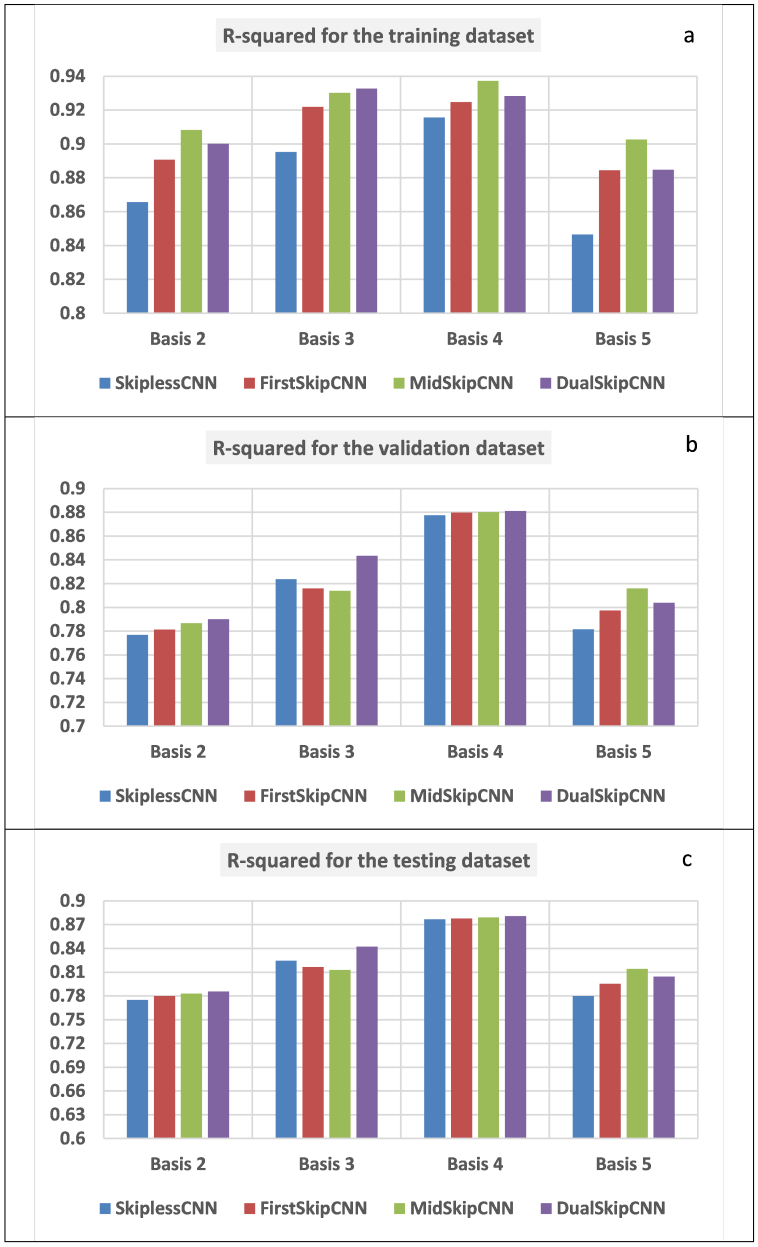


Figure 9.5: R^2 performance comparison of the CNN models with and without skip connections, for (a) the training dataset, (b) the validation dataset, and (c) the testing dataset.

Figure 9.6 presents three bar charts giving the MSE results obtained. The intuition here was that the MSE results would reflect the R^2 results (Figure 9.5). From Figure 9.6 a, it can be seen that for the training subset, all three skip connection schemes serve to decrease the MSE of SkiplessCNN. Additionally, MidSkip and DualSkip produce a better performance than FirstSkip. From Figures 9.6 b and c, it is shown that for the Basis 3 model

MSE fractionally increases for the validation and testing subsets when including FirstSkip or MidSkip. In other words, these two skip connection schemes do not favorably influence the Basis 3 models. For instance, FirstSkipCNN increases MSE of SkiplessCNN from 0.0371 to 0.0387 in the case of the validation dataset. DualSkip decreases MSE of SkiplessCNN for all multiscale basis functions.

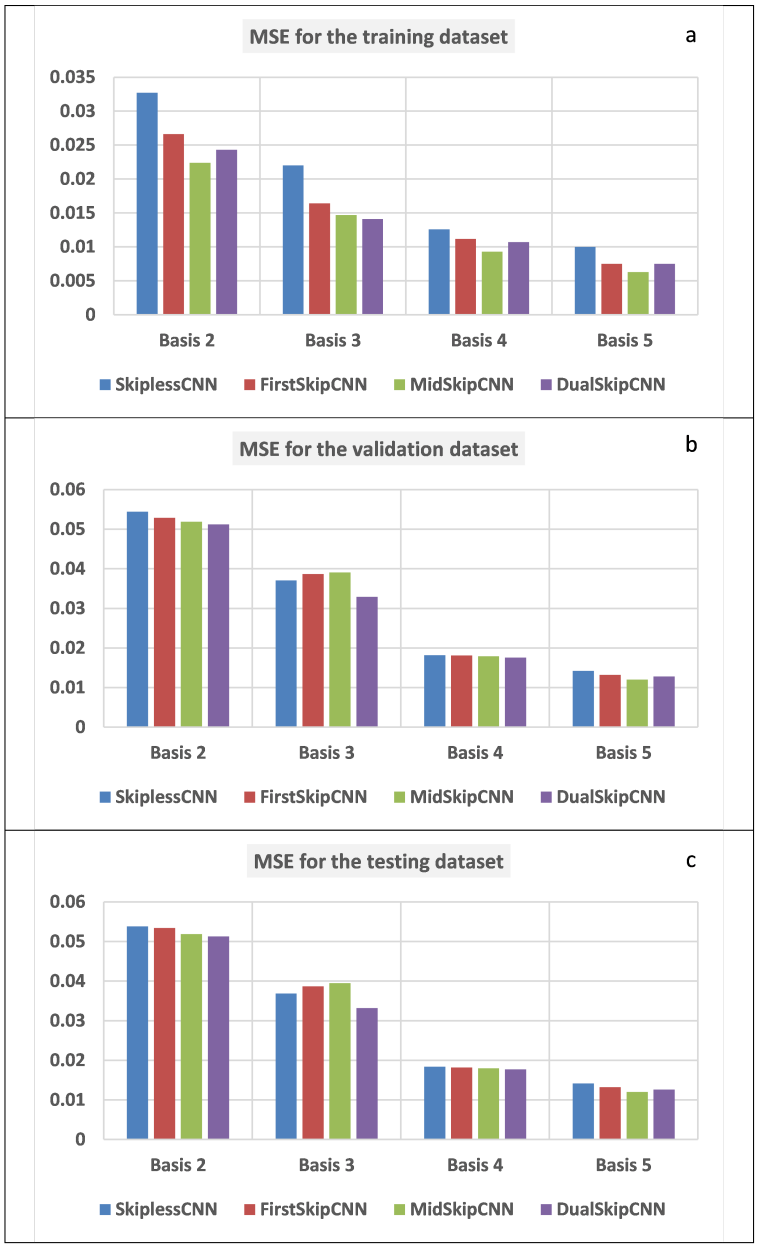


Figure 9.6: MSE performance comparison of the CNN models with and without skip connections, for (a) the training dataset, (b) the validation dataset, and (c) the testing dataset.

In addition to the graphical comparisons presented in Figures 9.5 and 9.6, the R^2 and MSE values are listed in full in Table 9.3. For each subset and multiscale basis function, the best result is highlighted.

Table 9.3: Prediction error analysis of the CNN models with and without skip connections developed by AMSGrad applied to the extended dataset.

Subset	**Model**	**R^2**				**MSE**			
		Basis 2	Basis 3	Basis 4	Basis 5	Basis 2	Basis 3	Basis 4	Basis 5
Training	SkiplessCNN	0.8657	0.8952	0.9156	0.8466	0.0327	0.0220	0.0126	0.0100
	FirstSkipCNN	0.8908	0.9219	0.9247	0.8844	0.0266	0.0164	0.0112	0.0075
	MidSkipCNN	**0.9083**	0.9302	**0.9372**	**0.9026**	**0.0224**	0.0147	**0.0093**	**0.0063**
	DualSkipCNN	0.9002	**0.9327**	0.9283	0.8847	0.0243	**0.0141**	0.0107	0.0075
Validation	SkiplessCNN	0.7770	0.8237	0.8777	0.7816	0.0544	0.0371	0.0182	0.0142
	FirstSkipCNN	0.7814	0.8160	0.8798	0.7974	0.0529	0.0387	0.0181	0.0132
	MidSkipCNN	0.7867	0.8139	0.8802	**0.8160**	0.0519	0.0391	0.0179	**0.0120**
	DualSkipCNN	**0.7900**	**0.8434**	**0.8811**	0.8038	**0.0512**	**0.0329**	**0.0176**	0.0128
Testing	SkiplessCNN	0.775	0.8247	0.8769	0.78	0.0538	0.0369	0.0184	0.0142
	FirstSkipCNN	0.7801	0.8164	0.8778	0.7954	0.0534	0.0387	0.0182	0.0132
	MidSkipCNN	0.7829	0.8127	0.8791	**0.8141**	0.0519	0.0395	0.018	**0.012**
	DualSkipCNN	**0.7857**	**0.8422**	**0.8809**	0.8044	**0.0513**	**0.0332**	**0.0177**	0.0126

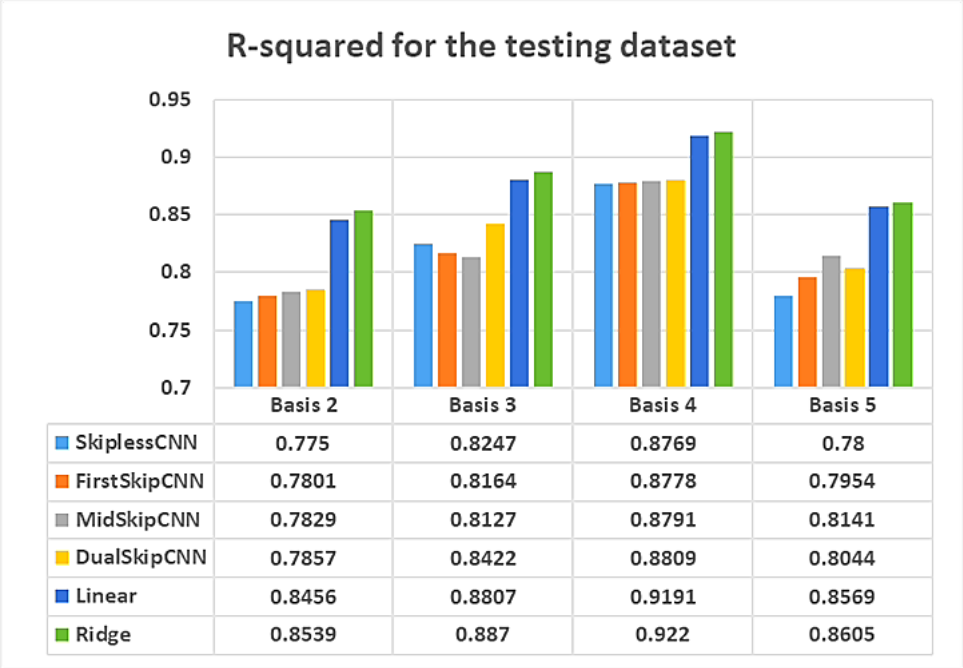
The results obtained for FirstSkipCNN imply that transferring feature maps from earlier convolutional layers to final ones has a very positive effect on the training dataset. This architecture has a marginally positive impact on the validation and testing subsets for the Basis 2, 4, and 5 models, but an adverse impact on the Basis 3 model. In other words, the corresponding skip connection tends to make the predictive model focus more on capturing the underlying trend of the training (seen) subset.

