STATISTICAL AND MACHINE LEARNING MODELS FOR THE EVALUATION OF GEOPHYSICAL AND GEOMECHANICAL DATA

Anil Kumar

STATISTICAL AND MACHINE LEARNING MODELS FOR THE EVALUATION OF GEOPHYSICAL AND GEOMECHANICAL DATA

Submitted in partial fulfillment of the requirements
of the degree of

Doctor of Philosophy

of the
Indian Institute of Technology Bombay, India
and
Monash University, Australia

by

Anil Kumar

Supervisors:

Prof. Kumar Hemant Singh
A/Prof. Mohan Yellishetty
Prof. Trilok Nath Singh

The course of study for this award was developed jointly by
Monash University, Australia and the Indian Institute of Technology Bombay, India
and was given academic recognition by each of them.
The programme was administrated by The IITB-Monash Research Academy

(2022)

Dedicated to my parents

DECLARATION

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Student Name: **Anil Kumar**
IITB ID:
Monash ID:

Approval Sheet

The thesis entitled "Statistical and Machine Learning Models for the Evaluation of Geophysical and Geomechanical Data" by Anil Kumar is approved for the degree of **Doctor of Philosophy**.

Sagarika Mukhopadhyay
Professor
Department of Earth Sciences
Indian Institute of Technology Roorkee
(External Examiner)

Munukutla Radhakrishna
Professor
Department of Earth Sciences
Indian Institute of Technology Bombay
(Internal Examiner)

Kumar Hemant Singh
Professor
Department of Earth Sciences
Indian Institute of Technology Bombay
(IITB Supervisor)

Mohan Yellishetty
Associate Professor
Department of Civil Engineering
Monash University, Australia
(Monash Supervisor)

Trilok Nath Singh
Professor
Department of Earth Sciences
Indian Institute of Technology Bombay
(IITB Co-Supervisor)

J. Adinarayana
Professor
Centre of Studies in Resources Engineering
Indian Institute of Technology Bombay
(Chairperson)

Date:
Place:

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my advisors, Prof Kumar Hemant Singh, Prof Mohan Yellishetty and Prof Trilok Nath Singh, for being an incredible guide, teacher and support and for providing me with an opportunity to work at immensely intellectual environments at Indian Institute of Technology Bombay and Monash University. I want to thank them for their encouragement and ideas, without which this work would not have been possible.
I would like to thank my research committee member, Prof. Munukuntla Radhakrishna, for his valuable feedback and suggestions during my progress seminars.
I would also like to thank Prof. Stuart D.C. Walsh for his enlightening discussions during the development of the reduced order modelling chapter.
Pursuing my doctoral studies at the IITB-Monash Research Academy has given me a very enriching experience. I thank all the working staff at the academy for making this journey a fulfilling one. At last, I would like to thank my lab mates who helped me look into some of the problems from their perspective. I take this opportunity to express my sincere thanks and appreciation to all those people.

Anil Kumar

Abstract

Physical laws govern geophysical processes, and to study them, they are generally formulated as a *forward model*. This forward model is often used as a synthetic data generator in an *inversion* setup wherein an optimizer minimizes the difference between the synthetic and field observed data. When either part of the setup is computationally large, it can delay the generation of solutions and, consequently, the interpretation process. We understand that some classes of algorithms from machine learning, such as supervised and dimensionality reduction algorithms, can help get around these situations. The main idea behind this work is to motivate the inclusion of machine learning algorithms into conventional geophysical modelling workflow to speed up the data generation process and aid automatic interpretation. In the first part of the thesis, we outline the theory underlying some general deep learning algorithms that are later used in the chapters. In the second part, we develop deep learning models for two different applications.

In the first application, synthetic data is generated using an acoustic wave simulator to study the response of pore geometry on an acoustic wavefield. We developed a semantic segmentation framework using a modified U-Net architecture that could directly output the pore structure from the acoustic volume. It was found that while the overall pore structure model could be inferred with a Dice coefficient accuracy of 92%, the trained model struggled to predict the individual classes.

In the second application, we develop an LSTM based reduced order model for predicting the future states of a hydro-mechanical system. The data for this model was generated using a linear elastic finite element solver. High non-linearity was introduced into the model by varying the pore pressure contribution in different parts of the computational domain. It was found that a 3-layer deep RNN was required to maintain a 99.99% accuracy in the predicted states. We understand that the development of such a model can help evaluate the present and near-future states of the stability of the slopes.

Keywords: Semantic segmentation, Reduced order model, Geophysical simulation, Geomechanical simulation, Machine learning

Contents

List of Figures

List of Tables

Abbreviations

3D-MPA	Multi Proposal Aggregation for 3D point-clouds
4D-XRM	4 Dimensional X-ray microscopy
ADAGRAD	Adaptive gradient
ADAM	Adaptive moment estimation
AGCN	Attention-based graph convolution networks
AI	Artificial intelligence
BGD	Batch gradient descent
Bi-LSTM	Bi-directional long short-term memory
BL	Body load
BPTT	Backpropagation through time
CAD	Computer-aided design
CEC	Constant error carousel
CNN	Convolutional neural network
CPU	Central processing unit
DFN	Discriminative feature network
DGCN	Deep graph convolutional network
DL	Deep learning
DMD	Dynamic mode decomposition
FCNN	Fully connected neural network
FEM	Finite element model

FOS	Factor of safety
GCN	Global convolutional network
GeoNet	Geometric Neural Network
GPU	Graphics processing unit
GRU	Gated recurrent unit
HMC	Hamiltonian Monte Carlo
HMM	Hidden Markov model
HPN	Hierarchical parsing network
ICA	Independent component analysis
K-SVD	Kernel singular value decomposition
Kd-tree	K dimensional tree
LSTM	Long short-term memory
MAP	Maximum a-posteriori estimation
MBGD	Mini-batch gradient descent
ML	Machine learning
MLE	Maximum likelihood estimation
NADAM	Nesterov accelerated adaptive moment estimation
NAG	Nesterov accelerated gradient
NUTS	No-U-Turn sampler
O-cnn	Octree- based convolutional neural networks
OccuSeg	Occupancy-aware 3D instance segmentation
PCA	Principal component analysis
PCNN	Point-cloud convolutional neural network
PDE	Partial differential equation
PGCRNet	Pont global context reasoning network

POD	Proper orthogonal decomposition
PSANet	Point-wise spatial attention network for scene parsing
PSPNet	Pyramid scene parsing network
ReLU	Rectified linear unit
ResNet	Residual neural network
RGB	Red green blue
RL	Reinforcement learning
RMSPROP	Root mean squared propagation
RNN	Recurrent neural netowrk
ROM	Reduced order model
RSNet	Remote sensing deep neural network
SFCN	Shape fully convolutional networks
SGD	Stochastic gradient descent
SGPN	Similarity group proposal network
SPG	Super-point graph
SVD	Singular value decomposition
VGG-16	Visual geometry group's 16 layer model
VoxSegNet	Volumetric CNNs for semantic part segmentation

Chapter 1

Introduction

Geoscientific data can provide one of the best modelling adventures for a scientist. Processing of data in geoscientific disciplines often involves the usage of computers to *determine the most* of the interior of the earth, where direct penetration tests are unpractical and/or uneconomic. Standard geophysical prospecting techniques, such as electromagnetics, seismic, and gravity/magnetic help gather a small signature, in the form of field data, of the underlying processes that generated it. Scientists often try to match these observations to well known analytical solutions to claim their findings. But it must not be forgotten that this data is just a tiny representation of the process or group of processes that could have generated it, so there is always an associated uncertainty. The processes that we study in geophysics and geomechanics require us to use the laws of physics that govern them, followed by encoding the process in the form of a *forward model*. These models are often used as synthetic data generators to be compared against the field observed data in an *inversion* setup. Although doable by experts, it is often cumbersome for a human to apply and record the outputs and interpret the data at each stage of processing, especially when the dimension of data is large, necessitating the application of computer programs while the generated results are checked in the interim at each intermediate step. When the data is small, an expert can safely avoid any computational procedure to arrive at the result. But, the current trend in data acquisition, their type, variety and amount make it an utmost requirement to learn and apply custom developed computer codes to extract more information from them.

Machine learning (ML) is a branch of science where computer algorithms learn from data. While it has shown human level accuracy in tasks such as object detection, computer game play and natural language processing, it is slowly being included in the mainstream data pipeline for Geoscience studies and a variety of geophysical data processing applications. For example the authors in [1] imple-

mented a physics informed neural network architecture from well to seismic tie using CNNs. A deep learning based segmentation model was realised using the SegNet architecture in [2] to approximate the inverse operator from 3D apparent resistivity data. A CNN network was designed to impose regularisation to inverse problems in [3]. Authors in [4] employed a U-Net model to interpret ground penetrating radar data for archaeo-geophysical applications. Authors in [5] devised a CNN based earthquake early warning system that could learn the source and site effects from the waveform data to forecast the intensity. Five architectures were compared for reservoir characterisation using geophysical well logs in [6]. A deep learning based seismic impedance inversion was carried out in [7] using a Cycle-GAN architecture. An automatic fault detection algorithm was realised in [8; 9] that used 2D slices extracted from 3D seismic amplitude data. A CNN framework was designed in [10] to predict the shear wave velocity profile from surface wave data. Automatic velocity picking was attempted in [11] using semblance maps. Seismic reservoir characterisation was done using deep CNNs in [12]. Geophysical inversion to detect near surface cavern was done in [13] using deep frameworks.
ML algorithms, in general, require a large amount of data, adequately preprocessed for their application. As a matter of fact, we in geophysics and geomechanics perform many synthetic studies in the form of simulation and forward models that can be leveraged to train an ML algorithm. This requires knowledge of the underlying physics of the process and the machinery of the ML algorithm to work in tandem. Although the human interpretation of data in this field is indispensable and necessary for gaining confidence in the obtained results, nonetheless, artificial intelligence can be made to complement the interpreter at various stages of data processing. In the light of the above, this thesis attempts to depict some of the advantages of using ML algorithms in processing synthetic data. We demonstrate this in the form of two important and exciting applications pertaining to geophysics and geomechanics.

1.1 Conventional Data modeling

Modelling of any geophysical process involves a physical theory that connects a discrete set of Earth parameters $m \in \mathfrak{M} \subset \mathbb{R}^M$ to the experimentally obtained data $d \in \mathfrak{D} \subset \mathbb{R}^N$. The theory gives us a mathematical operator g to predict the data d for a given set of parameters.

$$d = g(m) + \eta \tag{1.1}$$

where η represents the sum of all errors during data acquisition and the errors due to physics not accounted for in theory. The operator g can have several forms depending on the process used to represent the theory. For example, in a 1D setup, a time domain electromagnetic field modelling g represents an operator that takes the resistivity and width of layers of a layered Earth model and solves Maxwell's equations to give the electric and magnetic fields. An optimisation algorithm uses the forward model g to solve these equations numerically to obtain the best possible set of parameter pairs that yields the least error against the observed data. Another example that g can be representative of is a well-designed Green's function serving as the analytical solution to wave propagation problems [14; 15].
An equivalent example from the geomechanical field is the modelling of the stability of the borehole. Here, the operator g of Eq. 1.1 represents the set of partial differential equations governing the stress and strains in a finite computational domain. The parameters m of the model are the engineering quantities used to specify the different types of materials constituting the domain. Advanced numerical modelling techniques are often used to solve these kinds of problems [16; 17].

Solving Eq. 1.1 generally yields multidimensional data that can be interpreted for a variety of purposes. Before performing the data analysis, an important step is to arrange the data points in a specific order and position in the data space. From the data science perspective, the criteria for positioning these points may not have any practical significance. For example, a particular data ordering may assign different weights to different data features. This ordering might not seem important as it may not give better solutions, but it may still ease some parts of the matrix calculations. Notably enough, dealing with geophysical and geomechanical data using ML algorithms often requires projecting the data points to a lower dimension so that essential features embedded within are captured. While this is true for most real-world ML applications, it is paramount for geo-type data, especially when the amount of given data is large.
Data in Geosciences often demand multidisciplinary approaches to processing, analysis and interpretation. While geomechanics, in particular, deals with the mechanism of soil and subsurface, geophysics spans a broader scope with subfields of study such as gravity, magnetic, seismic, electromagnetic, radioactivity, well-logging and various chemical and thermal processes. Modelling in any of these areas involves acquiring information from the field and fitting a synthetic model to it. The modeller has complete control over the numerical experiment being conducted by varying the parameters of the synthetic model. Partial Differential Equations (PDEs) are ubiquitous tools in these domains. The data acquired

from the field are primarily used for calibrating these computational models. As modellers, we understand that real data has its challenges that present themselves in the form of noise, heterogeneity, arbitrary computational domain shapes, and nonlinearity, which often result in the ill-posedness of the inverse problem.
With the success of artificial intelligence in scientific applications such as computer vision, language understanding, and logical reasoning, it is evident that this technology can help enhance the conventional optimisation based models. The ML aided models have proven to be capable of learning from the data, deducing patterns and making future predictions. The non-triviality of geophysical and geomechanical data offers a perfect avenue for its application.

1.2 Machine Learning for Data modeling

Eq. 1.1 depicts a general problem in imaging commonly encountered in geophysics. The equation can also be represented in a form that is more common in the Deep Learning (DL) literature:

$$d = \mathcal{N}(g(m)) \tag{1.2}$$

where $\mathcal{N}(\cdot)$ represents a Gaussian distributed noisy measurements of the data generating process where the noise term is no more additive, unlike Eq. 1.1. The model is in much use in computational imaging [18] tasks such as image inpainting and super-resolution [19]. Other image application areas are tomographical reconstruction, radar imaging, magnetic resonance imaging [20] and X-ray tomography [21]. Conventionally the reconstruction task involved some prior information about m in terms of sparsity [22; 23; 24; 25] or in terms of geometry [26; 27; 28; 29; 30]. In conventional processing, a good estimation of m has to be a good fit and adhere to the prior information. This is generally posed as an optimisation problem, often involving a regulariser that sets a trade-off between the generalisation capacity and overfitting of the algorithm. This approach has recently been changing with the arrival of deep learning techniques. A deep learning algorithm can leverage a large set of labelled data to create a regularised mapping between the dependent and the independent variables. A number of such applications can be found in signal recovery, image super-resolution, compressed sensing, and image denoising [31; 32; 33; 34; 35; 36; 37].

Supervised ML algorithms require labelled examples to map the covariates to the target variables. A simple example is identifying the digits in the area zip

codes, also called pin codes in some countries. A naive approach to such a problem would involve hand-crafting the rules to detect specific lines, curves, and shapes in specific picture regions. Though simple to understand, this approach leads to a large number of rules and still does not provide good results, if not poor. A smarter way of achieving fewer errors can be a ML-based algorithm wherein we supply input images with different writing styles of the numeric digits printed on them and provide their corresponding labelled outputs, i.e. the actual one-hot vectors of the actual answer. Symbolically, we can say that the image $X \in (x_1, x_2, \ldots, x_n)$ and its label $y \in \vec{y}$, where $\vec{y}$ is a one-hot binary vector of length 10 whose element $y_i = 1$ if the supplied query digit belongs to i^{th} position. These example sets are used to train the parameters of an ML model. Creating the labelled set is typically done by a human expert and is time-consuming. The way the algorithm tunes its parameters is by minimising a cost function. The algorithm takes a new image as the input and generates an output encoded in the same manner as the labels y in the supervised set. The algorithm then checks the value of the cost function corresponding to the current output and generates an error value used to tune the parameters. This step is also referred to as the learning/training phase. After the training phase, the trained model is typically checked against a validation set containing digits that have not been shown to the model yet. The capability of the trained model to predict the correct labels in this new set is called its generalisation capability.

The central aim of training any ML algorithm is to improve its generalisation capability. An important step towards achieving this is the preprocessing stage. This step takes care of the unequal variance in the input feature variables and is done to enhance its generalisation capacity. Because now the variability of each new example is fixed/equalised, the task of pattern recognition becomes easier. Another operation that is done as a part of preprocessing is feature extraction. This part helps enhance training by transforming the input variables into new features. A simple example can be squaring the input variable x to be appended as a new feature, wherein the model's output y is some function of (x, x^2). Here, the base variable x is acted upon by function $f(x) = x^2$ to produce a new feature variable x^2. Combining this new variable with the original variable x helps regress faster and ease the optimiser's burden. It is obvious that feature transformation/extraction also helps boost computation in graphics processing units (GPUs), typically preferred for high-end processing. An example of such a requirement is processing streaming video data for face detection. A direct feed of the frames in the stream might unnecessarily overload the GPUs in identifying complex facial features; instead, the frame could pass through a deterministic feature extractor first and then

the feature vector could be passed on to the ML algorithm. The feature extractor could focus on important areas of the frame that would increase its discriminatory function enabling face detection. An example could be of detecting the RGB values possessing a particular range of intensity that are symbolic of a face [38; 39; 40]. A similar set of other features could be hand-designed to enable efficient face detection. Because this step greatly reduced the amount of data to be processed, it is also sometimes called the dimensionality reduction step. As this is subtractive, step care must be taken to prevent the discarding of relevant features; otherwise, the entire training procedure would suffer. It is understood that the parameters of the feature extractor must be tuned *a-priori* to enable efficient extraction – this is yet another area of active research.

There are at least two categories in which the ML algorithm could be grouped – a classifier or a regressor. In problems where the target variable retains specific values or classes, the problem can be framed as a classification problem; on the other hand, when the value of the output variable lies in the continuous range, the problem can be formulated as a regression problem. Yet, another class of problem exists in the ML literature that consists of the unsupervised ML algorithms. In this class of problems, the algorithm is subjected to identifying relevant/important features in the input variables, i.e. without any labelled set, y. Common tasks here are density estimation, where the algorithm is assigned the job of estimating the probability distribution of the given dataset; clustering, wherein the algorithm is expected to discover members of a similar group based on some similarity measure; dimensionality reduction – where the algorithm's job is to reduce the dimension of the data without losing essential information within. This is also called data compression in some literature.

Yet another branch of ML algorithms deals with the task of choosing suitable actions given a specific situation. The algorithms here maximise the rewards obtained from selecting a particular action. This branch has been suitably named reinforcement learning (RL) [41; 42]. This is also the case where the user does not supply any form of supervised inputs except the design of a suitable reward function whose return varies according to the choice made by the playing *agent*. This is usually accomplished by a trial and error process in a given environment. The states and the set of actions are typically pre-designed. The selected action not only affects the current reward but also affects the subsequent rewards. Therefore, the design of the reward function is quite important. Successful implementation can be seen in a neural network that is trained using the RL technique to play games like chess, go, and backgammon [43; 44; 45]. The rewards must be chosen carefully to maximise efficiency as quickly as possible. As they call it in the RL

literature, an algorithm must tune the trade-off between exploration and exploitation. Biasing towards any one of them can yield poor results.

1.3 Motivation of Research

Geoscientific problems present a plethora of modelling opportunities. In geophysical modelling, we mainly deal with optimisation, wherein we aim to evaluate the variables of interest given their responses. These problems are primarily nonlinear and often possess uncertainty in their solutions. For example, in modelling the elastic response of a rock core sample, we are especially interested in knowing the internal pore structure and its constituents. Minute details such as tiny pores do not result in appreciable changes in the elastic responses. So, our methodology would be wasted on this problem if more attention is given to resolving the tiny pores' effect. There is a need that the algorithms used to optimise the variables of interest must be smart enough to retain as much detail as possible while still ignoring the irrelevant parts. Another inherent problem is non-uniqueness, wherein multiple pore network configurations and constituents can result in similar elastic responses. In this scenario, the algorithms must be able to work out the differences between the responses in at least some higher dimension, if not the same. At this juncture, it is essential to note that geoscientists and particularly the reservoir engineers are primarily interested in approximate solutions to these problems such that the bulk of the domain can be comprehended/estimated for asset evaluation. On the other hand, in geomechanics, we often build a numerical model to study physical phenomena represented using a set of partial differential equations. The discretisation of the equations results in a large system of algebraic equations that need to be solved iteratively. This size is directly proportional to the resolution of the computational domain, and fine results mandate large meshes or grids to better approximate the actual solution. This is because direct matrix inversion is impractical as it would require enormous physical memory, which is generally not feasible. Therefore, discretisation methods like Finite differences, Finite Elements and Finite Volume exist that can help divide the computational domain into small elements and solve sub-problems separately. An important issue with solving such a large system of equations is the time required to get a single solution. At this juncture, the dimensionality reduction algorithms might help speed up the modelling pipeline. This class of algorithms can help reduce the dimensionality of the problems using reduced rank representations. While these algorithms capture most of the variances in the data, it's up to the user how they tune it such that

the resolution in the variables of interest is retained. To this end, linear dimensionality reduction methods have been developed and are being successfully used to deal with and interpret high-dimensional data. However, although the linear methods are fast in factorising these large amounts of data, decomposition ought to be ambiguous when the data lies in the nonlinear space. Therefore, one must choose these data reduction techniques judiciously.
Therefore, we see some specific properties within the data that can direct our efforts towards designing new and efficient algorithms, which can perform equally well, if not better, than the current approaches. Hence, this work demonstrates the development of those data-driven models that adapt to the problem specified as the user requirements.

1.4 Aims and Objectives

The main idea behind this work is to motivate the inclusion of ML algorithms into conventional geophysical and geomechanical modelling workflow. We aim to solve two different problems that utilise modern deep learning algorithms to accomplish the set tasks. We divide the work into two main chapters – chapter 3 and chapter 4.
In chapter 3, we have worked with Convolutional Neural Network (CNN) architectures to characterise the pore networks in carbonate rocks. To do this, we define the following objectives:

1. To prepare the volumetric image data of rock core samples
    i. To convert the raw data into workable formats
    ii. To study various preprocessing and binarisation algorithms
    iii. To extract the pore networks from rock core samples
2. To analyse the extracted pore networks using acoustic wave simulation
    i. To realise an acoustic wave simulator using finite difference technique
    ii. To analyse the patterns exhibited by different pore network models
3. To map the pore network models to their acoustic responses using deep neural network
    i. To develop a U-Net architecture for mapping the acoustic volumes to the causative pore network model

ii. To evaluate the performance of the developed network using different metrics

In chapter 4, we show how Long Short-Term Memory (LSTM) based Recurrent Neural Networks (RNN) can be used to realise a Reduced Order Model (ROM) for accelerated simulation of a hydro-mechanical system. Hence, the objectives are:

1. To develop a numerical model of the hydro-mechanical system
   i. To define the computational domain for the study
   ii. To generate state space data from few full-order simulation
2. To develop a Multistep Prediction algorithm using deep neural network
   i. To create a supervised training and testing dataset
   ii. To reduce the dimension of data using proper orthogonal decomposition
   iii. To realise a multi-step prediction algorithm using single-step prediction scheme
3. To perform some statistical analyses of the developed ROM
   i. To compare the predicted states to the observed states
   ii. To analyse the performance of the developed ROM

1.5 Organisation of the Remaining Chapters

The current chapter gave a general introduction to the ML algorithms that fit into conventional data processing workflow. The rest of the thesis can be organised as follows.

Chapter 2: General Algorithms in Deep Learning and Statistical Estimation. In this chapter, we provide a brief overview of some of the general deep learning algorithms that we use in the forthcoming chapters. We also mention different loss functions that can be used for training these algorithms. We also mention the statistical estimation techniques such as Maximum *a-posteriori* Estimation and Maximum likelihood estimation.

Chapter 3: A Data-Driven Approach to Porosity Segmentation for Carbonates. This chapter presents the results of predicting the pore network model from acoustic volume data. We go through the several data preparation steps as it is the most crucial part before illustrating the training of the U-Net. We first show how raw grayscale image data can be processed using image processing algorithms

such as contrast equalisation, median filtering and Sauvola thresholding. We then use the extracted pore networks to prepare the computational domains for acoustic wave simulations. We then discuss the simulation results for various types of pore configurations. The acoustic volumes and the extracted pore networks are then used to train the U-Net model to perform semantic image segmentation. We discuss some salient points in training the convolutional neural network with supervised data. In the end, we quantify the performance of the trained U-Net on some test samples.

Chapter 4: LSTMs based Accelerated Simulation for Hydro-Mechanical Systems. This chapter shows how an LSTM-based recurrent neural network can be configured as a multi-step prediction algorithm. We show how a million degrees of freedom state space can be compressed using a matrix factorisation algorithm such as the singular value decomposition. We then emphasise different normalisation strategies for the weights and boundary conditions before training the network. Next, we discuss the single-step training algorithm for the recurrent neural network, which ultimately is used to realise multi-step predictions. We show the training and validation accuracy and evaluate the network's performance with a reduced number of layers. We then show the reconstruction results in the original state space and compare them with the original simulations. At last, we also quantify the range of the pseudo-full-order simulations that can be generated using the ROM.

Chapter 5: Conclusions and Future Work. In this chapter, we summarise our contributions and discuss the scope of future research.

Chapter 2

General Algorithms in Deep Learning and Statistical Estimation

2.1 Deep Learning Algorithms

Deep learning algorithms refer to the artificial neural networks formed by combining multiple hidden layers between the input and the output layer. They provide a robust framework for supervised learning where the task is to map an input vector to an output target vector. The layers can form various units, often called neuronal units, to realize specific tasks. We discuss them in brief in this chapter.

2.1.1 Convolutional Neural Networks

The convolutional neural networks (CNNs) are inspired by the human visual ventral vision system and are similar to its architecture and functional tasks. Although the forms have been derived from many different fields, the fundamental framework is still from the branch of Neuroscience. The scientific work of [46] laid the foundation for modern day's CNNs after their years of analysis analyzing the mammalian visual system. While the pioneers received their Nobel prize in 1981, the main experiment that led to the discovery was on cat eye cells' response to lights of different wavelengths. The experiment involved fixing electrodes to the anaesthetized cat's brain and measuring its responses to various visual stimuli. It was discovered that the individual neurons of the visual cortex system were only activated for specific patterns in the input image. The deep architecture that we use today resembles the visual cortex. The CNNs were first developed for image processing applications and have been quite successful. The current text does not

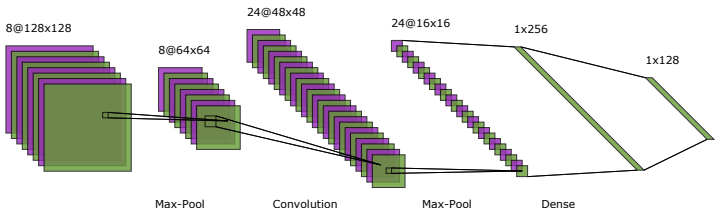


Figure 2.1: A typical CNN

intend to be an exhaustive source to understand these networks, so, for a more detailed study, the readers are referred to [47; 48; 49; 50; 51]. Apart from their success in their native image processing applications, they have also been applied successfully in the tasks of time series classification [52; 53; 54; 55; 56] applications, natural language processing and the comprehending video streaming data. In the forthcoming paragraphs, we explain the working of a CNN. Throughout the text, the data is assumed to be 2-dimensional.

A typical CNN is shown in Fig. 2.1. The CNN architecture can be composed of three types of layers – fully connected, convolutional layer, and pooling layer. Each type has its own set of rules for propagating the errors forward or backwards. While there are no precise rules on how to structure a CNN, broadly speaking, the conventional architecture can be understood to be comprised of two parts – first being the feature extractor that involves combinations of convolutional and pooling layers and the second being the classifier/regressor that consists of the fully connected layers. There are some exceptions to this broad categorization, as can be seen in the recent developments such as the U-Net architecture [57].

2.1.1.1 Convolutional layer

This layer performs the convolution operation to the input image. The important parameters here are the number of filters called the kernel. The output from this operation is a set of other images called feature maps. The convolutional layers are often cascaded with each other to realize a deep network for complex visual analysis tasks such as human pose identification. The input to a convolutional layer can either be an input image or a feature map in a deep net. For a 2D image, the shape of the kernel is typically more than three and less than the length/width of the image. The feature map is computed by the convolution operation of the kernel over parts of the image using a moving-window type approach. The size of the step taken to move this window to any dimension is called the stride and is typically taken to be 1 or 2 or, in some cases, higher if the image is larger in a

given dimension. These operations shall be taken up in detail in the forthcoming sections. The layer also has another parameter called padding. Padding is nothing but the appending of zero values on all four sides of the input image. This is done explicitly to match the size of a given dimension.
The size of the feature map is dependent on the size of the kernel and the padding length. This number is given by Eq. 2.1

$$p = ((h - 1)/2) \tag{2.1}$$

Here, h is the width of the square kernel, and p is the padding size. The feature is reduced by a factor of $2p$. Although the feature map reduces the dimension of the input image, it captures important patterns in the input. Also, the reduction is compensated by extracting many filters from the image. The feature map is computed using the kernel applied to all the image inputs. This essentially introduces redundancy, and the maps may carry shared information.

The value of the stride can also effect the size of the feature map. The formula for the output feature map is given by Eq. 2.2

$$w_{out} = \frac{w_{in} - f + 2p}{s} + 1 \tag{2.2}$$

Here, w_{out} is the dimension of the output image, w_{in} is the dimension of the input image, f is the size of the filter or kernel, p is the size of the padding and s is the stride value. A stride value of 1 does not affect the size of the output feature map. The convolution operation creates a composition of feature maps, each having a size dictated by Eq. 2.2. The number of such features in the output tensor/image is specified as the number of kernels. A large number of kernels ensures, though not guaranteed, capturing relevant features from a diverse set of a dataset. A deep network consists of more than one convolutional layer. The feature maps in each of the individual layers tend to capture features that progressively are complex in nature, appearance and shape. For example, the first layer might only capture the edges and line like features in the input image. The second layer might capture the 2D structures composed of these 1D lines, and so on and so forth. Higher in the hierarchy, one can expect the layers to learn features that are very abstract in nature. The reason is the nonlinear activations that are typically combined with these networks at the end of a layer. We next list a number of different types of layers that are commonly used with the CNNs.

2.1.1.2 Activation layer

The universal approximation quality of the deep net is imparted by the functions called activations which act on the input weights and apply a nonlinear transformation before outputting the weights to the next layer. Some commonly used activations are ReLU, tanh, sigmoid. A linear activation also exists that simply outputs the same weights as the inputs, i.e. it does not modify the information and acts like an identity function.

2.1.1.3 Pooling layer

The pooling layer is typically used to downsample the input data. Two of the most famous types of pooling layers exist. They are max-pooling and average pooling. The max-pooling is nothing but the extraction of the maximum value in the kernel patch when the convolution operation is being done. Similarly, the average pooling operation calculates the mean of the pixel values of the kernel patch. The schematics are shown in Fig. 2.2 and 2.3. An alternative way of seeing these layers is that they can be considered to guarantee invariancy to small changes in the inputs. This operation further reduces the number of trainable parameters and also gains better generalizable capabilities. Another way of downsampling can be realized using strides which is defined as the number of pixels to be skipped before applying the convolutional operation. The average pooling approach gets the mean of the pixels in the kernel patch. This type of pooling is typically used between the last convolutional layer and the final output layer (e.g. softmax for classification). The global average pooling can help process images with different sizes because their extracted feature maps can be transformed into the same output size and shape.

2.1.1.4 Transposed convolutional layer

Back converting from latent space to image space is an important task in machine learning. This is generally brought about by using multiple upsampling layers. One of the most used methods for doing so is the use of the transposed convolutional layer, which can be considered as the backward pass of a forward convolution. A cleaner way to understand this upsampling strategy is to view the transposed convolution as a forward convolution on the partially stridden input image with added zeros.

2.1.1.5 Upsampling layer

The upsampling operation can also be realized with a combination of a convolutional layer and an activation layer. It essentially is a non-parametric interpolation method that directly rescales the input volume to a higher dimension. These interpolation methods can be either the nearest neighbours, bilinear or cubic interpolation methods. This rescaled input volume is used as an input to the convolutional layer such that the size of the output remains the same. This is brought about by using a stride value of 1, and padding equals zero. Nonlinear activations generally follow these layers. This type of upsampling is generally costlier as it is performed on a higher dimensional volume. This method avoids generating checkerboard type artefacts inherent in the other upsampling methods.

2.1.1.6 Normalisation layer

Training the deep networks via Stochastic gradient descent (SGD) is often plagued with the problem of covariate shift. This occurs when the distribution of the input layer constantly changes with respect to the output layer. Several normalization methods have been devised to curate this. The normalization layer is used to centre the input data. It does this operation by subtracting the mean and dividing by the standard deviation, which essentially reduces the covariate shift. Commonly used normalization schemes are batch normalization and layer normalization. The feature maps are combined into a single input volume where the axes are called the minibatch axis (N), the channel axis (C) and the spatial axis (H, W). The three normalizations are calculated as follows:

- Batch normalisation is done along the (N,H,W) axis
- Layer normalisation is done along the (C,H,W) axis
- Instance normalisation is carried along the (H,W) axis

The activation layer is mostly preceded by the normalization layer. This results in better stabilizing the activation weights' distribution. One established advantage of batch normalization is that it accelerates the deep network's training procedure. Weight initialization becomes less important with batch normalization as it greatly improves the backpropagation with higher learning rates. Also, as it adds some noise by statistical centring of the data, a natural regularisation is naturally applied, making the network use less of the dropout fraction. The use case for layer normalization is mostly seen when training the recurrent networks (contrary

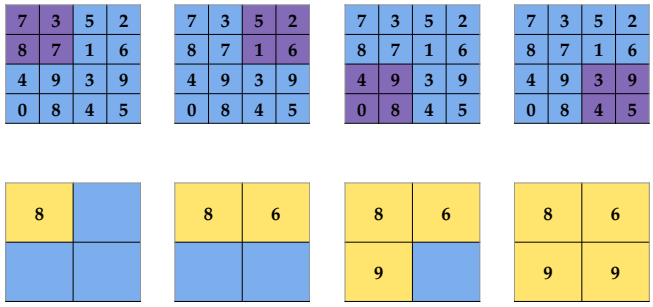


Figure 2.2: Example of max-pooling

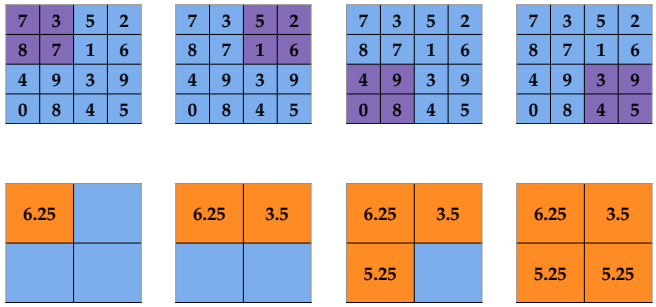


Figure 2.3: Example of average-pooling

to CNNs), in which layer normalization is generally not possible due to parameter reuse and variable length of inputs. Instance normalization must be preferred when using small batch-size.

2.1.1.7 Training CNNs

Optimizing a CNN is more complex than that of a fully connected neural network (FCNN). The reason for this is the various combinations of different types of layers that are cascaded from the input to the output layer. The following equations and paragraphs summarise the training procedure.

The image is first input to the input layer, and this marks the start of the forward propagation. This image undergoes a lot of transformations before appearing through the output layer. This output is compared against the observed image to compute an error which is then used to backpropagate to tune the weights of the network. The tuning of the weights is mostly done by the gradient descent algorithm, though modern variants are now available, though the principle remains the same.

The forward propagation of the image/signal in a CNN can be given by the fol-

lowing Eq. 2.3:

$$x_{ij}^{l} = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} w_{ab} y_{(i+a)(j+b)}^{l-1} \tag{2.3}$$

where, $i, j \in (0, N-m+1)$, l denotes the current layer number, w_{ab} are filter weights and $y_{(i+a)(j+b)}^{l-1}$ are the previous layer's output. The final layer in a CNN network computes the output using the following Eq. 2.4:

$$y_{ij}^{l} = g(x_{ij}^{l}), \tag{2.4}$$

where g is a nonlinear activation.
The backward-propagation in a CNN can be given using the following equations in this section. The CNN computes the derivatives using the automatic differentiation technique [58]. This enables computing the derivatives of any arbitrary function. Now, given the forward propagation has already finished, the derivative of the error w.r.t. the current convolutional layer can be given as $\frac{\partial C}{\partial y_{ij}^{l}}$, then the effect of the filter/kernel weights can be given as

$$\frac{\partial C}{\partial w_{ab}} = \sum_{i=0}^{N-m} \sum_{j=0}^{N-m} \frac{\partial C}{\partial x_{ij}^{l}} \frac{\partial x_{ij}^{l}}{\partial w_{ab}} \tag{2.5}$$

From Eq. 2.3 we can substitute the value as $x_{ij}^{l} = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} w_{ab} y_{(i+a)(j+b)}^{l-1}$. Therefore,

$$\frac{\partial C}{\partial w_{ab}} = \sum_{i=0}^{N-m} \sum_{j=0}^{N-m} y_{(i+a)(j+b)}^{l-1} \tag{2.6}$$

Using the chain rule, we can compute:

$$\frac{\partial C}{\partial x_{ij}^{(l)}} = \frac{\partial C}{\partial y_{ij}^{l}} \frac{\partial y_{ij}^{l}}{\partial x_{ij}^{l}} = \frac{\partial C}{y_{ij}^{l}} \frac{\partial}{\partial x_{ij}^{l}} (g'(x_{ij}^{l})) = \frac{\partial C}{\partial y_{ij}^{l}} g'(x_{ij}^{l} \tag{2.7}$$

Because the value $\frac{\partial C}{\partial y_{ij}^{l}}$ is already computed, the delta term can be computed as follows:

$$\frac{\partial C}{\partial y_{ij}^{l-1}} = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} \frac{\partial C}{\partial x_{(i-a)(j-b)}^{l}} \frac{\partial x_{(i-a)(j-b)}^{l}}{\partial y_{ij}^{l-1}} \tag{2.8}$$

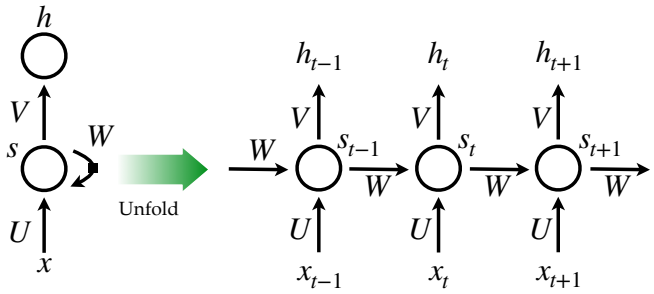


Figure 2.4: A schematic Recurrent Neural Network; Towards the left we have a standard RNN where the state vector is denoted by s; Towards the right is the unfolded version depicting how the state is built over time

Eq. 2.8 computes the propagation of error into the previous layer. Then using Eq. 2.3, we can see compute that $\frac{\partial x^l_{(i-a)(j-b)}}{\partial y^{l-1}_{ij}} = w_{ab}$

$$\frac{\partial C}{\partial y^{l-1}_{ij}} = \sum_{m-1}^{m-1} \frac{\partial C}{\partial x^l_{(i-a)(j-b)}} w_{ab} \tag{2.9}$$

Eq. 2.9 looks very similar to the familiar convolution operation, but it must be thought of as a convolution of error with reversed kernel/filter. An important point worth noticing with regard to the pooling layer and the forward propagation is that the layer simply compresses the information by downsampling an input matrix shown in Fig. 2.2. Conversely, in the backpropagation case, the error backpropagates to the element with the strongest information content; so if the operation is max-pooling, the error is backpropagated to the largest element of the input matrix.