Compared to FirstSkip, flowing information from the middle convolutional layer to the last layer via the MidSkip skip connection has a more positive impact on all basis functions models of the training subset and the Basis 2, 4, and 5 models of the validation and testing subsets. This suggests that the feature maps of the middle convolution process contain important information.

Adding two simultaneous skip connections (DualSkip) favorably affects all basis functions with respect to the training, validation, and testing subsets. By comparing the architectures and results produced using MidSkip and DualSkip, the positive role of transferring raw feature maps is understandable. Therefore, enriching the last convolutional blocks with information hidden in the neighboring layers is more efficient than enriching them using earlier convolutional blocks near the input layer.

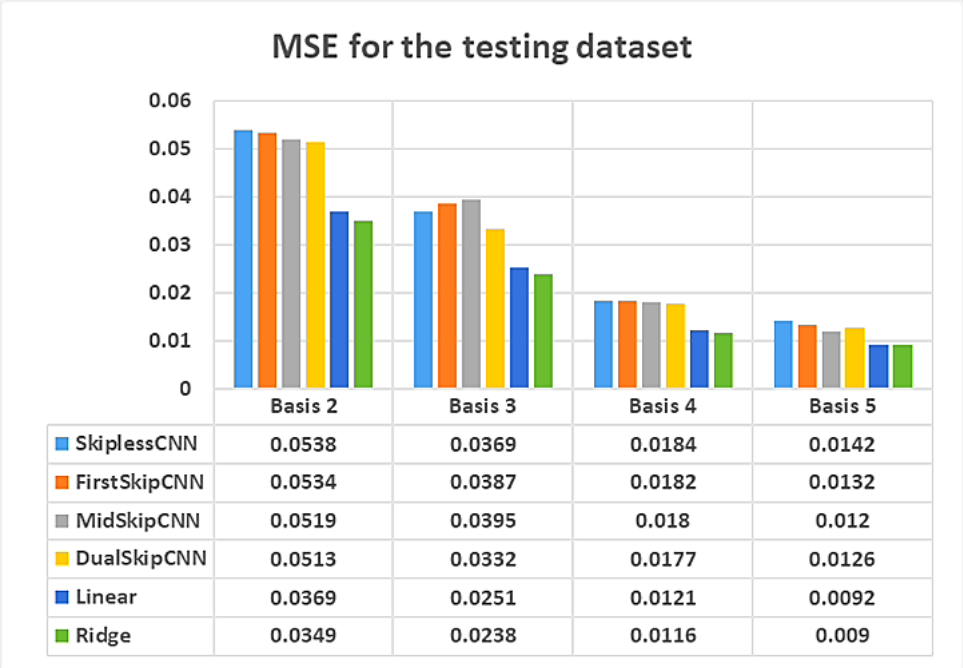
9.5 Performance Comparison of the DEL-based Models and CNN Models based on the Testing Subset

It is apparent from Figures 9.7 and 9.8 that the ensemble models built by either linear regression or ridge regression perform substantially better than the individual models on the testing subset. In the case of applying linear regression, R^2 and MSE lie in the range of 0.8456 to 0.9191, and 0.0092 to 0.0369, respectively. As expected, the results reveal that ridge regression works marginally better than linear regression with an R^2 ranging from 0.8539 to 0.9220, and ranging from MSE of 0.0090 to 0.0349, due to having more evenly distributed weights.



	Basis 2	Basis 3	Basis 4	Basis 5
SkiplessCNN	0.775	0.8247	0.8769	0.78
FirstSkipCNN	0.7801	0.8164	0.8778	0.7954
MidSkipCNN	0.7829	0.8127	0.8791	0.8141
DualSkipCNN	0.7857	0.8422	0.8809	0.8044
Linear	0.8456	0.8807	0.9191	0.8569
Ridge	0.8539	0.887	0.922	0.8605

Figure 9.7: Prediction error analysis of the DEL-based and CNN models applied to the testing data subset, expressed in terms of R^2.



	Basis 2	Basis 3	Basis 4	Basis 5
SkiplessCNN	0.0538	0.0369	0.0184	0.0142
FirstSkipCNN	0.0534	0.0387	0.0182	0.0132
MidSkipCNN	0.0519	0.0395	0.018	0.012
DualSkipCNN	0.0513	0.0332	0.0177	0.0126
Linear	0.0369	0.0251	0.0121	0.0092
Ridge	0.0349	0.0238	0.0116	0.009

Figure 9.8: Prediction error analysis of the DEL-based and CNN models applied to the testing data subset, expressed in terms of MSE.

9.6 Uncertainty Quantification of DualSkipCNN

In Section 9.4, it was found that among the three skip connection-based models defined, DualSkipCNN was the most effective as it increased the R^2 value by 0.0034–0.0381 and decreased the MSE value by 0.0006–0.0084 compared to the standard structure i.e., SkiplessCNN. To be more specific, except for Basis 5, R^2 of others is above 0.9 with regards to the training subset. The values obtained for MSE lie within the range of 0.0075 to 0.0243. The constructed models perform suitably for the validation subset, with an R^2 of 0.7900 to 0.8811 and an MSE of 0.0128 to 0.0512. Because the validation and testing subsets were selected from a similar distribution of data, it is anticipated to have almost the same results over the testing samples: an R^2 of 0.7857 to 0.8809, and MSE 0.0126 of to 0.0513.

According to Table 9.4, the dropout after two FC layers enhances performance over all subsets for all multiscale basis functions. For the training subset, it has the maximum effect on the model for Basis 3 and the minimum effect for Basis 4. For Basis 3, R^2 increases from 0.9327 to 0.9584, and MSE decreases from 0.0141 to 0.0113. There is an R^2 increase from 0.9283 to 0.9326 and an MSE decrease from 0.0107 to 0.0101 for Basis 4.

Adding dropout to the initial architecture has generally a marginally positive effect on the validation and testing samples. The range of R^2 and MSE is 0.7919-0.8858 and 0.0120-0.0507 for validation. R^2 and MSE lie in the range of 0.7881-0.8839 and 0.0121-0.0508 for testing.

As a general result, it is evident that the use of dropout has a positive impact on the

performance of the developed models across the training subset, regardless of the basis function used. Furthermore, it also demonstrates a similar positive impact over the validation and testing subsets for the Basis 3 and Basis 5 models. However, for Basis 2 and Basis 4, there is only a marginal difference between the performance of the $\text{DualSkipCNN}_{\text{initial}}$ and $\text{DualSkipCNN}_{\text{dropout}}$ models. The probable reasons for this could be attributed to the high-dimensional regression problem investigated in this study and the complexity and non-linear nature of DL models. Nonetheless, even this slight improvement in the models' performance could help reduce overfitting and enhance generalization in the constructed $\text{DualSkipCNN}_{\text{dropout}}$ models. Additionally, it can significantly affect the pressure distribution obtained through the basis functions.

Table 9.4: Performance comparison of $\text{DualSkipCNN}_{\text{initial}}$ and $\text{DualSkipCNN}_{\text{dropout}}$ models in terms of R^2 and MSE.

Subset	**Model**	**R^2**				**MSE**			
		Basis 2	**Basis 3**	**Basis 4**	**Basis 5**	**Basis 2**	**Basis 3**	**Basis 4**	**Basis 5**
Training	$\text{DualSkipCNN}_{\text{initial}}$	0.9002	0.9327	0.9283	0.8847	0.0243	0.0141	0.0107	0.0075
	$\text{DualSkipCNN}_{\text{dropout}}$	**0.9113**	**0.9584**	**0.9326**	**0.9089**	**0.0211**	**0.0113**	**0.0101**	**0.0058**
Validation	$\text{DualSkipCNN}_{\text{initial}}$	0.7900	0.8434	0.8811	0.8038	0.0512	0.0329	0.0176	0.0128
	$\text{DualSkipCNN}_{\text{dropout}}$	**0.7919**	**0.8620**	**0.8858**	**0.8155**	**0.0507**	**0.0290**	**0.0170**	**0.0120**
Testing	$\text{DualSkipCNN}_{\text{initial}}$	0.7857	0.8422	0.8809	0.8044	0.0513	0.0332	0.0177	0.0126
	$\text{DualSkipCNN}_{\text{dropout}}$	**0.7881**	**0.8622**	**0.8839**	**0.8132**	**0.0508**	**0.0290**	**0.0173**	**0.0121**

According to Table 9.5, the SD values lie within 0.0181-0.158, 0.0179-0.152, 0.0169-0.104, and 0.0121-0.086 for the CNN models with dropout developed for Basis 2, 3, 4, and 5 based on the training subset. The SD obtained for the Basis 4 and 5 models is lower than that for the Basis 2 and 3 models. For all basis functions, most samples have an SD lower than 0.05. For instance, 221,006 out of 304,511 samples for Basis 3 are in the range of 0-0.05. In general, SD exceeds 0.15 only for 547 samples.

With regards to the validation subset, the developed models for Basis 2, 3, 4, and 5 have an SD range of 0.0268-0.174, 0.0237-0.124, 0.019-0.171, and 0.012-0.097, respectively. Generally, only 27 out of 34,421 samples have an SD higher than 0.15. The model built for Basis 5 has the best performance in terms of uncertainty, with 24,276 samples having an SD of lower than 0.05 and 10,145 samples with an SD of 0.05-0.1. Following this, the models developed for Basis 4 and 3 perform well. The model designed for Basis 2 has the worst performance, with only 2577 samples having an SD between 0 and 0.05.

The SD values for the testing subset lie within 0.025-0.169, 0.024-0.142, 0.020-0.113, and 0.012-0.098 for the CNN models with dropout developed for Basis 2, 3, 4, and 5. The trend is consistent with that of the validation subset, whereby the model for Basis 5 performs the

best, and the one for Basis 2 performs the worst. Additionally, except for 24 cases for Basis 2, there are no samples with an SD higher than 0.15.

Table 9.5: Reliability of the developed DualSkipCNN$_{\text{dropout}}$ models for Basis 2, 3, 4, and 5 using MC dropout in terms of SD.

Subset	**SD**	**Basis 2**	**Basis 3**	**Basis 4**	**Basis 5**
Training	[0-0.05)	197,572(0.649)	221,006(0.726)	238,107(0.782)	261,537(0.859)
	[0.05-0.1)	99,364(0.326)	81,227(0.267)	63,189(0.208)	42,974(0.141)
	[0.1-0.15)	7143(0.024)	2163(0.007)	3215(0.01)	-
	≥ 0.15	432(0.001)	115	-	-
Validation	[0-0.05)	2577(0.075)	4679(0.136)	7475(0.217)	24,276(0.705)
	[0.05-0.1)	19,296(0.561)	29,395(0.854)	26,937(0.783)	10,145(0.295)
	[0.1-0.15)	12,522(0.364)	347(0.01)	8	-
	≥ 0.15	26	-	1	-
Testing	[0-0.05)	2245(0.079)	3984(0.138)	6232(0.216)	20,725(0.718)
	[0.05-0.1)	16,321(0.565)	24,599(0.852)	22,641(0.784)	8154(0.282)
	[0.1-0.15)	10,289(0.356)	296(0.01)	6	-
	≥ 0.15	24	-	-	-

As mentioned earlier, the output is in the form of a 900×1 vector, which is too big to show in a graph. Additionally, basis functions in the mixed GMsFEM are defined in one coarse grid element, which includes 9 fine grids. Figure 9.9 gives the 30 values obtained for each of the nine points using MC dropout for a coarse grid with the matrix permeability of 1 mD (as a representative sample). The average of 30 outputs (for each point) is considered the model's output. The figure demonstrates that the values are close to each other (some overlap) and have a very low SD.

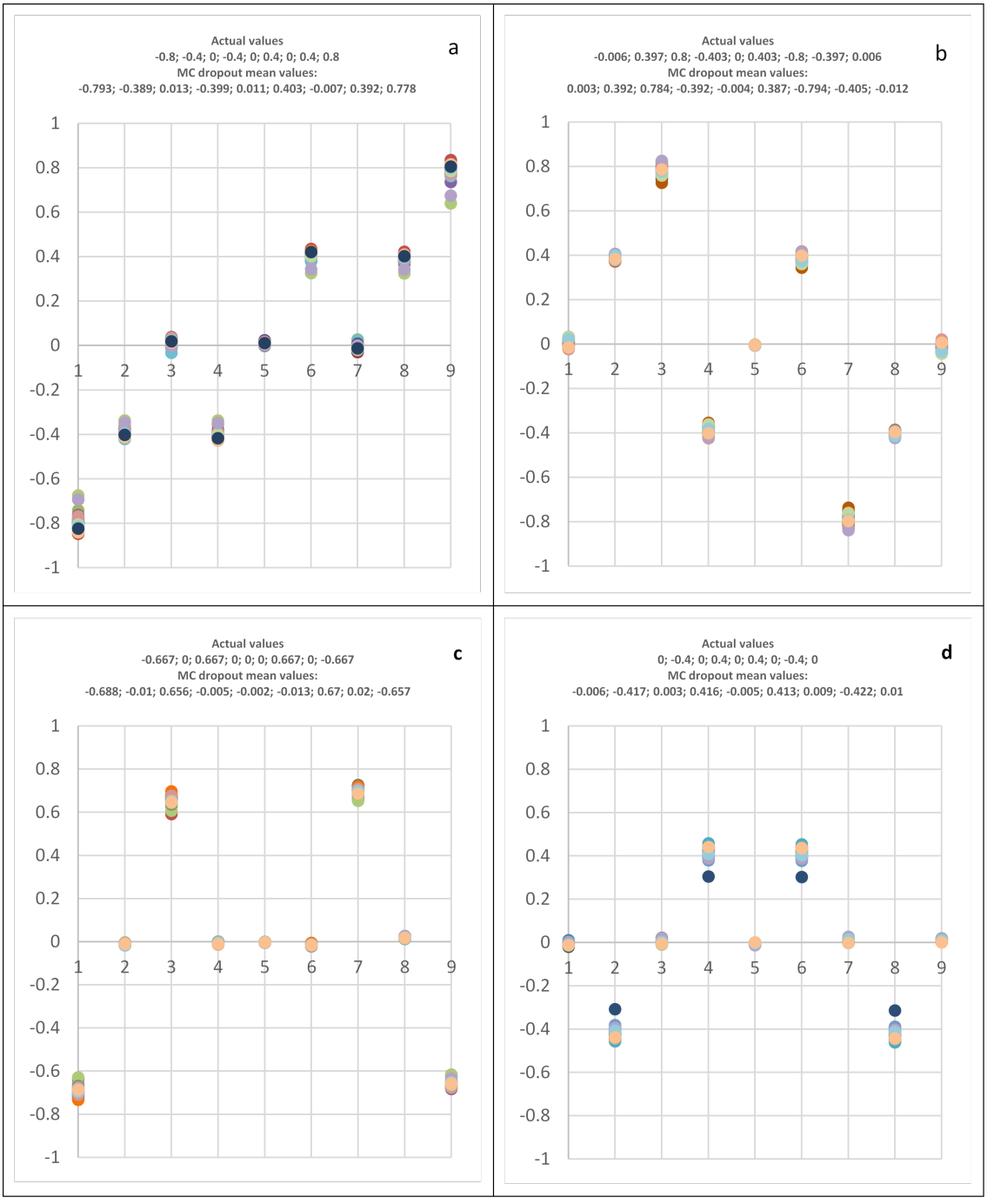


Figure 9.9: Dispersion of values for a representative coarse grid for: (**a**) Basis 2, (**b**) Basis 3, (**c**) Basis 4, and (**d**) Basis 5.

9.7 Performance Evaluation of the FNO Models

There are two main hyperparameters in FNO: the number of channels and modes. The former defines the width of the FNO network, referring to the number of features learned in each layer. The latter defines the number of lower Fourier modes retained when truncating the Fourier series. The size of the grid space controls the maximum allowable number of modes. In this research, five values were evaluated for the width: 20, 60, 100, 140, and 180, and four cases were evaluated for the mode: 5, 10, 15, and 20.

Figure 9.10 a reveals that the FNO models generated very similar errors, based on MSE when calculated based on initial pressure values (actual non-normalized values) for the training data when the number of modes is 10, 15, or 20. However, the errors increased slightly for models configured with modes = 5. The R^2 values for the training data varied from 0.9945 to 0.9971, according to Figure 9.10 b. As a general result, all models were able to predict pressure with acceptable error levels for the training subset.

Figures 9.10 c,d display the FNO results for the testing subset. The model with modes of 5 generated the poorest prediction performance, i.e., highest MSE and lowest R^2. As width increased (with modes held at 5), MSE decreased from 109.9231 to 86.3347 and R^2 increased from 0.7661 to 0.8163. When the number of modes was increased to 10, the FNO performance improved. Additionally, an increase in width had a positive effect on accuracy when modes were held at 10. The model with modes = 15 performed better than models with modes of 5 and 10, as it generated MSE and R^2 displaying ranges of 42.1611–65.5664 and 0.8605–0.9103, respectively. In general, the prediction performance of the FNO model with modes of 20 overlapped with that of modes of 15. Considering all twenty cases, the model with modes = 15 and width = 100 generated the best performance with an MSE of 1.4087 and R^2 of 0.997 for the training subset, and an MSE of 42.1611 and R^2 of 0.9103 for the testing subset. In addition to the graphical comparisons (Figure 9.10), the MSE and R^2 values achieved by all FNO cases evaluated are listed in Table 9.6.

To assess whether downsampling has a positive or negative impact on the FNO model performance with respect to small-shape data (in the dataset modeled: 30×30), a downsampling rate was set to 2. By applying that rate, the data shape was reduced to 15×15, which led to poor prediction results. For example, with modes = 10 and width = 100, the FNO model achieved pressure predictions with an MSE of 27.3128 and R^2 of 0.9411 for the training subset, and with an MSE of 410.7709 and R^2 of 0.1259 for the testing subset. As to be expected, further downsampling of the initial case caused prediction accuracy to deteriorate further. A likely explanation for this outcome is that the size of the grid space controls the maximum allowable number of modes. This means that by downsampling, the allowable number of FNO modes also decreases. Meantime, because CNN acts on discretized vectors, downsampling with CNN is not reasonable.