2.1.2 Recurrent Neural Networks

While neural networks, in general, are versatile in mapping nonlinear relations between targets and covariates, they do possess a few shortcomings with sequential data. One of the fundamental assumptions underlying the plain multilayer perceptron networks is the assumption of independence in data samples. One exception to this assumption is the sequential data. Some of the examples in this category are human speech, stock price and weather temperature data. All these data exhibit dependence between sequential data points. Multilayer perceptron networks fail to exploit the information from this sequential dependence. A naive approach to dealing with such data is to collect multiple data points in time as one data point. This is analogous to the moving window approach. Some of the

works using this approach with sequential data can be seen [59; 60; 61]. But a crucial point in tuning such networks is to find the optimal window size - this is important; a smaller window size won't capture longer dependencies, while a large window would contribute to extra information that may sometime appear as noise. One more limitation to the window based method may arise when there are very long dependencies ranging over hundreds of timesteps. Additionally, the plain NNs are not too welcoming for variable length sequences. This requirement is especially important for tasks like language and machine translation.

In the recent past, Hidden Markov Models (HMMs) were being used for sequence modelling tasks. HMMs, first developed by [62], work by mapping a set of observed states to a set of hidden states. Probability distributions are used to define the relationships between hidden states and observed states. These models follow the Markov property, which states that the value at the current state is only dependent on the immediately previous state. This is one of the severe limitations of HMMs that prevents them from capturing long-range dependencies. Another is that of the space complexity of an HMM that grows quadratically with the number of states. This is significantly less than the fully connected versions that grow exponentially. Some of the notable applications of HMM can be found in [63; 64; 65; 66; 67; 68].

The recurrent neural network is the modern way of handling sequential data. The RNN cell tackles the long term dependency by maintaining a memory of all the previous seen elements in the training sequence. This is done using the state vector s in the hidden units. A simplistic form of the RNN can be seen in Fig. 2.4. The curved dotted line and its pair in the solid line depict the feedback loop that connects all the hidden neurons across the sequence length in time. At each timestep $t = t$, the RNN cell receives the current element x_t and the hidden state s_{t-1} to update the current state s_t. The final output h_t is then updated, ready to be fed to the next stage. This structure is crucial to maintain the dependency of h_t on all previous states.

As seen is In Fig. 2.4 a weight matrix U is maintained between the input and the hidden layers. Between one hidden state to another another weight matrix W is maintained. And between hidden state and the output another weight matrix V is maintained. All the operations can be summarized using the following equations.

$$\begin{aligned} s_t &= \sigma(U_{x_t} + W_{s_{t-1}} + b_s) \\ h_t &= softmax(V_{s_t} + b_h) \\ y_k &= \frac{e^{a_k}}{\sum_{k'=1}^{K} e^{a_{k'}}}, k \in (1, \ldots, K) \end{aligned} \tag{2.10}$$

Eq. 2.10 refers to the operations done on the state s. Symbols b_s, b_h are the bias vectors. The softmax function ensures the output probabilities sum up to 1, similar to a multiclass classification problem. Observing closely, one can notice that a standard RNN 2.4 is itself a deep neural network. The unfolded version makes it clear that a number of layers make up the networks, and the same set of weights are applied and updated at each timestep. This feature enables the RNN to process variable length input sequences. Due to the feedback connection and the way s_t is updated, the information flow can take place across many timesteps, thus enabling it to maintain memory in itself.

2.1.2.1 Training Recurrent Neural Networks

The backpropagation through time (BPTT) is used to train an RNN. Considering the unfolded version shown in Fig. 2.4 we can see that it acts like a deep network that implies it can be trained using the BPTT algorithm. Theoretically, the RNNs can learn very long-range dependencies in time, given the multilayer analogy of the unfolded network. The training should result in the tuning of weights so as to retain the relevant information in the memory. However, in practice, the training procedure is not trivial due to some well-known issues. In fact, authors in [69] have identified that training a vanilla RNN (Fig. 2.4) performs very poorly even for a sequence with short lags, as short as 10. The BPTT algorithm propagates the output errors through long timesteps. The recurrent edge maintains the same set of weights across time that eventually results in multiplying the gradients at timestep t through timestep $t - n$ backwards in time. This results in a problem called the "vanishing gradient problem" when the product becomes too small; conversely, when the product is too large, the problem is known as the "exploding gradient problem". These are some of the significant problems with regard to training a vanilla RNN as shown in Fig. 2.4. These are highly determined by the weight matrices and the types of activations used. There have been several suggestions to tackle these problems. Authors in [70; 71; 72] suggested the use of clipping the gradients to mitigate the training problem; [70; 73] also suggested the use of regularisation term $L1$ and $L2$ to prevent the overfitting issue.

2.1.2.2 Long short-term memory units

In order to get around the training difficulty, an improved architecture was suggested by [74]. This architecture is called the Long-short Term Memory (LSTM)

unit. As stated in the paper, the LSTMs have proven to be useful. LSTM solve the problem of vanishing gradients by substituting the simple neurons with advanced architectures called gates in the LSTM unit. A basic LSTM unit is shown in Fig. 2.5. Some of the attributes of the LSTM unit can be summarised as below:

- CEC: Constant error carousel, this refers to a fundamental unit that has a recurrent connection of unit weight. This can be seen in the Fig. 2.5 as the feedback loop. This acts as the internal state responsible for maintaining the memory.
- Input gate: This aids in the input activation. This is mostly a hyperbolic tangent.
- Output gate: This acts as the output activation. This is mostly sigmoid.
- Forget gate: This acts as the forget activation. This is mostly sigmoid.

The gates control the access to the CEC unit. A value equal to one in the input gate allows new information to be added to the CEC. Similarly, a value equal to one in the output gate allows the information to flow out of the unit. A value near zero blocks the gates allowing the information to remain intact inside the cell. This selective opening and closing of the gates allows the error propagation to flow across long sequences. This gated version of the CEC is effective in dealing with the vanishing gradient problem. The problem of exploding gradients is taken care of by clipping the gradients. This set of modifications leads to better learning of the long term sequences than conventional vanilla RNNs. Notably, the forget gates also serve an additional function. It has been found that training these units with continuous and very long streams which do not have specified start and end markers makes the unit unstable. The network is no more is capable of identifying the temporal patterns and/or cycles. One remedy to this behaviour was then identified. The solution was to reset the state of the unit after each training set. This resetting must be done before a new sequence is fed to the unit. This is exactly what the forget gates are included for, by [75]. These special gates force the state to reset after each training set is passed through. Some of the properties of the LSTMs with the inclusion of forget gates are given below. Referring to Fig. 2.5 we state:

- The input sequence x_t at the current timestep and the output from timestep $(t-1)$ given by h_{t-1} are passed through the *tanh* activation function after summing operation

- Then, the input gate receives x_t, and recurrent input h_{t-1} computes the weighted sum and applies a sigmoid activation. A product of the result i_t and z_t is taken to be input into the memory cell
- The forget gate enables the unit to let go of the information that is old enough and is no longer required. This operation is important when new data is available, and the network must be trained with this new data as well. A reset in the state must be performed before doing so. So, the forget gate receives the inputs x_t and hidden state h_{t-1} and applies a sigmoid function to the weighted inputs - then a product of this result and the cell's previous state s_{t-1} is taken to "forget" the irrelevant information.
- Then, the CEC unit with the feedback loop receives the input after forgetting the not so important information from the previous timestep and, at the same time, accepting relevant information from the current timestep
- Then, the output gate acts by computing the weighted sum of the inputs x_t and h_{t-1} to apply a sigmoid activation. This controls what and how much information flows out of the LSTM cell.
- Finally, the output is taken after passing the current state, s_t through a $tanh$ activation and taking a product with the result of the output gate, o_t.

The above mentioned operations can be summarized as below:

$$\begin{aligned} z_t &= tanh(W^z x_t + R^z h_{t-1} + b^z \\ i_t &= \sigma(W^i x_t + R^i h_{t-1} + b^i \\ f_t &= \sigma(W^f x_t + R^f h_{t-1} + b^f \\ o_t &= \sigma(W^o x_t + R^o h_{t-1} + b^o \\ s_t &= z_t \odot i_t + s_{t-1} \odot f_t \\ h_t &= tanh(s_t) \odot o_t \end{aligned} \tag{2.11}$$

Here, Eq. 2.11 denote the following operations of the input, input gate, forget gate, output gate, the cell state and the final output, respectively.

They process a single datapoint in time. They maintain a hidden state vector which acts as a memory unit for the past states. They are well capable of retaining information that is important across several timesteps; hence are well equipped to capture long term dependencies. A single memory cell makes use of both the past and the current state to make a forecast for the future. And the most interesting thing is that the model learns and tunes to do all these tasks itself while undergoing

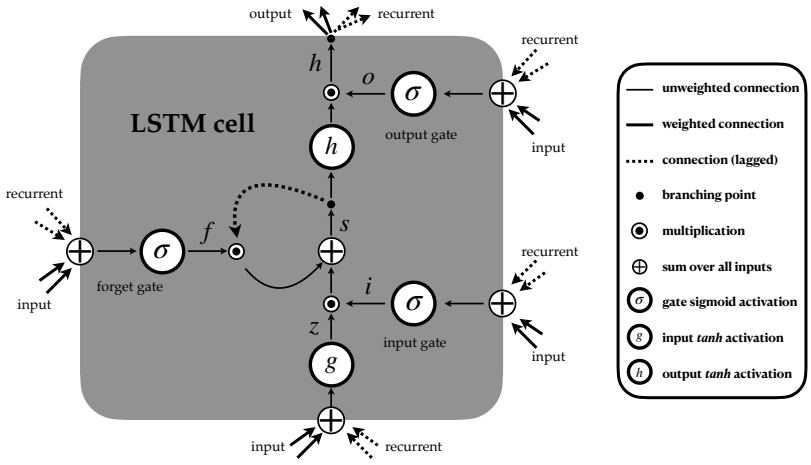


Figure 2.5: A standard Long short-term memory cell

training. But the RNNs need a fixed window size, although they can handle variable-size inputs.

Recurrent networks find immense applications in areas of language modelling, machine translation and image captioning. LSTM cells are an advanced form of RNNs where they allow the gradient to flow through all the temporal states making use of an extra latent state called the memory cell state. There exist a few variations in the LSTM, like the LSTM with a peephole connection that enables learning precise time dependencies and the gated recurrent units (GRUs). These are less computationally expensive compared to the original LSTMs.
Other applications related to geosciences can be found in hydrocarbon detection from well-logs [76], simulating reservoir behavior [77; 78], and prediction of reservoir physical parameters [79].

2.1.3 Encoder-Decoder Frameworks

The different forms of neural networks described in sections 2.1.1 and 2.1.2 suffice for most purposes, but there may be tasks that need custom designed networks. For example, one may want to learn a compressed representation/latent vector of the input image similar to dimensionality reduction techniques and then use this compressed representation to reconstruct the input. This compression and reconstruction help deduce relevant and important features. The compressed representation is usually smaller in dimension than the input image size. The main idea behind this technique is that the latent representation/vector would capture

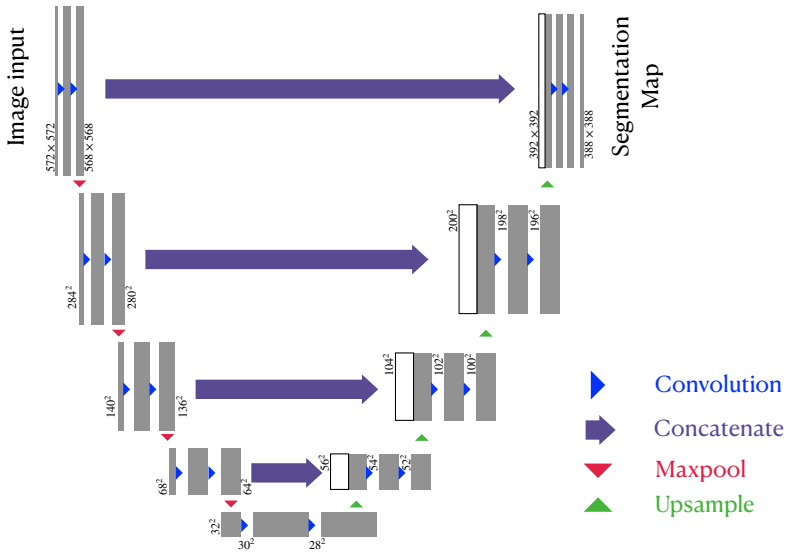


Figure 2.6: The U-Net architecture

just enough information so as to recreate the data itself. This type of network consists of two parts - the encoder and the decoder. The type of network that recreates the input data is called an *Autoencoder* and will be explained in the next section. One of the interesting applications of these models is semantic segmentation [80; 81; 82; 83; 84]. With regard to the flexibility that these modularised networks offer is that one can train multiple decoders for one single encoder, each designed for a specific task, and vice-versa. For example, in a machine translation application where one language needs to be translated to many other languages, there can be a decoder for each one of them. Another application can be of calculating a fixed length representation of variable length inputs. RNNs are commonly used for these types of tasks. Next, we define some common encoder-decoder architectures in the next subsections.

2.1.3.1 The U-Net Architecture

Developed by [57] the U-Net architecture takes its name from the shape it bears when drawn in on a 2D plane. The architecture can be seen in Fig. 2.6 was designed for the purpose of semantic image segmentation and localization for Biomedical purposes. While conventional CNNs are excellent in image classification tasks, e.g. [80]. The U-Net is composed of an encoder-decoder framework with skip connec-

tions and is also capable of localization. By localization, we mean the segmented parts retain their physical position in space after the task of segmentation completes. In addition to that, and unlike conventional deep network requirements, the U-Net is able to work with less amount of labelled data. The added capability comes from the use of upsampling layers with a large number of feature maps (typically in thousands for a 3D image of size $800 \times 800 \times 800$) that help carry context information to higher layers in the hierarchy.
The U-Net architecture can be seen in Fig. 2.6 where we observe it has two parts that look like the two halves of the English letter "U". The left half is the encoder, and the right half is the decoder. The encoder offers a contracting path to the input image while the decoder expands on the contracted image to form the output having the same dimension as that of the input. The contracting encoder consists of the conventional convolutional operations of size 3×3 and max pooling followed by a downsampling operation of stride of 2. The downsampling step creates double the number of feature maps. On the other hand, in the expanding decoder path, the feature maps are upsampled with a convolutional operation of size 2×2. This operation reduces the number of feature maps by 2, which are then concatenated with a copy of the corresponding downsampling operation of the encoder part. The result is again followed by a convolutional operation to map the feature vectors to a multiclass vector. The main idea behind the contracting operation is that the network must capture the context of the image, similar to a classification network. The localization effect is brought about by the "direct" skip connections that connect the downsampling layers to the upsampling layers in the architecture.

2.1.3.2 The *seq*2*seq* Architecture

As the name suggests, the *seq*2*seq* model derives its name from the phrase "sequence to sequence". This architecture was developed for processing pairs of sequences, frequently encountered in machine translation tasks. The architecture was first developed by [85; 86; 87; 88]. Interesting applications have been found since then in the tasks such as image captioning [89], dialogue systems [90], and speech recognition [91; 92]. Though not directly related to this thesis, the model was previously considered for the development of the Reduced order model as described in chapter 4.
The vanilla *seq*2*seq* describes the conditional probability of the output sequence y given the input sequence x as given by Eq. 2.12:

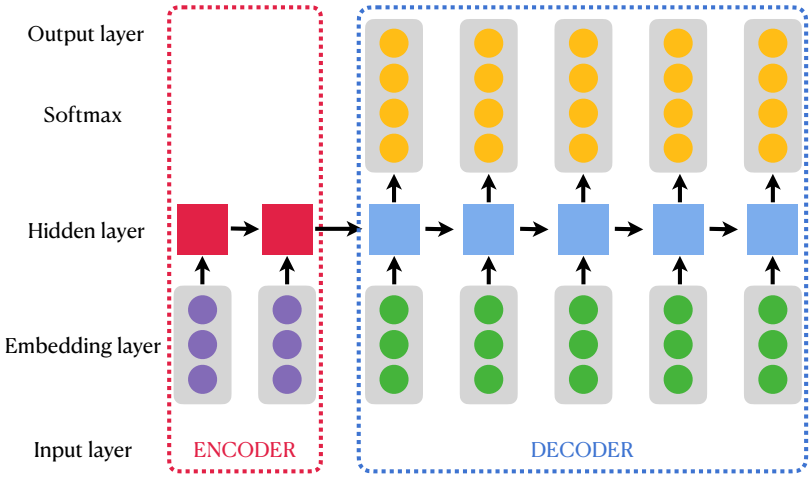


Figure 2.7: The *seq2seq* architecture

$$p(\mathbf{y}|\mathbf{x}) = \prod_{j=1}^{J} p(y_j|\mathbf{y}_1^{j-1}, \mathbf{x}) \tag{2.12}$$

This is also an example of the encoder-decoder framework. The job of the encoder is to receive an input sequence (usually one token at a time) to generate a fixed length latent representation/vector. The decoder then maps back the generated latent vector to a token of the output sequence. While RNNs are typically used for machine translation tasks, for image captioning, the encoder stage may be framed by a CNN. The framework can be seen in Fig. 2.7. In this figure, both the encoder and decoders are realized using RNNs. Usually, two separate embeddings are created for each of the stages. The encoder stage takes as input the words from the input sequence and encodes them into a latent vector. The end of the input is usually marked with a special character. This special character helps identify the starting point for the decoder. The decoder RNN then starts from the last latent state of the encoder to begin the translation.

Training is usually carried out with true source and target sequences when the output error is backpropagated through time across the entire model. The loss function for training is given by Eq. 2.13:

$$L = -\sum_{(\mathbf{x},\mathbf{y})\in S} \log p(\mathbf{y}|\mathbf{x}) \tag{2.13}$$

The equation depicts the negative log-likelihood of the pair of sequences in the training data S. During testing, the model generates a sequence that is probabilistic and more likely to be the output.

2.2 Dimensionality Reduction and Matrix Factorization Algorithms

Data is ubiquitous. These are used to represent the observations of an experiment or a field survey. Often, there are multiple sensors/devices that are employed to capture the observations. The data from each of these devices can be arranged in the form of a vector representing different features or aspects of the phenomenon. When these features are independent of each other, they can often be called degrees of freedom [93]. But this is seldom the case, and the result is that we end up using multiple devices to capture the phenomena in the hope that it increases the signal-to-noise ratio to uncover the inherent properties within the data. Using multiple sensors to capture single phenomena leads to redundancy in information which ultimately manifests as modelling complexities to the use. Apart from being redundant, the data is also sometimes irrelevant that may persist as modelling overload by taking up unnecessary computation. And on top of these, there lies the problem of "curse of dimensionality". This problem arises when the high dimensional space is more sparse than being informative. So, when avoiding dimensionality reduction techniques, the sparse data soon blows up the data storing space and still may be "poorly informative" about the ongoing phenomena.

The branch of dimensionality reduction has evolved over the decades to overcome the above-mentioned problems in multi-dimensional data. The idea is to devise a compact parameterization of the phenomena that generate the data. This way, dimensionality reduction algorithms can be thought of as a set of transformations that can accurately characterize the important events in the dataspace by reducing very high dimensional data to a fairly low-level representation. Fig. 2.8 and Fig. 2.9 illustrate the idea nicely. The few advantages of this reduction are –

- Reduction in computational costs
- Better data representation
- Preserving relevant information

The data in the real world is mostly three dimensional. As a part of data processing, multiple features are extracted from them to make it work through the

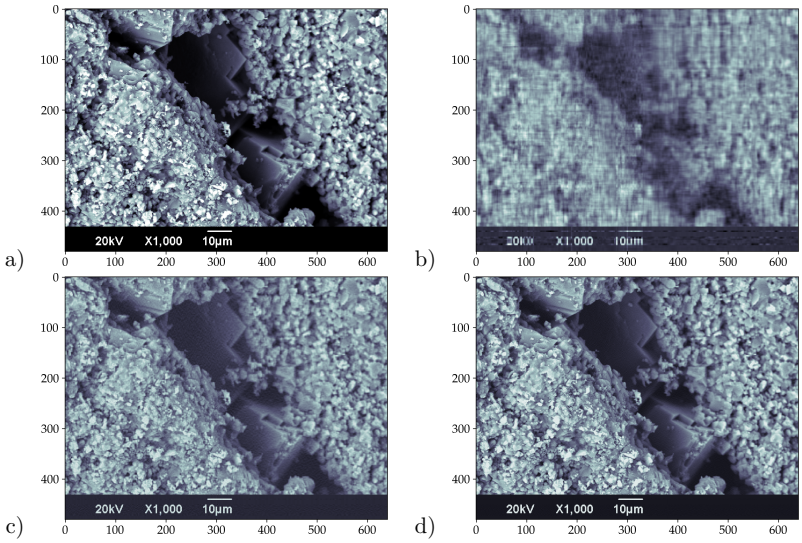


Figure 2.8: From a) Original image; From b) to d): Reconstruction of a scanning electron microscopy image with increasing number of principal components – $20, 120, 220$

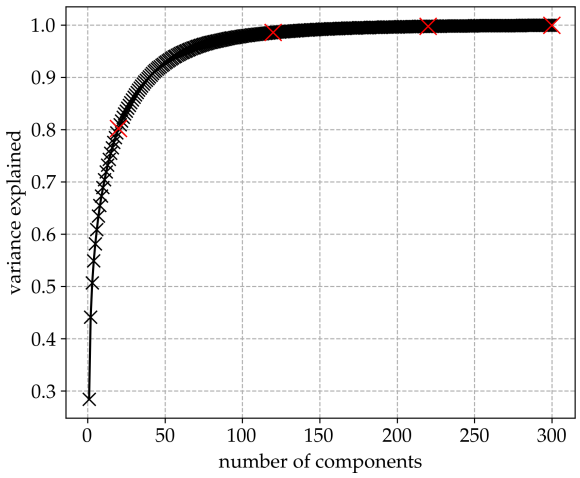


Figure 2.9: Reconstruction accuracy; Higher is better

machine learning algorithm. As will be shown in chapter 4, a large state space of a hydro-mechanical simulation resides on a smaller data dimension compared to its original space, where the degrees of freedom are in millions.

The problem of dimensionality reduction lies at the base of machine-learning algorithms. These algorithms are frequently employed to solve tasks in pattern recognition, image segmentation and the more general data compression. There exists a class of methods when the data appears to be linear in nature. One such algorithm is the Principle Component Analysis or the PCA. It accounts for the variance in the data and uses it as the basis to "load" its several components. The space where these components are evaluated is called the principle component space, which is often much lower than the original data space. There are other algorithms as well that exploit different properties of the data to reduce the dimension. For example, there is a variant of the basic Singular Value Decomposition called the Kernel-Singular Value Decomposition algorithm or the K-SVD [25]. This algorithm makes use of the "kernel" to account for the nonlinearity in the data. Among the linear techniques, there are quite a few like the Dictionary learning algorithm [94], Matching Pursuit [95] and the Factor Analysis algorithm [96]. Often, the data acquired for Geoscientific analysis is high-dimensional, although the causative sources might be much less in numbers. These algorithms try to reduce the space of the observed data by working out a procedure like matrix transformations to get down to a dimension equal to the number of sources that generated the data. Technically, we say that the observed data, though high dimensional, lies in a low-dimensional manifold. All dimensionality reduction algorithms are designed to obtain the best possible reduction. Probably, the reason for ending up at a high dimension space is that the user cannot ever know the exact location of the generative source in the data space. Eventually, when the measurements are made at a location which is far from the causative source, there arises a need to compensate for the distance, which results in making multiple dimensions.

There exist other classes of algorithms which are specifically targeted to address the nonlinearity in data, such as the Local Linear Embedding, Laplacian Eigenmaps, and the Isomap algorithm [97; 98]. These algorithms offer greater flexibility in fitting the nonlinear data. But there are a few issues when applying these algorithms. First among them is that these algorithms map the points in the original higher dimension and the lower dimension in a one-to-one manner, i.e. say, the embedding that is created needs an interpolation technique to extend the generalizability to initially unavailable points. This process can be termed an out-of-sample extension. The second issue is that these algorithms emphasize the properties of

the training data only, i.e. to say they mostly maximize the inter-class distance of the training data but do not perform so well for the data in the test sets. There exist several methods to increase the predictive performance on the test data [99]. The nonlinearity in data presents special challenges because the performance depends not only on the computed embedded space but also on the interpolator used during training. One possible solution to extend the learned embedding is the use of smooth functions such as radial basis functions [100]. Also, it becomes obvious that learning the embedding and the interpolation parameters are done in a joint manner rather than sequential. In this thesis, we emphasize linear dimensionality reduction techniques only.

2.2.1 Proper Orthogonal Decomposition

The Proper Orthogonal Decomposition (POD) is an ubiquitous matrix factorization method and has applications ranging from numerical simulation [101; 102], model reduction [103; 104; 105], model inversion [106; 107; 108] to dimensionality reduction [109; 110]. Let the high dimensional data matrix be denoted as Y. The POD can be realised by the Singular Value Decomposition gives three matrices as the products of its operation on Y, given as:

$$Y = USV^T \tag{2.14}$$

where, $U = [u_1, \ldots, u_m] \in \mathbb{R}^{m \times m}$ and $V = [v_1, \ldots v_n] \in \mathbb{R}^{n \times n}$ contain the left and right singular vectors. The singular values are stored in the S matrix as:

$$S = \begin{bmatrix} D & 0 \\ 0 & 0 \end{bmatrix} \tag{2.15}$$

where, $S \in \mathbb{R}^{m \times n}$. The diagonal D contains the singular values in decreasing order such that $s_1 \geq S_2 \ldots s_d > 0$. The vectors $\{u_i\}_{i=1}^{d}$ and $\{u_i\}_{i=1}^{d}$ denote the eigenvectors obtained from matrices YY^T and Y^TY, respectively. The data matrix in Eq. 2.14 can be approximated using a lower rank estimate, given as:

$$\hat{Y} = U^{(d)} D (S^d)^T \tag{2.16}$$

We ideally choose the value of d as a trade-off between compression and estimation quality.

2.2.2 Sparse Coding and Dictionary Learning

Redundancy in data can be exploited to code a signal/image for a compressed representation [111; 112]. For example, in image applications, the algorithm can look into the statistical distribution of the pixels to come up with a compressed representation. Given a signal $y \in R^n$, one can represent it using a linear combination of some basis vectors.

$$y \approx Dx \tag{2.17}$$

where the vectors are represented by the columns in the $D^{n \times k}$ matrix x^k contains the weight or coefficient of each of the vectors. Sparse coding relies on the fact that only a few of such columns are required to reconstruct the given signal while the coefficients of others are near zero. The optimization function for sparse coding a signal can be given as:

$$\min_x ||x||_0 : ||y - Dx||_2^2 \leq \epsilon \tag{2.18}$$

where the x^k is called the sparse representation of the signal y and ϵ is a very small value. The sparsity in x is enforced by the l_0 norm. Conventionally, image compression algorithms have been using some pre-defined basis such as the Fourier basis and the Wavelet basis [113; 114]. The dictionary learning algorithm works by finding a sparse dictionary or atoms of the given matrix in an overcomplete manner. The method has been proven to give satisfactory results in image denoising tasks.

$$(D^*, x^*) = \arg\min_{D,x} = \frac{1}{2}||y - Dx||_2^2 + \alpha||D||_1^1 \tag{2.19}$$

where, α is the regularization constant. In this optimization function the algorithm aims to estimate the two matrices D^*, x^* whose dot product gives the least error with the observed signal y. The optimization algorithm used for such tasks are orthogonal matching-pursuit and least angle regression with coordinate descent [115; 116].

2.2.3 Independent Component Analysis

The independent component analysis (ICA) algorithm [117] is used to identify the underlying generative sources of a given signal where the given signal is a linear combination of the sources. It also assumes that the sources be independent and non-Gaussian. Although a clear explanation of this algorithm would take several pages, we try to give out the main idea behind its working. In particular, we

demonstrate the working of a practical version of the ICA algorithm called the Fast ICA. let the observed data be represented as $y = (y_1, y_2, \ldots, y_m)^T$. This m-dimensional vector is zero-mean. One wants to estimate a suitable transformation of y such that result has some desired properties, here being the non-Gaussianity.

$$s = Wy \tag{2.20}$$

where, $s = (s_1, s_2, \ldots, s_n)^T$ is a n-dimensional transform and W is the weight matrix. The FastICA algorithm mandates demeaning and whitening of the data prior to its application. The optimizer tests for the convergence of the algorithm by checking for the non-Gaussianity of each s_i, where i denotes the i^{th} component of the mixed signal y. The objective that the algorithm maximizes is given in the following form:

$$J_G(w) = [E\{G(w^T y)\} - E\{G(\nu)\}]^2 \tag{2.21}$$

where, w is an m-dimensional weight vector and ν is a Gaussian variable of zero-mean and unit variance. Minimizing this function J_G, gives a vector $\hat{w}$ which can be tested for being a likely estimator using some "contrast functions". The author [118; 119] gives some choices of this contrast functions as given below.

$$G_1(u) = \frac{1}{a_1} \log \cosh(a_1 u), g_1(u) = \tanh(a_1 u) \tag{2.22}$$

$$G_2(u) = -\frac{1}{a_2} \exp -a_2 u^2/2, g_2(u) = u \exp -a_2 u^2/2 \tag{2.23}$$

$$G_3(u) = \frac{1}{4} u^4, g_3(u) = u^3 \tag{2.24}$$

where, $a_1 \in [1, 2]$ and $a_2 \approx 1$. Some. recommendations on using the $G's$ is as follows. G_1 should be good for general purpose applications, G_2 must be used for robustness or when the components are super-Gaussian and for sub-Gaussian independent components the choice of G_3 is good. Some of the applications of ICA are mostly found for dimensionality reduction, denoising and unmixing of hyperspectral data as in [120; 121; 122]. It has also been used in reservoir geomorphology [123; 124] and detection of coal bed methane [125].

2.2.4 Autoencoders

The neural networks are good at discriminative and generative modelling tasks. There is a class of neural networks that work in an unsupervised setting that can be used for dimensionality reduction - the Autoencoders. There are other varia-

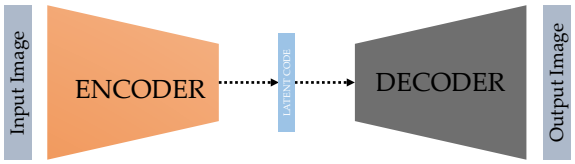


Figure 2.10: A schematic autoencoder architecture

tions as well, such as the Deep Boltzmann Machines and Deep-Belief Networks. For a detailed discussion on those, the readers are encouraged to go through the Deep Learning Book [126]. The Autoencoder is a neural network that learns to reconstruct itself. Fig. 2.10 shows the schematic of an autoencoder. The motivation behind such an architecture is that in the pursuit of reconstructing the input data, the network would learn a representation that is much small in size compared to the input. The reduction in the representation is enforced by the "bottleneck" present in the middle of the network Fig. 2.10. This process of compression can be understood as follows - let the input x be parameterized as $h = f(x)$ where h is the latent vector or the code that needs to be learned. The process can be realized by putting multiple layers from the input layer to the bottleneck called the encoder. Once the information is contained in h, the next set of layers from the bottleneck to the output try to reconstruct the input image as $\hat{x} = g(h)$, where $\hat{x}$ is the network's estimate of the input and $g(h)$ is the functional representation of the decoder. Hence, two parts are called the encoder and the decoder. The loss function used here is the squared error loss because the network needs to compare the input vector and the reconstruction in an element-wise fashion. The equations can be given in the following form:

$$h = f(x) \tag{2.25}$$

$$\hat{x} = g(h) \tag{2.26}$$

$$L_{autoenc}(x, \hat{x}) = ||\hat{x} - x||^2 \tag{2.27}$$

An important point w.r.t. the loss function is that additional constraints may be applied to it to reduce the representational capacity of the network, which in turn can help avoid learning an identity function. Other constraints may facilitate regularisation. We now list three simple forms of autoencoders that serve different purposes.

- **Sparse Autoencoder**: This type of Autoencoder simply adds an extra penalty term $\Omega(h)$ for enforcing sparsity. So, the loss function now reads as

follows:

$$L(x, \hat{x}) + \Omega(h) \tag{2.28}$$

where $\hat{x}$ is the decoder's output. These types of autoencoders are mostly feature extractors that can be used by other machine-learning algorithms as a classifier. The extra term $\Omega(h)$ can be thought of as a regulariser whose task is to make the network respond to the sparse features present in the input dataset.

- **Denoising autoencoder**: Similar to the sparse Autoencoder, the denoising autoencoder also minimizes a modified loss function given by the following equation:

$$L(x, \tilde{x}) \tag{2.29}$$

where $\tilde{x}$ is a corrupted copy of the input. This can be done by adding some form of noise to the input image. Hence the name - these autoencoders are supposed to remove the noise from the corrupted signal and recover the original information.

- **Contractive autoencoder**: This type of autoencoder, introduced in [127], expresses a regularizing term on the latent vector in the following form:

$$\Omega(h) = \lambda || \frac{\partial f(x)}{\partial x} ||^2 \tag{2.30}$$

The term $\Omega(h)$ is the Frobenius norm or the sum of squared elements of the Jacobian matrix of the partial derivatives computed in the encoder part.

The autoencoders are good at information retrieval and dimensionality reduction tasks. Reduction in the data space can greatly improve the performance of classification algorithms. Smaller data spaces imply fewer computation times and fewer memory requirements. These are some of the benefits of using them with fundamental machine-learning algorithms.

2.3 Miscellaneous

2.3.1 Optimising Deep Networks

Gradient descent is a popular choice for optimizing deep neural networks. There exists a handful of different variations to the basic gradient descent algorithm. We

list the main ones here from [128].
Let the cost function be $J(\theta)$ where $\theta \in R^d$ is the model parameter. The gradient descent aims to minimize this function by updating θ in the opposite direction of its gradient $\nabla_\theta J(\theta)$ w.r.t. the parameters. The optimizer has another parameter called η that sets the stepsize the optimizer takes at each update. This parameter determines "how big" of an update is to be done once the gradients are computed. Summarily, the optimizer tends to move in the direction dictated by the gradient of the cost function until it reaches a saddle point or local optimum.
Literature cites many variants of the basic gradient descent. We list a few of them in this section. In the following paragraph we denote $X(i), y(i)$ to denote the i^{th} predictor and target, respectively.

- **SGD** - The basic variant of the gradient descent algorithm is the stochastic gradient descent and it derives its name from the way it uses the amount of data. It updates the parameters for each new sample in the training set. The exact cost function is given as:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; X^{(i)}, y^{(i)}) \tag{2.31}$$

  The SGD can be used in an online mode, i.e. it can perform its update with new incoming data as the update works with a single example at a time. And because it takes one example at a time, it needs to perform frequent and high variance updates which results in the cost function value fluctuating rapidly. The rapid updates also have the added benefit of escaping the local minima in case of non-convex loss functions. When the function has multiple non-smooth regions, the algorithm may suffer from "overshooting", i.e. it may miss out on a potentially good local optimum. The way around here is to decrease the step size η. Shuffling the examples is often accompanied within the main loop to avoid biasing.

- **BGD** - The Batch gradient descent or the BGD is the most simplistic version of the gradient descent algorithm. It differs from the SGD in the amount of data it processes at a time. The BGD takes in the entire data into the memory to perform the parameter update. Therefore the cost function it minimizes is given as:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta) \tag{2.32}$$

  Using this variant can be problematic for large datasets as it puts a constraint on the memory requirements. This variant can also be very sluggish and

requires all data to be present at a time; this implies it cannot be used in an online mode.

- **MBGD** - The mini-batch gradient descent or the MBGD is a culmination of both the above mentioned variants. Instead of treating one example at a time it takes a batch of size n to perform the gradient and update operations. It takes the form given by the equation:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; X^{(i:i+n)}, y^{(i:i+n)}) \tag{2.33}$$

  There are two main advantages of the approach - it leads to better and more stable convergence by reducing the variance in parameter updates; it can be tweaked for better matrix optimizations that are common to other optimization algorithms.

- **SGD with Momentum** - Momentum based SGD is an improvement to the simplistic SGD by introducing a "momentum" term γ.

$$\nu_t = \gamma\nu_{t-1} - \eta \cdot \nabla_\theta J(\theta) \tag{2.34}$$

$$\theta = \theta - \nu_t \tag{2.35}$$

  The value of γ is usually set to 0.9 or somewhat near. The parameter updates are accelerated in the direction opposite to the gradient, while for others, the acceleration is lessened.

- **NAG** - Nesterov Accelerated Gradient (NAG) [129], when combined with gradient descent, enables the base algorithm to have a notion of accelerating the updates as the case may be. For example, the momentum term $\gamma\nu_{t-1}$ is used to update the parameters; the next step is usually to calculate the velocity term ν, which can be used as the argument to the cost function J as shown in the equation below. This gives a better approximation of the optimal point. $\theta - \gamma\nu_{t-1}$ to approximate the next optimal location.

$$\nu_t = \gamma\nu_{t-1} - \eta \cdot \nabla_\theta J(\theta - \gamma\nu_{t-1}) \tag{2.36}$$

$$\theta = \theta - \nu_t \tag{2.37}$$

  The formulation helps anticipate the update and prevents going too fast or too slow. This has been highly beneficial for training the RNNs.

- **ADAGRAD** - One scope of improvement with the NAG optimization is that it can be done on a parameter-wise basis. The ADAGRAD [130] or

the Adaptive Gradient algorithm has been designed for the same purpose. It adapts the updates for each parameter separately. The result is that we have larger updates for less frequent changes in the parameters and smaller updates for parameters with large changes. The following equations summarise the ADAGRAD steps:

$$g_{t,i} = \nabla_\theta J(\theta_{t,i}) \tag{2.38}$$

$$\theta_{t+1,i} = \theta_{t,i} - \nu \cdot g_{t,i} \tag{2.39}$$

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \cdot g_{t,i} \tag{2.40}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t \tag{2.41}$$

One of the advantages of using ADAGRAD is that it relieves the user from manually tweaking the learning rates. At the same time, one disadvantage of using this variant is the addition of the positive term in the denominator. This term keeps increasing during training which makes the fraction infinitesimally small after some time. After this stage, the algorithm becomes less effective.

- **ADADELTA** - The ADADELTA [131] variant overcomes the shortcomings of the ADAGRAD algorithm by reducing the effect of monotonically decreasing learning rate. This is done by restricting the accumulation of past squared gradients to some fixed value, w. The algorithm maintains a running average of $E[g^2]_t$ at timestep t that just depends on the previous average and the current gradient.