The prediction performance is improved by adding only MLP to the original Fourier layers while maintaining the previous hyperparameters such as the number of channels and modes, number of Fourier layers, batch size, learning rate, number of epochs, and gamma. MSE For the training subset decreases from 1.4087 to 0.7163, and the R^2 value increases

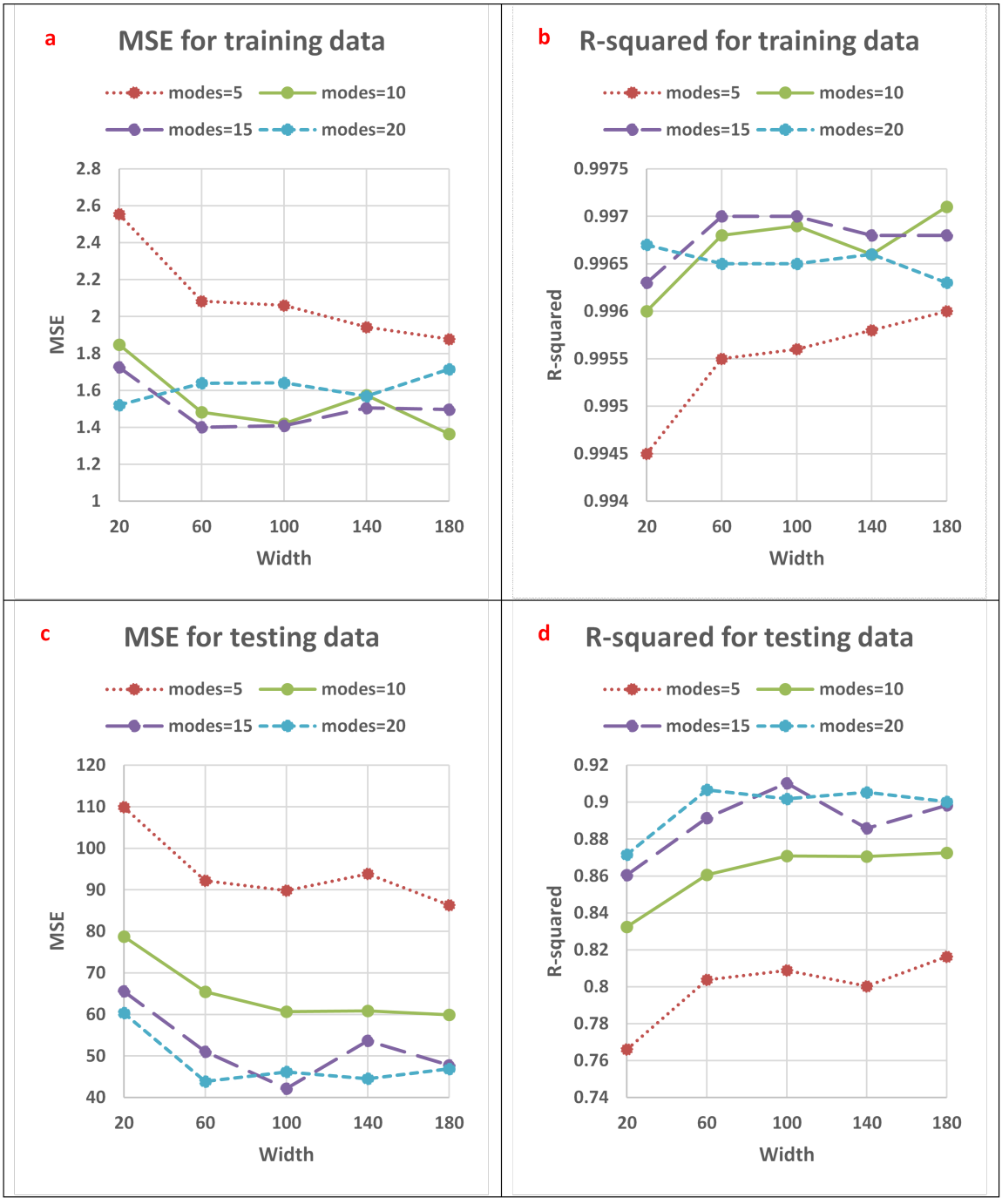


Figure 9.10: Prediction error graphical analysis of the developed FNO models applied to the training/testing data subsets in terms of MSE and R^2.

Table 9.6: Performance of the developed FNO models with different modes and widths based on MSE and R^2.

mode	width	MSE (training)	R^2 (training)	MSE (testing)	R^2 (testing)
5	20	2.5543	0.9945	109.9231	0.7661
5	60	2.0832	0.9955	92.2014	0.8038
5	100	2.0605	0.9956	89.8219	0.8089
5	140	1.943	0.9958	93.8539	0.8003
5	180	1.878	0.996	86.3347	0.8163
10	20	1.8483	0.996	78.7648	0.8324
10	60	1.4814	0.9968	65.4587	0.8607
10	100	1.4196	0.9969	60.6803	0.8709
10	140	1.5745	0.9966	60.8775	0.8705
10	180	1.3643	0.9971	59.8904	0.8726
15	20	1.7253	0.9963	65.5664	0.8605
15	60	1.4007	0.997	51.0625	0.8914
15	**100**	**1.4087**	**0.997**	**42.1611**	**0.9103**
15	140	1.505	0.9968	53.6779	0.8858
15	180	1.4966	0.9968	47.783	0.8983
20	20	1.5206	0.9967	60.3367	0.8716
20	60	1.6387	0.9965	43.8621	0.9067
20	100	1.6409	0.9965	46.167	0.9018
20	140	1.5687	0.9966	44.5223	0.9053
20	180	1.7145	0.9963	46.8985	0.9002

from 0.997 to 0.9985. With regards to the testing subset, MSE decreases from 42.1611 to 11.9111, and R^2 increases from 0.9103 to 0.9747. Therefore, the new FNO architecture has a significant effect on the testing subset, addressing the issue of overfitting observed with the initial FNO architecture.

The prediction performance of the CNN model is similar to that of the FNO models in terms of R^2 with regard to the training subset (Table 9.7). MSE generated by the CNN model (0.3074) is slightly less than that generated by the FNO models (0.7163 and 1.4087). Nonetheless, the FNO models clearly provided superior results in terms of R^2 and MSE when

the trained models were applied to the testing data subset. The results (Table 9.7) suggest that whereas the trained FNO model is well-fitted to the dataset, the trained CNN model is somewhat over-fitted to the same dataset.

Table 9.7: Performance comparison of the best-performing FNO models and the CNN model in terms of MSE and R^2.

model	**MSE (training)**	**R^2 (training)**	**MSE (testing)**	**R^2 (testing)**
$FNO_{standard}$ (mode = 15 and width = 100)	1.4087	0.997	42.1611	0.9103
FNO_{MLP} (mode = 15 and width = 100)	0.7163	0.9985	11.9111	0.9747
CNN	0.3074	0.9993	86.1818	0.8166

In order to improve visualization of the pressure changes occurring over the defined shapes, three examples are illustrated for selected training (Figure 9.11) and testing (Figure 9.12) subsets. The plots in the left-side columns display the permeability fields, for representative sample grids. The plots in the left-central columns display the pressure distribution derived by FEM (considered to be true distribution). The plots in the right-central columns display the predicted pressure distributions of the best-performing FNO_{MLP} model developed. The plots in the right-side columns display the pressure difference between the FEM and FNO_{MLP} outputs $[p_{(FNO_{MLP})} - p_{(FEM)}]$. Generally, there was a very close match between the true pressure distributions and those predicted by the FNO_{MLP} model, especially for the training dataset.

9.8 Summary

This chapter presented a statistical and graphical performance analysis of the developed models. To assess the accuracy of the models in predicting the multiscale basis functions, two statistical error metrics were employed: R^2 and MSE. The pattern available in a coarse block was tracked for graphical investigation. The models' uncertainty was quantified using the measure of SD. R^2 and MSE were utilized to evaluate the FNO-based models' ability to predict pressure distribution. Moreover, a graphical investigation was carried out on the entire porous medium. The next chapter provides an overview of the main findings of the work presented in this thesis and offers some suggestions for future work.

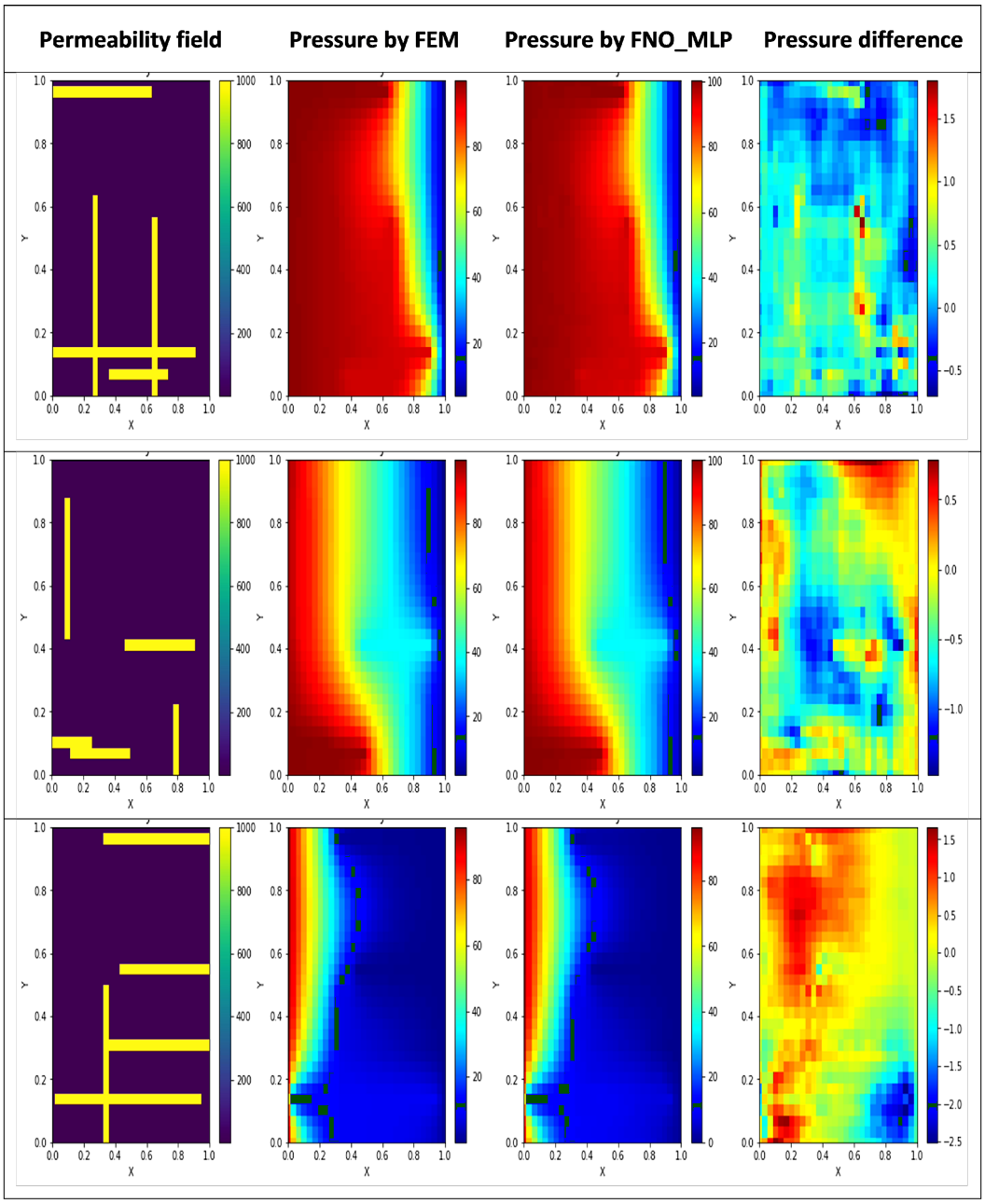


Figure 9.11: A comparison between the actual pressure distributions and those obtained by FNO_{MLP} for three representative training subset samples. The pressure difference is based on a point-by-point absolute error. Outputs are displayed as rectangles rather than squares due to a scaling issue.