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2 \tag{2.42}$$

Enumerating all the steps once again gives us the following equations.

$$\Delta\theta_t = \eta \cdot g_{t,i} \tag{2.43}$$

$$\Delta\theta_{t+1} = \theta_t + \Delta\theta_t \tag{2.44}$$

$$\Delta\theta_t = -\frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t \tag{2.45}$$

$$\Delta\theta_t = -\frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t \tag{2.46}$$

$$\Delta\theta_t = -\frac{\eta}{RMS[g]_t} g_t \tag{2.47}$$

In order to match the units from the last update, an exponentially decaying average of squared parameter updates is defined:

$$E[\Delta\theta^2]_t = \gamma E[\Delta\theta^2]_{t-1} + (1-\gamma)\Delta\theta_t^2 \tag{2.48}$$

$$RMS[\Delta\theta]_t = \sqrt{E[\Delta\theta^2]_t + \epsilon} \tag{2.49}$$

The final update rule is given as:

$$\Delta\theta_t = -\frac{RMS[\Delta\theta]_{t-1}}{RMS[g]_t} g_t \tag{2.50}$$

$$\theta_{t+1} = \theta_t + \Delta\theta_t \tag{2.51}$$

With the development of the ADADELTA update rule, the need to set the default learning rate could be avoided.

- **RMSPROP** - The RMSPROP is an adaptive learning method that seeks to eradicate ADAGRAD's decreasing learning rates issues. The equations for this are given as:

$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_t^2 \tag{2.52}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t \tag{2.53}$$

Ideal values for the γ, η have been proposed as 0.9 and 0.001 respectively.

- **ADAM** - The Adaptive Moment Estimation or ADAM [132] that also performs parameter-wise update. It keeps a record of both the past squared gradients v_t like the ADADELTA and RMSPROP and past gradients m_t like the Momentum algorithm. The equations can be given as follows:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \tag{2.54}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \tag{2.55}$$

In these equations the m_t, v_t are the first and second moment estimates of the gradients. The two vectors are initialized with zeros and are always biased towards it. This also happens when the decay rates are small. In order to

overcome the issues they introduce the bias correction terms as:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{2.56}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{2.57}$$

The above two equations are then used to update the parameters as given below:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t \tag{2.58}$$

The suggested value for $\beta_1, \beta_2, \epsilon$ are 0.9, 0.999, $1e-8$, respectively.

- **NADAM** - The Nesterov Accelerated Adaptive Moment Estimation [133] combines the ADAM and NAG's qualities. We already know that RMSPROP calculates the exponentially decaying average of the past squared gradients ν_t and Momentum calculates the exponentially decaying average of past gradients m_t. As seen earlier, the momentum update rule is given by:

$$g_t = \nabla_{\theta_t} J(\theta_t) \tag{2.59}$$

$$m_t = \gamma m_{t-1} + \eta g_t \tag{2.60}$$

$$\theta_{t+1} = \theta_t - m_t \tag{2.61}$$

Expanding the third equation gives us

$$\theta_{t+1} = \theta_t - (\gamma m_{t-1} + \eta g_t) \tag{2.62}$$

This shows that momentum allows taking a step in the direction of the previous momentum vector and a step in the current gradient's direction. The NAG enables a better step prior to gradient calculation. We can see that only g_t needs to be modified to arrive at NAG.

$$g_t = \nabla_{\theta_t} J(\theta_t - \gamma m_{t-1}) \tag{2.63}$$

$$m_t = \gamma m_{t-1} + \eta g_t \tag{2.64}$$

$$\theta_{t+1} = \theta_t - m_t \tag{2.65}$$

Dozat suggested a one time update to the NAG's update equation as follows:

$$g_t = \nabla_{\theta_t} J(\theta_t) \tag{2.66}$$

$$m_t = \gamma m_{t-1} + \eta g_t \tag{2.67}$$

$$\theta_{t+1} = \theta_t - (\gamma m_t + \eta g_t) \tag{2.68}$$

Now, in order to get the Nesterov momentum to ADAM we can replace the previous momentum vector with the current one. Recalling ADAM update rule:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{2.69}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{2.70}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}} + \epsilon} \hat{m}_t \tag{2.71}$$

Expanding the second equation with $\hat{m}_t$ and m_t gives the following:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}} + \epsilon} \left(\frac{\beta_1 m_{t-1}}{1 - \beta_1^t} + \frac{(1 - \beta_1) g_t}{1 - \beta_1^t} \right) \tag{2.72}$$

We note that $\frac{\beta_1 m_{t-1}}{1-\beta_1^t}$ is the bias corrected estimate of the momentum vector of the last time step and can be substituted by $\hat{m_{t-1}}$:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}} + \epsilon} \left(\beta_1 \hat{m}_{t-1} + \frac{(1 - \beta_1) g_t}{1 - \beta_1^t} \right) \tag{2.73}$$

The Nesterov momentum can now be added by substituting the bias corrected estimate of the momentum vector of the previous timestep $\hat{m}_{t-1}$ with the bias corrected estimate of the current momentum vector $\hat{m}_t$ giving us the final update equation as:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \left(\beta_1 \hat{m}_t + \frac{(1 - \beta_1) g_t}{1 - \beta_1^t} \right) \tag{2.74}$$

2.3.2 Statistical Inference

2.3.2.1 Maximum Likelihood Estimation

The maximum likelihood estimation or MLE provides a simple statistical tool to estimate the parameters of given statistical model. To find these estimates we need a *likelihood function* of the given model. The likelihood function can be

thought of as a simultaneous density function which when used in the MLE, the observations are held fixed and the parameters are set to vary. The main idea behind its mechanism is to search for the most likely values of the parameters that would generate the given data/observations. Therefore, this problem can be defined as an optimisation problem where the values of the parameters are optimised to match the observations. Or, in simpler words we need to maximize the likelihood function with respect to the parameters. If the probability density function is given by $f(x|\theta)$ where θ denotes the parameter vector of length q and x is a set of random samples produced from this density function, then the joint probability of x can be given as

$$f(x_1, x_2, \ldots, x_n|\theta) = f(x_1|\theta) \cdot f(x_2|\theta) \cdot \cdots \cdot f(x_n|\theta) = \prod_{t=1}^{n} f(x_t|\theta) \tag{2.75}$$

Each sample x_t is assumed to be independent and identically distributed (IID). Now, the likelihood function is the same probability density function with x remaining fixed the θ is allowed to change. It is given as

$$L(\theta|x_1, x_2, \ldots, x_n) = \prod_{t=1}^{n} f(x_t|\theta) \tag{2.76}$$

For many practical purposes and numerical computation it is necessary to work with the logarithms of the likelihood function. Other reason being the product gets converted to a sum and the computation of derivatives becomes easy. The log-likelihood is gives as

$$LL(\theta|x_1, x_2, \ldots, x_n) = \log\left(\prod_{t=1}^{n} L(\theta|x_t)\right) = \sum_{t=1}^{n} \log L(\theta|x_t) \tag{2.77}$$

The computation of MLE is done by finding the θ that maximizes the value of $LL(\theta|x)$ function. This is done by taking the derivatives of the function w.r.t. the parameters, as given here and setting it zero. The resulting system of equations can be solved numerically to get the most optimal values of the parameters.

$$\frac{\partial LL(\theta|x)}{\partial \theta_{\mathbf{j}}} = 0, j = 1, \ldots q \tag{2.78}$$

Some of the important properties of the MLE can be stated as follows:

- When the number of observations is high the MLE approaches the true parameter value
- When using the Gaussian Distribution to define the probability density func-

tion of a multi-parameter system the covariance matrix is proportional to the inverse of the Fisher information matrix.

- The MLE of any function of θ, $\tau(\theta) = \tau(\theta_{MLE})$ if the MLE of θ is θ_{MLE}. This is also called the invariance property.
- The probability density functions must be known prior to estimation

2.3.2.2 Maximum a posteriori Estimation

Suppose $\mathbf{x} \in \mathbb{R}^{\mathbf{n}}$ be the data generated by an unknown process. Let $\mathbf{y}$ be a partially observed signal related to $\mathbf{x}$ such that the likelihood function is given as $p(\mathbf{y}|\mathbf{x})$. Also, let us assume that the problem of reconstructing $\mathbf{x}$ from $\mathbf{y}$ is ill-posed. This leads to high uncertainties in the estimation of x. The Bayesian way of dealing with this problem involves the use of prior or expert knowledge about the distribution of $\mathbf{x}$. In particular, $\mathbf{x}$ is treated as a random vector having a prior distribution $p(\mathbf{x})$ that can be combined with the observed data, i.e. $p(\mathbf{y}|\mathbf{x})$ using the Bayes' theorem. The posterior distribution is given as the following:

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{\int_{R^n} p(\mathbf{y}|\mathbf{x})p(\mathbf{x})dx} \tag{2.79}$$

When $\mathbf{x}$ is a high dimensional quantity, inference becomes impossible and as a result we just use the point estimates that capture some of the information about $\mathbf{x}$. Thus, the MAP estimator of $\mathbf{x}$ is given by

$$\hat{\mathbf{x}}_{MAP} = \arg\max_{\mathbf{x} \in R^n} p(\mathbf{x}|\mathbf{y}) \tag{2.80}$$

The integrable term in the denominator of the Eq. 2.79 is generally done using Markov chain Monte Carlo techniques. The choice of sampler becomes crucial for high dimensional inference. The modern samplers that are good with this task are the Hamiltonian Monte Carlo (HMC) and the No-U-Turn Sampler or NUTS [134].

2.4 Summary

In this chapter, various algorithms that are used to develop the models in chapters 3 and 4 were studied. The *seq*2*seq* model shown in section 2.1.3.2 was previously considered for the development of the Reduced order Model in chapter 4, but it did not show an improvement in the multistep prediction tests and hence could not be used further. The various gradient descent optimization algorithms were each tested for optimizing the developed deep networks in the forthcoming chapters

3 and 4. Among all, the ADAM optimizer (2.3.1) was considered for the final implementation. Autoencoders were previously considered to learn more relevant features in reduced space of the states in order to enhance the multistep prediction algorithm in chapter 4, but the implementations were not successful. For the dimensionality reduction part in chapter 4, the ICA (2.2.3), the sparse encoding (2.2.2) and POD (2.2.1) were tried. Out of all the three, the POD scheme turned out to be more robust than the others and was used in the final implementation.

Chapter 3

Application-I: A Data-Driven Approach to Porosity Segmentation for Carbonates

3.1 Introduction

In this chapter, we develop a way of analysing complex pore networks in carbonate rock samples using a simulation-based study. Understanding the variation of pressure wave velocities with the shape and size of the pores is a nontrivial problem [135]. The problem arises because of the way the pore networks are developed with time. Geological processes play a key role in defining the petrophysical content and heterogeneity of the rocks. Some of them are attributed to deposition and diagenetic processes that largely control the rock texture viz, grain size, arrangement and uniformity, the clay content and the proportion of minerals. All of these affect the shape and size of the pore networks in a rock. Different materials combine to generate a pore structure that is unique from others of its kind. The shapes, in turn, enable the fluid to take the shape of the pore, which in turn affects the wave phenomena such as reflection, refraction and scattering [135; 136]. Although the current chapter focuses on wave properties in carbonate rock samples, the phenomena are quite general in other allied studies as well.

There have been a few investigations in this regard. Here we list a few works involving studies of wave properties and pores in the context of reservoirs. The authors in [137] found that porosity and clay content were the most important factors affecting the wave velocity in sandstones. In [138], they found that there exists a relation between the porosity and collectively the mineralogy, grain sorting and angularity. The authors in [139] introduced porosity and frame flexibility

factors to quantify the effects on elastic wave propagation. In [140] it was stated that the fluid's presence could help dissipate the travelling wave's energy along the contact plane between the fluid and solid. In [141] the authors modelled the energy partitions in the refracted and reflected parts of the wave for a poroelastic seabed model. Elastic wave velocities were investigated under different saturated conditions in [142] where they found that thin pores affected the elastic moduli more than rounded pores, and the presence of fluid tended to affect the compressional velocity more than the shear velocity. In [143] the authors asserted the use of pore geometry factor prior to calculation of elastic moduli of the rock in the effective medium modelling approaches like the differential effective medium models. The effect of pore size and shape on acoustic velocities was summarised for travertines in [144]; it was found that the petrophysical properties such as porosity, permeability and acoustic velocity are related to seismic reflection data. The authors in [145] had studied the effect of pore fluids and connectivity on reflected wave amplitudes. It was suggested in [146] that the microstructural properties of the carbonate reservoir could be predicted by the seismic quality factor.
In the current work, we base our study on acoustic simulation data. Simulation of the acoustic wave instead of elastic waves is justified given that we are mostly interested in the variation of P-wave properties here. Additionally, it was recommended in [147; 148] that the simulation of elastic waves is computationally intensive and can be substituted by its acoustic counterpart when feasible. It is also reasonable to study only the P-waves when the observed data mainly consists of primary wave data as in [149]. In this chapter, we describe the development of a simulation-based study to evaluate the effects of pore geometry on acoustic wavefields. We believe that it is important to build a relation between the pore network structure and the acoustic response of the pore structure model.

3.2 Extraction of Pore Network Models

The first-hand data for studying the pore networks can be obtained using 4D-XRM equipment. The raw 3D volumetric images can be processed to study the different geometrical aspects of the pores. The maximum and minimum resolution at which the rock samples were scanned was $40\mu m$ and $10\mu m$. The cylindrical samples were cored from D1-field of the Bombay offshore basin and consisted of carbonates. Figure 3.1 shows the flowchart for processing the raw images. The current work proposes a deep learning based method for porosity segmentation. In order to generate the training data for supervised learning, the extraction must be as accurate as possible to the real pore network. Every step in the given flowchart

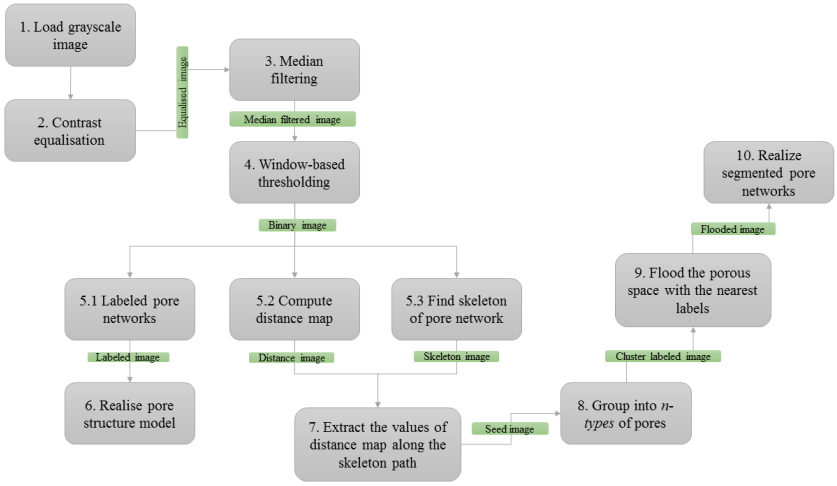


Figure 3.1: Segmentation flowchart: 1) Load the raw grayscale image; 2) Perform contrast equalization on the grayscale image; 3) Apply median filtering on the contrast equalized image; 4) Perform a window-based Sauvola threshold operation to obtain a binary image; 5) Treat the binary image as foreground ($label_{pore} = 1$) and background ($label_{matrix} = 0$); Compute the distance map of the binary and extract the skeleton of the network; 6) Realize a pore structure model; 7) Extract distance map values along the skeleton paths; 8) Perform k-means clustering on the extracted distance values to generate n labels with the cross-sectional area as feature; 9) Perform watershed/flooding transform to propagate the labels in the foreground; 10) Realize a segmented pore network model

is important to realise a good segmented image. Although there always exists some level of subjectivity in processing the images, we often make qualitative judgements about the parameter values at each step, which is mostly sufficient to proceed with the extraction.

3.2.1 Extraction of Pore Networks

3.2.1.1 Volumetric Raw Image Preprocessing

As can be seen in Fig. 3.2 each of the rock samples appear different from the other. Table 3.1 shows simple statistics describing the grayscale values of the samples. The variation can often be attributed to the generation of different ranges of grayscale levels during the imaging process. This marks just one of the few issues encountered in processing 3D rock images. The entire process of volumetric image processing, as shown in Fig. 3.1 can be grouped into three parts. We closely follow the steps taken in [150] and name the steps as – preprocessing, pre-segmentation and segmentation. For simplicity's sake, we work with 3D cubes

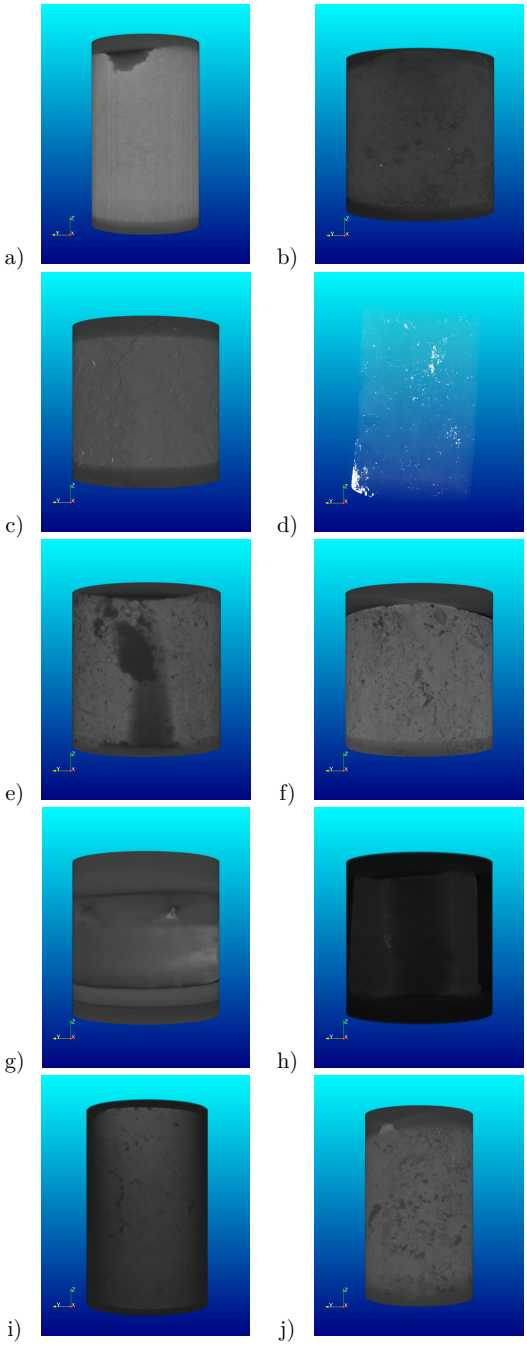
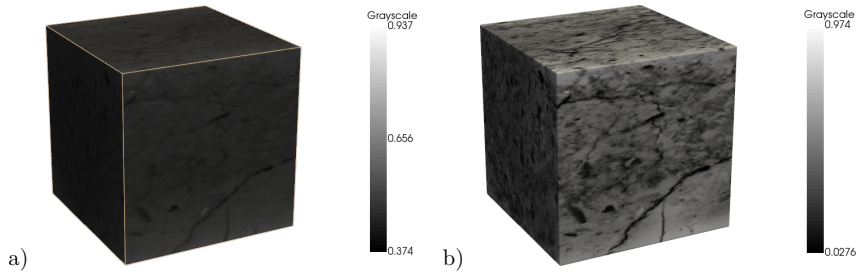


Figure 3.2: From a) to j): Raw grayscale rock samples numbered 1 to 10 have been scanned using 4D-XRM equipment. Each sample is of an individual physical dimension. The colour variation and the contrast can be easily identified and compared with each other

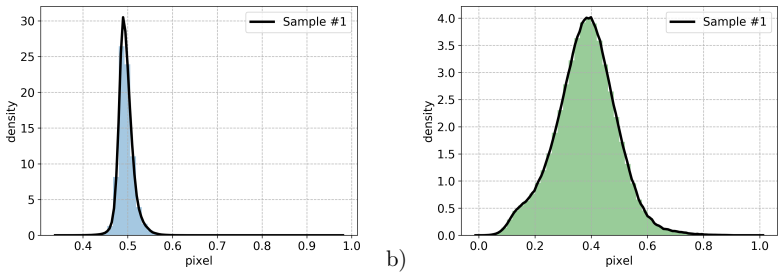
Table 3.1: Exploratory statistics of voxel data of rock core samples

Sample No.	Length (cm)	Diameter (cm)	Upper	Lower	Mean	Std.
1	3.02	2.52	255	0	69.87	52.48
4	2.51	2.52	221	0	52.43	32.16
5	2.51	2.52	237	0	68.04	40.82
9	3.74	2.52	255	0	39.10	60.84
12	3.74	2.52	224	0	66.50	39.49
13	5.06	3.80	255	0	75.94	46.84
18	6.29	2.53	255	0	80.56	55.40
21	5.06	3.80	221	0	20.52	14.13
104	5.03	3.80	255	0	105.92	72.37
105	4.92	3.80	240	0	22.06	36.21



a) b)

Figure 3.3: a) Raw grayscale sample before contrast equalization; b) Grayscale variation after contrast equalization; The color variation and the contrast can be easily identified and compared with each other in the colorbars



a) b)

Figure 3.4: a) Kernel density estimate of raw grayscale values; b) Kernel density estimate after contrast equalization; The width at the bottom clearly justifies the clarity of features seen in Fig. 3.3.

extracted from full cylindrical core samples. This greatly eases operations such as slicing and matrix calculations.

Loading the images into the computer marks the beginning of the preprocessing step. The raw grayscale images often suffer from poor dynamic range. Contrast equalisation is a possible remedy in such situations. In order to have a rough estimate of the pixel value ranges, a kernel density estimation was performed on the samples prior to and after post-contrast equalisation steps. The difference can be seen in Fig. 3.3a) and b). Next, in order to obtain a good segmentation result, it is highly recommended to remove all types of noise in these images. This noise arises due to improper handling of instruments and/or improper selection of input parameters. Their removal is essential as it can mask the small pore features present in the images. As described in [151], an image can be decomposed into two parts – the pure pixels part and the noise.

$$I\left[x, y, z\right] = J\left[x, y, z\right] + \eta[x, y, z] \tag{3.1}$$

where x, y, z represent the coordinates of the pixel, J represents the original noise-free image and I, the corrupted version of the image. The most common type of noise found in 3D volumetric data is of the impulse type or the gamma type. Following [19] these can be defined using probability density functions as in:

$$Impulse(x) = f(x) = \begin{cases} a\exp(-ax), x \geq 0 \\ 0, x < 0 \end{cases} \tag{3.2}$$

$$Gamma(x) = f(x) = \begin{cases} \frac{a^b x^{b-1}}{(b-1)!}\exp(-ax), x \geq 0 \\ 0, x < 0 \end{cases} \tag{3.3}$$

There exist multiple ways of removing such noises from the images as suggested in [152; 153; 154; 155; 156]. Notably, a simple yet effective technique to remove impulse and gamma noises is the application of median filter [157; 158].

As shown in the flowchart we also perform a contrast equalization as a preprocessing step prior to the application of the median filter as in [159]. A structuring element of size $3 \times 3 \times 3$ was used for the median filter. The application of the median filter marks the execution of the preprocessing step. The result of the application of the median filter is shown in Fig. 3.7. Once the images pass the median filter stage they are subjected to the pre-segmentation stage marked by the application of the Sauvola [160] thresholding algorithm.

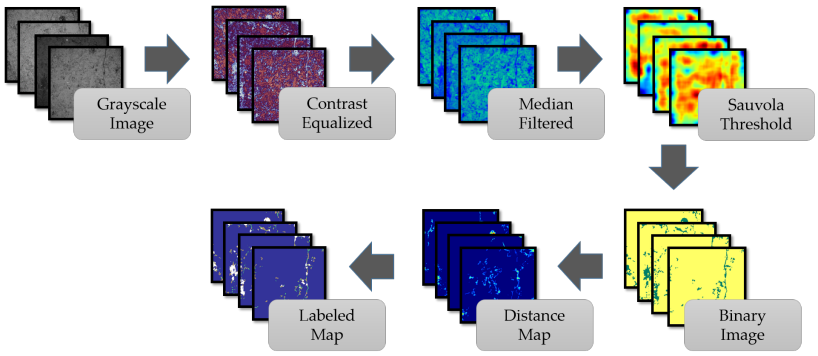


Figure 3.5: A 2D illustration of sample output images after performing operations shown in Fig. 3.1.

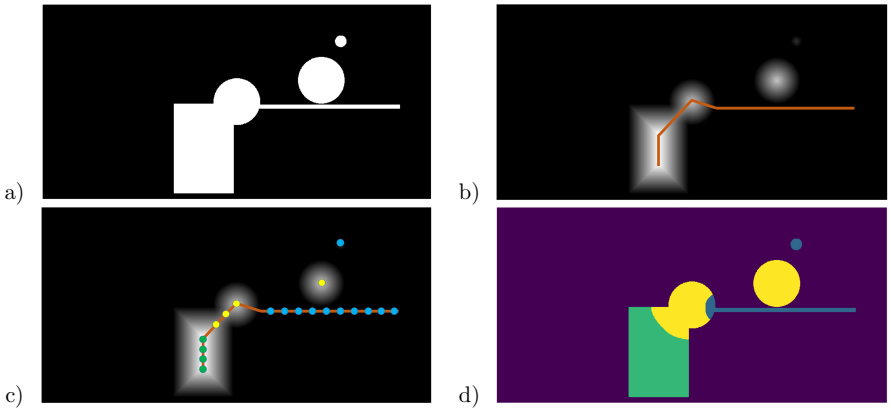


Figure 3.6: A 2D illustration of segmenting poretypes based on relative cross-sectional areas using k-means clutering operation: a) Binary image (white represents the porous region); b) Distance map of the porous region with skeleton marked by brown color; c) Marker generation based on k-means clustering algorithm - same color dots represent regions of similar cross-sectional areas; d) Final segmented image - regions of same color denote similar cross-sectional areas

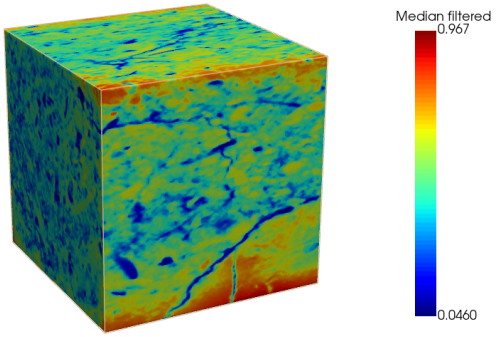


Figure 3.7: A median filtered image

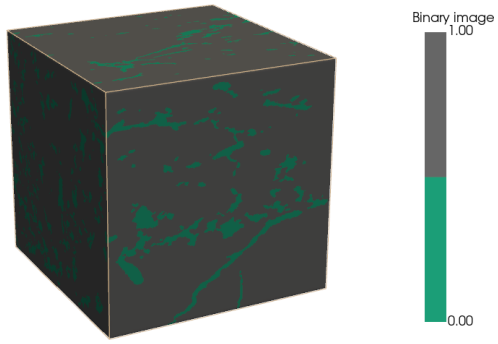


Figure 3.8: A Sauvola thresholded image

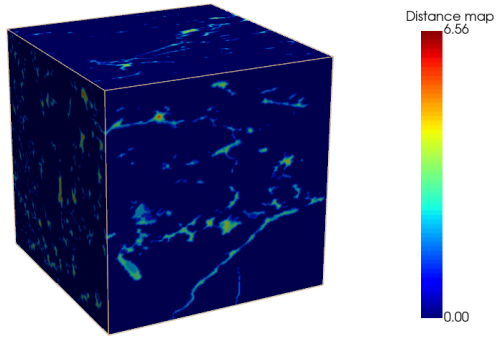


Figure 3.9: Distance map of the binary image

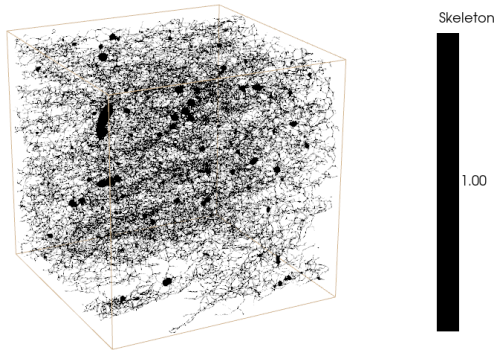


Figure 3.10: Skeleton of the binary image

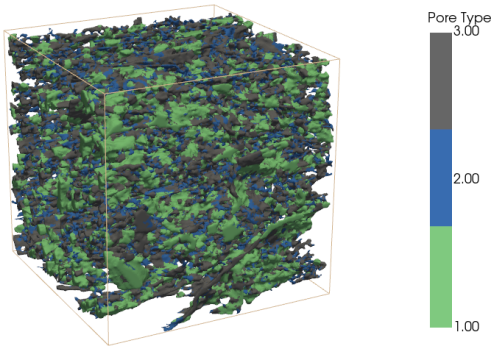


Figure 3.11: Porosity segmented image

Table 3.2: Calibrated Porosity

Sample	Porosity observed	Porosity computed
1	12.0	11.81
4	8.5	8.26
5	11.2	10.70
9	23.3	23.10
12	13.6	13.25
13	11.4	10.89
18	7.70	6.85
21	16.1	15.96
104	14.0	13.72
105	24.5	24.44

3.2.1.2 Sauvola Binarization Algorithm

The conventional method of binarising an image uses the global thresholding technique. This procedure helps reduce the complexity of the image and gain information about the interesting regions. In the context of porosity extraction, the procedure assigns a pixel as a background if its grey value is smaller than the threshold value, and if its value lies above, it gets assigned as a foreground. The literature cites three main methods of thresholding – global, local and adaptive. The global method uses a single threshold value for partitioning. The Otsu technique is based on discriminant analysis and is one of the most used global thresholding methods. It determines a threshold by maximising the interclass variance [161; 162]. On the other hand, the local method uses several threshold values for smaller sub-images. The adaptive method differs from both in the way it computes a threshold for each pixel in the image. Continuous efforts have been put into optimising this procedure, but it still remains a challenge. Local and adaptive methods use windows or sub-images for calculating a local threshold value for each [163]. Determination of this window size is the most important step in these techniques. The assumption

Table 3.3: Sauvola Binarization Parameters

Sample	Window size (k)	Weight (w)
1	37	0.2
4	31	0.4
5	31	0.4
9	41	0.4
12	27	0.4
13	31	0.3
18	35	0.3
21	41	0.3
104	33	0.3
105	31	0.3

of regular distribution of intensity values leads to non-optimal binarisation as in [162] technique. Two of the important adaptive techniques are described next.

Niblack Binarization: This method heuristically chooses a threshold value for a given window. The method relies on the pixel values of the neighbourhood. The Niblack formula [164] determines a threshold by adding a weighted local standard deviation to the local mean of the window. The problem of window size determination is inherent in this technique as well. The formula for threshold is given by

$$T(x, y, z) = m(x, y, z) + k \times s(x, y, z) \tag{3.4}$$

where T is the threshold, m is the mean, s is a standard deviation of the pixels in the current window, and k is the weight for each pixel at x, y, z.

Sauvola Binarization: The Sauvola binarization [160] method uses the method of integral images for binarization [160; 162]. Originally developed for documents, the method has proven to be equally good for other binarisation tasks. It owes an improvement over the Niblack technique. The technique readily handles bad illumination and irregular distribution of pixel intensities. The threshold is calculated using the following equation:

$$T(x, y, z) = m(x, y, z) \times \left[1 + k \cdot \left(\frac{s(x, y, z)}{R} - 1\right)\right] \tag{3.5}$$

Where T is the threshold, R is the dynamic range of gray level of the pixels, m is the mean and s is a standard deviation of the pixels in the current window. The application of the Sauvola binarization step marks the end of the pre-segmentation stage. The result of the Sauvola binarization can be seen in Fig. 3.8. The labora-

tory obtained porosity (Table 3.2) was used as a guide to decide on the values of the Sauvola binarization parameters shown in Table 3.3.

3.2.2 Pore Segmentation

The steps of segmentation are schematically shown in Fig. 3.6. Once the images pass the Sauvola binarisation stage, they are subjected to a distance transform filter [165]. The prerequisite to the application of this filter is the division of the input image into foreground and background, where the foreground represents the pore network, and the background is the rock matrix. The distance transform filter computes the distance of every pixel in the foreground to the nearest background pixel. This way, the distance map stores the values of the area of the cross-section along the length of the pore path. Among the many distance calculation functions, the most popular ones are the following:

$$chessboard(p,q) = \max(|p_1 - q_1|, |p_2 - q_2|) \tag{3.6}$$

$$cityblock(p,q) = |p_1 - q_1| + |p_2 - q_2| \tag{3.7}$$

$$euclidean(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2} \tag{3.8}$$

where, $(p_1, q1), (p_2, q2)$ are the pixel values at two given points. The result of the application of this step is the creation of a maxima inside every catchment basin [166] in the foreground. The result of the distance transform can be seen in Fig. 3.9.

Once the distance map is computed, the images are subjected to skeletonisation [167]. The purpose of extracting the skeleton is to extract values of the distance map along the skeleton paths, as shown in Fig. 3.10. The extracted distance values are then stacked and flattened into a 1D vector. This 1D vector is then used by the k-means algorithm [168] to cluster the values into three groups. This grouping divides the foreground/pore part into three groups according to the cross-sectional widths of the pores by assigning unique labels to each group. This way, the labels of the pores are stored as a function of the cross-sectional widths of the pore paths giving us a labelled image where each label signifies the width of the "local" pore. These labels are then propagated to other parts of the foreground using the watershed algorithm [169]. A sample of the segmented image is shown in Fig. 3.11. After segmentation, the extracted pore networks are used to define the computational domains for acoustic wave simulations, as described in the following sections.

3.3 Acoustic Wave Analysis

3.3.1 The Acoustic Wave Model

In order to solve the acoustic wave equation, the pseudospectral time-domain method has been proven to be superior to the conventional finite difference time domain methods [170; 171]. We begin by stating the governing equations.

3.3.1.1 Governing Equations

The momentum and mass conservation equations can be written as the following:

$$\rho_0 \frac{\partial u}{\partial t} + \nabla p = \rho \frac{\partial u}{\partial t} - \frac{1}{2}\rho_0 \nabla (u^2) \tag{3.9}$$

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho_0 u) = -\nabla \cdot (\rho u) \tag{3.10}$$

where p, ρ, u are the acoustic pressure, density and particle's velocity. ρ_0 is the ambient acoustic density. Solving equations 3.9 using standard finite difference techniques can be difficult. The use of spectral methods provides a superior alternative to this. In general, it is sufficient to assume the heterogeneity effects on the wave field to be of second order, implying any higher order terms would be discarded. The work by [172] suggested the second order terms be re-written for Eq. 3.9 in terms of acoustic Lagrangian density. The new equation then takes the form:

$$\rho_0 \frac{\partial u}{\partial t} + \nabla p = -\nabla L \tag{3.11}$$

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho_0 u) = \frac{1}{c_0^2}\frac{\partial L}{\partial t} + \frac{1}{c_0^2}\frac{\partial L}{\partial t} + \frac{1}{\rho_0 c_0^4}\frac{\partial p^2}{\partial t} \tag{3.12}$$

$$L = \frac{1}{2}\rho_0 u^2 - \frac{p^2}{2\rho_0 c_0^2} \tag{3.13}$$

where, L is the second order Lagrangian density.

In the case when only cumulative nonlinear effects are required to be modelled the Lagrangian density can be set to zero. We then have the following pair of equations to solve:

$$\frac{\partial u}{\partial t} + \frac{1}{\rho_0}\nabla p = 0 \tag{3.14}$$

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho_0 u) = \frac{1}{\rho_0 c_0^4}\frac{\partial p^2}{\partial t} \tag{3.15}$$

Solving Eq. 3.14 using the spectral method involves re-writing the nonlinear con-

Table 3.4: Material Assignment

Material type	Velocity (m/s)	Density (kg/m^3)
Rock Matrix	5200	2700
Fluid Medium	1500	1000

Table 3.5: Qualitative description of the pore networks

Type	Description
Case #1	Pores proximal to one of the surface
Case #2	Scattered and Branched Pore Network
Case #3	Dominating Crack Pore around the middle
Case #4	Matrix dominated

vective term in the mass conservation term as a spatial gradient. The reason for this is that the spectral gradients can be calculated using spectral methods, while the computation of temporal gradients requires standard finite difference equations. The complete set of coupled partial differential equations can be obtained in the form of 3.11 after a number of substitutions from 3.9, as:

$$\frac{\partial u}{\partial t} = -\frac{1}{\rho_0}\nabla p \tag{3.16}$$

$$\frac{\partial \rho}{\partial t} = -(2\rho + \rho_0)\nabla \cdot u - u \cdot \nabla \rho_0 \tag{3.17}$$

$$p = c_0^2 \left(\rho + d \cdot \nabla \rho_0 + \frac{B}{2A}\frac{\rho^2}{\rho_0} - l\rho \right) \tag{3.18}$$

The loss term l is given by the following fractional Laplacian as:

$$l = \tau \frac{\partial}{\partial t}(\nabla^2)^{y/2-1} + \eta(-\nabla^2)^{(y+1)/2-1} \tag{3.19}$$

where τ, η are the absorption and dispersion coefficients. The power law attenuation for acoustic waves has been found suitable for rocks [173; 174]. For a detailed description of the above-mentioned equations, the reader is referred to the excellent article by [171; 175; 176] and a review by [177].