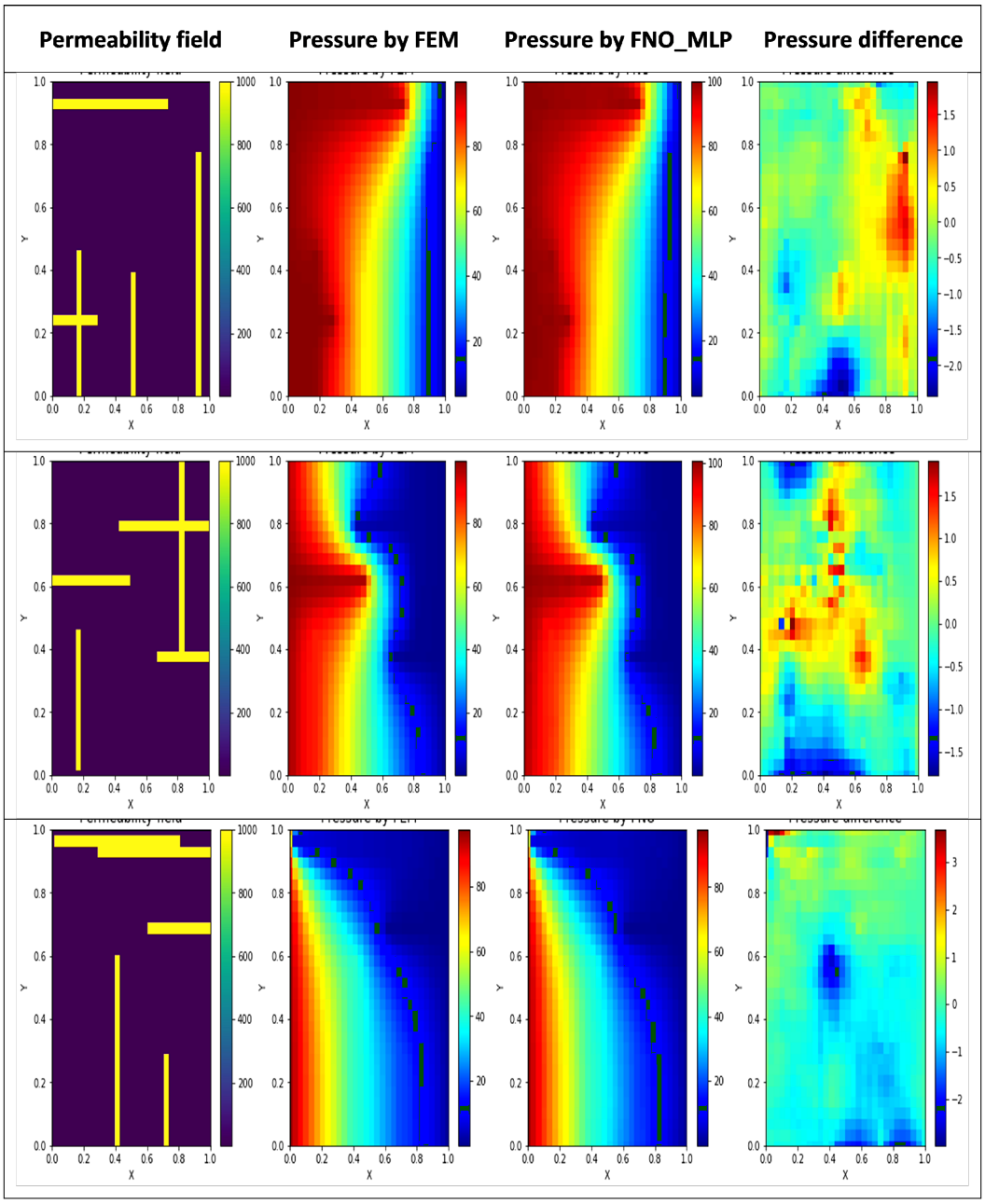


Figure 9.12: A comparison between the actual pressure distributions and those obtained by FNO_{MLP} for three representative testing subset samples. The pressure difference is based on a point-by-point absolute error. Outputs are displayed as rectangles rather than squares due to a scaling issue.

Chapter 10

Conclusion

10.1 Introduction

This chapter presents the conclusion of the research presented in this thesis and is organized as follows. In Section 10.2, a summary of the thesis is provided. The main findings and contributions with respect to the main research question and the subsidiary research questions are provided in Section 10.3. Finally, Section 10.4 offers some recommendations for potential future work.

10.2 Summary of the Thesis

This thesis began with Chapter 1, discussing the importance of fractures in numerical modeling of heterogeneous petroleum reservoirs. Numerous PDEs must be solved to produce multiscale basis functions in the mixed GMsFEM. This leads to significant computational overhead. Considering the widespread acceptance and effectiveness of DL models, the primary research question addressed in this thesis was how to most effectively replace PDE solvers with DL models for predicting basis functions. Moreover, the suitability of FNO, acting on infinite-dimensional spaces, was investigated to directly predict pressure distribution. To limit the scope of the research, the computational domain was defined as $\Omega = [0, 1]^2$.

As mentioned in Chapter 2, there are three common methods: (i) reduced order modeling, (ii) upscaling, and (iii) multiscaling, to address computational challenges in modeling fine grid systems. The chapter provided a detailed explanation of the mixed GMsFEM, a new multiscaling method. Furthermore, the chapter surveyed published research on the use of data-driven methods for reduced order modeling, upscaling, and multiscaling domains.

Chapter 3 provided details of three separate datasets: two for multiscale basis functions (Basis 2, 3, 4, and 5) and another for pressure. In both cases, the only input required was the permeability field. The initial data generated for multiscale basis functions comprised 249,375 samples, which increased to 376,250 samples for the extended dataset. However, generating data for pressure was challenging, resulting in 1700 samples, which was consider-

ably lower than the number of samples for the basis functions. The chapter also discussed the necessary preparations involved in generating the data, such as removing duplicates, scaling input/output, and changing the initial dimension of input/output.

In Chapter 4, a similar standard CNN configuration named SkiplessCNN, with five convolutional layers and two FC layers, was presented to predict each of the four basis functions (Basis 2 to 5). The number of kernels in the convolutional layers 1 to 5 was 5, 10, 15, 20, and 25, respectively. Each FC layer contained 2000 neurons. The input (permeability field) was in the format of 100×9, and the output was in the format of 900×1. Statistical analysis revealed that training the standard network with AMSGrad resulted in a slightly better performance compared to Adam as an optimization algorithm based on the initial dataset. In addition, it was found that having a sufficient amount of data for the validation and testing subsets could help decrease overfitting, based on the results obtained from the extended dataset.

Chapter 5 was dedicated to investigating the effectiveness of skip connections in DNNs of low complexity, such as SkiplessCNN. To this end, three different skip connection schemes were added to the base CNN structure: (i) FirstSkip, (ii) MidSkip, and (iii) DualSkip. FirstSkip added a single skip connection from the first convolutional layer to the last convolutional block. MidSkip included a single skip connection from the middle convolutional layer to the last layer. In DualSkip, two skip connections from the middle convolutional layer to the last and the second-to-last layers were considered. The results showed that all three skip connections were effective, with DualSkip being the most effective among them.

To avoid the continuous endeavor required to adjust the architecture of individual networks, or the nature of the propagation to improve the accuracy of DL models, DEL was introduced and applied in Chapter 6. In this technique, four models - SkiplessCNN, FirstSkipCNN, MidSkipCNN, and DualSkipCNN - were used as the base learners and combined separately using linear regression and ridge regression as part of the stacking technique. The results demonstrated that the combined models meaningfully outperformed the individual models for all basis functions based on the testing subset.

DL models are powerful prediction tools but often neglect the issue of uncertainty. On the other hand, it was found that DualSkipCNN performed better than FirstSkipCNN and MidSkipCNN. Therefore, MC dropout was applied in Chapter 7 to investigate the reliability of DualSkipCNN. A dropout ratio of 0.05 was used for the FC layers, and the analysis was performed separately for each of the multiscale basis functions (Basis 2 to 5). The obtained SD range of 0.012–0.174 confirmed the robustness of MC dropout in terms of epistemic uncertainty, in addition to the high degree of accuracy (R^2 of 0.7881–0.9584 and MSE of 0.0113–0.0508).

Classical NNs are designed to learn mappings between finite-dimensional input and output spaces. This makes them mesh-dependent and confined to a particular discretization. To reduce these constraints, mesh-independent networks have been developed that learn mappings between functions. In Chapter 8, two FNO models were developed to predict pressure distribution on a 30×30 problem using 1700 generated samples. A CNN model was also developed based on the same dataset. The results confirmed the superior prediction

performance of the FNO models to that of the CNN model.

Chapter 9 provided a detailed analysis of the statistical-graphical results obtained, accompanied by a discussion of them. The chapter addressed various topics related to the multiscale basis functions, such as comparing the performance of Adam and AMSGrad based on the initial dataset, investigating the impact of dataset size on AMSGrad-based models' performance, exploring the influence of skip connections on the SkiplessCNN's performance, comparing the performance of DEL-based and CNN models on the testing subset, and quantifying the uncertainty of DualSkipCNN. Moreover, the chapter presented the performance of two FNO models and a new type of CNN model designed to predict pressure distribution.

10.3 Main Findings

This section provides the main findings of the work presented in this thesis in the context of the research questions. It commences by considering the subsidiary research questions and then goes on to address the main research question:

1. **Is DL able to accurately reconstruct four distinct multiscale basis functions in the mixed GMsFEM in terms of statistical-graphical investigation, given its impressive performance with respect to datasets involving nonlinear relationships in recently published research in a range of scientific and engineering fields?**

   As the problem being studied was of a supervised 2D type, four different CNN models were developed for the multiscale basis functions (Basis 2 to 5) of the mixed GMsFEM. A similar CNN configuration was achieved for each of Basis 2 to 5, consisting of five convolutional and two FC layers. Graphically, all models precisely followed the observed trend in each coarse block. The statistical results indicated that, based on the total initial dataset, the AMSGrad optimizer with an R^2 of 0.8434–0.9165 and MSE of 0.0078–0.0206 performed slightly better than Adam with an R^2 of 0.8328–0.9049 and MSE of 0.0109–0.0261. However, both optimization algorithms somewhat suffered from overfitting, especially regarding Basis 3. Moreover, extending the dataset resulted in a slight decrease in AMSGrad-based models' performance for the training subset. However, it had a positive effect on the validation and testing subsets, which could help decrease overfitting.

2. **Will skip connections significantly affect the performance of Deep Neural Networks (DNNs) of low complexity or whether their inclusion has little or no effect?**

   The analysis was performed using three distinct skip connections named FirstSkip, MidSkip, and DualSkip. The results obtained for the FirstSkipCNN architecture implied that transferring feature maps from earlier convolutional layers to final ones had a significantly positive effect on the training dataset. This architecture had a marginally positive impact on the validation and testing subsets for Basis 2, 4, and 5 models,

but adversely affected the Basis 3 model. Compared to FirstSkip, flowing information from the middle convolutional layer to the last layer via the MidSkip skip connection had a more positive impact on all basis functions models of the training subset and the Basis 2, 4, and 5 models of the validation and testing subsets. However, it had a negative effect on the Basis 3 model, similar to FirstSkip. Adding two simultaneous skip connections (DualSkip) favorably affected all basis functions with respect to the training, validation, and testing subsets. Therefore, enriching the last convolutional blocks with information hidden in the neighboring layers was more efficient than using earlier convolutional blocks close to the input layer to enrich them.

3. **Does combining multiple deep learners into an ensemble improve the accuracy of DL algorithms?**

   Four models, SkiplessCNN, FirstSkipCNN, MidSkipCNN, and DualSkipCNN, were used as base learners. These models were combined using linear regression and ridge regression separately as part of the stacking technique. Based on the results, the ensemble models built using either linear regression or ridge regression performed substantially better than the individual models on the testing subset. In the case of linear regression, the R^2 values ranged from 0.8456 to 0.9191, with corresponding MSE values ranging from 0.0092 to 0.0369. As expected, the results revealed that ridge regression worked marginally better than linear regression with an R^2 ranging from 0.8539 to 0.9220, and a corresponding MSE ranging from 0.0090 to 0.0349, due to having more evenly distributed weights.

4. **How does incorporating Uncertainty Quantification (UQ) methods improve the reliability of the CNN models in predicting new data points?**

   To investigate the reliability of the DualSkipCNN model in terms of epistemic uncertainty, MC dropout was applied as a computationally efficient approach. A dropout ratio of 0.05 was used for the FC layers, and the analysis was performed separately for each of the multiscale basis functions (Basis 2 to 5). The SD range of 0.012–0.174 confirmed the robustness of MC dropout, in addition to the high degree of accuracy (R^2 of 0.7881–0.9584 and MSE of 0.0113–0.0508).

5. **Can FNO models accurately perform on small-shape data problems to predict pressure distribution?**

   A 30×30 uniform mesh problem was considered with 1400 samples for the training dataset and a further 300 samples for the testing dataset. Among all twenty cases, it was found that the original FNO model with modes = 15 and width = 100 performed the best with an MSE of 1.4087 and R^2 of 0.997 for the training subset, and an MSE of 42.1611 and R^2 of 0.9103 for the testing subset. Furthermore, downsampling was found to have a negative impact on the FNO model performance. The addition of an MLP to the original Fourier layers significantly improved the prediction performance. For the training subset, MSE decreased from 1.4087 to 0.7163, and the R^2 value increased from

0.997 to 0.9985. With regards to the testing subset, the MSE decreased from 42.1611 to 11.9111, and R^2 increased from 0.9103 to 0.9747. Therefore, the new FNO architecture addressed the issue of overfitting observed with the initial FNO architecture. While the prediction performance of the CNN model was similar to that of the FNO models for the training subset, it was found to be somewhat over-fitted when applied to the testing subset.

Returning to the main research question that this thesis sought to answer:

To what extent can standard DL models effectively reconstruct the multiscale basis functions of the mixed GMsFEM, and what steps can be taken to improve their performance and reliability? Moreover, is the mesh-independent approach of Fourier Neural Operator (FNO) able to accurately predict pressure distribution?

While standard DL models demonstrated good performance on the training subset, they encountered challenges in effectively reconstructing the multiscale basis functions of mixed GMsFEM on the testing subset. To mitigate the issue of overfitting, various techniques and methods were employed, such as allocating more samples to the validation and testing subsets, incorporating skip connections, and combining deep learners. Additionally, the reliability of the best skip connection-based model was assessed using a computationally effective method. Finally, the mesh-independent approach of FNO yielded promising results in accurately predicting pressure distribution, particularly by integrating an MLP with the Fourier layers. Viewing this work as an image (matrix)-to-image (matrix) regression problem, the constructed data-driven models may have applications beyond reservoir engineering, such as hydrogeology and rock mechanics.

10.4 Future Work

While the current study presents a valuable contribution towards understanding the application of data-driven methods in petroleum reservoirs, there are several avenues for future research:

1. **Evaluation of the models developed for the multiscale basis functions using a new dataset:** The models were trained, validated, and tested using consistent permeability value ranges: specifically, $K_{\mathrm{m}} = 1$, 2, 3, 4, and 5 mD, and $K_{\mathrm{f}} = 500$, 750, 1000, 1250, 1500, 1750, and 2000 mD. It is advisable to test the models with varied values. For instance, data samples might be generated based on K_{m} values of 1.5, 2.5, 3.5, and 4.5 mD, and K_{f} values of 625, 875, 1125, 1375, 1625, and 1875 mD. By adopting this approach, the potential for overfitting can be more effectively examined.

2. **Exploring various configurations for the computational domain:** In this study, a fine grid system with a uniform 30×30 mesh was employed. Also, a sparser, uniform 10×10 mesh represented the coarse grid network. With the mixed GMsFEM, there

is potential to establish configurations utilizing a greater number of multiscale basis functions. Hence, delving into larger systems, such as 100×100 and 500×500 fine grid systems, could be intriguing. Considering the input/output dimensions, it is anticipated that more intricate models will be required, involving additional convolutional layers and a higher number of kernels than those utilized in this study. Consequently, the demand for more robust computing capabilities will arise.

3. **Extension to 3D porous media with inclined fractures and a wider range of permeability:** The scope of the present study was restricted to 2D porous media with vertical and horizontal fractures. It is recommended to extend it to 3D porous media and incorporate inclined fractures. Moreover, it would be worthwhile to consider a wider range of permeability values for both the matrix and fracture. These adjustments would provide a more comprehensive representation of subsurface conditions.

4. **Exploration of more advanced ensemble techniques:** The effectiveness of the stacked generalization method using linear regression and ridge regression was confirmed on the given task. However, there is still room for improvement in the testing subset. To further enhance performance, exploring more advanced ensemble techniques could be one direction to consider. Additionally, developing other types of skip connection-based CNN models and using them as base learners could also provide valuable insights for further improving the performance of the stacking ensemble.

5. **Using more suitable indices for UQ:** Several indices, such as entropy, Negative Log Likelihood (NLL), and SD were defined to quantify epistemic uncertainty. However, the values obtained for entropy and NLL were not particularly meaningful or informative. Therefore, it would be beneficial to explore alternative statistical measures that are more applicable and appropriate for conveying information about uncertainty more accurately, which would ultimately improve the overall reliability of the models.

6. **Interpretability and explainability of CNNs:** CNNs are frequently criticized for their lack of interpretability, which can make it difficult to understand how the model arrives at its predictions. In contrast, explainability refers to the ability to provide reasons or justifications for the model's decisions, effectively explaining why a certain prediction was made for a given input. Prioritizing research and development efforts towards increasing the interpretability and explainability of CNNs could improve the ability to identify biases or errors in the model's predictions, build trust in the model's decisions, and identify areas for optimization and improvement.

7. **Incorporating PINN into FNO architecture:** As opposed to data-driven NNs such as FNO, which rely exclusively on the provided data points, PINNs use the PDE itself as a data source. In PINNs, PDEs are explicitly encoded into the NN via automatic differentiation algorithms. The weighted summation of MSE of the PDE residuals, BCs, ICs, and possibly known solution points could then be minimized as a loss function based on the NN parameters. Therefore, it could be beneficial to combine

PINN and FNO to find out how the performance changes compared to a stand-alone FNO. In this sense, the model uses available data and/or physics constraints to learn the solution operator, conquering the limitations of purely data-driven and physics-based techniques.

8. **Generating more data for pressure distribution:** Data generation for pressure distribution was a challenging and time-consuming task. Consequently, the FNO models were developed using only 1700 samples with fixed values of $K_{\mathrm{m}} = 1$ md, $K_{\mathrm{f}} = 1000$ md, and $N_{\mathrm{f}} = 5$. Therefore, it is advisable to generate a more diverse and comprehensive dataset by considering a wide range of K_{m}, K_{f}, and N_{f} values.

Bibliography

[1] Jie Chen, Eric T Chung, Zhengkang He, and Shuyu Sun. Generalized multiscale approximation of mixed finite elements with velocity elimination for subsurface flow. *Journal of Computational Physics*, 404:109133, 2020.

[2] Keinosuke Fukunaga and Warren LG Koontz. Application of the karhunen-loeve expansion to feature selection and ordering. *IEEE Transactions on computers*, 100(4): 311–318, 1970.

[3] Gene H Golub, James M Ortega, et al. *Scientific computing and differential equations: an introduction to numerical methods*. Academic press, 1992.

[4] ZiCheng Tao, ZhiNan Cui, JingQin Yu, and Majid Khayatnezhad. Finite difference modelings of groundwater flow for constructing artificial recharge structures. *Iranian Journal of Science and Technology, Transactions of Civil Engineering*, 46(2):1503–1514, 2022.

[5] Reza Fathollahi, Saman Hesaraki, Arsam Bostani, Ehsan Shahriyari, Hamid Shafiee, Pooya Pasha, Fateme Nadalinia Chari, and Davood Domiri Ganji. Applying numerical and computational methods to investigate the changes in the fluid parameters of the fluid passing over fins of different shapes with the finite element method. *International Journal of Thermofluids*, 15:100187, 2022.

[6] Asif Afzal, C Ahamed Saleel, K Prashantha, Suvanjan Bhattacharyya, and Mohammed Sadhikh. Parallel finite volume method-based fluid flow computations using openmp and cuda applying different schemes. *Journal of Thermal Analysis and Calorimetry*, 145(4):1891–1909, 2021.

[7] Che Han, Yu-Lan Wang, and Zhi-Yuan Li. Numerical solutions of space fractional variable-coefficient kdv–modified kdv equation by fourier spectral method. *Fractals*, 29(08):2150246, 2021.

[8] Akanksha Bhardwaj and Alpesh Kumar. A meshless method for time fractional nonlinear mixed diffusion and diffusion-wave equation. *Applied Numerical Mathematics*, 160:146–165, 2021.

[9] Ekaterina Stalgorova and Louis Mattar. Analytical model for history matching and forecasting production in multifrac composite systems. In *SPE Canadian Unconventional Resources Conference*. OnePetro, 2012.