3.3.2 Description of the Simulation

The simulation of acoustic wave propagation is computationally expensive and demands high-end computing clusters for large meshes. In the current study, we work with small sub-cubes derived from large cubes. The dimensions of the large cubes were $256 \times 256 \times 256$ voxels in respective axes. We extract an overcomplete set of 400 smaller cubes of dimensions $64 \times 64 \times 64$ for the purpose of this study. The

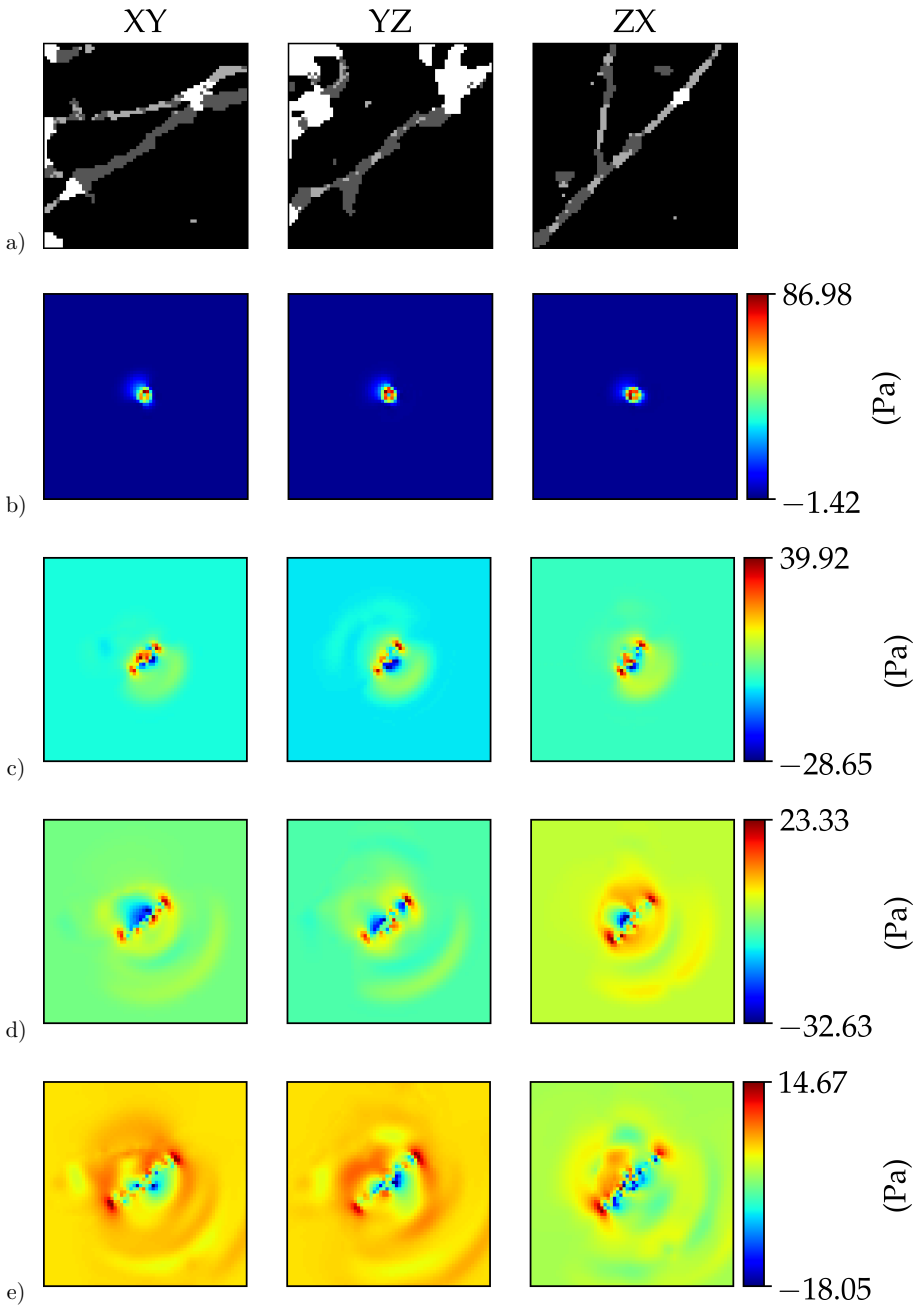


Figure 3.12: a) An instance of the pore network as seen from XY, YZ and ZX planes after extracting the middle slice in the perpendicular axis to the plane; b) to e) Progress of wave fields with increasing time, $t(\mu s) = 0.53, 0.90, 1.26, 1.63$

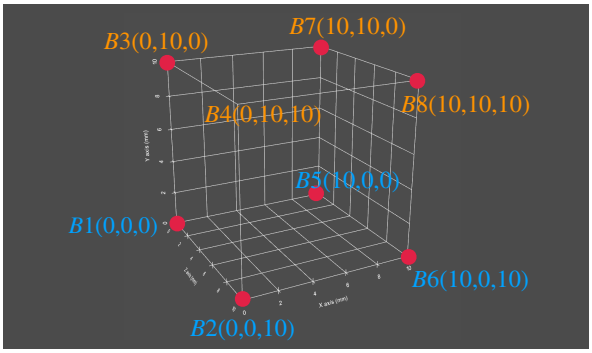


Figure 3.13: Probe locations on the cube

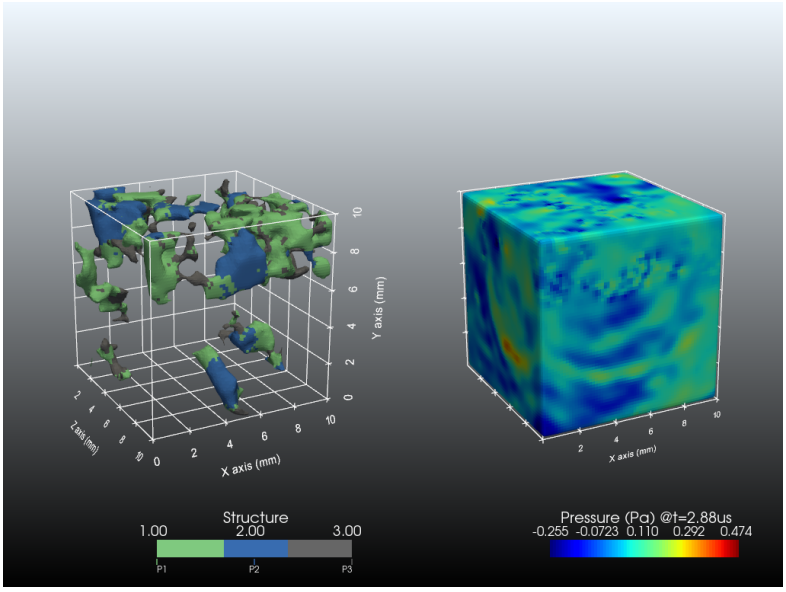
64 voxels equal $10mm$ in each direction. The centre of the cube has the coordinate $(5mm, 5mm, 5mm)$. The source of the pulse is located at the centre of each of the simulations with a source frequency equals to $30kHz$. This frequency was chosen to be near the frequency of practical sensors used in well-logging equipment. Although one would ideally set up the simulation befitting an actual well-logging scenario and use the bulk velocity response of the formation to train the neural network so that compressed sensing of the causative pore structure can be realized, instead we have taken a preliminary step towards such advanced application by considering the full acoustic volume information from each simulation. The reason for this is twofold – first, the compressed sensing technique would require the use of generative adversarial frameworks and high end hardware and two, the simulation of an ideal scenario would require a detailed case study. The amplitude of the source pulse is taken to be 10.50Pa. Each simulation was designed to run for 1281 timesteps or $11.53\mu s$. For post-processing, every voxel in the cube is monitored during the simulation. Absorbing boundary conditions are applied at the six boundaries of each cube, where the widths of the absorbing layer were taken to be 20 voxels. Therefore, in total, the computational domain has the dimension of $84 \times 84 \times 84$ voxels. The simulation generated acoustic volumes were of size $64 \times 64 \times 64$. For simplicity's sake, the acoustic volumes corresponding to timestep, $t_{step} = 1165$ or $t = 10.49\mu s$ were used during training. The reason for taking a higher timestep is that by this time, the wave would have encountered multiple reflections and refraction and would carry more information about the causative pore configurations. We also put eight probes $B1, \ldots, B8$ at the corners of each cube, as shown in Fig. 3.13. This is done to analyse the received signals.

3.3.3 Effect of Pore Shapes on travelling Wavefields

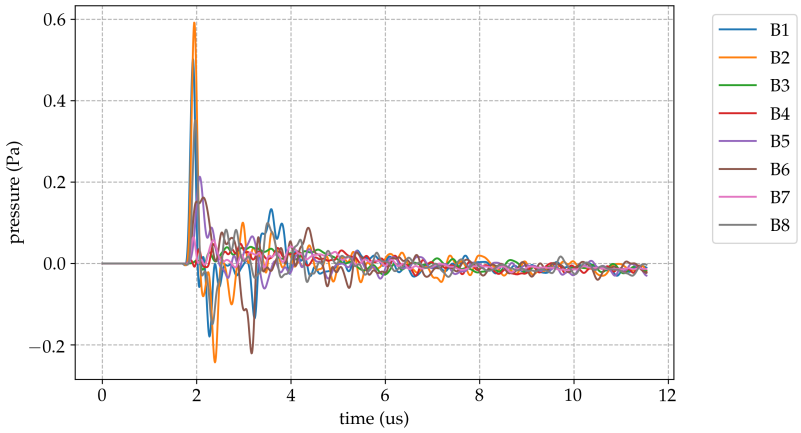
A sample propagation of the acoustic wave is shown in Fig. 3.12. The first, second and third columns show the progression of the wave in three central planes of the cube. The first row shows the plane of the pore network. The subsequent rows show how the acoustic wave undergoes reflection and refraction along the propagation path. The slices are shown for specific timesteps as depicted in the figure caption. In every simulation, the material property was specified using velocity and density as given in Table 3.4. The values were chosen after evaluating a number of numerical experiments. We have presented here one among those cases. The mineralogy represented by such values can be indicative of calcite dominated carbonate rock sample with cementation effects and low porosity.
In order to have an idea about the change in wave field due to the pore network, we demonstrate 4 cases, each with different geometry. An acoustic wave simulation was run on each of the 400 cubes, with the source of the pulse being located at the centre $x = 5mm, y = 5mm, z = 5mm$. We list the four types as Case $\#1, 2, 3$ and 4. The description of the types is given in Table 3.5. We disregard the material type at the location of the source, i.e. the pulse could originate either at the rock matrix medium or the fluid medium. Part of the reason for this is to generate a dataset with as much variation as possible because, in general, higher variation ensures better generalisation.
A complex pore network scattered across the cube's volume will present more low-velocity regions for the acoustic wave to travel. This will result into complex interference patterns and can be observed on the cube surface. This justifies the use of a statistical metric, the standard deviation for being indicative of the complexity of the pore network. Later, in the coming sections, we show the standard deviation values for each case in the form of graphs. Although every simulation was run for 1281 timesteps, we just show one timestep for each case here to prevent clutter. But, we do show the complete signals received at the 8 probing points $B1 \dots B8$ for all four cases. The probe locations are shown in Fig. 3.13. We report the signals received at these locations for all 1281 timesteps.

3.3.3.1 Case #1

In the first case we consider pore network as shown in Fig. 3.14a). Shown towards the left is the part of the pore network on which the acoustic simulation was run. Towards the right is the wave field obtained at time $t = 2.88\mu s$. The pressure amplitudes received at the probing points are shown in Fig. 3.14b). The upper and lower values are around 0.6Pa and -0.2Pa. The coherent part of the wave



a)



b)

Figure 3.14: Case #1: a) Part of the pore network due to sub-cube #350 of sample #105; Pressure wave field at $t = 2.88\mu s$ exhibited by the sub-cube; b) Waveform of signals received at 8 corner probing points $B1 \dots B8$ of the sub-cube

appears to be small compared to the broader coda part. The higher magnitude is received by probe $B2$.

3.3.3.2 Case #2

In the second case, the pore network is more scattered within the cube, as shown in Fig. 3.15a). The corresponding wave field at time $t = 2.88\mu s$ is shown towards the right. At the bottom of Fig. 3.15b) we show the signal pattern due to the network. The waveforms appear to have a constant range of magnitudes, with $B1, B5$ being an exception. This can be seen at timestep $t = 5\mu s$ to $t = 7\mu s$. The high and low magnitudes are around 0.8Pa and -0.06Pa. These are small compared to case #1. This can be attributed to more scattered pores occupying more volume. Larger volumes of low-velocity regions dissipate the wave energy more. Moreover, the coherent part of the received signals cannot be clearly demarcated and are rather unclear w.r.t. their boundaries. Even the coda part merges with the coherent part with the same levels of disorder.

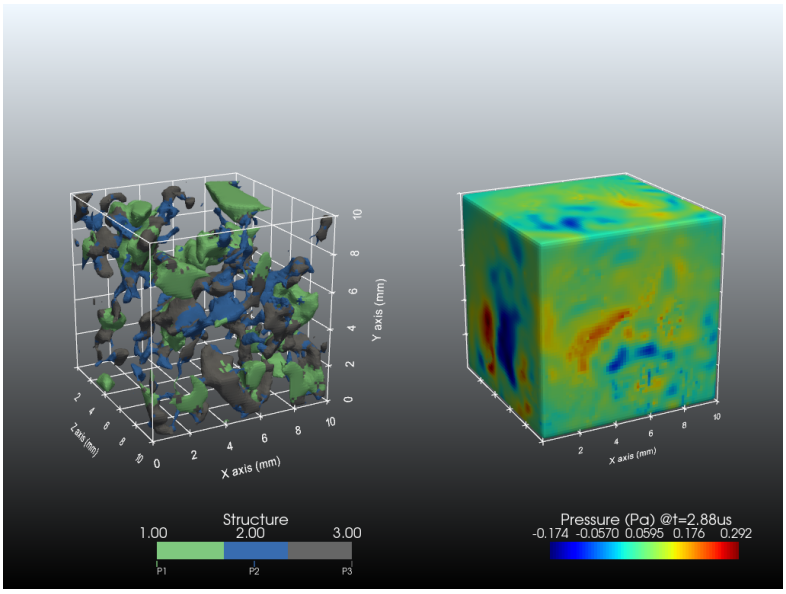
3.3.3.3 Case #3

In the third case, we consider a pore network as shown in Fig. 3.16a). The network exhibits a crack type void around the centre of the cube in the ZX plane. The corresponding wave field at time $t = 2.88\mu s$ is shown towards the right of the figure. The highest magnitude of 1.5Pa is recorded by the probe $B8$. Perhaps, the probable reason for this was the occurrence of more constructive interference of the reflections from different parts of the pore network. The coherent part of the waveforms can be clearly seen in all the probes. Probes $B4, B8$ distinguish themselves from others w.r.t. amplitudes and oscillations.

3.3.3.4 Case #4

In the fourth case, we consider a pore network as shown in Fig. 3.17a). The network is unique in the sense that most part of it is empty, signifying the wave has more high-velocity regions to travel due to the rock matrix. The corresponding wave field at $t = 2.88\mu s$ shows clear wavefronts on the surfaces. One can also see the interference occurring at the side and top planes towards the bottom of Fig. 3.17b) we can see the signals received by the 8 probes. The high magnitude of 0.15Pa was recorded by $B7$ and seconded by $B2$. As can be seen in the top right part of the figure, destructive interference dominates most of the wave interactions. The colour bar indicates the situation accordingly. The high red values occupy less space signifying more low values had to be accommodated to show the amplitudes.

a)



b)

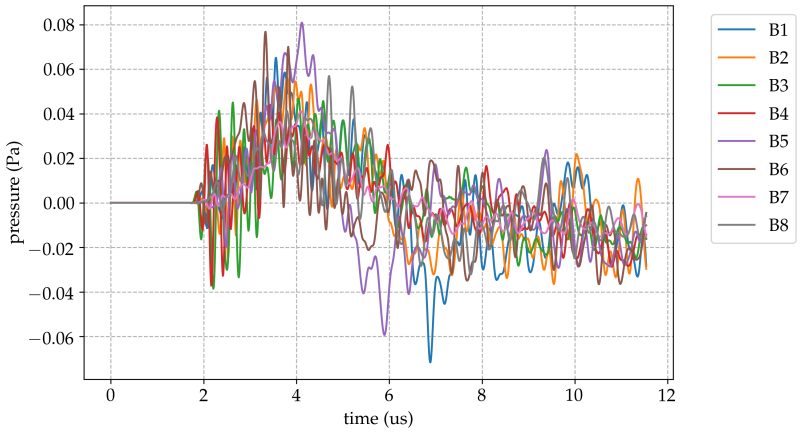
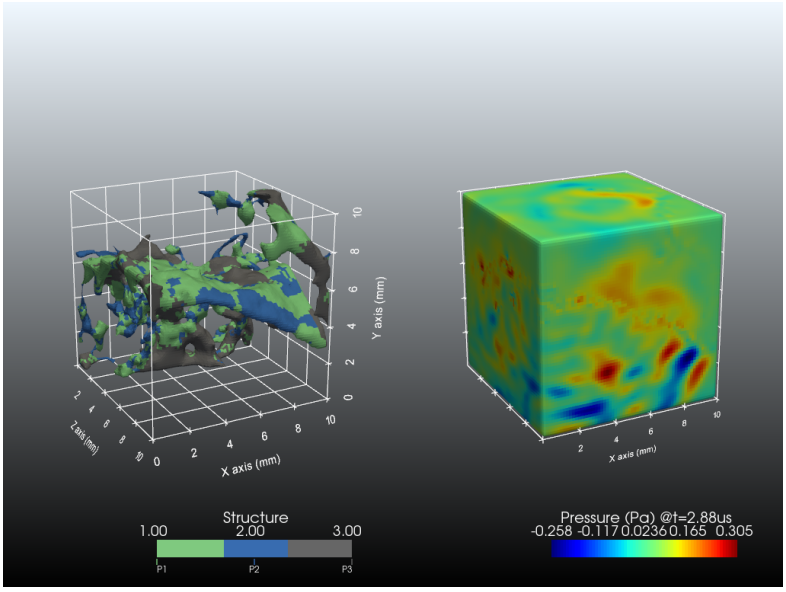


Figure 3.15: Case #2: a) Part of the pore network due to sub-cube #290 of sample #13; Pressure wave field at $t = 2.88\mu s$ exhibited by the sub-cube; b) Waveform of signals received at 8 corner probing points $B1 \dots B8$ of the sub-cube

a)



b)

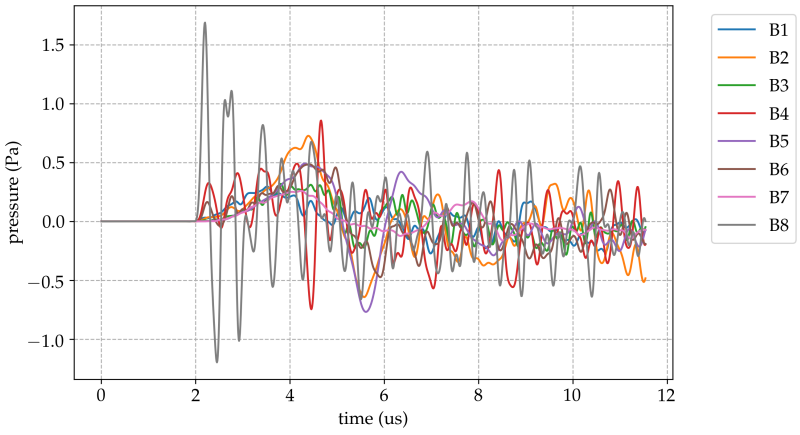
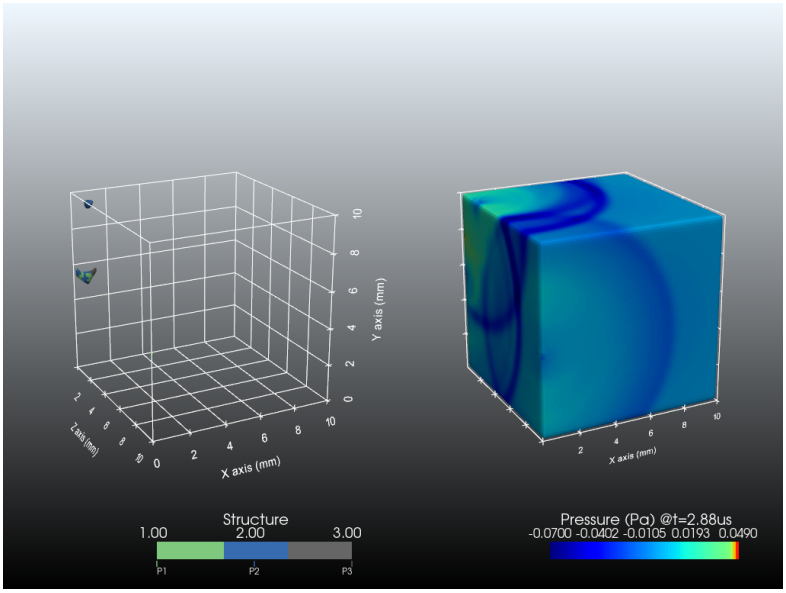


Figure 3.16: Case #3: a) Part of the pore network due to sub-cube #16 of sample #109; Pressure wave field at $t = 2.88\mu s$ exhibited by the sub-cube; b) Waveform of signals received at 8 corner probing points $B1 \dots B8$ of the sub-cube

a)



b)

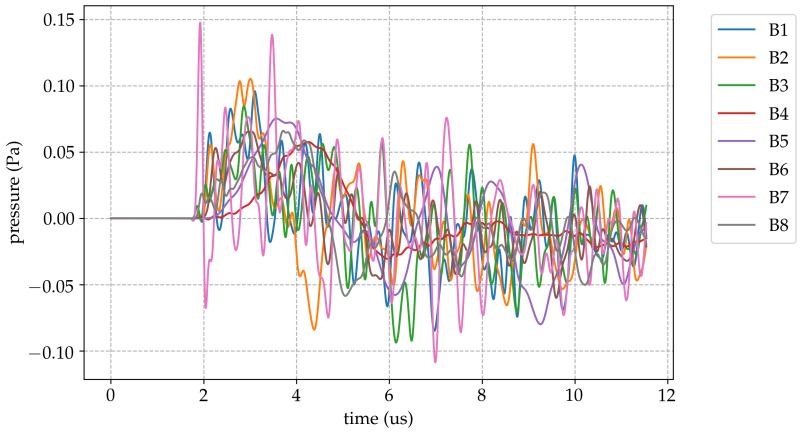


Figure 3.17: Case #4: a) Part of the pore network due to sub-cube #19 of sample #5; Pressure wave field at $t = 2.88\mu s$ exhibited due to the pore network in the sub-cube; b) Waveform of signals received at 8 corner probing points $B1 \dots B8$ of the sub-cube

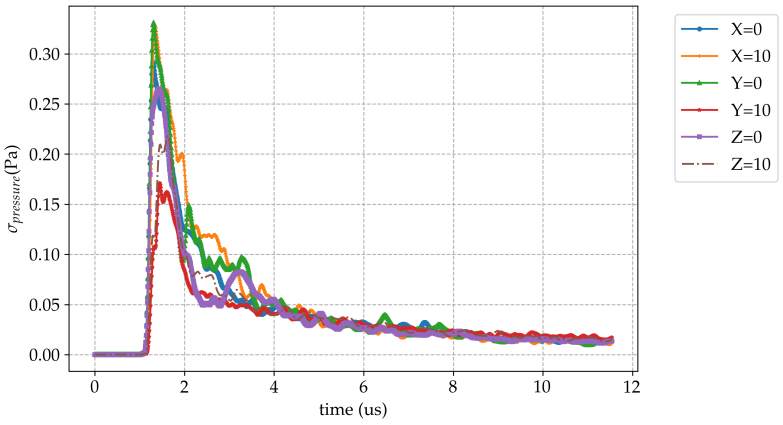


Figure 3.18: Standard deviation in the pressure amplitudes on the 6 surfaces of sample #105 and pore network instant #350

The two high peaks of $B7$ clearly indicate two constructive interference. As the patterns are quite clear on the surfaces of the cube, such clear interactions are valid and are evident in the waveform plots.

3.3.4 Statistical measure of Signal Amplitudes

The wave fields experience multiple reflections and refraction due to the complex pore networks in their travelling paths. As a result, we obtain complex waveforms at the surface of the cube as well. We use standard deviation as a measure of disorder on the surfaces. The values are obtained by computing the standard deviation of received amplitudes at each timestep on each of the surfaces. We show the deviation of the received amplitudes as a function of time in figures 3.18 though 3.21. It is quite evident from the plots that the pores help attenuate the wave amplitudes. The highest deviation is shown in Fig. 3.20. This corresponds to the third case study. The large dominant crack pore around the middle results in more deviation of the un-dissipated amplitudes of the received signals. The lowest deviation is reported by 3.19 with a peak deviation of 0.1. This corresponds to case study 2, where we have scattered and branched pore networks. The distributed nature of the pores helps attenuate the received amplitudes more in the other 3 cases. The peak deviation in signal amplitudes for case study 1 is reported to be 0.3, as shown in Fig. 3.18. And for case study 4, it was 0.14, as shown in Fig. 3.21.

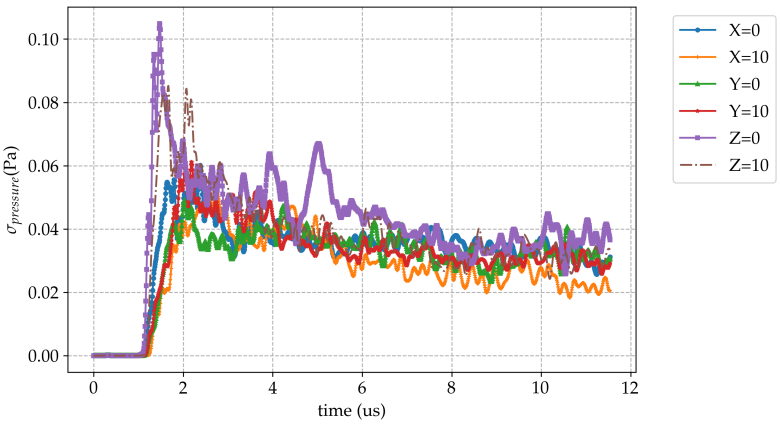


Figure 3.19: Standard deviation in the pressure amplitudes on the 6 surfaces of sample #13 and pore network instant #290

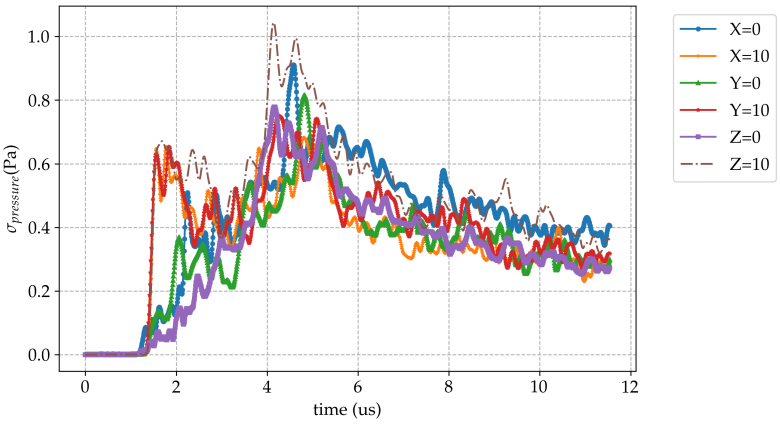


Figure 3.20: Standard deviation in the pressure amplitudes on the 6 surfaces of sample #109 and pore network instant #16

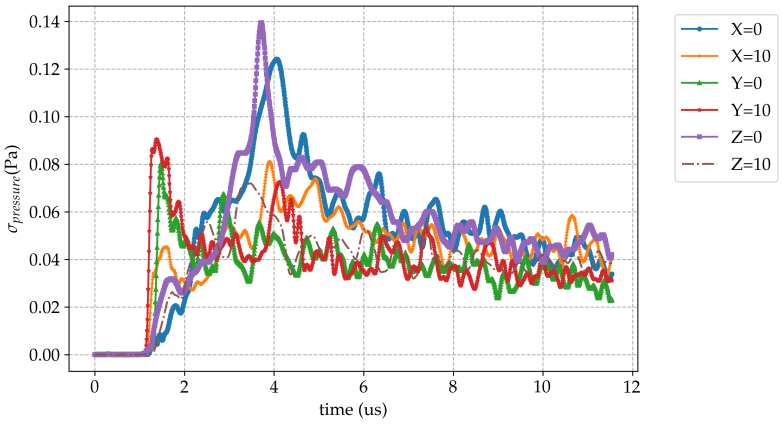


Figure 3.21: Standard deviation in the pressure amplitudes on the 6 surfaces of sample #5 and pore network instant #19

3.4 Segmentation and Localization of Pore Networks

In this section, we consider the task of predicting the pore structure models from the acoustic volumes. Segmentation consists of dividing the image into several disjoint parts according to some rules. The division may or may not pursue the idea of the content of the image. On the other hand, semantic segmentation aims to divide an image into smaller segments that are semantically meaningful, i.e. the individual parts carry some meaning with the label they have been assigned with [178]. It is to be noted that semantics is a specialised field of study in *linguistics* that concerns meaning in language studies [179]. There have been quite a number of groups working on semantic image segmentation problems. The works we cite here do not intend to be exhaustive as there is continuous development in the field. Also, for the sake of completeness, we describe some of the works that were developed for 2D images first. CNNs and RNNs are the fundamental building blocks underlying these semantic segmentation schemes for images. In the following, we describe the type of neural network used and the salient features of the architectures.

The task of scene labelling was taken up in [180], where Recurrent Neural Nets were realised using different instances of a convolutional neural networks. The application was not so successful as it resulted in heavy computational loads when there were multiple instances of training. In works by [181] the authors used fully connected networks with skip connections to join the multiscale activations from

the CNNs, in the final layer. DeepLab version 1 was devised by [81] that combined convolutional nets with dilations and a fully connected conditional random field. The fully connected nets in series led to costly computations and a reduction in performance. For the purpose of medical image segmentation, an encoder-decoder framework called the U-Net was suggested by [57] with skip connections connecting the same hierarchical levels in both parts. Due to the avoidance of any fully connected layers, the computations were economical. Similar to the U-Net, the Segnet was suggested by [80]. The skip connections here were only used to transmit the pooling indices. Another encoder-decoder framework was suggested by [182] whose encoder part was derived from VGG-D-16L. The computations were efficient due to no fully connected layers. Efficient pixel-wise labelling was achieved by [183] using a new CNN framework that could use dilated convolution to assemble contextual information from multiple scales. Authors in [184] integrated the conditional random field to the CNN to realise a deep learning system that would benefit from the qualities of both; but due to the presence of the recurrent block, the computations were limited. The integrative approach was also taken by [185] where they used layers of pyramidal inputs to be combined with feature maps to be finally received by the upsampling or concatenation blocks to form the final feature map into a dense layer; they too had used the conditional random fields with the CNN. A graph-structured solution was suggested by [186] for 2D still images but turned out to be inefficient due to the graph and LSTM processing layers. A directed acyclic graph-structured solution was suggested by [187] that was intended to process long range semantic dependencies between images. Due to the consecutive processing of data, the architecture was not efficient.

Chen's [81] Deeplab was improved in its version 2 [188] with similar performance. Zhao et al. [189] suggested PSPNet with a performance similar to Shelhamer's [181] but was relatively fast. Deeplab second version was improved in version 3 [190] by removal of the dense layers. Luo et al. [191] suggested a dual network approach where one network predicted the labels from the inputs from another network doing the semantic segmentation part. The performance of the network was similar to Chen's Deeplab version 2. A masked recurrent-convolutional layer architecture was provided by [192] with almost real-time segmentation performance. The Global Convolutional Network (GCN) was suggested by [193] where they suggested using large kernels to fuse together high and low-level features; they emphasised on using them for the simultaneous task of classification and localisation. A U-Net based solution was suggested by [194] that consisted of many deconvolutional layers guaranteeing fine segmentation results. Attention-based networks called the Discriminative Feature Network (DFN) were suggested by [195] where they em-

ployed two networks to handle the global contextual information and the border information. In works by [196], multiscale information was fused in a bidirectional fashion using LSTM units; once the LSTM would train, the features were combined hierarchically. A multiple LSTM based solution named Hierarchical Parsing Net (HPN) was suggested by [197] where a contextual feature encoder was followed by a convolutional layer with reduced efficiency. The authors in [198] designed a context encoding module where dense feature maps obtained from ResNet were used by the fully connected layers to extract contextual information, and a convolutional layer was used as the final prediction layer, limiting the computational efficiency. The PSANet of [199] used a spatial attention map on points attached to a pre-trained convolutional network; this served to relax the local neighbourhood constraint of the CNNs; the design was limited in efficiency due to the additional attention framework.

The field of 3D image segmentation can be classified into three groups –

a) Semantic segmentation - These image segmentation techniques aim to divide a given image into meaningful parts. There are different ways the data can be represented to perform semantic segmentation. The first one is the point data. In these classes, the points in the point clouds are either processed by graph convolution networks or point convolution networks or the simplest multilayer perceptron networks. Some of the works are listed here. The AGCN was developed by [200] that was realised using a point attention layer for capturing local features. The PGCRNet was developed by [201] that could model context dependencies among different categories. A new graph pooling convolution was incorporated in [202]. The DeepGCN was developed by [203] and could adapt to residual connections between layers of the network. The PointNet [204] was developed to consume unstructured point cloud data directly for the purpose of classification and segmentation, and parsing. The SPG [205] was developed to parse large scale scenes in a graph of super points. The DGCN [206] used edge convolutions for feature extraction. They also used a new update strategy for the graph.
The PointCNN [207] incorporated a novel point convolution layer in the development. The FlexConv [208] used the novel flexible point convolution and max-pooling without subsampling to work on large datasets. The KPConv [209] also called the kernel point convolution, could work on direct representation of point clouds. The PCNN [207] used the KDTree to process the point clouds in a non-structured manner. The PWCNN [210] introduced a new point-wise convolution operator. The DPC [211] introduced a dilated

point convolution operator with a special emphasis on receptive field size calculation. The RSNet [212] introduced a method to model local dependencies in point clouds using a slice pooling, RNN and slice unpooling layer. Inherent contextual features were exploited in 3P-RNN [213] making use of pyramid pooling and a pair of RNNs to explore long-range dependencies. The authors in [214] built on the Point Net [204] to incorporate the large scale spatial context. Segmentation in unstructured point clouds was addressed in [215] by grouping within point clouds and introducing novel loss functions.

Another subclass within the semantic segmentation category is due to depth maps extracted from RGB images. The authors in [216] augmented the RGB images with estimated depth information for increased accuracy in segmentation. Similar approaches to augmentation in the pursuit of segmentation accuracy have been reported. Fully connected nets were employed by [178; 216; 217; 218; 219; 220]. Variants of CNNs were used in [221; 222; 223; 224; 225; 226; 227; 228; 229]. Encoder-decoder frameworks were employed in [84; 230]. Graph networks were employed by [231]. LSTMs/RNNs were employed by [232; 233].

There also exist voxel-based semantic segmentation approaches wherein 3D convolutional networks are used for pattern recognition. [234] used the 3D CNN to avoid any hand-crafted features to label the points in a 3D cloud. It was well adapted for large datasets. Authors in [235] maintained the global consistency by combining the fully FCNNs, trilinear interpolation and conditional random fields. A different approach was adopted by [236] where they used a variational autoencoder to encode the local geometry within each voxel. Further, in order to handle sparse data, they used the radial basis function to compute continuous local representations in each voxel. The authors in [237] combined the convolutional nets with Deep Q-network and RNN for efficient parsing and localisation of data in point clouds. Another approach was suggested by [238] to process unorganised point cloud data. The unorganised data was converted to organised representations to make the processing more memory efficient, which the authors claimed that it made learning better features on benchmark datasets. In a work by [239], the authors introduced a 3D data completion network realised using generative CNN. The network has been claimed to handle large datasets while outputting semantic labels to much greater accuracy compared to other techniques. The goal in [240] was to exploit the sparsity in input to represent the data using a set of unbalanced octrees where each leaf node represented

a pooled feature. The performance of the network was demonstrated using the resolution on 3D tasks like object orientation/classification and labelling. In [241], spatially sparse convolutional networks were developed for semantic segmentation on point clouds.

b) Instance segmentation - The 3D instance segmentation techniques further distinguish between different instances within the same class. Instance segmentation was realised using a Generative Shape Proposal Network on point cloud data by making the network emphasise geometric understandings in [242]. Authors in [243] performed instance segmentation on RGB depth scans, which leveraged the geometric as well as colour information of the scans to do the segmentation. The network designed in [244] consisted of a backbone network followed by two parallel network branches to perform bounding box regression and point mask prediction. The design has been claimed to perform well on ScanNet and S2DIS datasets while still being computationally efficient. A network named SGPN was proposed in [245] in which the technique involved a similarity matrix that would be indicative of the similarity between each pair of points in the embedded feature space. The design was tested on various 3D scenes of point clouds. The 3D-MPA was designed in [214] for point clouds where a graph convolutional network was used to learn high-level features by allowing every point in the cloud to vote for its object centre. The proposal vector is comprised of a semantic label, a set of associated points, an object score and aggregation features. Their work avoided the non-maximum suppression and grouped all proposals based on the learned aggregation features. The network was tested on ScanNetV2 and S3DIS datasets. The authors in [246] introduced a detection-free and grouping-free technique, for instance, segmentation. They constructed an instance grouping loss for the network training purpose.
Among the proposal-free methods, we list a few of them here. The authors in [247] introduced a proposal free method based on sparse convolution and point affinity prediction, which indicated the likelihood of two points belonging to the same instance. They included a clustering algorithm as well to cluster the points in a voxel into instances based on the predicted affinity and topology of the mesh. The semantics was taken care of by the semantic prediction part of the network. In [248], dense 3D voxel grid data were targeted. Their focus was on shape recognition of individual object instances. This was accomplished by learning an adequate feature embedding stage followed by estimation of directional information of the instance's centre of

mass in each voxel. The author claimed to be able to demarcate clear boundaries. A Non-maximum suppression based strategy was suggested in [249] where the design focussed on the void space between objects in the scene. A two-branched network was designed to extract point features and predict semantic labels. The network was tested on the ScanNet v2 dataset and the S3DIS dataset. The authors in [250] have proposed to combine the local point geometry with global context information and a proposal free method to group the feature spaces into semantic instances. Another approach to point cloud semantic segmentation was made in [206] wherein the design enabled a mutual enhancement of the semantic and instance segmentation tasks. The authors in [251] developed a multi-task pointwise network that could perform the task of semantic and instance segmentation simultaneously. A graph-based solution was formulated in [252] was used along with 3D convolutional neural networks to obtain discriminative embeddings for each 3D instance. Also, an attention-based k-nearest neighbour was proposed to assign different weights for the neighbours. In OccuSeg [253], the scheme could produce an occupancy signal and an embedding representation simultaneously. They could also prevent over-segmentation using a clustering algorithm.

c) Part segmentation - The 3D part segmentation is still more difficult than either the semantic segmentation or the instance segmentation counterparts. Although there is an increased level of fineness associated with part segmentation, some semantic segmentation deep networks can still be used for part segmentation tasks. Here, we list some of the developments in part segmentation tasks. In [254] the authors have introduced an image based fully convolutional network and conditional random fields to obtain faithful segmentation of 3D shapes. VoxSegNet, introduced in [255] consists of an attention feature aggregation module to adaptively select features from different abstraction scales. Their method operated in voxel data format. The authors in [256] introduced a point grid with a 3D convolutional network that incorporated a fixed number of points within each grid cell, enabling the network to learn high order local estimations to better represent the local geometry. In [257], the authors had adopted a teacher-student training procedure wherein the VolNet (working in 3D data) was used to teach the GeoNet to extract 3D features from a 2D figure. Their data was synthesised in image-volume pairs. The authors in [258] used a conditional random field network to fuse the shape representations of two other networks that

worked with geometrical face normal data and face normal histogram data. MeshCNN was described in [259] where the authors have illustrated how a specially designed convolutional neural network can directly apply pooling and convolution operations on mesh edges. In [260] the authors had introduced a recursive neural network that could perform hierarchical segmentation of 3D shapes represented by point clouds. The work in [261] improvises on the PointNet architecture to accommodate the local details of the local neighbourhood. They introduced two new operations - one for learning local 3D geometric structures and the other for local high-dimensional feature structures using recursive feature aggregation on a nearest-neighbour graph of 3D points. The SFCN network in [262] utilised the voting based multilabel graph cut method to optimise the segmentation results. The authors of [263] introduced the special spider convolutional operation by parameterising a family of convolutional filters using simple products of a step function and Taylor polynomials. New graph convolution operations were introduced in [264] that were dynamically calculated from the learned features on point cloud data. A novel Kd-trees based network was proposed in [265] that could perform multiplicative transformations with shared parameters according to the subdivisions obtained from point clouds. Shape segmentation was performed using o-cnn [266], an octree based convolutional neural network that utilised the average normal vectors of a 3D model to perform convolution operations on the octants occupied by a 3D shape. Their implementation was claimed to be feasible for high-resolution 3D models. A unique 3D point-capsule network was introduced in [267] that could process sparse 3D point clouds while still maintaining the spatial arrangements of the input data. Their approach was a unique combination of the dynamic routing within the network and a peculiar 2D latent space that could perform object classification and segmentation tasks nicely. The work in [268] made use of self-organising maps to represent orderless point clouds as single feature vectors. The advantages reported in their work list fast training times.