[10] Hui Zeng, Dongyan Fan, Jun Yao, and Hai Sun. Pressure and rate transient analysis of composite shale gas reservoirs considering multiple mechanisms. *Journal of Natural Gas Science and Engineering*, 27:914–925, 2015.

[11] Jinghao Ji, Yuedong Yao, Shan Huang, Xiongqiang Ma, Shuang Zhang, and Fengzhu Zhang. Analytical model for production performance analysis of multi-fractured horizontal well in tight oil reservoirs. *Journal of Petroleum Science and Engineering*, 158: 380–397, 2017.

[12] Suran Wang, Linsong Cheng, Yongchao Xue, Shijun Huang, Yonghui Wu, Pin Jia, and Zheng Sun. A semi-analytical method for simulating two-phase flow performance of horizontal volatile oil wells in fractured carbonate reservoirs. *Energies*, 11(10):2700, 2018.

[13] Jin-Chang Li, Bin Yuan, Christopher R Clarkson, and Jian-Quan Tian. A semi-analytical rate-transient analysis model for light oil reservoirs exhibiting reservoir heterogeneity and multiphase flow. *Petroleum Science*, 2022.

[14] Ana C Azerêdo, Luís V Duarte, and Alexandre P Silva. The challenging carbonates from the pre-salt reservoirs offshore brazil: facies, palaeoenvironment and diagenesis. *Journal of South American Earth Sciences*, 108:103202, 2021.

[15] Jincong He, Jiang Xie, Xian-Huan Wen, and Wen Chen. An alternative proxy for history matching using proxy-for-data approach and reduced order modeling. *Journal of Petroleum Science and Engineering*, 146:392–399, 2016.

[16] Cong Xiao, Hai-Xiang Lin, Olwijn Leeuwenburgh, and Arnold Heemink. Surrogate-assisted inversion for large-scale history matching: Comparative study between projection-based reduced-order modeling and deep neural network. *Journal of Petroleum Science and Engineering*, 208:109287, 2022.

[17] Mostafa Ganjeh-Ghazvini. The impact of viscosity contrast on the error of heterogeneity loss in upscaling of geological models. *Journal of petroleum science and engineering*, 173:681–689, 2019.

[18] Xuefeng Liu, Jingxu Yan, Xiaowei Zhang, Lingtan Zhang, Hao Ni, Wei Zhou, Baojun Wei, Chaoliu Li, and Li-Yun Fu. Numerical upscaling of multi-mineral digital rocks: Electrical conductivities of tight sandstones. *Journal of Petroleum Science and Engineering*, 201:108530, 2021.

[19] Zhiming Chen and Thomas Hou. A mixed multiscale finite element method for elliptic problems with oscillating coefficients. *Mathematics of Computation*, 72(242):541–576, 2003.

[20] Patrick Jenny, Seong H Lee, and Hamdi A Tchelepi. Adaptive multiscale finite-volume method for multiphase flow and transport in porous media. *Multiscale Modeling & Simulation*, 3(1):50–64, 2005.

[21] Malgorzata Peszynska. Mortar adaptivity in mixed methods for flow in porous media. *Int. J. Numer. Anal. Model*, 2(3):241–282, 2005.

[22] Hadi Hajibeygi, Giuseppe Bonfigli, Marc Andre Hesse, and Patrick Jenny. Iterative multiscale finite-volume method. *Journal of Computational Physics*, 227(19):8604–8621, 2008.

[23] Benjamin Ganis, Danail Vassilev, ChangQing Wang, and Ivan Yotov. A multiscale flux basis for mortar mixed discretizations of stokes–darcy flows. *Computer Methods in Applied Mechanics and Engineering*, 313:259–278, 2017.

[24] Zhengkang He, Huangxin Chen, Jie Chen, and Zhangxin Chen. Generalized multiscale approximation of a mixed finite element method with velocity elimination for darcy flow in fractured porous media. *Computer Methods in Applied Mechanics and Engineering*, 381:113846, 2021.

[25] Marco A Cardoso and Louis J Durlofsky. Linearized reduced-order models for subsurface flow simulation. *Journal of Computational Physics*, 229(3):681–700, 2010.

[26] Mohammad-Reza Azad, Abolghasem Kamkar-Rouhani, Behzad Tokhmechi, and Mohammad Arashi. Hierarchical simultaneous upscaling of porosity and permeability features using the bandwidth of kernel function and wavelet transformation in two dimensions: Application to the spe-10 model. *Oil & Gas Science and Technology–Revue d'IFP Energies nouvelles*, 76:26, 2021.

[27] Yalchin Efendiev, Juan Galvis, and Thomas Y Hou. Generalized multiscale finite element methods (gmsfem). *Journal of computational physics*, 251:116–135, 2013.

[28] Thomas Y Hou and Xiao-Hui Wu. A multiscale finite element method for elliptic problems in composite materials and porous media. *Journal of computational physics*, 134(1):169–189, 1997.

[29] Zhangxin Chen. *Reservoir simulation: mathematical techniques in oil recovery*. SIAM, 2007.

[30] J Nagoor Kani and Ahmed H Elsheikh. Reduced-order modeling of subsurface multiphase flow models using deep residual recurrent neural networks. *Transport in Porous Media*, 126:713–741, 2019.

[31] J Nagoor Kani and Ahmed H Elsheikh. Dr-rnn: A deep residual recurrent neural network for model reduction. *arXiv preprint arXiv:1709.00939*, 2017.

[32] Rui Jiang and Louis J Durlofsky. Implementation and detailed assessment of a gnat reduced-order model for subsurface flow simulation. *Journal of Computational Physics*, 379:192–213, 2019.

[33] Mahdi Ghadiri, Azam Marjani, Reza Daneshfar, and Saeed Shirazian. Model order reduction of a reservoir simulation by sod-deim. *Journal of Petroleum Science and Engineering*, 200:108137, 2021.

[34] Jemimah-Sandra Samuel and Ann Helen Muggeridge. Fast modelling of gas reservoir performance with proper orthogonal decomposition based autoencoder and radial basis function non-intrusive reduced order models. *Journal of Petroleum Science and Engineering*, 211:110011, 2022.

[35] MV Behl and M Tyagi. Data-driven reduced-order models for volve field using reservoir simulation and physics-informed machine learning techniques. *SPE Reservoir Evaluation & Engineering*, pages 1–15, 2023.

[36] Roman Bohne. Machine-learning algorithms for the computation of upscaled permeabilities. Master's thesis, NTNU, 2018.

[37] Xupeng He, Ryan Santoso, and Hussein Hoteit. Application of machine-learning to construct equivalent continuum models from high-resolution discrete-fracture models. In *International Petroleum Technology Conference*. OnePetro, 2020.

[38] Hannah P Menke, Julien Maes, and Sebastian Geiger. Upscaling the porosity–permeability relationship of a microporous carbonate for darcy-scale flow with machine learning. *Scientific Reports*, 11(1):2625, 2021.

[39] Nandita Doloi, Somnath Ghosh, and Jyoti Phirani. Super-resolution reconstruction of reservoir saturation map with physical constraints using generative adversarial network. In *SPE Reservoir Characterisation and Simulation Conference and Exhibition*. OnePetro, 2023.

[40] Chenhong Zhu, Jianguo Wang, Shuxun Sang, and Wei Liang. A multiscale neural network model for the prediction on the equivalent permeability of discrete fracture network. *Journal of Petroleum Science and Engineering*, 220:111186, 2023.

[41] Shing Chan and Ahmed H Elsheikh. A machine learning approach for efficient uncertainty quantification using multiscale methods. *Journal of Computational Physics*, 354:493–511, 2018.

[42] Patrick Jenny, SH Lee, and Hamdi A Tchelepi. Multi-scale finite-volume method for elliptic problems in subsurface flow simulation. *Journal of computational physics*, 187 (1):47–67, 2003.

[43] Min Wang, Siu Wun Cheung, Eric T Chung, Yalchin Efendiev, Wing Tat Leung, and Yating Wang. Prediction of discretization of gmsfem using deep learning. *Mathematics*, 7(5):412, 2019.

[44] Min Wang, Siu Wun Cheung, Wing Tat Leung, Eric T Chung, Yalchin Efendiev, and Mary Wheeler. Reduced-order deep learning for flow dynamics. the interplay between deep learning and model reduction. *Journal of Computational Physics*, 401:108939, 2020.

[45] Eric T Chung, Yalchin Efendiev, Wing Tat Leung, Maria Vasilyeva, and Yating Wang. Non-local multi-continua upscaling for flows in heterogeneous fractured media. *Journal of Computational Physics*, 372:22–34, 2018.

[46] Yating Wang and Guang Lin. Efficient deep learning techniques for multiphase flow simulation in heterogeneous porousc media. *Journal of Computational Physics*, 401: 108968, 2020.

[47] Eric Chung, Yalchin Efendiev, Wing Tat Leung, Sai-Mang Pun, and Zecheng Zhang. Multi-agent reinforcement learning accelerated mcmc on multiscale inversion problem. *arXiv preprint arXiv:2011.08954*, 2020.

[48] Zecheng Zhang, Eric T Chung, Yalchin Efendiev, and Wing Tat Leung. Learning algorithms for coarsening uncertainty space and applications to multiscale simulations. *Mathematics*, 8(5):720, 2020.

[49] Yating Wang, Siu Wun Cheung, Eric T Chung, Yalchin Efendiev, and Min Wang. Deep multiscale model learning. *Journal of Computational Physics*, 406:109071, 2020.

[50] Nanzhe Wang, Dongxiao Zhang, Haibin Chang, and Heng Li. Deep learning of subsurface flow via theory-guided neural network. *Journal of Hydrology*, 584:124700, 2020.

[51] Eric Chung, Wing Tat Leung, Sai-Mang Pun, and Zecheng Zhang. A multi-stage deep learning based algorithm for multiscale model reduction. *Journal of Computational and Applied Mathematics*, 394:113506, 2021.

[52] Xiaofei Guan, Lijian Jiang, and Yajun Wang. Multiscale model reduction for stochastic elasticity problems using ensemble variable-separated method. *Journal of Computational and Applied Mathematics*, 421:114895, 2023.

[53] F Pooya Nejad, Mark B Jaksa, M Kakhi, and Bryan A McCabe. Prediction of pile settlement using artificial neural networks based on standard penetration test data. *Computers and geotechnics*, 36(7):1125–1133, 2009.

[54] Razieh Razavi, Amin Bemani, Alireza Baghban, Amir H Mohammadi, and Sajjad Habibzadeh. An insight into the estimation of fatty acid methyl ester based biodiesel properties using a lssvm model. *Fuel*, 243:133–141, 2019.