3.4.1 Semantic Segmentation for inference of Pore Networks

The problem of predicting the class of every voxel is what is sought from the deep network. This forms a crucial step in the characterisation of the pore network using acoustic analysis. In the current context, a good segmentation is said to occur when a pore is labelled as a pore and the matrix as the matrix. The primary objective

of semantic segmentation is the independent segregation of objects of interest. This is especially important because segmentation is fundamentally important to the development of rock physics models. The literature cites immense content on image segmentation but does not recommend a single best method [269].
Seeing the complexity of the shapes of the pores which carbonates exhibit, we choose the basic U-Net architecture [57] and modify it for our own purposes. One of the more compelling reasons for its selection is the type of input data. We seek to extract the pore networks from the acoustic volumes. These acoustic/impedance volumes are often the end products of a conventional seismic processing stage [270] and the wave signatures of the objects the wave has interacted with. Interpreters use the method of inversion to delineate the sweet spots from these volumes [270]. But this process is often time-consuming with no guarantee of the optimal solution. At this juncture, as outlined in the next sections, we propose to directly infer the pore networks from acoustic volumes. Of course, one may argue about the frequency of the seismic data acquisition, that the used frequency is too small to resolve the heterogeneity at the core scale. But in this chapter, we are more interested in taking the first steps toward direct interpretation and would like to incorporate the intricate modelling requirements as a future project.

3.4.2 The Modified U-Net Architecture

In this thesis, we are working with computational semantics in the context of 3D volumetric images of rock sample data. We make use of the basic U-Net architecture to modify it for segmentation purposes. Out input images are acoustic wavefields from which we would like to extract the pore networks that caused the patterns in the wavefields.

The U-Net architecture was primarily developed for medical image segmentation tasks [57]. The architecture has proven to be highly beneficial for a low amount of labelled data, such as that of medical images [271]. The second benefit offered by the architecture is that it is reasonably fast to train [271]. The explanation of the basic U-Net is given in section 2.1.3.1. The 3D equivalent, as relevant in the current study, replaces all operations with their 3D equivalents.
We now describe the model used for semantic segmentation of acoustic images. The original U-Net consisted of 4 blocks of convolutional layers prior to the bottleneck. And correspondingly, there were four blocks of upsampling layers in the decoder part of the network. On the contrary, we had to reduce each part of the U-Net to 3 blocks only. Fig. 3.22 shows the modified architecture for acoustic image segmentation. The convolution block uses a filter size of $3 \times 3 \times 3$ and and

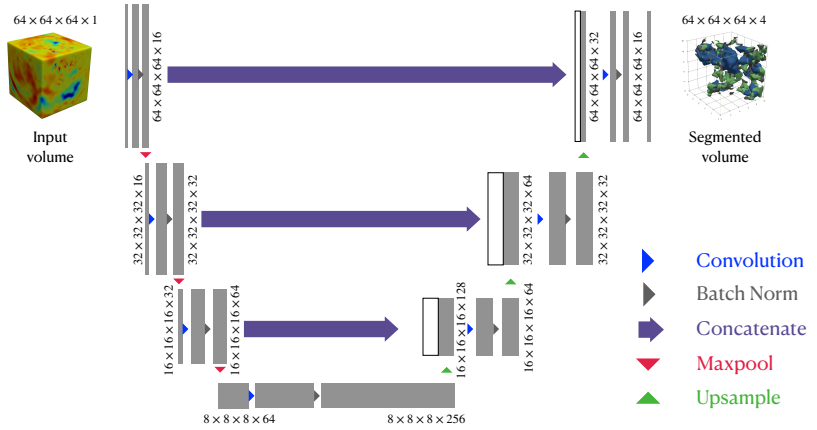


Figure 3.22: The modified U-Net architecture for semantic segmentation of the acoustic volume

stride of $2 \times 2 \times 2$. The number of filters used for the first, second and third layers is $16, 32, 64$ in the encoder part of the network. Similarly, in the decoder part, the number of filters decreased from $64, 32, 16$. The latent/bottleneck consisted of 256 filters. Every convolution operation was followed by a ReLU layer, batch-normalisation layer, a max-pooling layer and a dropout layer. The dropout value used was 0.1 in each of the blocks. The output of the network was passed on to a softmax function to output the class probabilities of each class in the 3D sample image.

3.4.3 Training and Validation

In order to measure the prediction quality, four suitable metrics were used. They are Accuracy, Cross-entropy, Dice-coefficient and Soft Dice-coefficient. The Accuracy and Cross-entropy metrics are label based metrics, i.e. individual labels are compared in the predicted and observed labels. So, if $y, \hat{y}$ are the observed and predicted labels, the metric is calculated as:

$$Accuracy(y, \hat{y}) = \frac{1}{N} \sum_{i=0}^{N-1} 1(\hat{y}_i = y_i) \tag{3.20}$$

$$Cross\_entropy(y, \hat{y}) = -\frac{1}{N} \sum_{i=0}^{N-1} y_i * \log \hat{y}_i \tag{3.21}$$

The Dice-coefficient [272; 273; 274] and Soft-Dice coefficient are special mea-

Table 3.6: Training Metrics

Epoch	Accuracy	Cross-entropy	Dice Coeff.	Soft-Dice Coeff.
1	blue!250.89	red!250.39	blue!250.71	blue!250.63
2	0.94	0.25	0.84	0.76
3	0.94	0.24	0.86	0.78
4	0.95	0.21	0.88	0.79
5	0.95	0.21	0.88	0.80
6	0.96	0.19	0.90	0.81
7	0.96	0.20	0.89	0.81
8	0.96	0.19	0.90	0.83
9	0.96	0.14	0.92	0.84
10	0.96	0.15	0.92	0.84
11	0.97	0.13	0.92	0.85
12	0.97	0.14	0.92	0.85
13	0.97	0.13	0.93	0.85
14	0.97	0.16	0.92	0.85
15	0.97	0.14	0.93	0.87
16	0.97	0.16	0.92	0.86
17	0.97	blue!250.12	0.93	0.87
18	red!250.97	0.13	red!250.94	red!250.87
19	0.97	0.14	0.93	0.87
20	0.97	0.17	0.93	0.87

sures of similarity defined over sets. If T, P are the true binary mask and predicted binary mask, respectively, then the metrics are given as:

$$Dice\_coefficient(T, P) = \frac{2 \times |T| \cap |P|}{|T| + |P|} \tag{3.22}$$

The Soft-Dice coefficient is the same as the original Dice coefficient, except that in this, the predicted masks are not rounded prior to computation. This is unlike the Dice coefficient, where the predicted mask is first rounded and then compared with the observed true mask.
The loss function used to train the model can be given as:

$$loss_{Dice} = 1 - \frac{2\sum_i^N p_i t_i}{\sum_i^N p_i^2 + \sum_i^N t_i^2} \tag{3.23}$$

where p_i, t_i are the predicted probability and the value of the ground truth, respectively, for each voxel. This loss function has been useful in semantic segmentation tasks with imbalanced occurrences of target classes [275].

Table 3.7: Validation Metrics

Epoch	Accuracy	Cross-entropy	Dice Coeff.	Soft-Dice Coeff.
1	0.88	0.53	0.66	0.57
2	0.93	0.36	0.82	0.74
3	0.95	0.20	0.87	0.78
4	0.89	0.37	0.82	0.73
5	0.96	0.18	0.89	0.81
6	0.80	0.51	0.71	0.66
7	0.95	0.20	0.89	0.83
8	red!250.96	blue!250.12	red!250.92	0.82
9	0.44	4.00	0.39	0.39
10	0.57	1.49	0.50	0.51
11	0.48	4.13	0.43	0.43
12	0.38	red!257.51	0.34	0.34
13	blue!250.37	7.25	blue!250.33	blue!250.33
14	0.96	0.20	0.91	red!250.85
15	0.58	2.12	0.51	0.51
16	0.52	2.43	0.46	0.46
17	0.41	6.48	0.37	0.37
18	0.41	5.59	0.37	0.37
19	0.61	1.42	0.53	0.53
20	0.73	0.79	0.66	0.63

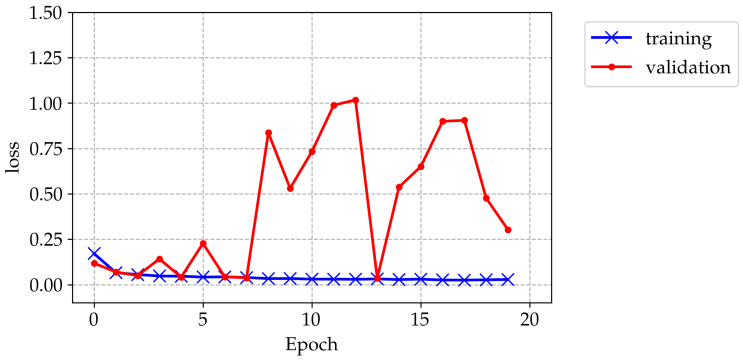


Figure 3.23: Epoch vs Training and Validation loss

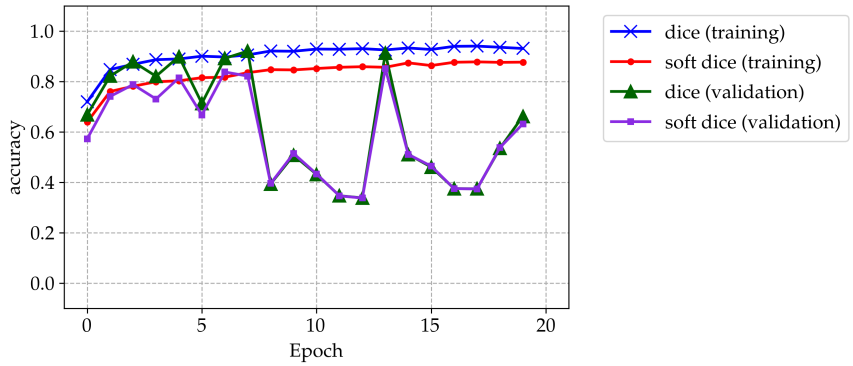


Figure 3.24: Epoch vs Dice and Soft-Dice accuracy

3.5 Results and Discussion

The main objective of this chapter was to map the pore networks present in carbonate rock core samples to their acoustic responses. We discuss the results of all the objectives in this section.

Volumetric image processing of rock core samples

As a starting point, the chapter has shown how to process the 3D volumetric images of rock core samples. After the acquisition of digital images from a 4D-XRM instrument, the images require a few steps of processing before they become useful for pore network extraction.

Referring to Fig. 3.2 we can see that the 3D images show quite an amount of variation in their grayscale values. These variations manifest as contrast changes when analysed visually. Specifically, the samples in figures 3.2b), 3.2h) and 3.2i) appear to have low grayscale values in the 0-255 range, imparting to their dark appearances. Figure 3.2a) shows a relatively low variation of intensities and appears greyish in colour from top to bottom. Sample 3.2d) and 3.2j) are louder with their representation. We find that the scanning of sample 3.2d) resulted in pixel values whose volume rendering just left scatters of pixel values in the cylindrical volume. On the other hand, in 3.2j), one can easily identify the presence of pores along the length of the sample. This is an example of a good scan.

Due to the variation of intensities in different samples, a simple contrast equalisation was applied to make the grayscale values stretch over the entire range. As one can see in Fig. 3.3a) the cube has been extracted from 3.2b) and whose grayscale lies in the range 0.374 to 0.937. After the contrast stretching, the pixel

a)

b)

c)

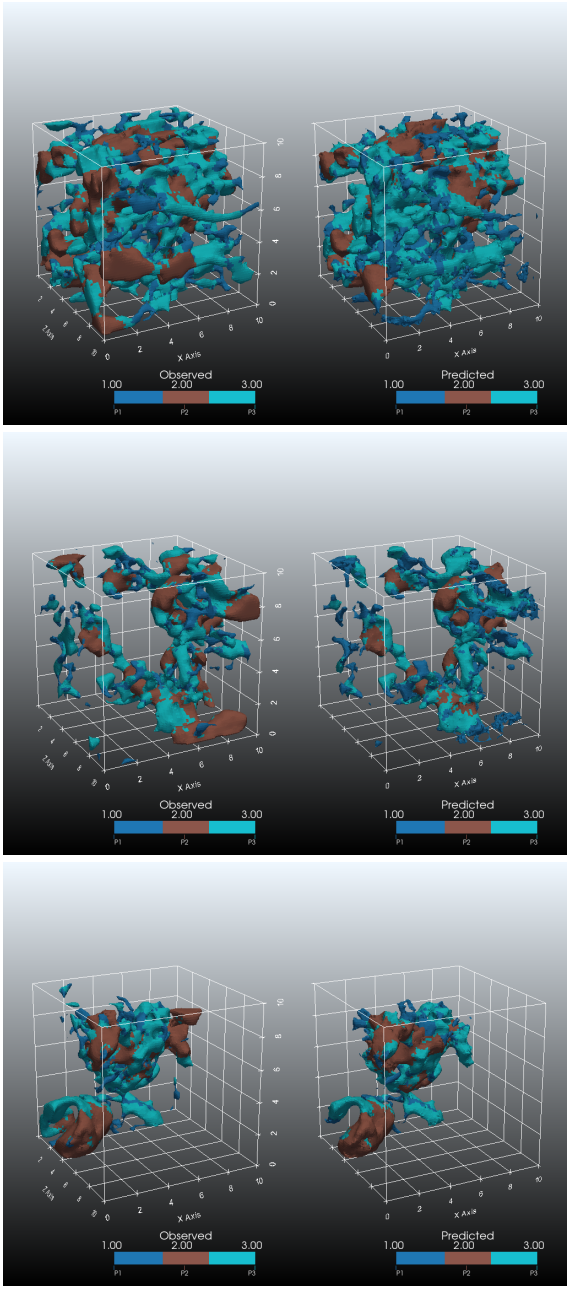
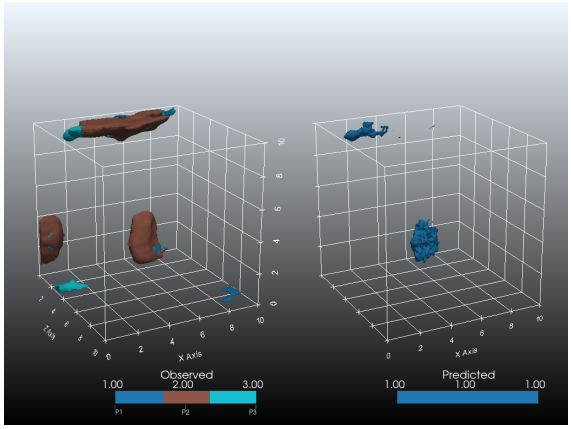
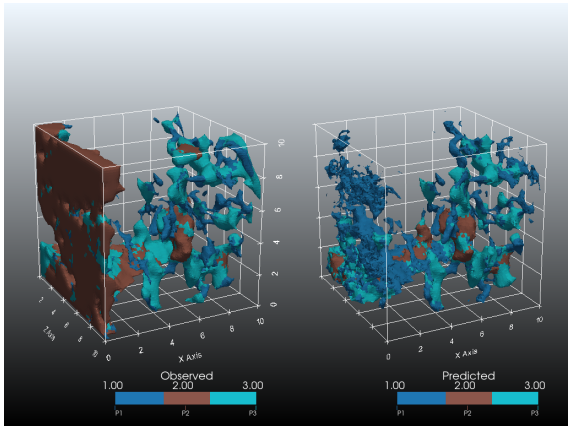


Figure 3.25: From a) to c): Examples of some optimally predicted pore networks

a)



b)



c)

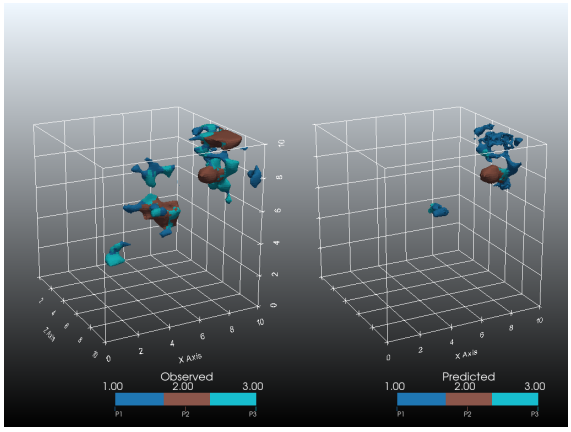


Figure 3.26: From a) to c): Examples of some non-optimal predicted pore networks

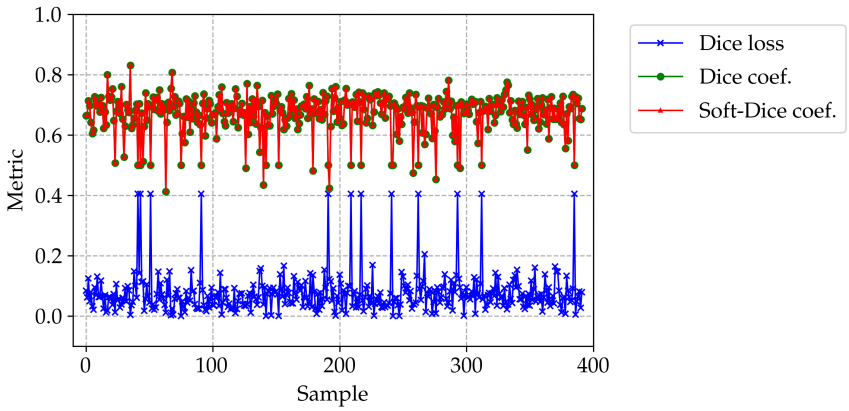


Figure 3.27: Prediction metrics on 392 test samples that were not used for training. Dice loss, Dice coefficient and Soft-Dice coefficients are calculated for 4 classes - Matrix, P1, P2, P3.

values have an extended scale that lies in the range 0.0276 and 0.974. Consequently, the porous regions are more visible now, easing the next stages of processing. We also show the distribution of the pixel values in Fig. 3.4a) and 3.4b). Figure 3.4b) clearly shows a stretched scale of the values.

Often, the images are also plagued with some noise, as depicted in Eq. 3.1 through 3.3. These noises can be suppressed through the use of median filters. An application of a $3 \times 3 \times 3$ structuring element was used for the filters. The result can be seen in Fig. 3.7. The image has been shown in "jet" colourmap for clarity; otherwise, it is essentially greyscaled. The median filtered image is subjected to the Sauvola thresholding algorithm to obtain a binary image as shown in Fig. 3.8. The Sauvola binarisation algorithm requires two parameters to perform the binarisation. Table 3.3 shows the various window sizes and weights used for different samples. In the current context, the effect of binarisation is the extraction of the pore network. The porosity computed in the binarisation step must be closely matched with the porosity obtained from the laboratory. For this purpose, the computed porosity was calibrated against the lab derived porosity, and the result is shown in Table 3.2.

Another goal of volumetric image processing was to extract the pore networks from rock samples in a semi-automated manner. For this, the binary image was operated by a distance filter given by Eq. 3.6 to obtain the distance image. This map essentially computed the distance of every foreground pixel to the nearest

background pixel. We used the Euclidean distance function for this purpose. The maximum distance was computed to be 6.56 units. The binary image was also used for computing the skeleton of the pore network, as shown in Fig. 3.10. As we can see, the skeleton is too complex to be interpreted manually. Therefore, we seek to implement a semi-automated algorithm for obtaining the segmentation.

The watershed algorithm is an established method for image segmentation [276]. It is a flooding algorithm that propagates the labels of the "seeds" to every pixel in the foreground. These seeds are something that the user provides manually to the algorithm. These can also be generated automatically using a peak detector algorithm. In the current work, automation is achieved using the k-means clustering algorithm. We discuss the steps here. The skeleton image 3.10 holds the coordinates of the skeleton of the pore network. These coordinates are used to extract the distance values from the distance map of the sample. Because the distance map holds the distance of every foreground pixel to the nearest background pixel, it, in a way, encodes the cross-sectional width of the pore path throughout the network. This single strategy can be used to attain a semi-automatic segmentation. The distance values of the skeleton image are subjected to k-means clustering with a predefined number of clusters as 3. This number corresponds to 3 types of pores based on relative cross-sectional widths in the pore network. The output of the clustering algorithm is a skeleton map where every pixel now has either of the three labels. We use these labels as seeds to propagate their values to all the pixels in the foreground. The strategy is well illustrated in Fig. 3.6. We can see in Fig. 3.6a) that a binary image has been synthetically designed, which consists of simple geometrical objects. The cross-sectional widths are used by the clustering algorithm to assign labels in Fig. 3.6c), which are then flooded throughout the foreground. The resulting figure, as shown in Fig. 3.6d) gives a segmentation based on their relative cross-sectional widths.
The above-discussed procedure is followed for the 3D binary image for each cubic sample. The result of the segmentation is shown in Fig. 3.11. The three types of pores are assigned three unique labels as shown in different colours in the image.

Acoustic Wave Analysis

After the extraction of the pore networks, the cube samples are subjected to acoustic wave simulation. To keep things simple, we consider only two types of bodies inside the cube – the matrix and the pores. The pores are considered to be completely occupied with a single fluid. The materials are specified using two properties – the medium's velocity and the density. The values used in the simulation are

given in Table 3.4. The matrix has been assigned a high velocity of $5200m/s$ and density $2200kg/m^3$. The fluid medium has been assigned a velocity of $1500m/s$ and a density of $1000kg/m^3$. Every simulation on each of the cubical samples was run for 1281 timesteps, which corresponds to $11.38\mu s$. A demonstration of the propagation of the wave has been shown in Fig. 3.12. The cubes are $64 \times 64 \times 64$ voxel sized. First row in 3.12a) shows the structure of the pores present on the three planes – XY, YZ, and ZX. The row numbers 3.12b) through 3.12e) show the propagated waves at times, $t = 0.53\mu s, 0.90\mu s, 1.26\mu s, 1.63\mu s$. In the figure, we can see that the wavefront undergoes reflection when it encounters a low-velocity region of the pore. The beginning of the reflection occurs in Fig. 3.12c) and continues in Fig. 3.12d) and e).

In order to get a quantitative estimate of the wave amplitudes, we set up 4 cases of different pore networks and probed the 8 corners of the simulation domains. The graphs for the same are shown in Fig. 3.14, 3.15, 3.16, 3.17. The 4 cases are described in Table 3.5. In case 1, the pore bodies are near the top surface. The highest amplitude recorded is by probe $B2$, which equals 0.6 Pa. The waves get attenuated after time, $t = 4\mu s$. In case 2, the pore network is branched and more scattered. The wave amplitudes recorded by the probes are much smaller of the order of 1e-2 Pa. The reason for this might be the scattered pore paths cause the waves to reflect and undergo destructive interference causing a decrease in amplitudes. In case 3, there is a dominating crack pore in the middle, parallel to the ZX plane. The amplitudes recorded in this case are higher than in cases 1 and 2. The highest amplitude was recorded at around 1.5 Pa by the $B8$ probe. After time $t = 6\mu s$, the amplitudes decrease in amount. Case 4 stands out from the previous 3 cases in the sense that the cube has just 2 small pore bodies located parallel to the XY plane towards the back. The corresponding wave pattern shows clear wavefronts travelling on the surface of the cube. On analysing the wave amplitudes of the probes, we find that most of the power is lost and must have dissipated into the absorbing layers. The highest amplitude recorded is about 0.15 Pa by probe $B7$.

In addition to the 8 probes, we also compute an extra metric of the wave amplitudes for quantitative analysis. In order to quantify the patterns, we take the standard deviation of the amplitudes at the 6 surfaces of the cube for the length of the simulation. Overall, there exists a lot of dissimilarity in the observed patterns for the 4 different cases. The highest variation is exhibited by sample 109 in Fig. 3.20 which is approximately equal to 1.0 Pa. The second highest variation is

shown by sample 105 in Fig. 3.18 and is approximately equal to 0.3 Pa. Samples 5 and 13 stand third and fourth with the standard deviation values, approximately equal to 0.14 Pa and 0.10 Pa. This analysis shows that the wave fields received at different parts of the computational domain carry information about the pore network structure inside. We seek to retrieve these causative pore networks from their corresponding wavefields in the next part of the discussion.

Mapping of Pore-networks to Acoustic Responses

We modify the basic U-Net architecture into a simpler form and reduce the tensor sizes for our purpose. We work with smaller cubes of size $64 \times 64 \times 64$. This also helped reduce the depth of the network to 7, inclusive of the bottleneck. Figure 3.22 shows the shapes of the tensors at the input of each layer. Here, the input data used are the acoustic volume generated by a point source from the centre of the cube of size $10mm \times 10mm \times 10mm$. The waves, after emanating from the centre, encounter low acoustic velocity zones in the form of pores. Due to this, the waves undergo reflection and refraction to ultimately exhibit specific wavefield patterns, as discussed in section 3.3.3. The training of the U-Net is computationally expensive, and it takes around 2 hours to train on 3280 samples. We validate the training on 392 samples. We use 4 accuracy metrics for monitoring the training of the deep network. The loss function used for training is given in Eq. 3.23. This loss function takes care of the imbalance in classes.

The U-Net was trained for 20 epochs, whose results are shown in Table 3.6. The results have been rounded off 2 decimal places, although the coloured cells mark the high(red) and low(blue) values of the metrics. As one can see, the Accuracy metric starts off with a value of 0.89 and at the end of 20^{th} epoch attains 0.97. Correspondingly the Cross-entropy metric starts off with a value of 0.39 and ends at 0.17. On the other hand, the Dice coefficient and the Soft-Dice coefficient start at 0.71, 0.63 and end at 0.93, 0.87. This suggests that the former two metrics are not optimal indicators of the training in the context of semantic segmentation. Had we used these for monitoring the network training, the results would not have been good enough as we obtained with the latter two. The Dice and Soft-Dice coefficients are better metrics for the purpose. Choice of the Dice loss is also optimal as it readily takes care of the imbalance in classes. We can observe similar characteristics in the validation metrics as well. The validation metrics are shown in Table 3.7. Out of the 20 epochs, the network achieved the best performance on epochs 8 and 14, as marked in the table. We can also see the loss graphs in Fig. 3.23. The loss values attain the best values corroborate with the Dice and

Soft-Dice coefficient values. We can also see the graphs of the accuracy metrics in Fig. 3.24 for the training and validation stages.

Now we discuss the predictive performance of the trained network. In Fig. 3.25a) through 3.25c) we can see the model could predict the overall pore network nicely, but the individual classes are somewhat non-optimal. Nonetheless, however, we assume, from the interpreter's perspective, that the overall structure is more important than the individual pore types. In Fig. 3.26a) through 3.26c) we can see some of the non-optimal predictions. In the first part 3.26a) the leftmost observed brown pore is completely missed in the predictions. The model is faintly able to capture the other two pores on the top and right. In the second-row 3.26b) the large brown pore on the left is completely missed in the predictions, while the other parts of the network appear to be nicely predicted. In the third-row 3.26c) the middle, top, and bottom pores are missing from the predictions, and only the right top pore has been faintly predicted. Finally, we plot the Dice metrics viz. loss, Dice coefficient and the Soft-Dice coefficient for all the validation samples in Fig. 3.27 to have an idea about the performance of the model on unseen data. We can conclude that the model has attained 70% accuracy on the validation set from the graphs.

Chapter 4

Application-II: LSTMs based Accelerated Simulation for Hydro-Mechanical Systems

4.1 Introduction

Hydro-mechanical simulations are widely used to study the slope-stability problem. These multiphysics simulations capture the mechanical and hydraulic response of a system. Conventional slope-stability analysis techniques use the *limit equilibrium method* criterion as the main principle to determine the factor of safety. To name a few, we have, the ordinary method of slices [277], the Bishop's modified method [278], the force equilibrium methods [279], the Janbu's generalised method of slices [280], the Morgenstern and Price method [281], the Spencer's method [282]. Chart based slope-stability schemes have also been developed by [281; 283; 284] and [285]. In general the slope-stability analysis can be based on either *total stress* or the *effective stress* condition. The total stress-based method ignores the effect of *pore pressures* and the undrained shear strength determines the soil strength. For the effective stress based method, the failure can be modelled using one of the many soil failure criteria such as the Mohr-Coulomb failure criterion [286], the Hoek-Brown failure criterion [287], the Drucker-Prager failure criterion [288], the Matsuoka-Nakai failure criteria and the Lade-Duncan failure criterion [289; 290]. The most commonly used of all these failure criteria is the Mohr-Coulomb failure criteria. The criteria are given using the following equations:

$$I_1 = \sigma_1 + \sigma_2 + \sigma_3 \tag{4.1}$$

[1]Parts of this chapter are published here: 10.1007/s00603-021-02668-9

$$I_3 = \sigma_1 * \sigma_2 * \sigma_3 \tag{4.2}$$

$$J_2 = \frac{1}{6}\left[(\sigma_1 - \sigma_2)^2 + (\sigma_2 - \sigma_3)^2 + (\sigma_3 - \sigma_1)^2)\right] \tag{4.3}$$

In the above equations if σ_1, σ_2 and σ_3 denote the maximum, intermediate and minimum principal stresses, I_1 and I_3 the first and third stress invariants and J_2 the second deviatoric stress invariant, then the Mohr-Coulomb failure criteria can be stated as follows:

$$\frac{\sigma_1 - \sigma_3}{\sigma_1 + \sigma_3 + 2 \cdot c \cdot cot(\phi)} = sin\phi, \tag{4.4}$$

where ϕ and c denote the cohesion and internal angle of friction, respectively.
The more versatile and contemporary approach to slope-stability is the finite element model (FEM) based approach. Some of the works that have used FEM for slope-stability analyses can be attributed to, [291; 292; 293; 294; 295; 296; 297; 298; 299; 300; 301; 302]. The underlying reasons for its versatility can be restated from [293], as:

1. The method relaxes the shape and orientation of the slip surface. As a simulation progresses, the failure naturally occurs in regions with low shear strength than the applied shear stresses.

2. The method avoids making assumptions about the internal forces, as is the case with limit equilibrium based methods. The governing equations enforce the equilibrium until failure.

3. The method is informative about the pre-failure states of the slopes, given that the actual soil and material parameters are used.

4. Stages of failure progression can be observed up-till overall shear failure occurs.

With the above stated advantages, the current work extends the FEM based slope-stability analysis system using deep learning algorithms to overcome some of its limitations and enhance the interpretability of the results.

4.2 Development of a Reduced Order Model

Geophysical systems are complex entities that involve coupled multiphysics processes. Numerical simulations describing these systems are often extremely computationally intensive. Nevertheless, to understand the sensitivities of these systems,

map the potential impact of variable boundary conditions, or engineer appropriate geotechnical solutions, it may be necessary to run such models hundreds or thousands of times. In many cases, the only way to achieve this is through the development of Reduced Order Models (ROMs) – numerical methods designed to capture the fundamental response of a system under a limited range of boundary conditions for significantly less computational effort than would otherwise be required.

Most of the data-driven model reduction has been aimed at applications in fluid mechanics. Two popular model reduction strategies employed in this area are Dynamic Mode Decomposition and Proper Orthogonal Decomposition. Dynamic mode decomposition (DMD), first developed by [303] performs both temporal and spatial model reduction. As described in [304], DMD works by identifying transient coherent structures and their associated rates of growth and decay. The DMD has been reported useful for data-driven analysis of fluid systems [305; 306; 307]. The DMD method has also been used in system identification [305]; estimation of flow fields [308], and studying of neural pattern recordings [309]. An alternative reduction strategy is Proper Orthogonal Decomposition (POD), first described in [310]. POD is a singular-value decomposition based technique that seeks to find a set of optimal basis functions using snapshots of the state space from the simulation data. These bases are then used to build a ROM. The POD method has been used as a reduction technique across many different applications [311; 312; 313; 314; 315; 316]. However, unlike DMD, POD methods require a means to advance the state space once a suitable set of bases have been found. Some recently developed ROMs combine deep learning methods with data reduction to develop a model that can advance the state space through time. [317] employed a residual recurrent neural network to study the subsurface multiphase flow problems, while [318] used deep recurrent networks based on long short-term memory units for turbulent flow control problems. Both of these works employed POD as the dimensionality reduction technique to reduce the basis of the underlying high fidelity physics simulations. [319] used the full-scale physics simulation data from transient flows. Their work developed two architectures – sequential and residual artificial neural networks and compared them to the Galerkin projection method. The standard Burgers equation was solved to provide the full-scale simulations and decomposed using POD. They found that their residual framework was better in extrapolating and interpolation along the temporal dimension. [320] found that deep learning based ROMs outperformed the Galerkin based approaches in electrophysiological simulations. A recurrent neural network based ROM was developed by [321] to predict the aerodynamic forces on an airfoil due to structural interaction and vary-

ing gust loads. The authors found that their long short-term memory units based on temporal evolution were accurate enough under controlled conditions. They used the discrete empirical interpolation method as the dimensionality reduction technique. A different strategy was adopted by [322], who used deep learning to recommend and use local ROMs on an anisotropic elastoplastic problem. The study in [323] used multilayer networks along with local ROMs to investigate a porous media flow problem. They found that the network was good enough to establish an input-output map between the fine and coarse grid representation of the simulated results. The dimensionality reduction was made using a non-local multi continuum upscaling technique.

In this chapter, we describe the development of a data-driven ROM designed to capture the hydro-mechanical behaviour in multi-material media. The ROM is trained on multiple simulations of a high-fidelity multiphysics model. The underlying model simulates the coupling between the varying fluid load and the mechanical deformation of the bulk matrix and can faithfully reproduce different forms of deformation encountered on slopes. However, this flexibility makes the fully-fledged model quite computationally expensive – inhibiting its practical use to only a few different scenarios at a time. Instead, developing a ROM trained from the high fidelity results allows the simulation of many different future scenarios over longer periods than would otherwise be possible. Here, data reduction is achieved using a proper orthogonal decomposition of the simulation results. The data matrix represents the state space vectors in the rows and their temporal evolution along the columns. Application of the proper orthogonal decomposition reduces this matrix into a tractable set of basis weights. The challenge then becomes one of predicting the temporal evolution of these weights in response to the applied loading conditions. In this work, we consider neural networks that consist of long short-term memory units [74] for the purpose of training and predicting the evolution of states in time. This class of units have several advantages over more generic recurrent neural networks. One of the key features of these networks is that they overcome the "vanishing gradient" problem by introducing a gating mechanism that prevents the exponential decay of gradients [324]. Additionally, they also help to learn long term dependencies or lags in the input sequence data.

The following sections describe the development of the Reduced Order Model and provide examples illustrating its use. We start with a brief description of the underlying Hydro-Mechanical model. We then describe the steps involved in training the ROM. Next, we give examples comparing the predictions of the ROM to those from the original high-fidelity simulations. Finally, we provide an example illustrating the use of the reduced order model to track the effects of variable loading

conditions on the behaviour of the Loy-Yang mine structure.

4.2.1 The Hydro-Mechanical Model

Fully coupled hydro-mechanical simulations require sophisticated nonlinear solvers to capture the complex relationship between load due to fluid and a material's mechanical response. Such simulations may involve detailed meshes comprising millions of degrees of freedom. As a result, modelling these systems can be quite onerous, with the number of runs usually limited to a few realisations for practical purposes. Nevertheless, it may be necessary to understand a system's behaviour over long timescales or subject to a broad range of boundary conditions in many cases. In such cases, a reduction strategy is necessary to obtain models that are able to predict the relevant physical response for significantly less computational effort and time. We outline the key steps involved in the development of the ROMs and highlight some of the potential challenges. To illustrate the approach, we derive a ROM for the Loy-Yang mine structure subjected to different loading conditions. While the high-fidelity simulations used to create the ROM require several hours of central processing unit (CPU) time on a dedicated workstation, the developed reduced order model is able to conduct scenario assessments involving a thousand simulations in a matter of minutes on a single multicore CPU.
The principle of effective stress was introduced by [325]. It is defined as a part of total stress that governs the deformation of the soil or rock. It can also be assumed that the total stresses can be described as a sum of effective stresses and pore pressure as given by the Eq:

$$\sigma_{ij} = \sigma'_{ij} + \alpha p \delta_{ij}, \tag{4.5}$$

where σ_{ij} is the total stress, σ'_{ij} is the effective stress, p is the pore pressure α is the Biot's coefficient and δ_{ij} is an indicator function whose value is governed as in ($\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ otherwise). For the value of $\alpha = 1$, the effective stress principle reduces to

$$\sigma_{ij} = \sigma'_{ij} + p\delta_{ij} \tag{4.6}$$

This form of used to express the effective stress principle [16; 326] which is based on Terzaghi's work from [325; 327]. This form is mostly justified because the compressibility of solid particles is very small when compared to the compressibility of porous material.

4.2.1.1 Governing Equations

We use a simple linear and isotropic elastoplastic formulation for describing the deformation. The constitutive relation can be given as the following:

$$\sigma'_{ij} = D(\epsilon_t - \epsilon_p), \tag{4.7}$$

where ϵ_t is the total strain, ϵ_p is the plastic strain and the term $\epsilon_t - \epsilon_p$ denotes the elastic strain ϵ_{el}. Here, D is the elasticity matrix expressing the material properties. Expanding equation 4.7 gives us the following form:

$$D = \begin{bmatrix} \lambda+2\mu & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda+2\mu & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & \lambda+2\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{bmatrix} \tag{4.8}$$

where, λ, μ are the Lame's parameters. These parameters can further be expressed as:

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)} \tag{4.9}$$

$$\mu = \frac{E}{2(1+\nu)}, \tag{4.10}$$

where, E, ν are the Young's modulus and Poisson's ratio, respectively.
The mechanical equilibrium equation is given as

$$\nabla \cdot \sigma = \vec{\boldsymbol{F}} \tag{4.11}$$

The elastic strain tensor is given as:

$$\epsilon_{el} = \frac{1}{2}\left(\nabla\vec{\boldsymbol{u}} + (\nabla\vec{\boldsymbol{u}})^T\right) \tag{4.12}$$

4.2.2 Creation of the 3D Domain: The Loy-Yang Mine Structure

Finite element analysis requires an accurate geometry. Computer-aided design software are often used to parameterise the boundaries of the domain to realise a smooth representation and aid fast FEM simulation [328]. In order to run the FEM on the Loy-Yang mine shaped computational domain, we need to create its

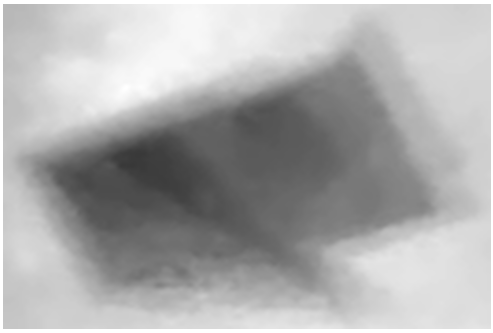


Figure 4.1: Depth map of the Loy-Yang mine obtained from satellite imagery. Darker regions denote surface depth while lighter regions tend to be on the surface.

3D mesh. The steps for creating the same are discussed next.

4.2.2.1 Surface Representation of the Loy-Yang Mine

Surfaces can be represented in implicit, explicit and parametric forms [329]. We use the parametric form here to define the uneven Loy-Yang mine topography. The parametric form of any surface can be given as

$$(x, y, z) = (x(u, v), y(u, v), z(u, v)), \tag{4.13}$$

for $u_0 \leq u \leq u_1$ and $v_0 \leq v \leq v_1$. More generally, the domain can be defined as $(u, v) \in D$.

The basic depth map of the Loy-Yang mine structure was obtained from satellite imagery as depicted in Fig. 4.1. The colours in the figure are on the grayscale and range between $0 - 255$. These pixel values signify the depth of the mine body at different parts of the 2D image. The task at hand is to obtain a parametric representation of this uneven surface.

The depth map is not an accurate representation of the actual physical depths. Therefore we need to estimate a surface that best approximates this surface data. Let the pixel values of Fig. 4.1 be the samples of a graph surface given by the Eq 4.14

$$z = f(x, y) \tag{4.14}$$

Starting with a parametric model of this surface, the surface reconstruction problem can be reduced to a regression problem where the graph surface $z = f(x, y)$ needs to be determined that best fit the sampled pixel data. The regression equa-

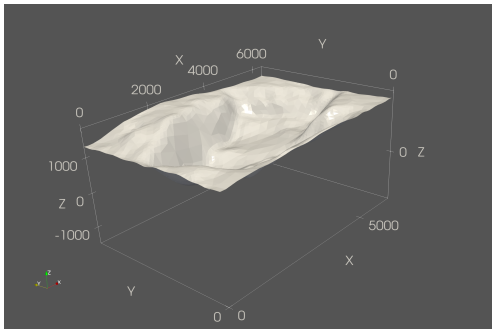


Figure 4.2: Parametric representation of the top surface of the Loy-Yang mine structure

tion can be given as:

$$X^2 = \sum_{i=1}^{n} (z_i - f(x_i, y_i; a_1, a_2, \dots a_m))^2 \tag{4.15}$$

$$X^2 = \sum_{i=1}^{n} (z_i - f(x_i, y_i; a_1, a_2, \dots a_m))^2 + \alpha^2 \iint \frac{\partial^2 f}{\partial x^2} + 2\frac{\partial f}{\partial x}\frac{\partial f}{\partial y} + \frac{\partial^2 f}{\partial y^2}\, dx\, dy \tag{4.16}$$

Fitting a surface using this equation leads to an ill-posed problem because an infinite number of function(s) satisfy the equation. To pose the problem such that we obtain a unique surface requires some constraints. A natural choice is to select a function that approximates the data well and is also a smooth in itself. The constraints can be applied using Eq. 4.16. This equation is fundamentally same as Eq. 4.15 with an added smoothness with a coefficient α term. The value of $\alpha > 0$. This term is also called the regularisation term with α as the regularisation parameter. This parameter acts as the trade-off between the smoothness and the data approximation. Low values of α tend to fit the data more while compromising the smoothness and vice-versa. The result of the parametric representation of the top surface can be seen in Fig. 4.2. After the creation of the top surface layer, the copy of the same is translated in the vertical Z-axis to form a second layer, as shown in Fig. 4.3b). Next, the top surface is extruded in the negative Z-axis to realise a 3D body followed by a Boolean union operation. Next, in order to obtain a rounded structure, a bezier curve was drawn on the XY plane, which was then extruded into the Z-axis to define the 3D rounded structure. A Boolean difference operation was applied to realise the rounded body as shown in Figure 4.3d). Then in 4.3e) the whole structure was scaled down on the Z-axis to approximate the actual dimensions of the Loy-Yang mine structure. Fig. 4.3 summarises all the steps for creating the complete computation domain.

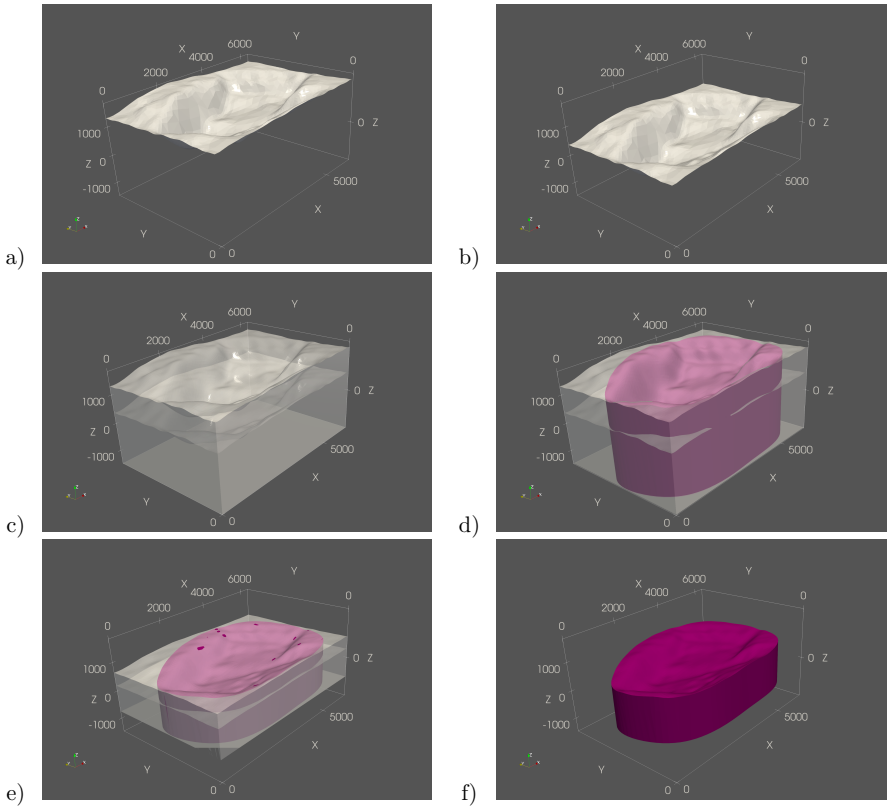


Figure 4.3: From a) to f): Top layer, Intermediate layer, Created Solid Body, Extruded bezier curve to realise a rounded structure inside the Solid Body, Scaled version of the rounded structure, Final LY-mine structure. In a) and b), we start with the parametric layer at the top, followed by a translated copy to make the intermediate layer. In c), a solid cuboid is created with its extent defined by the length and breadth of the parametric surface. A Boolean union operation helped unite different parts together. In d), a bezier curve is drawn on the XY plane and extruded to the bottom of the solid cuboid. We then use a Boolean difference operation between the solid cuboid and the extruded rounded structure in d), demarcated in pink colour. Then in e) this rounded structure is scaled along the Z-axis to match the real size of the LY-mine structure that can be seen in f)

Table 4.1: Pore pressure generating functions

Function	Range	Parameters
Sine	$x \in (-2\pi, 2\pi)$	Offset in x
Sawtooth	$x \in (-2\pi, 2\pi)$	Offset in x + Ramp/Triangle
Polynomial	$x \in (-2\pi, 2\pi)$	Offset in x + Degree $2, 3$

4.2.2.2 Loading Conditions

The pore water pressure is an important factor in determining the stability of soils. An excess of such quantity can lead the slopes toward failure. It is highly desirable to know in advance the state of the slope as a function of the pore water pressure. Training a system that can estimate/predict the state of the slope using this factor can be highly beneficial.

Performing a slope-stability dry run on the slopes of the Loy-Yang mine reveals that the current factor of safety lies in the interval $1.2 < FOS < 1.6$, which is a good range for all practical safety reasons. But to develop a robust ROM, we introduce a variation in the applied loading conditions. As is explained here, we apply 3 sets of body loads in different parts of the computational domain. It is assumed that designing a ROM with a variety of loading conditions will help enhance its generative performance on a range of inputs. For this, the computation domain is divided into 3 regions as shown in Fig. 4.5 a) through c). We define volumetric loading conditions to act on these regions of the domain.

When trained on high variance data, ML algorithms generally have better predictive performance. It goes without saying that with high variance comes longer training times. We synthesise 4 types of signals to describe the pattern in which the pore water pressure varies in the system. The parameters used to define the properties of the signal are given in Table 4.1. These signals were used to realise unique sets of full-order simulations. Patterns of the sample pore pressure generating functions are shown in Fig. 4.4.

4.2.3 State-space Description and Reduction

We consider the mechanical equilibrium equation 4.11 and solve for the state space variables. The forcing function $\vec{\boldsymbol{F}}$ is set to $\vec{0}$, and we directly contribute to the stress tensor σ's normal components by adding the varying pore pressures in the 3 different parts of the computational domain as shown in Fig. 4.5. We evaluate the effect of these varying conditions on the values of some of the important state variables like the – displacement, equivalent plastic strain, all components of the elastic strain tensor and all components of the stress tensor.

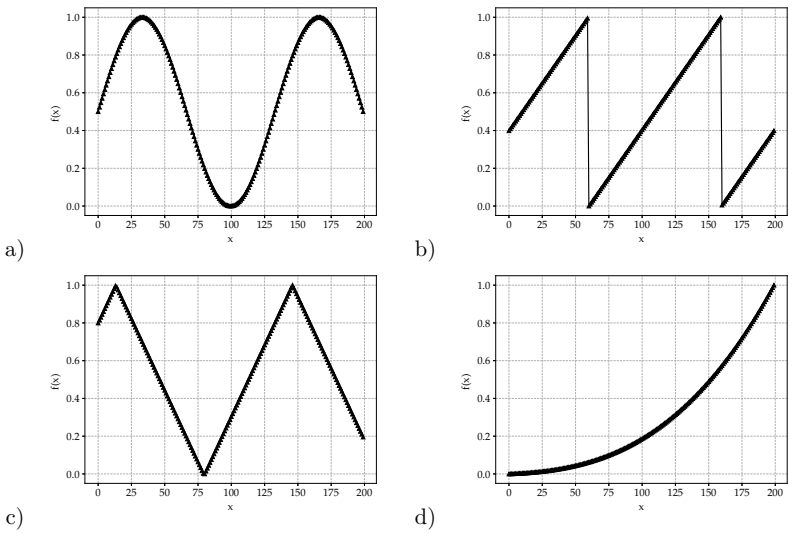


Figure 4.4: From a) to d): Sample of Sine function, Ramp function, Sawtooth function, Polynomial function; Different pore-pressure functions used as loading conditions in different parts of the computational domain. The length of the signal in the x axis is 200; this denotes the number of timesteps the loading condition is applied for.

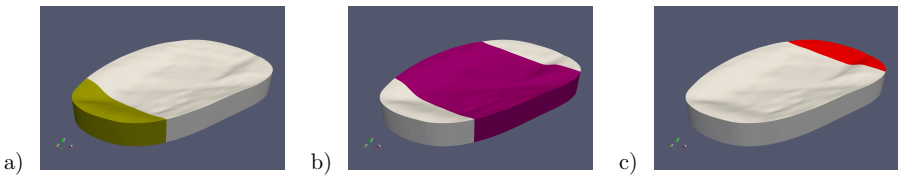


Figure 4.5: From a) to c): Part 1, Part 2, Part 3; Different parts of the computational domain for applying the pore pressure loading conditions

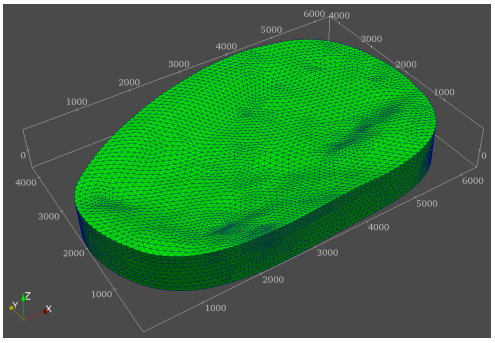


Figure 4.6: Computational mesh of the domain

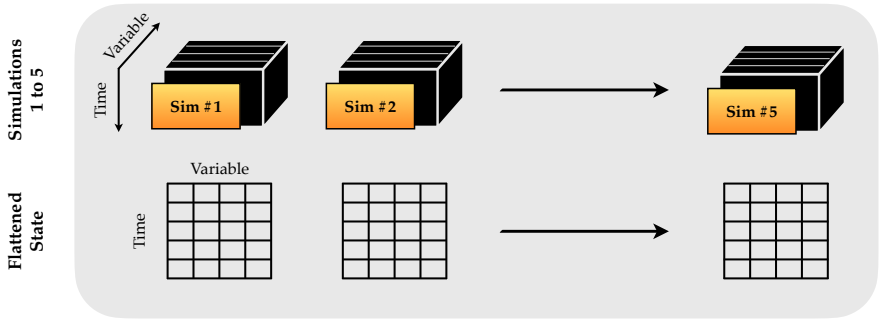


Figure 4.7: Schematic representation of a state. Top row shows actual simulation data. Bottom part shows a transposed data matrix where each row is a 1D long skinny vector realised by flattening the the state variables and stacked along the time axis.

Table 4.2: State variables and symbols

Variable	Symbol
Equivalent Plastic Strain	ϵ_{pl}
Displacement Magnitude	u
Strain Tensor Components	ϵ_{xx}, ϵ_{xy}, ϵ_{xz}, ϵ_{yy}, ϵ_{yz}, ϵ_{zz}
Stress Tensor Components	σ_{xx}, σ_{xy}, σ_{xz}, σ_{yy}, σ_{yz}, σ_{zz}

A state is defined as the collection of state vectors that hold the values of different physical variables. A state vector is a vector whose elements are the values of a state variable with their length equal to the nodes of the mesh on which they are solved. Fig. 4.6 shows the mesh of the computational domain. The type of mesh is tetrahedral. The physical variables are solved on the nodes of the mesh during each simulation. Schematically, the storing of the variables' states can be illustrated as in Fig. 4.7. As shown in the figure, there are two axes of a simulation – the time axis and the variable axis. The variable axis holds the values of the different physical variables, whereas the time axis holds their successive progressive states. In the second part of the figure, the number of rows denotes the number of time steps for which the simulation is run. The column holds a flattened state of a variable. Ideally, one would want to include simulations that are different from each other with regard to the variation in the physical variables; therefore, in our case we ensure to generate a unique simulation by varying the parameters of pore pressure generating functions as shown in Table 4.1. A unique set of 3 out of 4 signals were used for every simulated case.

Table 4.3 lists the values of the materials that were used during the simulation. The properties have been abbreviated as follows - cohesion (c); internal angle of friction (ϕ); Young's modulus (E); density (ρ) and unit weight (γ). In this work

Table 4.3: Material properties

Material Type	c (kPa)	ϕ (deg)	E (MPa)	ρ (kg/m^3)	γ (t/m^3)
Overburden	50.00	26.00	1860.00	1120.00	1.90
Coal	150.00	26.00	1560.00	1100.00	1.12

we have considered homogeneous material properties but the technique could well be extended to the heterogeneous equivalents as long as the degree of freedom of the computational domain remains constant across the analysis. Notably, for the the latter case the number of components in the POD would need to be increased to accommodate the spatial variations.

All the simulations were carried out in the opensource FEM solver - Multiphysics Object-Oriented Simulation Environment (MOOSE) [330] using the Porous Flow and Tensor Mechanics modules.

4.2.3.1 Model Reduction

The high fidelity model that describes the material behaviour can be used to generate a series of simulations that will define, train and test the reduced order model. Running a series of simulations typically takes hours to complete, depending upon the complexity of the physics and the resolution of the mesh. As is commonly done in machine learning models, the results of those simulations are divided into training, and testing sets [331]. For the examples considered here, we reserve around 60% of the simulation output for training and 40% for testing. A typical high-resolution simulation may contain several hundreds of thousands, if not millions of degrees of freedom. The first task in the model reduction is to reduce this to a more tractable number. This is achieved by using a proper orthogonal decomposition (POD) of the output states to identify the modes of deformation that are most important to the material's response.

From the state variables output from each training simulation sampled at discrete time intervals, we construct a matrix in which each column represents a separate simulation set, and each row represents a degree of freedom from the model itself. This leads to a matrix $\mathbf{M} \in \mathbb{R}^{N \times L}$, where N is the number of variables in the state space and L is the number of output states. The output degrees of freedom combine both hydrodynamic and mechanical parameters from the multiphysics simulation. For the examples presented in this chapter, we make no distinction between the state variables – all of the independent degrees of freedom are included in the rows of each column in a single vectorised representation. Note, however,

that spatial relationships are not represented directly in the matrix. Instead, these are implicitly captured through the mapping from the simulation outputs to the column vectors. Also, it should be noted that this matrix is extremely skewed, with the number of columns exceeding the rows by a large margin. While here, for the sake of brevity, we describe the operations as acting on the whole matrix, in practice, the calculations described below are performed using iterative methods that act on individual columns (simulation outputs) rather than the entire system.

Once the matrix $\mathbf{M}$ has been constructed, we then use the proper orthogonal decomposition to identify the modes of deformation that are most important to the material response. The decomposition method is based upon the SVD matrix factorisation algorithm wherein SVD finds an orthonormal basis by performing the Eigen decomposition of the data matrix:

$$\mathbf{M} = \mathbf{U}\Sigma\mathbf{V}^T \tag{4.17}$$

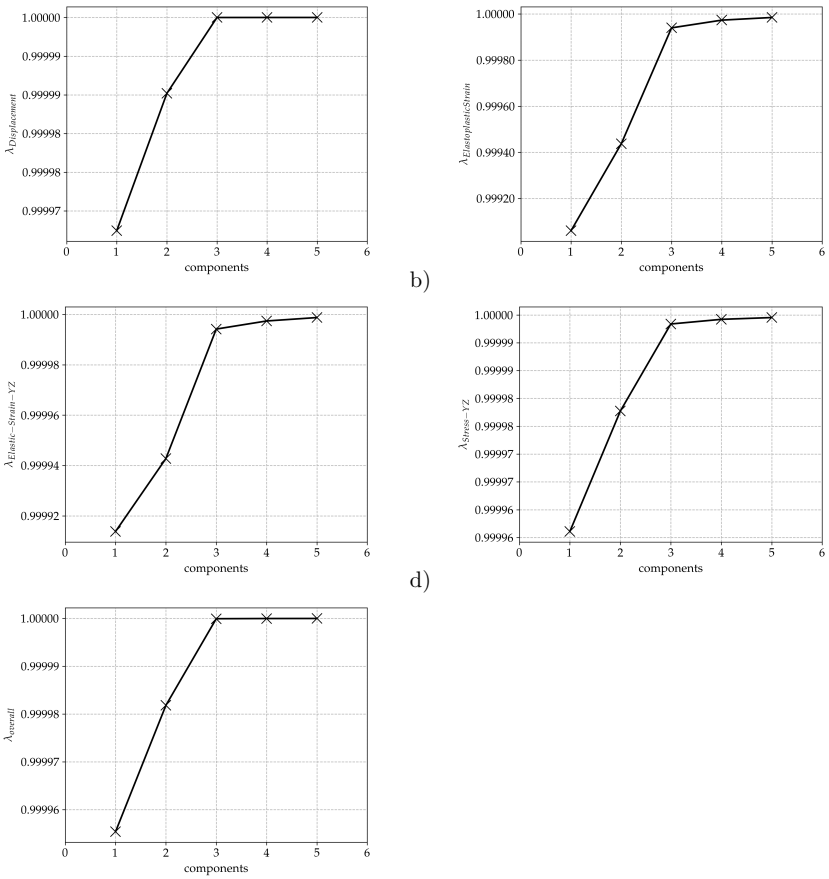
where $\mathbf{U}$ is an $N \times N$ orthogonal matrix representing particular modes describing the spatial distribution of the state variables, while $\mathbf{V}$ is an $L \times L$ orthogonal matrix describing the temporal response of the system. The matrix Σ is a rectangular $(N \times L)$ diagonal matrix, whose diagonal entries σ_{ii} contain the so-called singular values of $\mathbf{M}$.

Model reduction is achieved by selecting only the columns of $\mathbf{U}$ and $\mathbf{V}$ corresponding to the n-largest singular values in Σ for a predetermined number n of singular values. From this reduced number of singular values and their corresponding columns of the $\mathbf{U}$ and $\mathbf{V}$ matrices, $\mathbf{M}$ is approximated as

$$\tilde{\mathbf{M}} = \tilde{\mathbf{U}}\tilde{\Sigma}\tilde{\mathbf{V}}^T. \tag{4.18}$$

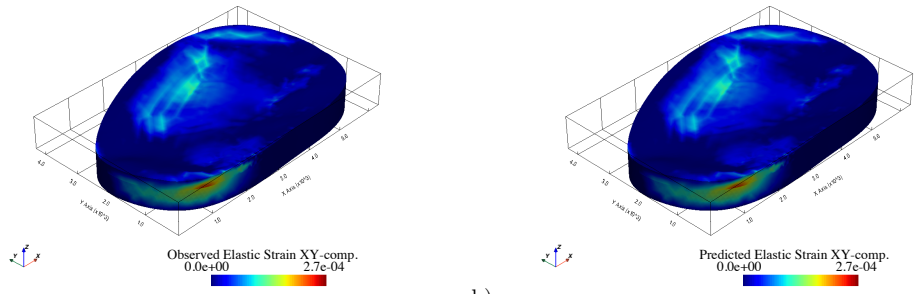
In this case, the required number of singular values are estimated by using the orthonormal vectors in the columns of the $\mathbf{U}$ matrix to first compress and then reconstruct the data in the *test set*. In this way, the number of singular components chosen from the decomposition of the training data is determined based on its ability to represent key components in the test data.

Thus, after collecting a representative sample of system states (by performing several simulations subject to representative combinations of loading conditions of interest), we then combine these results to create a state matrix $\mathbf{M}$. Next, the principal modes of deformation are identified by using the singular value decomposition. This produces a set of orthonormal state-vectors (the columns of $\tilde{\mathbf{U}}$) that form the basis for the reduced order representation of the system.



a) b) c) d) e)

Figure 4.8: From a) to e): Reconstruction accuracy of physical variables with increasing number of singular components



g) h)

Figure 4.9: a) and b): Demonstration of the reconstruction of observed and predicted states of elastic strain XY-component

The orthonormal state-basis vectors can be used to produce reduced-order representations of the system state at any time t, by taking the dot product of the transposed basis vectors $\tilde{\mathbf{U}}^T$ with the system state vector at that time, $\mathbf{s}(t)$:

$$\mathbf{w}(t) = \tilde{\mathbf{U}}^T \cdot \mathbf{s}(t)\,. \tag{4.19}$$

This yields a vector of weights, $\mathbf{w}(t) = [w_1, w_2, \ldots, w_n]^T$, describing the strength of the singular components at time (t). The system state $\tilde{\mathbf{s}}(t)$ can then be reconstructed from the weights as follows:

$$\tilde{\mathbf{s}}(t) = \tilde{\mathbf{U}} \cdot \mathbf{w}(t)\,. \tag{4.20}$$

The state variables from the reconstructed state can then be mapped back onto the original mesh from the high fidelity simulation to recover the spatial distribution. In Fig. 4.8 we can see graphs of the reconstruction accuracy versus the number of POD modes. It is evident that a minimum of 4 components are required to achieve the best reconstruction of the original states. But in the actual implementation 5 modes were used. An example of the reconstructed state can be seen in Fig. 4.9. While the singular value decomposition provides an efficient means to represent the system states, it does not describe how these states evolve over time. Importantly, it cannot predict how the system will respond to changing loading conditions. Instead, we train a recurrent neural network to predict the future state of the system as a function of the present state (described in terms of POD modes), the loading conditions applied in the present time step, and the loading conditions in the future timestep. Fig. 4.10 illustrates the adopted combined reduction process.

4.2.4 Generation of the Supervised Dataset

The application of RNN requires data to be converted into a supervised dataset. As discussed here, we work with the single-step prediction algorithm. This algorithm requires data to be prepared for a one-to-one prediction scheme. Training data from each of the simulation sets is generated using Eq. 4.19. This produces a series of principal component weights or the POD modes. Prior to training the neural network, the input data are subjected to a two-stage normalisation process. Normalisation is a standard requirement to train a neural network [332], or indeed when fitting any form of reduced order model [333; 334]. The first stage subtracts the median and scales the data according to the quantile range to remove the effects of outliers. This normalisation was chosen after evaluating its performance in comparison to standardisation that uses the mean and standard deviation. The

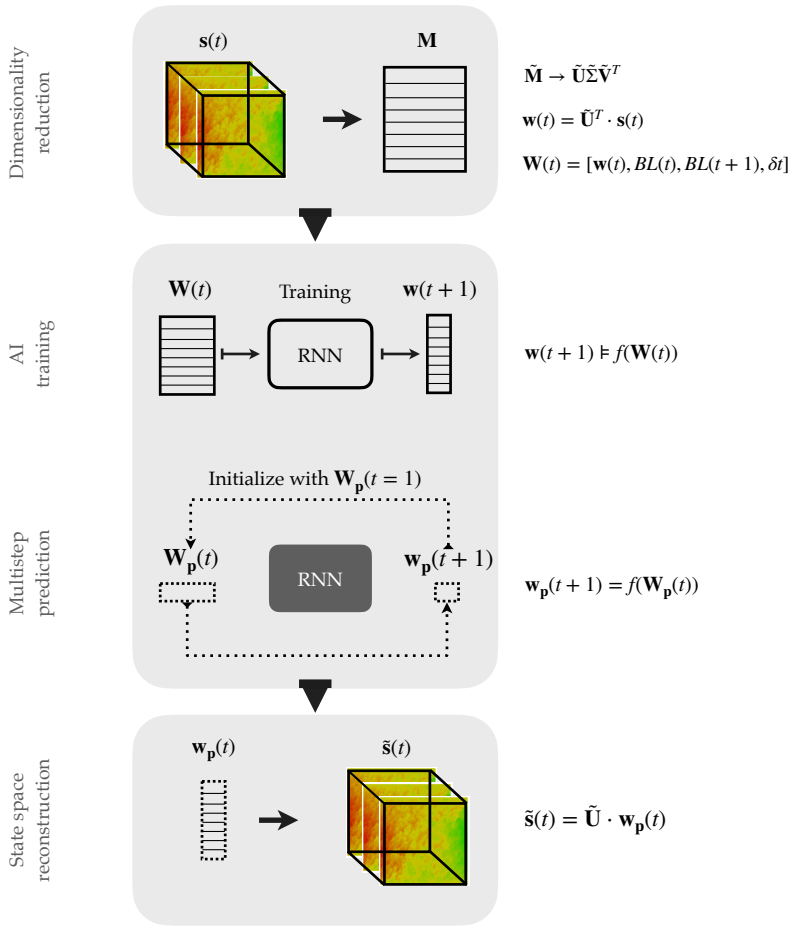


Figure 4.10: Illustration of the reduction process: starting with data matrix $\mathbf{M}$, we factorize it using POD and consider the left singular matrix $\tilde{\mathbf{U}}$. At each timestep t, we take the transpose of the state vector $\tilde{\mathbf{U}}^T$ and take the dot product with $\mathbf{s}(t)$ to get the weight vector $\mathbf{w}(t)$. These weights are combined into a vector with loading conditions and time difference δt to form a supervised training set for the RNN. After training, the network is ready for predicting on external test weights. Reconstructed state space is obtained by taking the dot product of $\tilde{\mathbf{U}}$ with the predicted weights $\mathbf{w}_p(t)$

former was chosen as the normaliser as the data did not quite follow the Gaussian distribution curve. Next, a simple min max scaling is used to scale the individual features in the training set between 0 and 1.

The goal of the training step is to develop a neural network that can predict the future state of the system—or more precisely, the future values of each of the principal components, based on the current values of the principal components, the current loading conditions, the future loading conditions, and the change in time. The neural network that we train uses a set of LSTM nodes to predict the evolution of each principal component. LSTM units are an advanced form of neuronal nodes with regulated memories that are controlled by "gates", represented by sigmoid functions [324].

4.3 Development of a Multistep Prediction Algorithm

4.3.1 The Deep Learning Model

The working of an LSTM cell is explained in section 2.1.2.2. We implement a three-layer deep RNN — where the number of layers and units in each were determined on the basis of several trials. The best-performing architecture is reported in Fig. 4.11. Each of the units in a layer receives an input from the previous layer. As the problem involves estimating the weight vectors at each timestep, we are seeking values in a continuous range, implying a regression type computation. The input is a vector $\mathbf{W}(t) = [\mathbf{w}(t), BL(t), BL(t+1), \delta t]$ and the output is a vector $\mathbf{w}(t+1)$. The final layer has a linear activation whose output is used to calculate the mean-squared error loss with respect to the target variables: weight vector $\mathbf{w}(t+1)$. The loss function is given by:

$$loss_{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{w}(t) - \hat{\mathbf{w}}(t))^2 \tag{4.21}$$

where, $\mathbf{w}(t)$ and $\hat{\mathbf{w}}(t)$ denote the observed and predicted weights. ADAM optimiser [132] is used to perform backpropagation in this deep network. The complete setup is illustrated in Fig. 4.11. Once the neural network has predicted the weights for a given timestep, they are first de-normalised (i.e. rescaled to their natural range) by reversing the operations of the normalisers employed prior to dot multiplication with $\tilde{\mathbf{U}}$, followed by the reconstruction of the corresponding state-space matrices.

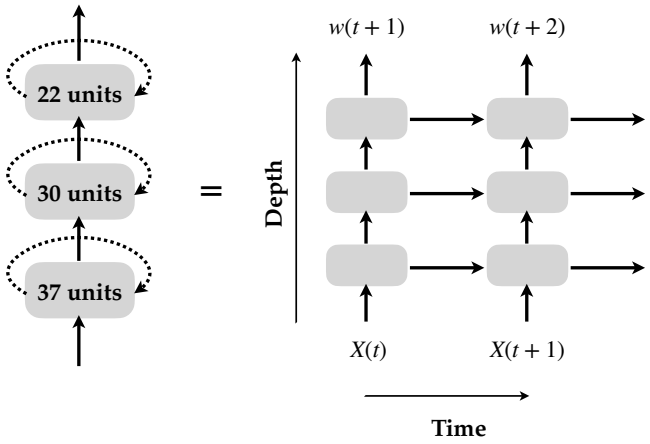


Figure 4.11: Deep network used to predict weights $\mathbf{w}(t{+}1)$ at $t = t{+}1$; Considering one training example from the training dataset, a weight vector comprising of SVD-derived weights $\mathbf{w}(t)$ of size $[5{\times}1]$ with loading conditions $BL(t)$, $BL(t+1)$ and time difference δt is input to the network that passes through a three-layer deep RNN consisting 37, 30, and 22 LSTM units, whose output is connected to a fully connected (FC) layer which outputs a weight vector of size 5 for calculating a mean squared error against the observed weights $\mathbf{w}(t+1)$ in the implemented regression setup

4.3.2 Development of Singlestep Prediction Algorithm

We show our analysis on a 3D mesh of the Loy-Yang mine structure, subject to a change in the pore pressure loading conditions. We apply 3 different loading conditions in the computational domain as shown in Fig. 4.5. We highlight the key steps and illustrate some of the challenges involved in the development here and in the forthcoming sections. However, it should be noted that the extension to more complex geometries/multiphysics and boundary conditions is straightforward.
There can be multiple ways of evolving the weights of a state ahead of time. We seek to develop a single-step prediction scheme for this purpose. Spatial information is implicitly encoded in the mapping of the mesh variables to the state vectors used in the SVD calculation. While different mesh' geometries change this mapping, they do not otherwise alter the other training and testing steps in the development of a ROM. The application of SVD on the state vectors results in a low dimensional weight vector, therefore, it becomes convenient to introduce the normalised squared error function to evaluate the predictive accuracy.

$$\Lambda_{\mathbf{w}_\mathbf{X}(t)} = \frac{\sum \left(\mathbf{w}_\mathbf{X}(t) - \hat{\mathbf{w}}_\mathbf{X}(t)\right)^2}{\sum (\mathbf{w}_\mathbf{X}(t))^2} \tag{4.22}$$

where, $\Lambda_{\mathbf{w}_\mathbf{X}(t)}$ denotes the normalised squared error of the predicted weight, $\hat{\mathbf{w}}_\mathbf{X}(t)$ w.r.t. the observed weight, $\mathbf{w}_\mathbf{X}(t)$ at timestep, t. Here, $X \in \{1, 2, 3, 4, 5\}$ and denotes the weight number obtained from the SVD calculation. After the training of the deep network, we use Eq. 4.22 to show how the predictions of the model compare to the original weights.
In the next section, we give an example of a potential application for this type of approach using the reduced order model to consider the impact of variable loading conditions on the system response.

4.3.2.1 Evaluating Single layered RNN Architecture

Choice of the deep network for development of a ROM can be very subjective. In general, increasing the number of layers increases the network's capacity to model complex functions [335; 336]. But there also lies a risk of making the network overfit to the training data. Therefore, it takes some amount of experimentation to arrive at a near-optimal design. In order to make the idea clear, we first implement a single layer RNN. This simple RNN is realised using the first layer of the deep network of Fig. 4.11, i.e. a single layer with 37 LSTM units. Training is done using the mean squared error loss function of Eq. 4.21. The training and validation losses of the network is shown in Fig. 4.12. We see that removal of the second and third layers reduces the network's capacity to model complex functions. The prediction results of the trained network are shown in Fig. 4.13 through Fig. 4.17. Each figure consists of a bottom part that shows the normalised squared error between the predicted and the observed weights. It should be noted that the prediction of weights corresponding to high singular values is more important than others. Therefore, the network must be made to adhere to this requirement. Failing to do so can make the predicted states to be less accurate, making the reduced order model less reliable.

From figures 4.13 to 4.17 are shown the predictions of weights w_1 through w_5 using single layered RNN. In each of the figures, towards the left are shown the results of single-step predictions, and towards the right, the predictions due to the multistep algorithm. It is very clear that the 37 LSTM units of the single layer do not offer enough network capacity to capture the multivariable dynamics of the 5 weight variables. The bottom graphs show the normalised squared difference of the predicted weights w.r.t. the observed weights. It is evident that converting these to the actual state space possesses the risk of inaccurate state reconstructions.

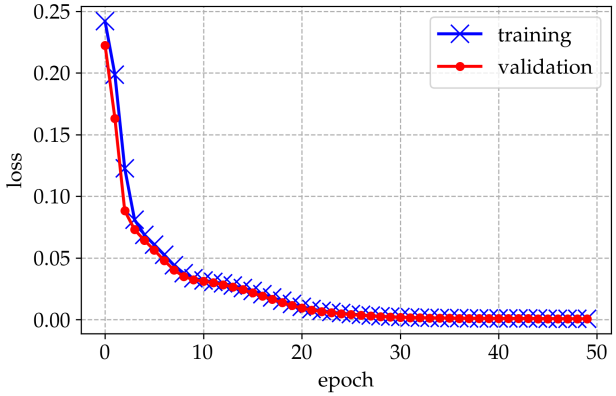


Figure 4.12: Training and validation curve for the RNN with single layer of 37 LSTM units

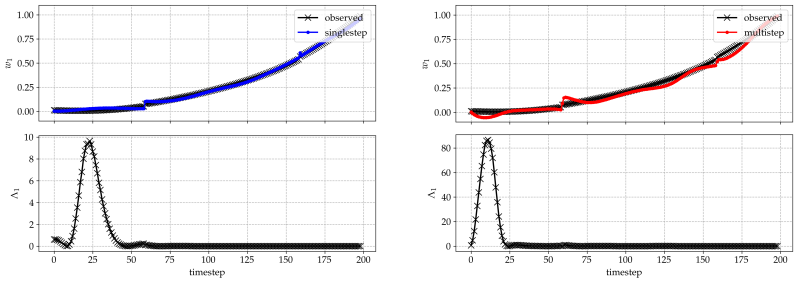


Figure 4.13: Prediction of w_1 with single layered RNN; Left: Singlestep, Right: Multistep

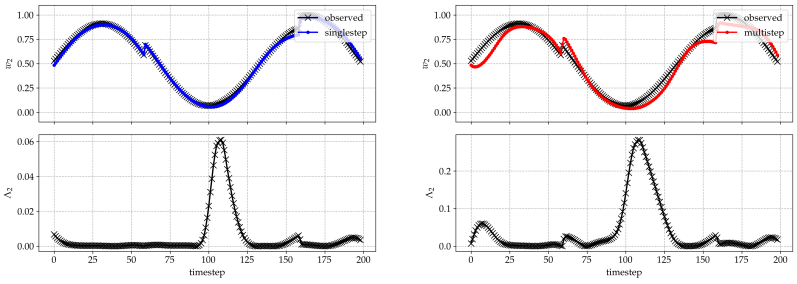


Figure 4.14: Prediction of w_2 with single layered RNN; Left: Singlestep, Right: Multistep

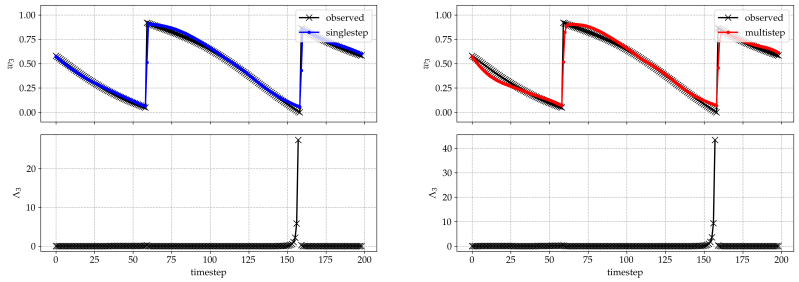


Figure 4.15: Prediction of w_3 with single layered RNN; Left: Singlestep, Right: Multistep

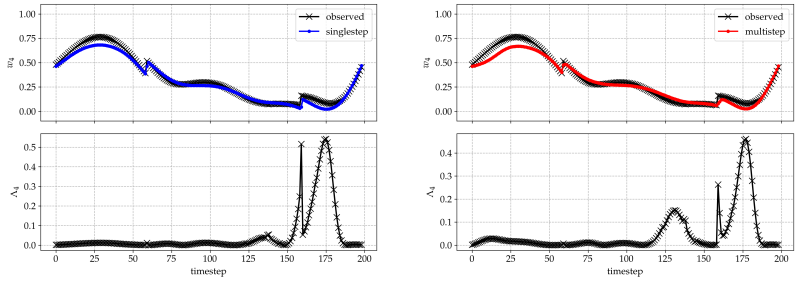


Figure 4.16: Prediction of w_4 with single layered RNN; Left: Singlestep, Right: Multistep

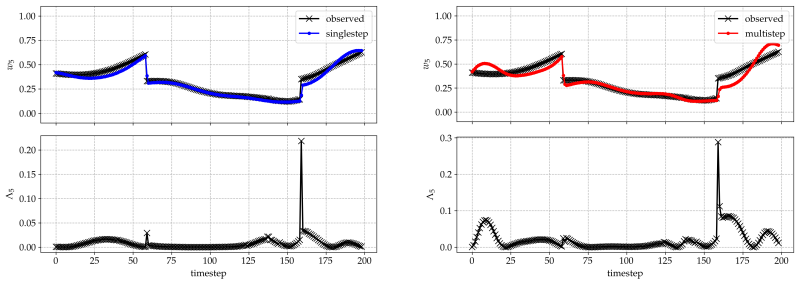


Figure 4.17: Prediction of w_5 with single layered RNN; Left: Singlestep, Right: Multistep

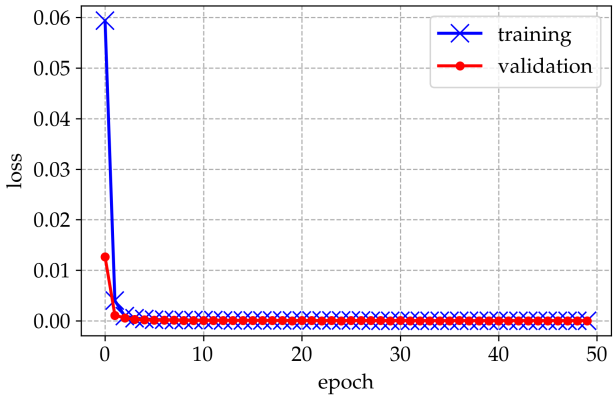


Figure 4.18: Training and validation curve for the 3 layered deep RNN shown in Fig. 4.11

4.3.2.2 Evaluating Deep RNN Architecture

After the evaluation of the performance of the single-layered RNN, we seek to test the deep RNN as shown in Fig. 4.11. The training is still done using Eq. 4.21 for 50 epochs. With the addition of two more layers, the network now has an increased capacity for modelling the input/output weights. Training and validation curves are shown in Fig. 4.18. The errors for both the training and validation fall at an increased rate in this case. In fact, the lower value of the validation error at the first epoch shows that the new architecture is more apt for the given input/output weight pairs. In addition to that, the low value of the errors on the Y-axis further supports the claim.