[55] Hamed Moosanezhad-Kermani, Farzaneh Rezaei, Abdolhossein Hemmati-Sarapardeh, Shahab S Band, and Amir Mosavi. Modeling of carbon dioxide solubility in ionic liquids based on group method of data handling. *Engineering Applications of Computational Fluid Mechanics*, 15(1):23–42, 2021.

[56] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks: an overview and application in radiology. *Insights into imaging*, 9:611–629, 2018.

[57] Mohamed Elgendy. *Deep learning for vision systems*. Simon and Schuster, 2020.

[58] Harry Pratt, Frans Coenen, Deborah M Broadbent, Simon P Harding, and Yalin Zheng. Convolutional neural networks for diabetic retinopathy. *Procedia computer science*, 90:200–205, 2016.

[59] Wenbo Zhang, Sifan Song, Tianming Bai, Yanxin Zhao, Fei Ma, Jionglong Su, and Limin Yu. Chromosome classification with convolutional neural network based deep learning. In *2018 11th international congress on image and signal processing, biomedical engineering and informatics (CISP-BMEI)*, pages 1–5. IEEE, 2018.

[60] Jiuyu Zhao, Fuyong Wang, and Jianchao Cai. 3d tight sandstone digital rock reconstruction with deep learning. *Journal of Petroleum Science and Engineering*, 207: 109020, 2021.

[61] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[62] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[63] Jürgen Schmidhuber. Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4(2):234–242, 1992.

[64] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[65] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

[66] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. Pmlr, 2013.

[67] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[68] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[69] Vincent Michalski, Vikram Voleti, Samira Ebrahimi Kahou, Anthony Ortiz, Pascal Vincent, Chris Pal, and Doina Precup. An empirical study of batch normalization and group normalization in conditional computation. *arXiv preprint arXiv:1908.00061*, 2019.

[70] 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.

[71] Zheng Hu, Jiaojiao Zhang, and Yun Ge. Handling vanishing gradient problem using artificial derivative. *IEEE Access*, 9:22371–22377, 2021.

[72] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[73] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[74] Michal Drozdzal, Eugene Vorontsov, Gabriel Chartrand, Samuel Kadoury, and Chris Pal. The importance of skip connections in biomedical image segmentation. In *International Workshop on Deep Learning in Medical Image Analysis, International Workshop on Large-Scale Annotation of Biomedical Data and Expert Label Synthesis*, pages 179–187. Springer, 2016.

[75] Hyeongyeom Ahn and Changhoon Yim. Convolutional neural networks using skip connections with layer groups for super-resolution image reconstruction based on deep learning. *Applied Sciences*, 10(6):1959, 2020.

[76] Lijun Wu, Xu Lin, Zhicong Chen, Jingchang Huang, Huawei Liu, and Yang Yang. An efficient binary convolutional neural network with numerous skip connections for fog computing. *IEEE Internet of Things Journal*, 8(14):11357–11367, 2021.

[77] Xin Jin, Yide Di, Qian Jiang, Xing Chu, Qing Duan, Shaowen Yao, and Wei Zhou. Image colorization using deep convolutional auto-encoder with multi-skip connections. *Soft Computing*, pages 1–16, 2022.

[78] G Yogeswararao, V Naresh, R Malmathanraj, and P Palanisamy. An efficient densely connected convolutional neural network for identification of plant diseases. *Multimedia Tools and Applications*, 81(23):32791–32816, 2022.

[79] Dong Chen, Fan Hu, P Takis Mathiopoulos, Zhenxin Zhang, and Jiju Peethambaran. Mc-unet: Martian crater segmentation at semantic and instance levels using u-net-based convolutional neural network. *Remote Sensing*, 15(1):266, 2023.

[80] Gang Wang, Jinxing Hao, Jian Ma, and Hongbing Jiang. A comparative assessment of ensemble learning for credit scoring. *Expert systems with applications*, 38(1):223–230, 2011.

[81] Gang Wang, Jianshan Sun, Jian Ma, Kaiquan Xu, and Jibao Gu. Sentiment classification: The contribution of ensemble learning. *Decision support systems*, 57:77–93, 2014.

[82] Minh Nguyen and Doina Logofătu. Applying tree ensemble to detect anomalies in real-world water composition dataset. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 429–438. Springer, 2018.

[83] Eiichiro Kanda, Bogdan I Epureanu, Taiji Adachi, Yuki Tsuruta, Kan Kikuchi, Naoki Kashihara, Masanori Abe, Ikuto Masakane, and Kosaku Nitta. Application of explainable ensemble artificial intelligence model to categorization of hemodialysis-patient and treatment using nationwide-real-world data in japan. *Plos one*, 15(5):e0233491, 2020.

[84] Kun Zhang, Ning Chen, Jian Liu, and Michael Beer. A gru-based ensemble learning method for time-variant uncertain structural response analysis. *Computer Methods in Applied Mechanics and Engineering*, 391:114516, 2022.

[85] Yicheng Zhou, Zhenzhou Lu, and Kai Cheng. Adaboost-based ensemble of polynomial chaos expansion with adaptive sampling. *Computer Methods in Applied Mechanics and Engineering*, 388:114238, 2022.

[86] Robert E Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

[87] Pratima Kumari and Durga Toshniwal. Extreme gradient boosting and deep neural network based ensemble learning approach to forecast hourly solar irradiance. *Journal of Cleaner Production*, 279:123285, 2021.

[88] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[89] GÖKSU Tüysüzoğlu and Derya Birant. Enhanced bagging (ebagging): A novel approach for ensemble learning. *International Arab Journal of Information Technology*, 17(4), 2020.

[90] Leo Breiman. Stacked regressions. *Machine learning*, 24(1):49–64, 1996.

[91] Xin Yin, Quansheng Liu, Yucong Pan, Xing Huang, Jian Wu, and Xinyu Wang. Strength of stacking technique of ensemble learning in rockburst prediction with imbalanced data: Comparison of eight single and ensemble models. *Natural Resources Research*, 30:1795–1815, 2021.

[92] Hyun-Joo Oh, Mutiara Syifa, Chang-Wook Lee, and Saro Lee. Land subsidence susceptibility mapping using bayesian, functional, and meta-ensemble machine learning models. *Applied Sciences*, 9(6):1248, 2019.

[93] Navid Kardani, Annan Zhou, Majidreza Nazem, and Shui-Long Shen. Improved prediction of slope stability using a hybrid stacking ensemble method based on finite element analysis and field data. *Journal of Rock Mechanics and Geotechnical Engineering*, 13 (1):188–201, 2021.

[94] Zhongqiang Liu, Graham Gilbert, Jose Mauricio Cepeda, Asgeir Olaf Kydland Lysdahl, Luca Piciullo, Heidi Hefre, and Suzanne Lacasse. Modelling of shallow landslides with machine learning algorithms. *Geoscience Frontiers*, 12(1):385–393, 2021.

[95] Lucas Abreu Blanes de Oliveira and Cleyton de Carvalho Carneiro. Synthetic geochemical well logs generation using ensemble machine learning techniques for the brazilian pre-salt reservoirs. *Journal of Petroleum Science and Engineering*, 196:108080, 2021.

[96] Bingguan Liu, Zhilin Wang, Yong Jin, Zhengjun Ge, Chenghao Xu, Haipeng Liu, and Hao Chen. Novel way to predict the mmp of a co2–oil system using stacking models. *Energy & Fuels*, 2023.

[97] Caifeng Zou, Luanxiao Zhao, Fei Hong, Yirong Wang, Yuanyuan Chen, and Jianhua Geng. A comparison of machine learning methods to predict porosity in carbonate reservoirs from seismic-derived elastic properties. *Geophysics*, 88(2):B101–B120, 2023.

[98] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya, et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297, 2021.

[99] Carl Edward Rasmussen, Christopher KI Williams, et al. *Gaussian processes for machine learning*, volume 1. Springer, 2006.

[100] Neil Lawrence and Aapo Hyvärinen. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *Journal of machine learning research*, 6(11), 2005.

[101] Neil D Lawrence and Andrew J Moore. Hierarchical gaussian process latent variable models. In *Proceedings of the 24th international conference on Machine learning*, pages 481–488, 2007.

[102] Michalis Titsias and Neil D Lawrence. Bayesian gaussian process latent variable model. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 844–851. JMLR Workshop and Conference Proceedings, 2010.

[103] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[104] Tao Zhao and Xiaoli Chen. Enrich the interpretation of seismic image segmentation by estimating epistemic uncertainty. In *SEG Technical Program Expanded Abstracts 2020*, pages 1444–1448. Society of Exploration Geophysicists, 2020.

[105] Runhai Feng, Dario Grana, and Niels Balling. Uncertainty quantification in fault detection using convolutional neural networks. *Geophysics*, 86(3):M41–M48, 2021.

[106] Evan Schankee Um, David Alumbaugh, Youzuo Lin, and Shihang Feng. Real-time deep-learning inversion of seismic full waveform data for co2 saturation and uncertainty in geological carbon storage monitoring. *Geophysical Prospecting*, 2022.

[107] Haibin Di and Aria Abubakar. Estimating subsurface properties using a semisupervised neural network approach. *Geophysics*, 87(1):IM1–IM10, 2022.

[108] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[109] A Ronald Gallant and Halbert White. There exists a neural network that does not make avoidable mistakes. In *ICNN*, pages 657–664, 1988.

[110] Adrian Silvescu. Fourier neural networks. In *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No. 99CH36339)*, volume 1, pages 488–491. IEEE, 1999.

[111] Shuang Liu. Fourier neural network for machine learning. In *2013 International Conference on Machine Learning and Cybernetics*, volume 1, pages 285–290. IEEE, 2013.

[112] Gege Wen, Zongyi Li, Kamyar Azizzadenesheli, Anima Anandkumar, and Sally M Benson. U-fno—an enhanced fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, 163:104180, 2022.

[113] Meer Mehran Rashid, Tanu Pittie, Souvik Chakraborty, and NM Anoop Krishnan. Learning the stress-strain fields in digital composites using fourier neural operator. *iScience*, page 105452, 2022.

[114] Zhijie Li, Wenhui Peng, Zelong Yuan, and Jianchun Wang. Fourier neural operator approach to large eddy simulation of three-dimensional turbulence. *Theoretical and Applied Mechanics Letters*, page 100389, 2022.

[115] Vladimir N Vapnik. An overview of statistical learning theory. *IEEE transactions on neural networks*, 10(5):988–999, 1999.

[116] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378: 686–707, 2019.

[117] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020.