Figures 4.19 through 4.23 show the prediction of weights $w_1 \dots w_5$ using the deep RNN. The curves are much better reproduced in this case. Once the weights are predicted with sufficient accuracy we seek to reconstruct the actual states from the weight. In figures, 4.24 through 4.27 are shown the reconstructions from the observed and predicted states of 4 different physical variables. We see that the reconstructions are accurately reproduced after the prediction.

4.4 Evaluation of the developed ROM

4.4.1 Analysis of Multiple Realisations

The previous sections discussed how to develop and test the reduced order model. Here, we demonstrate a potential application of the reduced order model by exploring the effect of different loading conditions on the behaviour of the 3D domain.

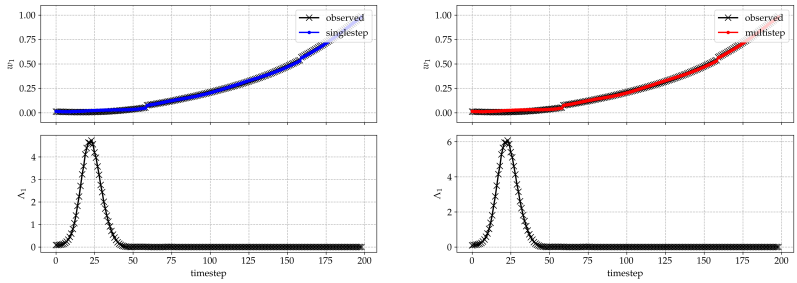


Figure 4.19: Prediction of w_1 with deep RNN; Left: Singlestep, Right: Multistep

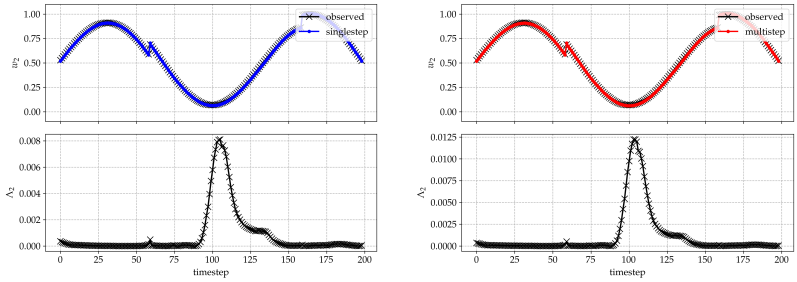


Figure 4.20: Prediction of w_2 with deep RNN; Left: Singlestep, Right: Multistep

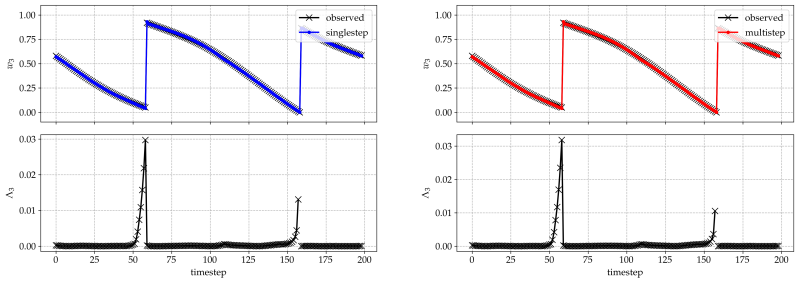


Figure 4.21: Prediction of w_3 with deep RNN; Left: Singlestep, Right: Multistep

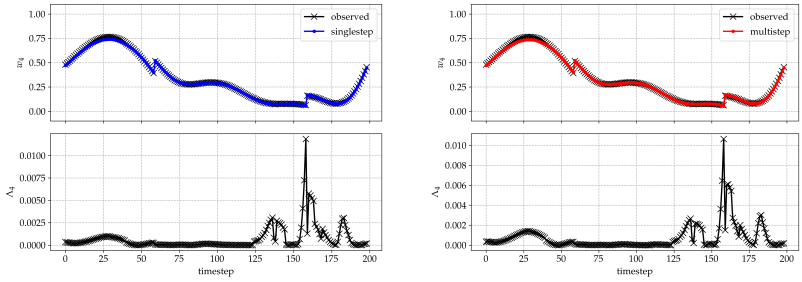


Figure 4.22: Prediction of w_4 with deep RNN; Left: Singlestep, Right: Multistep

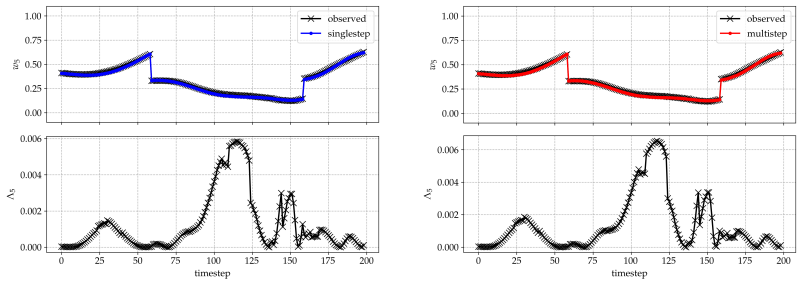
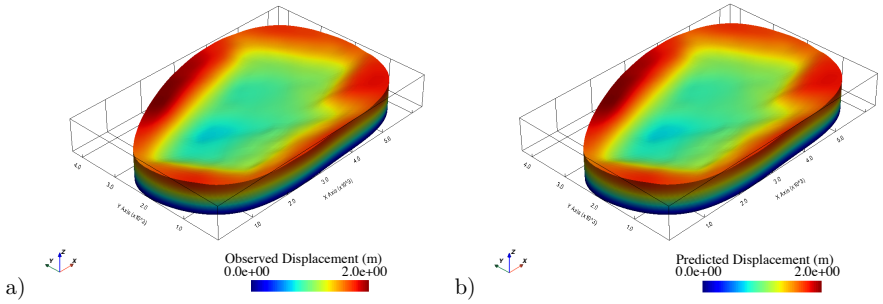
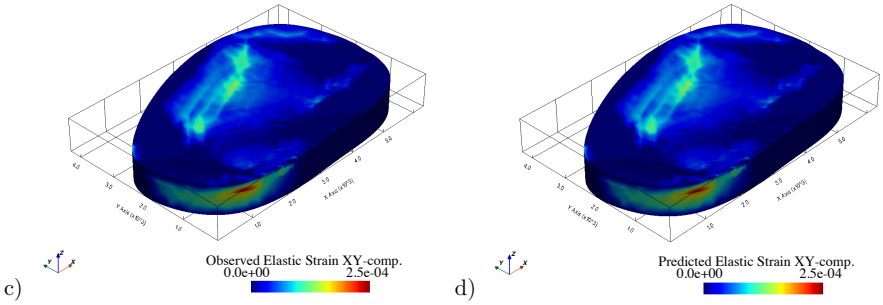


Figure 4.23: Prediction of w_5 with deep RNN; Left: Singlestep, Right: Multistep



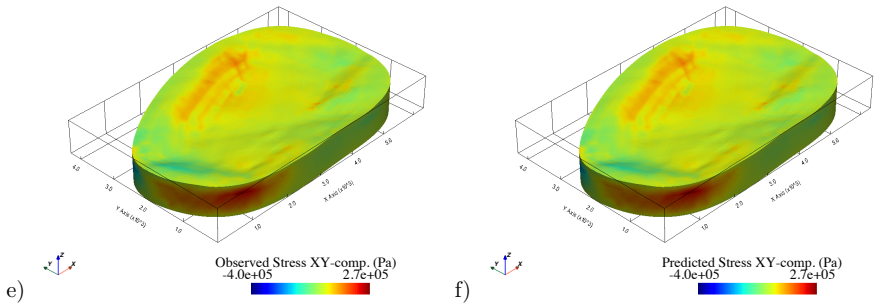
a) b)

Figure 4.24: a) and b): Reconstructions of observed and predicted states of displacement magnitudes



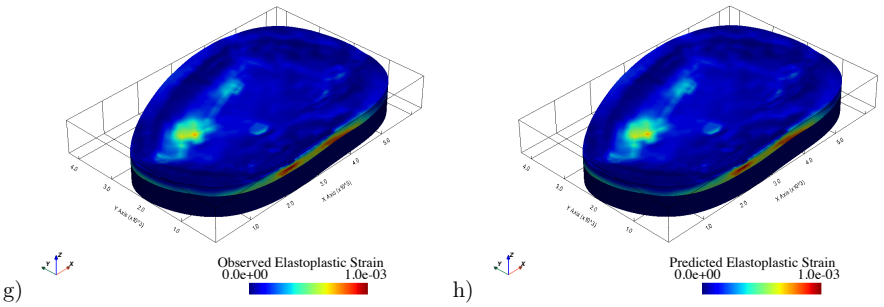
c) d)

Figure 4.25: a) and b): Reconstructions of observed and predicted states of elastic strain's XY component



e) f)

Figure 4.26: a) and b): Reconstructions of observed and predicted states of stress's XY component



g) h)

Figure 4.27: a) and b): Reconstructions of observed and predicted states of elasto-plastic strains.

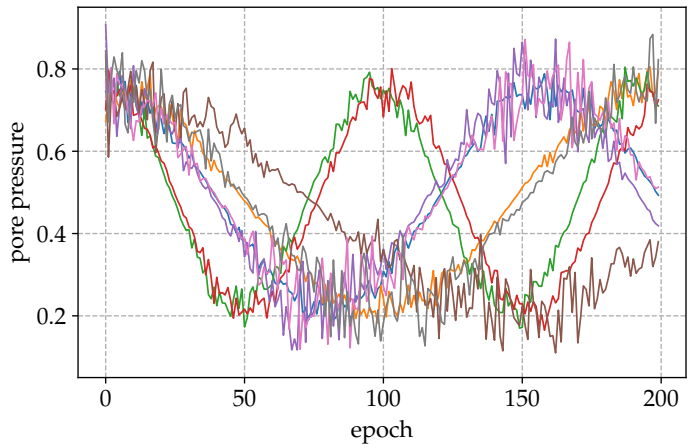


Figure 4.28: Instance of random pore pressure loading conditions. Examples of random loading conditions are used to illustrate the application of the reduced order model. Sinusoidal loading conditions with 4 different frequencies are corrupted with 5% Gaussian noise to realise a variety of loading conditions. For the purpose of visualisation, only 8 out of 1000 different are shown here. ROM was run for all 1000 simulated loading conditions.

We consider a single case study. The three sub-domains in Fig. 4.5 experience three different patterns of simulated pore pressure. An illustration of the pore-pressure loading conditions is given in Fig. 4.28. These signals were generated by corrupting the pure sinusoidal signals with 5% Gaussian noise to realise complex loading conditions.

A total of 1000 different simulations were conducted, where each simulation consisted of 200 timesteps. As the ROM only considers 5 principal components, the simulations could be completed in a few minutes on a single laptop – compared to multiple hours across a dedicated computing cluster for one high-fidelity simulation. Perhaps a less obvious but equally important feature of the ROM is that the amount of memory required to store each output state is also dramatically reduced. Only one set of principal components is required at each output time step, meaning that the full evolution of each of the 1000 simulations can be retained by querying the weights of each of the timesteps. For simulations of slope-stability, for example, this means that not only can the individual outputs be analysed for failure on mass, but the timesteps prior to failure events can be easily revisited to determine the causes of failure in each case.

These examples demonstrate not only the model's ability to run many realisations of the same system but also its ability to track the effect of uncertainty on particular features of the loading conditions (the rate of change in pore pressure in Fig. 4.28 for example).

There is also the question of how to interpret the results of these many realisations. As is often the case when running Monte Carlo simulations, the result of any one simulation is less important than the ensemble behaviour of the complete set. To represent this behaviour, it is common practice to report statistical measures of the full set of results, such as the mean and standard deviation of particular fields at each location. Examples of these results are shown in Fig. 4.29.

Nevertheless, while these results provide some insight into the system behaviour, it should be noted that distributions of mean values may not be typical of the overall system response (indeed, it may not even be a valid solution), while other measures (e.g. standard deviations) do not convey the skew or spread of the results. Median values and quantiles may provide more information. However, standard full-fidelity simulations, they may be difficult to obtain due to the need to store all simulation results. Furthermore, all of these measures fail to convey any information on the spatial correlation of the output.

A key advantage of the reduced order model is that it can be used to analyse variations in spatial distributions, not just local output, by considering the correlations between the different principal components. This is illustrated in Fig.

4.30. This figure plots the first five principal components obtained from each of the 1000 simulations against each other for a given timestep (here 100^{th} out of 200 timesteps).

These plots immediately demonstrate the correlation (or lack thereof) between the different principal components under the range of body loads. In the case of 4 random pore pressure loading conditions, while some principal components are largely uncorrelated (e.g. $w1$ $w2$; $w1$ $w3$), there are others (e.g. $w1$ $w4$; $w2$ $w4$) that are somewhat correlated.

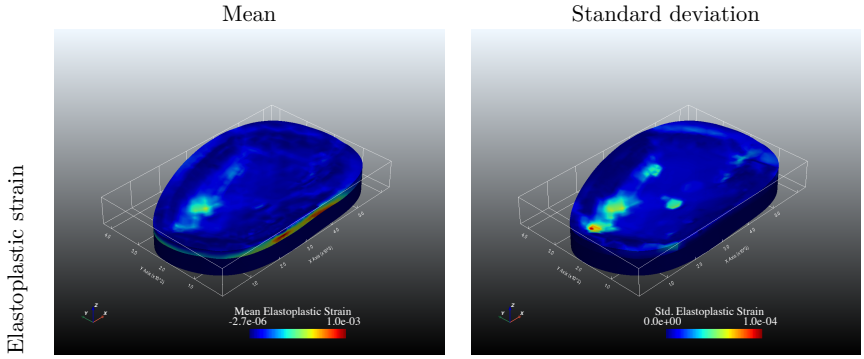


Figure 4.29: Mean and standard deviation of the plastic strain fields from 1000 simulations corresponding to pore pressures shown in Fig. 4.28 generated using the reduced order model.

4.5 Results and Discussions

The current chapter was dedicated to building an LSTM based deep learning model for accelerated simulation of hydro-mechanical systems. We describe the results of the various sections here.

Creation of the Computational Domain

Development of the ROM needs training data. This training data is usually provided by the full order numerical simulations. These simulations require a computational domain for the solution of the governing equations and specification of the loading conditions. Since we were up to develop the model of the Loy-Yang mine structure, the creation of the domain involved working with the actual elevation data of the mine. So, the starting elevation map of the top of the surface of the mine was extracted by specifying the latitude and longitude pairs. The extracted depth map is shown in Fig. 4.1. The darker regions are at a depth while lighter regions are towards the surface. It must be noted that elevation maps from satellite

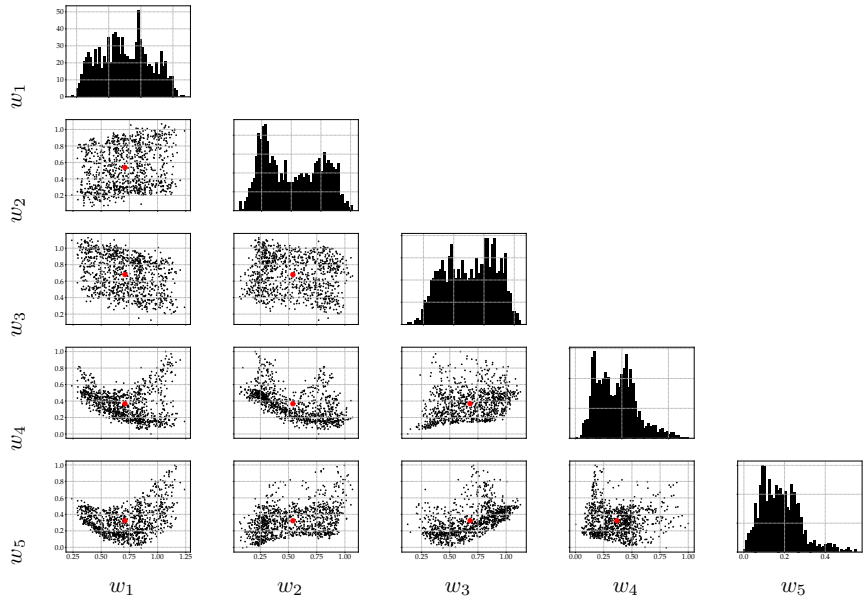


Figure 4.30: Distributions of principal component weights from 1000 simulations: Graphs on the main diagonal show histograms for each principal component weight, while off-diagonal plots show the scatter plots of pairs of weights from each simulation. Red dots indicate the mean values. Distributions correspond to the loading conditions shown in Fig. 4.28.

data are not accurate enough and, therefore, need to be processed to match the depth observations. After obtaining the depth map, the surface was parameterised as shown in Fig. 4.2. This step is especially important as finite element models work best with smooth meshes with good aspect ratios. This is generally available from a CAD design software. The parameterisation marks the first step in this design. In order to have an intermediate layer with the same surface topography, a copy of the top surface was created and placed at a depth below the original map. We can see the relative placements of the top and intermediate surfaces in Fig. 4.3a) and 4.3b). Next, an extrusion operation was done from the top surface, passing through the intermediate surface towards a greater depth to realise a 3D domain as shown in Fig. 4.3c). The sharp corners can sometimes be problematic for the convergence of FEM solvers; therefore, a curve was drawn on the surface, extruding it beyond the bottom and performing a Boolean difference operation. We can see the result in Fig. 4.3d). The pink part denotes the domain of interest. This step helped in the elimination of sharp corners. Also, in order to make the depth of the mine near-realistic, a scaling operation was done in the vertical Z-axis as shown in Fig. 4.3e). The final body was extracted by deleting the extra parts as shown in Fig. 4.3f). The final computational mesh of the Loy-Yang mine structure is shown in Fig. 4.6.

Numerical model for full order Hydro-Mechanical Simulations

The governing equations for the simulations are given in Eq. 4.7 through 4.12. As given by equation Eq. 4.5 the pore pressure directly affects the normal components of the stress tensor; we create pseudo pore pressure generating functions using 4 different mathematical functions as shown in Fig. 4.4. The details of the generative functions are given in Table 4.1. These functions help add nonlinearity to the applied loading conditions via their application on three different regions of the domain, as shown in Fig. 4.5. One of the compelling reasons for their usage instead of the real pore pressures from field measurements is that they help train the deep learning based ROM for a diverse scenario making it more robust to changes in loading conditions. A total of 5 full-order simulations were performed for testing the ROM, out of which 3 were used for training and 2 for testing. Each simulation consisted of 200 timesteps solving for 14 state variables as shown in Table 4.2. In Fig. 4.7 the first row shows the form of actual simulation data that is output from a FEM solver, and in the second row, we can see how the state variables are flattened to form one long skinny vector per row, and when stacked along time axis give us a 2D representation of the evolution of the states along the length of the column. On taking the SVD on the transposed state matrix, we

get a reduced representation in the form of 5 POD modes. In Fig. 4.8 we can see the reconstruction accuracy of different physical variables like the displacement, elastoplastic strain, elastic strain XY component, stress XY component and the overall combined state. High accuracy of the reconstruction is important for emulating a full order simulation. We can also see in Fig. 4.8a) through 4.8e) that the reconstruction accuracy varies for different state variables.

Development of Multistep Prediction Algorithm

In order to realise the functioning of the ROM, we first consider the development of a single-step prediction algorithm. The single-step prediction algorithm requires the simulation data to be converted into a supervised training set. The state space data obtained from full order simulations are converted to the weight space using the SVD because learning in the original data space is impractical for various reasons like – training times, data storage and memory requirements. Referring to Fig. 4.10 of dimensionality reduction block, the predictors are realised by combining the weights of the current timestep, $\mathbf{w}_t$, the loading conditions at timestep $BL(t)$, the loading conditions at timestep $BL(t+1)$ and the time difference δt. The combined vector is called the $\mathbf{W}(t)$. The target vector is just the weight vector $\mathbf{w}(t+1)$. By virtue of the creation of the supervised dataset for a single-step scheme, the number of training examples reduces by 1 because we cannot have data to predict the first timestep. In the Fig. 4.10 the second block of AI training, the RNN based single-step prediction algorithm is trained. In the second part, once the training is over, the algorithm is ready to predict the future weights $\mathbf{w_p}(t)$ given the current predictor vector $\mathbf{W}(t)$. The RNN used here is realised using LSTMs with 37, 30 and 22 units that form a 3 layer deep network. The final weights are output after being squashed using a dense layer. The training is performed using the backpropagation algorithm with a mean squared error loss. A total of 50 epochs were set for training and validation, as shown in Fig. 4.12.

Evaluating the Performance of the Developed ROM

In order to assess the quality of the ROM, we also trained another RNN architecture with just a single layer with 37 LSTM units. The results of the single-step prediction are shown in Fig. 4.13 through 4.17. In each of the figures, the graph on the left shows the result of the single-step prediction where the predicted weight (blue) and the observed weight (black) are plotted together. At the bottom, the plots show the normalised squared difference between the predicted weight and the observed weight. Towards the right, the results of the multistep prediction are given. In Fig. 4.13 we can see that single step prediction does a good job at

predicting $\mathbf{w}_1$ with the NMSE at the bottom being around ≈ 10 for the single-step case and ≈ 80 for the multistep case. Towards the right, the multistep scheme shows some discrepancy at timestep 60 and timestep 160. The transitions are not well captured; in other words, the network fails to learn the intricate/sharp transitions in weight's trajectory. In Fig. 4.14 the multistep scheme still shows the discrepancy but now at timestep 0 to 20 and at timestep 150 to 160. For the prediction of $\mathbf{w}_3$, both single-step and multistep schemes do a fair job, as seen in Fig. 4.15. But still, the error plot shows a high value for the multistep case. Similar trends are observed in figures 4.16 and 4.17 with a higher error for the multistep cases.

The evaluation of the deep RNN shows much improvement over the single layer RNN. In Fig. 4.19 the prediction of weight $\mathbf{w}_1$ appears to be equally good for both the single-step and the multistep schemes. The difference between the errors is also reduced when compared to the single layer RNN case. The error shows a discrepancy only at the beginning between timesteps 10 and 40. In Fig. 4.20 the maximum error is much reduced and is of the order of 1e-3 for the single-step case and 1e-2 for the multistep case. At the sharp transition that occurs at timestep 60, the deep model shows much improvement with just a glitch at timestep 60 in the error plot. It faces more difficulty in tracing the trajectory from high to low to high scenarios as in between timesteps 90 to 140. In Fig. 4.21 the errors are of the order 1e-2 with the multistep exceeding the single-step by just a fraction. The sharp transitions are not well captured probably because of the large jump in the values as seen at timesteps 50 to 60 and timesteps 155 to 160. In figures 4.22 and 4.23 the error at the bottom shows a much more complex error trajectory. Again, the low/high variations are not smooth. Also, the sharp changes are problematic. But the values of the errors are low enough to be ignored.

From figures 4.24 to 4.27 we can see the results of the reconstructed states of 4 of the state variables. Fig. 4.24 shows the reconstruction of the displacement field with high values ($\approx 2.0m$) at the top and no movement at the bottom. Both the observed and the predicted states are in good agreement with each other. The next Fig. 4.25 shows the XY component of the elastic strain with a high value ($\approx 2.5e-4$) towards the front and medium values $\approx 1.2e-4$ at the slopes towards the top part of the figure. The stress's XY component is shown in Fig. 4.26 with a high value of ($\approx 2.7e+5$) and low value of ($\approx -4e+5$). Both the observed and predicted states are in good agreement with each other. The states of the elastoplastic strain in Fig. 4.27 show that the Loy-Yang mine structure experiences the highest plastic strain in the light blue regions and is more likely to fail with adverse pore pressure values. Again the observed and the predicted states

are in good agreement.

Statistical Analysis and Multiple Realisations

Once the ROM is trained with the predefined set of the full-order simulations, it becomes ready to generate independent simulated states. For that, it requires an initial starting set of weights defined by a 1D vector of size 5 and a series of current and future loading conditions. We analyse the ROM generated states using the standard deviation and mean statistics. In Fig. 4.28 we show only 8 out of 1000 (for visualisation purposes) trajectories of pore pressure loading conditions with a randomly initialised weight vector. These are combined, normalised and arranged as a supervised set (similar to the one used during training) to be fed with the weight vector. When run in the ROM, 1000 different simulations could be realised within a matter of minutes. To be specific, it took around $253s$ to generate the 1000 simulations. The reconstruction took an extra 45 minutes time to save the ROM generated full-order simulations to the hard disk. We assessed the performance of the generated realisations by taking the mean and standard deviation of the physical variable – elastoplastic strain at timestep 100, whose reconstructed states are shown in Fig. 4.29. One can recall that the high strain is experienced in the light blue regions. The red regions are also prone to failure but require additional shear strength reduction analysis for determining the actual factor of safety. The standard deviation plot on the right shows the maximum variation $\approx 1.0e-4$, which is an order high of the mean state values. This implies that the ROM generated pseudo simulations have a high variance of generated states while still remaining near to the mean state. This is ideal for analysis with a fixed computational mesh such as that of the Loy-Yang mine structure. In Fig. 4.30 we show the 1000 states (represented as dots) in a 5-dimensional reduced weight space. The main diagonal shows the histogram of the weights (from top to bottom: $\mathbf{w}_1$ to $\mathbf{w}_5$). The red dot shows the location of the mean state in the thousand ROM generated states.

Chapter 5

Conclusions and Future Work

Data acquired in Geoscientific works has many challenges. Often, the processes responsible for generating them are complex, making the acquirer record instances of data using multiple probes. This leads to the generation of a dataset that possesses some important properties. Firstly, there is an inherent redundancy; multiple probes may have responded to some common signals by sharing their information with each other. Even if they have captured unique and independent parts of the process, they may still complement each other for completeness. In the real-world scenario, the sensors may have recorded a noisy version of the signal. This may be present throughout the recording or in some parts of it. A similar case was developed in chapter 3, where we saw that the probes kept at the corners of the cube showed unique wave signatures resulting from multiple reflections; it is obvious that neglecting even a single one would reduce the information required for inference of the pore networks. At the same time, there could have been multiple pore network models that would have generated similar acoustic responses; this presents a classic case of an ill-posed problem and suggests improvement to the conventional optimization-based data modelling workflows. Secondly, the size of the data may be too large to analyze, either by virtue of the scope of the experiment or the scale of the study. We saw in chapter 4 that the observed state space of a single simulation was too high of the order of ($1e6$). This makes it difficult to train a recurrent neural net. Reduction of data dimension becomes important in such cases. This has great significance in high-fidelity simulations of processes that are otherwise impossible to study. In the author's view, there exist at least three ways of working with these data.

The first one is the model-based data pipeline. In this mode, the modeller assumes a model for the data generating process. Using this mode can be helpful as it can quickly provide the result using analytical techniques, given that the process has been studied extensively and that there exists enough theory to support the equa-

tions guiding the analytical solution. The modeller may also choose this mode for simplifying a model. This generally ignores the intricacies involved in the data generating process. The reason can also lie in the incompetency of the modeller to handle complex and more detailed forms of the problem description. The second approach takes up the data-based pipeline. This is the modern way of dealing with data. In this approach, ML algorithms are provided with labelled/supervised data and are made to learn about the data transformation process from them. This often renders them "black-box" models. They do this by mapping a function between the inputs to outputs in an abstract manner by building complex relationships among them. Though the results are generally encouraging, explaining the trained model is quite difficult and may require some meta-modelling. The third approach takes the hybrid mode, wherein the model-based approach is combined with the data-based approach. This approach is often taken up when the data generation process is supported by a well-grounded theory, and one wants to accelerate certain parts of the solution. This, of course, introduces some amount of error (due to data-fitting stages) but can be suitably ignored with the interpreter's discretion and purposes. A partial reason for taking this approach may also be due to the subjectivity associated with interpretation. Machine learning algorithms could help resolve this by providing data-based approximations for certain parts of the solution hence avoiding any human bias.

Motivated by the issues of data modelling in geophysics and geomechanics, we have worked on two important problems in this thesis. We consider the latter two approaches and demonstrate their effectiveness along the modelling process. These have been chosen to cover the different aspects of data and perhaps provide a way of tackling some issues in real-world scenarios.

In Application-I, we have worked with data of dimensions of the order of $1e7$ while studying the acoustic responses of tiny pores in rock core samples. Effective medium techniques such as the "Differential Effective Medium" and the "Self-consistent" approaches are widely used to model the elastic response of the porous bodies by assuming uniform ellipsoidal and spheroidal inclusions [337]. Although these are quite established in rock physics modelling, we understand that there is still a need for an approach that deals with the arbitrary shapes of the pores and their inclusions. Therefore, we believe that this application is a novel way of including the effect of pore geometry on acoustic data. For this, the 3D volumetric core data were processed using a modified U-Net model. The U-Net consists of convolutional neural networks and has proven to be effective in processing the 3D image data. We took up the task of inferring the pore networks within the rock core sample from their acoustic responses. With the aid of ML-based processing,

we could infer the causative pore networks with the Dice coefficient accuracy of $\approx 92\%$ with complete acoustic information in the validation set.
In Application-II, we have dealt with data of the size of the order of $1e6$ when developing a reduced order model for hydro-mechanical simulation. Conventional finite element solvers can conveniently solve 2D problems within a matter of minutes. But the situation is not so trivial when the geometry is 3D. While the accuracy of FEM still remains the main attraction, the time that it takes to get to a single solution may range anywhere from tens of hours to several days. Moreover, the modeller often needs to study the response of the structure with multiple soil parameter values because this helps him design reinforcement strategies for making the structure more stable. Due to this, there is a need to accelerate the numerical model that would give a solution at much faster rates. Such a system was realized using recurrent neural networks in chapter 4. We performed 5 full-order simulations for developing the reduced order model. Out of all, 3 were used for training and 2 for testing. High non-linearity was added to the model by varying the pore pressure contribution to the stress tensor. It was found that for the given geometry of the Loy-Yang mine structure, the recurrent networks could recognize the pattern in the evolution of the states with sufficient accuracy ($\approx 99.99\%$). We understood that for high-fidelity simulations, the reduction of data dimension is paramount for the recurrent net to process the evolution of state space.

5.1 Conclusions

With regard to the results obtained in chapter 3, we conclude the following:

- The pore shapes and sizes in carbonates are too complex for human-level analysis. The heterogeneity is prevalent at all scales. The application of simple linearly fitted models must be avoided. As seen in the section 3.3.4 every cube exhibited different wave signatures. Modern deep learning algorithms are advanced enough to study these complex rocks.

- The extraction of pore networks from rock sample data requires at least 3 stages of image processing – the contrast equalization step, median filtering step and the binarisation step. The binarisation step can be accommodated by any binarization algorithm such as the Otsu or the Sauvola algorithm, although the deterioration of the region of interest must be considered prior to making a choice. In the current work, Sauvola binarisation was used.

- The segmentation step requires the binarized image to be processed by the distance transform, skeletonization and the watershed transform steps. A

semi-automated method of pore classification can be realized using ML approaches such as the k-means clustering and watershed algorithm.

- The acoustic wave analysis is an effective non-destructive technique for inferring pore networks. We just need the right tool that interprets the weak acoustic signatures. This can be accomplished by a U-Net semantic segmentation algorithm as done in the current work.

- Whenever permissible, an acoustic simulation must be used instead of the elastic wave simulation as the latter is time-consuming and resource-intensive. Because deep learning algorithms require a large set of labelled data using the latter can be prohibitive.

- Inference of pore network is a two-fold problem comprising classification and localization. The algorithm must classify every voxel of the image and additionally localize its extent in the 3D space. In the current approach, the U-Net architecture was suitably developed to infer the pore network model from the acoustic signatures of the rock core samples. The U-Net architecture could recognize the difference in wave patterns exhibited by the porous regions and matrix parts of the rock core sample.

- The U-Net was at least 92% successful in predicting and localizing the pores. It could differentiate between the low-velocity zones of the pores and the high-velocity zones of the rock matrix. During the simulation, both the parts were specified using the medium velocity and the medium density.

- For semantic segmentation tasks, common metrics such as the Accuracy and Binary Cross-Entropy are inadequate to reflect the training of a U-Net model. Instead, the Dice coefficient must be used in combination with the Dice loss to train such networks.

- In order to train the U-Net for the pore network inference task, the requirement of data is large – 3280 samples (of dimension $64 \times 64 \times 64$) for training for achieving a Dice coefficient accuracy of $\approx 94\%$ on the training set. This data size prohibits the use of finite element based wave simulation ideas as they are time-consuming. Instead, finite difference based wave simulation must be preferred.

- Visualization of the pore network is important for designing porosity segmentation algorithms.

With regard to Application-II in chapter 4 and the obtained results, we conclude the following:

- We understand that hydro-mechanical simulations are important for determining the stability of slopes. The interplay between the pore pressures and the stress and strains inside the slope structure needs to be understood to determine any risk.

- Conventional finite element software is accurate enough to capture the state variables but may take a long time to complete one simulation. Hence, there arises a need to hasten the simulation process for purposes like uncertainty analysis.

- Reduced order modelling, as in chapter 4 has been suggested as a way forward to accomplish this task. There are various elements to the development of a ROM, and the most important is the reconstruction quality of the original states, as this guarantees the accuracy of the state variables. The second important element is the speed of simulation.

- Proper orthogonal decomposition is a suitable way of dimensionality reduction that can effectively reduce the data dimension into a small workable weight space such that advanced ML algorithms can be applied. It was found that a simulation consisting of a million DOFs can be efficiently captured by just 5 POD modes, as seen in Fig. 4.8.

- After the dimensionality reduction step, the ROM development requires a time evolution strategy. This can be realized using a recurrent neural network, as is done in the current work.

- In order to realize a versatile ROM, it needs to be trained with simulation data with high variation. In order to accomplish this, 4 different types of the pore pressure signals were realized using sine, sawtooth and polynomial functions that independently contributed to the volumetric loading conditions in three different parts of the computational domain as shown in Fig. 4.5.

- In the current context of the development of a ROM for the Loy-Yang mine structure, the geometry doesn't change with time. Therefore, it was sufficient to perform a handful of full-order simulations. In this work, 5 numbers were sufficient. 3 out 5 were used for training and 2 for testing.

- The ROM development can be divided into 3 major parts – dimensionality reduction, AI training and state-space reconstruction. The AI training comes next in priority after the dimensionality reduction step. For this, we have used the long short-term memory (LSTM) units.
- It was found that a deep RNN architecture comprising 37, 30, 22 LSTM units was optimal for capturing the time evolution of the state space. The single-layered architecture with 37 units failed to capture the intricate transitions in the pore pressure trajectories. The accuracy of the two approaches can be evaluated in figures 4.13 through 4.23.
- Elastoplastic strain is an important variable to determine the high strain regions of the slope.
- The developed ROM can be assessed using statistical measures such as the mean and standard deviations of an ensemble of samples generated by the ROM, as can be seen in Fig. 4.29.
- The current implementation could generate 1000 simulations in a matter of a couple of minutes. The reconstruction time for the 1000 simulations could be large, though.

5.2 Scope of Future Work

We would like to extend our work and contribute largely towards areas of compressed sensing and model reduction with additional Multiphysics variables such as the thermal component. We understand that more research is required in these areas for designing smart models that auto-tune with more incoming data and are also capable of learning from them while still preserving the relevant information and knowledge within.

In particular, we would like to incorporate the attention mechanism into the data driven rock physics model in order to reduce the depth of sensing. The attention engine is anticipated to identify relevant and more important acoustic signatures that are representative of the underlying pore network structure and would hopefully give better accuracy.

In the reduced order modelling case, we would like to incorporate more failure models and material heterogeneity. Specifically, we are already working with generative adversarial networks to come up with a density estimation technique for the material part. We suppose the different failure modes can be accomplished

by incorporating a conditional node mechanism that informs the ROM about the failure model it is being trained with.

References

[1] D. Thanoon, J. Vamaraju, H. Yang, K. Wei, A. Vial Aussavy, and J. Chen, "Deep seismic2well tie: A physics-guided cnn approach to a classic geophysical workflow," in *SEG/AAPG/SEPM First International Meeting for Applied Geoscience & Energy*. OnePetro, 2021.

[2] M. Vu and A. Jardani, "Convolutional neural networks with segnet architecture applied to three-dimensional tomography of subsurface electrical resistivity: Cnn-3d-ert," *Geophysical Journal International*, vol. 225, no. 2, pp. 1319–1331, 2021.

[3] Y. Shi, X. Wu, and S. Fomel, "Deep learning parameterization for geophysical inverse problems," in *SEG 2019 Workshop: Mathematical Geophysics: Traditional vs Learning, Beijing, China, 5-7 November 2019*. Society of Exploration Geophysicists, 2020, pp. 36–40.

[4] M. Küçükdemirci and A. Sarris, "Deep learning based automated analysis of archaeo-geophysical images," *Archaeological Prospection*, vol. 27, no. 2, pp. 107–118, 2020.

[5] H. Zhang, D. Melgar, V. Sahakian, J. Searcy, and J.-T. Lin, "Learning source, path, and site effects: Cnn-based onsite intensity prediction for earthquake early warning," *Geophysical Journal International*, 2022.

[6] K. Zhu, Y. Du, Q. Wang, N. Ji, and L. Zhang, "A comparative study of five networks for reservoir classification based on geophysical logging signals," *IEEE Access*, vol. 8, pp. 197 776–197 786, 2020.

[7] H. Zhang, G. Zhang, J. Gao, S. Li, J. Zhang, and Z. Zhu, "Seismic impedance inversion based on geophysical-guided cycle-consistent generative adversarial networks," *Journal of Petroleum Science and Engineering*, p. 111003, 2022.

[8] B. Guo, L. Liu, and Y. Luo, "Automatic seismic fault detection with convolutional neural network," in *International Geophysical Conference, Beijing,*

China, 24-27 April 2018. Society of Exploration Geophysicists and Chinese Petroleum Society, 2018, pp. 1786–1789.

[9] Y. Ma, X. Ji, M. Nasher, B. Hassan, and Y. Luo, "Automatic fault detection with convolutional neutral networks," in *International Geophysical Conference, Beijing, China, 24-27 April 2018*. Society of Exploration Geophysicists and Chinese Petroleum Society, 2018, pp. 786–790.

[10] X. Chen, J. Xia, J. Pang, C. Zhou, and B. Mi, "Deep learning inversion of rayleigh-wave dispersion curves with geological constraints for near-surface investigations," *Geophysical Journal International*, vol. 231, no. 1, pp. 1–14, 2022.

[11] F. Sharifi, G. Schlosser, D. Quinn, E. Mohammed, and B. Far, "Convolutional neural network based geophysical model for automatic velocity picking in seismic data."

[12] G. Zhang, Z. Wang, and Y. Chen, "Deep learning for seismic lithology prediction," *Geophysical Journal International*, vol. 215, no. 2, pp. 1368–1387, 2018.

[13] A. Yakimenko, A. Morozov, and D. Karavaev, "Practical aspects of using a neural network to solve inverse geophysical problems," in *Journal of Physics: Conference Series*, vol. 1015, no. 3. IOP Publishing, 2018, p. 032148.

[14] F. J. Sánchez-Sesma, J. A. Pérez-Ruiz, M. Campillo, and F. Luzón, "Elastodynamic 2d green function retrieval from cross-correlation: Canonical inclusion problem," *Geophysical Research Letters*, vol. 33, no. 13, 2006.

[15] O. Matsuda and C. Glorieux, "A green's function method for surface acoustic waves in functionally graded materials," *The Journal of the Acoustical Society of America*, vol. 121, no. 6, pp. 3437–3445, 2007.

[16] A. Verruijt, *Computational geomechanics*. Springer Science & Business Media, 1995, vol. 7.

[17] O. C. Zienkiewicz, A. Chan, M. Pastor, B. Schrefler, and T. Shiomi, *Computational geomechanics*. Citeseer, 1999, vol. 613.

[18] H. H. Barrett and K. J. Myers, *Foundations of image science*. John Wiley & Sons, 2013.

[19] R. C. Gonzalez, *Digital image processing*. Pearson education india, 2009.

[20] J. A. Fessler, "Model-based image reconstruction for mri," *IEEE signal processing magazine*, vol. 27, no. 4, pp. 81–89, 2010.

[21] I. A. Elbakri and J. A. Fessler, "Statistical image reconstruction for polyenergetic x-ray computed tomography," *IEEE transactions on medical imaging*, vol. 21, no. 2, pp. 89–99, 2002.

[22] M. A. Figueiredo and R. D. Nowak, "A bound optimization approach to wavelet-based image deconvolution," in *IEEE International Conference on Image Processing 2005*, vol. 2. IEEE, 2005, pp. II–782.

[23] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 689–696.

[24] s. Yu, G. Sapiro, and S. Mallat, "Solving inverse problems with piecewise linear estimators: From gaussian mixture models to structured sparsity," *IEEE Transactions on Image Processing*, vol. 21, no. 5, pp. 2481–2499, 2011.

[25] M. Aharon, M. Elad, and A. Bruckstein, "K-svd: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on signal processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[26] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: nonlinear phenomena*, vol. 60, no. 1-4, pp. 259–268, 1992.

[27] R. M. Willett and R. D. Nowak, "Platelets: a multiscale approach for recovering edges and surfaces in photon-limited medical imaging," *IEEE Transactions on Medical Imaging*, vol. 22, no. 3, pp. 332–350, 2003.

[28] W. Marais and R. Willett, "Proximal-gradient methods for poisson image reconstruction with bm3d-based regularization," in *2017 IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*. IEEE, 2017, pp. 1–5.

[29] V. Katkovnik, A. Danielyan, and K. Egiazarian, "Decoupled inverse and denoising for image deblurring: variational bm3d-frame technique," in *2011 18th IEEE International Conference on Image Processing*. IEEE, 2011, pp. 3453–3456.

[30] J. I. Tamir, F. Ong, S. Anand, E. Karasan, K. Wang, and M. Lustig, "Computational mri with physics-based constraints: application to multicontrast

and quantitative imaging," *IEEE signal processing magazine*, vol. 37, no. 1, pp. 94–104, 2020.

[31] A. Mousavi, A. B. Patel, and R. G. Baraniuk, "A deep learning approach to structured signal recovery," in *2015 53rd annual allerton conference on communication, control, and computing (Allerton)*. IEEE, 2015, pp. 1336–1343.

[32] K. Kulkarni, S. Lohit, P. Turaga, R. Kerviche, and A. Ashok, "Reconnet: Non-iterative reconstruction of images from compressively sensed measurements," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 449–458.

[33] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 2, pp. 295–307, 2015.

[34] J. Sun, H. Li, Z. Xu *et al.*, "Deep admm-net for compressive sensing mri," *Advances in neural information processing systems*, vol. 29, 2016.

[35] A. Mousavi, G. Dasarathy, and R. G. Baraniuk, "Deepcodec: Adaptive sensing and recovery via deep convolutional neural networks," *arXiv preprint arXiv:1707.03386*, 2017.

[36] J. Zhang, W. Standifird, J. C. Roegiers, and Y. Zhang, "Stress-dependent fluid flow and permeability in fractured media: from lab experiments to engineering applications," *Rock Mech Rock Eng*, vol. 40, 2007.

[37] J. Rick Chang, C.-L. Li, B. Poczos, B. Vijaya Kumar, and A. C. Sankaranarayanan, "One network to solve them all–solving linear inverse problems using deep projection models," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5888–5897.

[38] P. Viola and M. Jones, "Face detection," *IJCV*, vol. 57, p. 2, 2004.

[39] Q. Sun, W. Huang, and J. Wu, "Face detection based on color and local symmetry information," in *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*. IEEE, 1998, pp. 130–135.

[40] E. Osuna, R. Freund, and F. Girosit, "Training support vector machines: an application to face detection," in *Proceedings of IEEE computer society conference on computer vision and pattern recognition*. IEEE, 1997, pp. 130–136.

[41] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[42] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[43] G. Tesauro, "Programming backgammon using self-teaching neural nets," *Artificial Intelligence*, vol. 134, no. 1-2, pp. 181–199, 2002.

[44] D. Silver, R. S. Sutton, and M. Müller, "Temporal-difference search in computer go," *Machine learning*, vol. 87, no. 2, pp. 183–219, 2012.

[45] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[46] D. H. Hubel and T. N. Wiesel, "Brain mechanisms of vision," *Scientific American*, vol. 241, no. 3, pp. 150–163, 1979.

[47] T. C. Kietzmann, P. McClure, and N. Kriegeskorte, "Deep neural networks in computational neuroscience," *BioRxiv*, p. 133504, 2018.

[48] K. R. Storrs and N. Kriegeskorte, "Deep learning for cognitive neuroscience," *arXiv preprint arXiv:1903.01458*, 2019.

[49] N. Kriegeskorte, "Deep neural networks: a new framework for modeling biological vision and brain information processing," *Annual review of vision science*, vol. 1, pp. 417–446, 2015.

[50] N. Kriegeskorte and T. Golan, "Neural network models and deep learning-a primer for biologists," *arXiv preprint arXiv*, 1902.

[51] D. L. Yamins and J. J. DiCarlo, "Using goal-driven deep learning models to understand sensory cortex," *Nature neuroscience*, vol. 19, no. 3, pp. 356–365, 2016.

[52] L. Sadouk and T. Gadi, "Convolutional neural networks for human activity recognition in time and frequency-domain," in *First International Conference on Real Time Intelligent Systems*. Springer, 2017, pp. 485–496.

[53] L. Sadouk, T. Gadi, and E. H. Essoufi, "A novel deep learning approach for recognizing stereotypical motor movements within and across subjects on the autism spectrum disorder," *Computational intelligence and neuroscience*, vol. 2018, 2018.

[54] O. Abdel-Hamid, L. Deng, and D. Yu, "Exploring convolutional neural network structures and optimization techniques for speech recognition." in *Interspeech*, vol. 2013. Citeseer, 2013, pp. 1173–5.

[55] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, and G. Penn, "Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition," in *2012 IEEE international conference on Acoustics, speech and signal processing (ICASSP)*. IEEE, 2012, pp. 4277–4280.

[56] Z. Wang and T. Oates, "Encoding time series as images for visual inspection and classification using tiled convolutional neural networks," in *Workshops at the twenty-ninth AAAI conference on artificial intelligence*, 2015.

[57] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[58] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: a survey," *Journal of Marchine Learning Research*, vol. 18, pp. 1–43, 2018.

[59] H. Al-Hamadi and S. Soliman, "Short-term electric load forecasting based on kalman filtering algorithm with moving window weather and load model," *Electric power systems research*, vol. 68, no. 1, pp. 47–59, 2004.

[60] D. Alberg and M. Last, "Short-term load forecasting in smart meters with sliding window-based arima algorithms," *Vietnam Journal of Computer Science*, vol. 5, no. 3, pp. 241–249, 2018.

[61] S. Siami-Namini, N. Tavakoli, and A. S. Namin, "A comparison of arima and lstm in forecasting time series," in *2018 17th IEEE international conference on machine learning and applications (ICMLA)*. IEEE, 2018, pp. 1394–1401.

[62] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state markov chains," *The annals of mathematical statistics*, vol. 37, no. 6, pp. 1554–1563, 1966.

[63] P. B. Quang, P. Gaillard, Y. Cano, and M. Ulzibat, "Detection and classification of seismic events with progressive multi-channel correlation and hidden markov models," *Computers & Geosciences*, vol. 83, pp. 110–119, 2015.

[64] S. E. Yuksel, J. Bolton, and P. Gader, "Multiple-instance hidden markov models with applications to landmine detection," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 12, pp. 6766–6775, 2015.

[65] K. Struminskiy, A. Klenitskiy, A. Reshytko, D. Egorov, A. Shchepetnov, A. Sabirov, D. Vetrov, A. Semenikhin, O. Osmonalieva, and B. Belozerov, "Well log data standardization, imputation and anomaly detection using hidden markov models," in *Petroleum Geostatistics 2019*, vol. 2019, no. 1. European Association of Geoscientists & Engineers, 2019, pp. 1–5.

[66] O. Missaoui, H. Frigui, and P. Gader, "Land-mine detection with ground-penetrating radar using multistream discrete hidden markov models," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 49, no. 6, pp. 2080–2099, 2010.

[67] L. Gutiérrez, J. Ibanez, G. Cortés, J. Ramírez, C. Benítez, V. Tenorio, and A. Isaac, "Volcano-seismic signal detection and classification processing using hidden markov models. application to san cristóbal volcano, nicaragua," in *2009 IEEE International Geoscience and Remote Sensing Symposium*, vol. 4. IEEE, 2009, pp. IV–522.

[68] T. Fjeldstad and H. Omre, "Bayesian inversion of convolved hidden markov models with applications in reservoir prediction," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 3, pp. 1957–1968, 2019.

[69] S. Hochreiter, "Recurrent neural net learning and vanishing gradient," *International Journal Of Uncertainity, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 2, pp. 107–116, 1998.

[70] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*. PMLR, 2013, pp. 1310–1318.

[71] R. Grosse, "Lecture 15: Exploding and vanishing gradients," *University of Toronto Computer Science*, 2017.

[72] A. H. Ribeiro, K. Tiels, L. A. Aguirre, and T. Schön, "Beyond exploding and vanishing gradients: analysing rnn training using attractors and smooth-

ness," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 2370–2380.

[73] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu, "Advances in optimizing recurrent networks," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 8624–8628.

[74] J. Schmidhuber and S. Hochreiter, "Long short-term memory," *Neural Comput*, vol. 9, 1997.

[75] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[76] M. Moghimihanjani and B. Vaferi, "A combined wavelet transform and recurrent neural networks scheme for identification of hydrocarbon reservoir systems from well testing signals," *Journal of Energy Resources Technology*, vol. 143, no. 1, 2021.

[77] A. Alakeely and R. N. Horne, "Simulating the behavior of reservoirs with convolutional and recurrent neural networks," *SPE Reservoir Evaluation & Engineering*, vol. 23, no. 03, pp. 0992–1005, 2020.

[78] D. Zhang, Q. Peng, J. Lin, D. Wang, X. Liu, and J. Zhuang, "Simulating reservoir operation using a recurrent neural network algorithm," *Water*, vol. 11, no. 4, p. 865, 2019.

[79] P. AN, D.-p. CAO, B.-y. ZHAO, X.-l. YANG, and M. ZHANG, "Reservoir physical parameters prediction based on lstm recurrent neural network," *Progress in Geophysics*, vol. 34, no. 5, pp. 1849–1858, 2019.

[80] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[81] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," *arXiv preprint arXiv:1412.7062*, 2014.

[82] S. Zheng, J. Lu, H. Zhao, X. Zhu, Z. Luo, Y. Wang, Y. Fu, J. Feng, T. Xiang, P. H. Torr *et al.*, "Rethinking semantic segmentation from a sequence-to-

sequence perspective with transformers," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 6881–6890.

[83] S. Minaee, Y. Y. Boykov, F. Porikli, A. J. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," *IEEE transactions on pattern analysis and machine intelligence*, 2021.

[84] C. Hazirbas, L. Ma, C. Domokos, and D. Cremers, "Fusenet: Incorporating depth into semantic segmentation via fusion-based cnn architecture," in *Asian conference on computer vision*. Springer, 2016, pp. 213–228.

[85] N. Kalchbrenner and P. Blunsom, "Recurrent continuous translation models," in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1700–1709.

[86] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in neural information processing systems*, vol. 27, 2014.

[87] J. Cho, M. K. Baskar, R. Li, M. Wiesner, S. H. Mallidi, N. Yalta, M. Karafiat, S. Watanabe, and T. Hori, "Multilingual sequence-to-sequence speech recognition: architecture, transfer learning, and language modeling," in *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2018, pp. 521–527.

[88] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[89] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: Lessons learned from the 2015 mscoco image captioning challenge," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 4, pp. 652–663, 2016.

[90] M. Shang, Z. Fu, N. Peng, Y. Feng, D. Zhao, and R. Yan, "Learning to converse with noisy data: Generation with calibration." in *IJCAI*, vol. 7, 2018.

[91] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," *Advances in neural information processing systems*, vol. 28, 2015.

[92] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates *et al.*, "Deep speech: Scaling up end-to-end speech recognition," *arXiv preprint arXiv:1412.5567*, 2014.

[93] I. Good, "What are degrees of freedom?" *The American Statistician*, vol. 27, no. 5, pp. 227–228, 1973.

[94] K. Kreutz-Delgado, J. F. Murray, B. D. Rao, K. Engan, T.-W. Lee, and T. J. Sejnowski, "Dictionary learning algorithms for sparse representation," *Neural computation*, vol. 15, no. 2, pp. 349–396, 2003.

[95] M. Gharavi-Alkhansari and T. S. Huang, "A fast orthogonal matching pursuit algorithm," in *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181)*, vol. 3. IEEE, 1998, pp. 1389–1392.

[96] S. Schaal, S. Vijayakumar, and C. Atkeson, "Local dimensionality reduction," *Advances in neural information processing systems*, vol. 10, 1997.

[97] D. L. Donoho and C. Grimes, "Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data," *Proceedings of the National Academy of Sciences*, vol. 100, no. 10, pp. 5591–5596, 2003.

[98] Y. Bengio, J.-f. Paiement, P. Vincent, O. Delalleau, N. Roux, and M. Ouimet, "Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering," *Advances in neural information processing systems*, vol. 16, 2003.

[99] L. H. Nguyen and S. Holmes, "Ten quick tips for effective dimensionality reduction," *PLoS computational biology*, vol. 15, no. 6, p. e1006907, 2019.

[100] N. D. Monnig, B. Fornberg, and F. G. Meyer, "Inverting nonlinear dimensionality reduction with scale-free radial basis function interpolation," *Applied and Computational Harmonic Analysis*, vol. 37, no. 1, pp. 162–170, 2014.

[101] L. Peng, G. Huang, X. Chen, and A. Kareem, "Simulation of multivariate nonstationary random processes: hybrid stochastic wave and proper orthogonal decomposition approach," *Journal of Engineering Mechanics*, vol. 143, no. 9, p. 04017064, 2017.

[102] X. Chen and A. Kareem, "Proper orthogonal decomposition-based modeling, analysis, and simulation of dynamic wind load effects on structures," *Journal of Engineering Mechanics*, vol. 131, no. 4, pp. 325–339, 2005.

[103] K. Willcox and J. Peraire, "Balanced model reduction via the proper orthogonal decomposition," *AIAA journal*, vol. 40, no. 11, pp. 2323–2330, 2002.

[104] J. F. Van Doren, R. Markovinović, and J.-D. Jansen, "Reduced-order optimal control of water flooding using proper orthogonal decomposition," *Computational Geosciences*, vol. 10, no. 1, pp. 137–158, 2006.

[105] A. Narasingam, P. Siddhamshetty, and J. S.-I. Kwon, "Handling spatial heterogeneity in reservoir parameters using proper orthogonal decomposition based ensemble kalman filter for model-based feedback control of hydraulic fracturing," *Industrial & Engineering Chemistry Research*, vol. 57, no. 11, pp. 3977–3989, 2018.

[106] J. He, Z. Chen, C. Zhao, X. Chen, Y. Wei, and C. Zhang, "Wave parameter inversion with coherent microwave radar using spectral proper orthogonal decomposition," *IEEE Transactions on Geoscience and Remote Sensing*, 2022.

[107] X. Li, X. Chen, B. X. Hu, and I. M. Navon, "Model reduction of a coupled numerical model using proper orthogonal decomposition," *Journal of Hydrology*, vol. 507, pp. 227–240, 2013.

[108] M. Dehghan and M. Abbaszadeh, "An upwind local radial basis functions-differential quadrature (rbf-dq) method with proper orthogonal decomposition (pod) approach for solving compressible euler equation," *Engineering Analysis with Boundary Elements*, vol. 92, pp. 244–256, 2018.

[109] H. Pourshamsaei, A. Nobakhti, and R. Jana, "Adaptive proper orthogonal decomposition for large scale reliable soil moisture estimation," *Measurement Science and Technology*, vol. 32, no. 11, p. 115026, 2021.

[110] A. Fic, R. A. Białecki, and A. J. Kassab, "Solving transient nonlinear heat conduction problems by proper orthogonal decomposition and the finite-element method," *Numerical Heat Transfer, Part B: Fundamentals*, vol. 48, no. 2, pp. 103–124, 2005.

[111] H. Barlow, "Redundancy reduction revisited," *Network: computation in neural systems*, vol. 12, no. 3, p. 241, 2001.

[112] Y. Wang, Q. Chen, C. Kang, Q. Xia, and M. Luo, "Sparse and redundant representation-based smart meter data compression and pattern extraction," *IEEE Transactions on Power Systems*, vol. 32, no. 3, pp. 2142–2151, 2016.

[113] G. Strang, "Wavelet transforms versus fourier transforms," *Bulletin of the American Mathematical Society*, vol. 28, no. 2, pp. 288–305, 1993.

[114] S. Mallat, *A Wavelet Tour of Signal Processing*, 2nd ed., ser. Wavelet Analysis Its Applications. Academic Press, 1999.

[115] M. Tan, I. W. Tsang, and L. Wang, "Matching pursuit lasso part i: Sparse recovery over big dictionary," *IEEE Transactions on Signal Processing*, vol. 63, no. 3, pp. 727–741, 2014.

[116] L. Rebollo-Neira and D. Lowe, "Optimized orthogonal matching pursuit approach," *IEEE signal processing Letters*, vol. 9, no. 4, pp. 137–140, 2002.

[117] P. Comon, "Independent component analysis, a new concept?" *Signal processing*, vol. 36, no. 3, pp. 287–314, 1994.

[118] A. Hyvarinen, "Fast ica for noisy data using gaussian moments," in *1999 IEEE international symposium on circuits and systems (ISCAS)*, vol. 5. IEEE, 1999, pp. 57–61.

[119] A. Dermoune and T. Wei, "Fastica algorithm: five criteria for the optimal choice of the nonlinearity function," *IEEE transactions on signal processing*, vol. 61, no. 8, pp. 2078–2087, 2013.

[120] J. Wang and C.-I. Chang, "Independent component analysis-based dimensionality reduction with applications in hyperspectral image analysis," *IEEE transactions on geoscience and remote sensing*, vol. 44, no. 6, pp. 1586–1600, 2006.

[121] N. Wang, B. Du, L. Zhang, and L. Zhang, "An abundance characteristic-based independent component analysis for hyperspectral unmixing," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 1, pp. 416–428, 2014.

[122] W. Xia, X. Liu, B. Wang, and L. Zhang, "Independent component analysis for blind unmixing of hyperspectral imagery with additional constraints," *IEEE transactions on geoscience and remote sensing*, vol. 49, no. 6, pp. 2165–2179, 2011.

[123] D. Lubo-Robles, "Development of independent component analysis for reservoir geomorphology and unsupervised seismic facies classification in the taranaki basin, new zealand." 2018.

[124] D. Lubo-Robles and K. J. Marfurt, "Independent component analysis for reservoir geomorphology and unsupervised seismic facies classification in the taranaki basin, new zealand," *Interpretation*, vol. 7, no. 3, pp. SE19–SE42, 2019.

[125] N. Wang, Q. M. Qin, C. Xie, L. Chen, and Y. B. Bai, "Coal-bed methane reservoir identification using the natural source super-low frequency remote sensing," in *2013 IEEE International Geoscience and Remote Sensing Symposium-IGARSS*. IEEE, 2013, pp. 4022–4025.

[126] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[127] S. Rifai, G. Mesnil, P. Vincent, X. Muller, Y. Bengio, Y. Dauphin, and X. Glorot, "Higher order contractive auto-encoder," in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2011, pp. 645–660.

[128] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[129] Y. Nesterov, "A method for unconstrained convex minimization problem with the rate of convergence o (1/kˆ 2)," in *Doklady an ussr*, vol. 269, 1983, pp. 543–547.

[130] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization." *Journal of machine learning research*, vol. 12, no. 7, 2011.

[131] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[132] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[133] T. Dozat, "Incorporating nesterov momentum into adam," 2016.

[134] M. D. Hoffman, A. Gelman *et al.*, "The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo." *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1593–1623, 2014.

[135] S. Prakoso, P. Permadi, and S. Winardhie, "Effects of pore geometry and pore structure on dry p-wave velocity," *Modern Appl Sci*, vol. 10, no. 8, pp. 117–133, 2016.

[136] S. Prakoso and M. Burhannudinnur, "Effects of pore complexity on saturated p-wave velocity and its impact in estimating critical porosity," *Arab Journal of Basic and Applied Sciences*, vol. 27, no. 1, pp. 335–343, 2020.

[137] D.-h. Han, A. Nur, and D. Morgan, "Effects of porosity and clay content on wave velocities in sandstones," *Geophysics*, vol. 51, no. 11, pp. 2093–2107, 1986.

[138] A. Nur, G. Mavko, J. Dvorkin, and D. Galmudi, "Critical porosity: A key to relating physical properties to porosity in rocks," *The Leading Edge*, vol. 17, no. 3, pp. 357–362, 1998.

[139] Y. F. Sun, "Pore structure effects on elastic wave propagation in rocks: Avo modelling," *Journal of Geophysics and Engineering*, vol. 1, no. 4, pp. 268–276, 2004.

[140] M. Kumar, M. Barak, and M. Kumari, "Reflection and refraction of plane waves at the boundary of an elastic solid and double-porosity dual-permeability materials," *Petroleum Science*, vol. 16, no. 2, pp. 298–317, 2019.

[141] A. Vashishth and M. Sharma, "Reflection and refraction of acoustic waves at poroelastic ocean bed," *Earth, planets and space*, vol. 61, no. 6, pp. 675–687, 2009.

[142] M. N. Toksöz, C. Cheng, and A. Timur, "Velocities of seismic waves in porous rocks," *Geophysics*, vol. 41, no. 4, pp. 621–645, 1976.

[143] M. Kumar and D.-h. Han, "Pore shape effect on elastic properties of carbonate rocks," in *SEG Technical Program Expanded Abstracts 2005*. Society of Exploration Geophysicists, 2005, pp. 1477–1480.

[144] J. Soete, L. M. Kleipool, H. Claes, S. Claes, H. Hamaekers, S. Kele, M. Özkul, A. Foubert, J. J. Reijmer, and R. Swennen, "Acoustic properties in travertines and their relation to porosity and pore types," *Marine and Petroleum Geology*, vol. 59, pp. 320–335, 2015.

[145] N. Bala and A. Arora, "Effect of pore connectivity on reflection amplitudes of an inhomogeneous wave in a composite porous solid saturated by two immiscible fluids," *Journal of Earth System Science*, vol. 127, no. 4, pp. 1–19, 2018.

[146] M. Pang, J. Ba, J. M. Carcione, S. Picotti, J. Zhou, and R. Jiang, "Estimation of porosity and fluid saturation in carbonates from rock-physics

templates based on seismic q," *Geophysics*, vol. 84, no. 6, pp. M25–M36, 2019.

[147] C. H. Chapman, J. W. Hobro, and J. O. Robertsson, "Correcting an acoustic wavefield for elastic effects," *Geophysical Journal International*, vol. 197, no. 2, pp. 1196–1214, 2014.

[148] K. Virta, "Numerics of elastic and acoustic wave motion," Ph.D. dissertation, Acta Universitatis Upsaliensis, 2016.

[149] Y. Rao and Y. Wang, "Seismic waveform simulation for models with fluctuating interfaces," *Scientific Reports*, vol. 8, no. 1, pp. 1–8, 2018.

[150] P. Sarkar, A. Kumar, K. H. Singh, R. Ghosh, and T. N. Singh, "Pore system, microstructure and porosity characterization of gondwana shale of eastern india using laboratory experiment and watershed image segmentation algorithm," *Marine and Petroleum Geology*, vol. 94, pp. 246–260, 2018.

[151] A. Buades, B. Coll, and J.-M. Morel, "A review of image denoising algorithms, with a new one," *Multiscale modeling & simulation*, vol. 4, no. 2, pp. 490–530, 2005.

[152] R. Herzog, "Computer control and the scanning electron microscope," in *Proceedings of 7th Annual Conference of Scanning Electron Microscopy Symposium*. IIT Research Institute, 1974, pp. 175–182.

[153] M. Kiani, K. S. Sim, M. E. Nia, and C. P. Tso, "Signal-to-noise ratio enhancement on sem images using a cubic spline interpolation with savitzky–golay filters and weighted least squares error," *Journal of microscopy*, vol. 258, no. 2, pp. 140–150, 2015.

[154] K. Sim, K. Law, and C. Tso, "Mixed lagrange time delay estimation autoregressive wiener filter application for real-time sem image enhancement," *Microscopy research and technique*, vol. 70, no. 11, pp. 919–927, 2007.

[155] K. Sim, N. Kamel, and H. Chuah, "Autoregressive wiener filtering in a scanning electron microscopy imaging system," *Scanning*, vol. 27, no. 3, pp. 147–153, 2005.

[156] E. Oho, A. Ogihara, and K. Kanaya, "A new non-linear pseudo-laplacian filter for enhancement of secondary electron images," *Journal of Microscopy*, vol. 159, no. 1, pp. 33–41, 1990.

[157] A. Tavanaei and S. Salehi, "Pore, throat, and grain detection for rock sem images using digitalwatershed image segmentation algorithm," *Journal of Porous Media*, vol. 18, no. 5, 2015.

[158] S. V. Vaseghi, *Advanced digital signal processing and noise reduction*. John Wiley & Sons, 2008.

[159] P. Jagatheeswari, S. Suresh Kumar, and M. Rajaram, "Contrast enhancement for medical images based on histogram equalization followed by median filter," 2009.

[160] J. Sauvola and M. Pietikäinen, "Adaptive document image binarization," *Pattern recognition*, vol. 33, no. 2, pp. 225–236, 2000.

[161] I. Bankman, *Handbook of medical image processing and analysis*. Elsevier, 2008.

[162] D. Bradley and G. Roth, "Adaptive thresholding using the integral image," *Journal of graphics tools*, vol. 12, no. 2, pp. 13–21, 2007.

[163] S. Uchida, "Image processing and recognition for biological images," *Development, growth & differentiation*, vol. 55, no. 4, pp. 523–549, 2013.

[164] W. Niblack, *An introduction to digital image processing*. Strandberg Publishing Company, 1985.

[165] N. Salman, "Image segmentation based on watershed and edge detection techniques." *Int. Arab J. Inf. Technol.*, vol. 3, no. 2, pp. 104–110, 2006.

[166] F. Meyer and S. Beucher, "Morphological segmentation," *Journal of visual communication and image representation*, vol. 1, no. 1, pp. 21–46, 1990.

[167] T.-C. Lee, R. L. Kashyap, and C.-N. Chu, "Building skeleton models via 3-d medial surface axis thinning algorithms," *CVGIP: Graphical Models and Image Processing*, vol. 56, no. 6, pp. 462–478, 1994.

[168] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the royal statistical society. series c (applied statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[169] G. P. Meyer, J. Charland, D. Hegde, A. Laddha, and C. Vallespi-Gonzalez, "Sensor fusion for joint 3d object detection and semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[170] J. Virieux, H. Calandra, and R.-É. Plessix, "A review of the spectral, pseudo-spectral, finite-difference and finite-element modelling techniques for geophysical imaging," *Geophysical Prospecting*, vol. 59, no. Modelling Methods for Geophysical Imaging: Trends and Perspectives, pp. 794–813, 2011.

[171] K. Wang, E. Teoh, J. Jaros, and B. E. Treeby, "Modelling nonlinear ultrasound propagation in absorbing media using the k-wave toolbox: experimental validation," in *2012 IEEE International Ultrasonics Symposium*. IEEE, 2012, pp. 523–526.

[172] S. I. Aanonsen, T. Barkve, J. N. Tjo/tta, and S. Tjo/tta, "Distortion and harmonic generation in the nearfield of a finite amplitude sound beam," *The Journal of the Acoustical Society of America*, vol. 75, no. 3, pp. 749–768, 1984.

[173] A. Hanyga and M. Seredyńska, "Power-law attenuation in acoustic and isotropic anelastic media," *Geophysical Journal International*, vol. 155, no. 3, pp. 830–838, 2003.

[174] S. P. Näsholm and S. Holm, "Linking multiple relaxation, power-law attenuation, and fractional wave equations," *The Journal of the Acoustical Society of America*, vol. 130, no. 5, pp. 3038–3045, 2011.

[175] B. E. Treeby and B. T. Cox, "k-wave: Matlab toolbox for the simulation and reconstruction of photoacoustic wave fields," *Journal of biomedical optics*, vol. 15, no. 2, p. 021314, 2010.

[176] B. E. Treeby, J. Budisky, E. S. Wise, J. Jaros, and B. Cox, "Rapid calculation of acoustic fields from arbitrary continuous-wave sources," *The Journal of the Acoustical Society of America*, vol. 143, no. 1, pp. 529–537, 2018.

[177] W. Cai, W. Chen, J. Fang, and S. Holm, "A survey on fractional derivative modeling of power-law frequency-dependent viscous dissipative and scattering attenuation in acoustic wave propagation," *Applied Mechanics Reviews*, vol. 70, no. 3, 2018.

[178] Y. Guo, Y. Liu, T. Georgiou, and M. S. Lew, "A review of semantic segmentation using deep neural networks," *International journal of multimedia information retrieval*, vol. 7, no. 2, pp. 87–93, 2018.

[179] R. A. Wilson and F. C. Keil, *The MIT Encyclopedia of the cognitive sciences (MITECS)*. MIT press, 2001.

[180] P. Pinheiro and R. Collobert, "Recurrent convolutional neural networks for scene labeling," in *International conference on machine learning*. PMLR, 2014, pp. 82–90.

[181] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[182] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1520–1528.

[183] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[184] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr, "Conditional random fields as recurrent neural networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1529–1537.

[185] S. Lin, C. Shen, A. Van Den Hengel, and I. Reid, "Efficient piecewise training of deep structured models for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3194–3203.

[186] X. Liang, X. Shen, J. Feng, L. Lin, and S. Yan, "Semantic object parsing with graph lstm," in *European Conference on Computer Vision*. Springer, 2016, pp. 125–143.

[187] B. Shuai, Z. Zuo, B. Wang, and G. Wang, "Dag-recurrent neural networks for scene labeling," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3620–3629.

[188] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[189] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2881–2890.

[190] L. C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 801–818.

[191] P. Luo, G. Wang, L. Lin, and X. Wang, "Deep dual learning for semantic image segmentation," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2718–2726.

[192] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[193] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun, "Large kernel matters–improve semantic segmentation by global convolutional network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4353–4361.

[194] J. Fu, J. Liu, Y. Wang, J. Zhou, C. Wang, and H. Lu, "Stacked deconvolutional network for semantic segmentation," *IEEE Transactions on Image Processing*, 2019.

[195] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "Learning a discriminative feature network for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1857–1866.

[196] D. Lin, Y. Ji, D. Lischinski, D. Cohen-Or, and H. Huang, "Multi-scale context intertwining for semantic segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 603–619.

[197] H. Shi, H. Li, F. Meng, Q. Wu, L. Xu, and K. N. Ngan, "Hierarchical parsing net: Semantic scene parsing from global scene to objects," *IEEE Transactions on Multimedia*, vol. 20, no. 10, pp. 2670–2682, 2018.

[198] H. Zhang, K. Dana, J. Shi, Z. Zhang, X. Wang, A. Tyagi, and A. Agrawal, "Context encoding for semantic segmentation," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 7151–7160.

[199] H. Zhao, Y. Zhang, S. Liu, J. Shi, C. C. Loy, D. Lin, and J. Jia, "Psanet: Point-wise spatial attention network for scene parsing," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 267–283.

[200] Z. Xie, J. Chen, and B. Peng, "Point clouds learning with attention-based graph convolution networks," *Neurocomputing*, vol. 402, pp. 245–255, 2020.

[201] Y. Ma, Y. , H. Liu, Y. Lei, and G. Wen, "Global context reasoning for semantic segmentation of 3d point clouds," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, pp. 2931–2940.

[202] H. Lei, N. Akhtar, and A. Mian, "Octree guided cnn with spherical kernels for 3d point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9631–9640.

[203] H. Li, M. Muller, A. Thabet, and B. Ghanem, "Deepgcns: Can gcns go as deep as cnns?" in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 9267–9276.

[204] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[205] L. Landrieu and M. Simonovsky, "Large-scale point cloud semantic segmentation with superpoint graphs," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4558–4567.

[206] X. Wang, S. Liu, X. Shen, C. Shen, and J. Jia, "Associatively segmenting instances and semantics in point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4096–4105.

[207] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "Pointcnn: Convolution on x-transformed points," *Advances in neural information processing systems*, vol. 31, 2018.

[208] F. Groh, P. Wieschollek, and H. Lensch, "Flex-convolution," in *Asian Conference on Computer Vision*. Springer, 2018, pp. 105–122.

[209] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "Kpconv: Flexible and deformable convolution for point clouds," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6411–6420.

[210] B.-S. Hua, M.-K. Tran, and S.-K. Yeung, "Pointwise convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 984–993.

[211] F. Engelmann, T. Kontogianni, and B. Leibe, "Dilated point convolutions: On the receptive field size of point convolutions on 3d point clouds," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 9463–9469.

[212] Q. Huang, W. Wang, and U. Neumann, "Recurrent slice networks for 3d segmentation of point clouds," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2626–2635.

[213] X. Ye, J. Li, H. Huang, L. Du, and X. Zhang, "3d recurrent neural networks with context fusion for point cloud semantic segmentation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 403–417.

[214] F. Engelmann, M. Bokeloh, A. Fathi, B. Leibe, and M. Nießner, "3d-mpa: Multi-proposal aggregation for 3d semantic instance segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 9031–9040.

[215] F. Engelmann, T. Kontogianni, A. Hermans, and B. Leibe, "Exploring spatial context for 3d semantic segmentation of point clouds," in *Proceedings of the IEEE international conference on computer vision workshops*, 2017, pp. 716–724.

[216] Y. Cao, C. Shen, and H. T. Shen, "Exploiting depth from single monocular images for object detection and semantic segmentation," *IEEE Transactions on Image Processing*, vol. 26, no. 2, pp. 836–846, 2016.

[217] A. Mousavian, H. Pirsiavash, and J. Košecká, "Joint semantic segmentation and depth estimation with deep convolutional networks," in *2016 Fourth International Conference on 3D Vision (3DV)*. IEEE, 2016, pp. 611–619.

[218] Y. Cheng, R. Cai, Z. Li, X. Zhao, and K. Huang, "Locality-sensitive deconvolution networks with gated fusion for rgb-d indoor semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3029–3037.

[219] J. Jiang, Z. Zhang, Y. Huang, and L. Zheng, "Incorporating depth into both cnn and crf for indoor semantic segmentation," in *2017 8th IEEE Interna-*

tional Conference on Software Engineering and Service Science (ICSESS). IEEE, 2017, pp. 525–530.

[220] D. Lin, G. Chen, D. Cohen-Or, P.-A. Heng, and H. Huang, "Cascaded feature network for semantic segmentation of rgb-d images," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1311–1319.

[221] P. Wang, X. Shen, Z. Lin, S. Cohen, B. Price, and A. L. Yuille, "Towards unified depth and semantic prediction from a single image," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2800–2809.

[222] J. Liu, Y. Wang, Y. Li, J. Fu, J. Li, and H. Lu, "Collaborative deconvolutional neural networks for joint depth estimation and semantic segmentation," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 11, pp. 5655–5666, 2018.

[223] N. Höft, H. Schulz, and S. Behnke, "Fast semantic segmentation of rgb-d scenes with gpu-accelerated deep neural networks," in *Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz)*. Springer, 2014, pp. 80–85.

[224] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik, "Learning rich features from rgb-d images for object detection and segmentation," in *European conference on computer vision*. Springer, 2014, pp. 345–360.

[225] H. Liu, W. Wu, X. Wang, and Y. Qian, "Rgb-d joint modelling with scene geometric information for indoor semantic segmentation," *Multimedia Tools and Applications*, vol. 77, no. 17, pp. 22 475–22 488, 2018.

[226] M. Jiang, Y. Wu, T. Zhao, Z. Zhao, and C. Lu, "Pointsift: A sift-like network module for 3d point cloud semantic segmentation," *arXiv preprint arXiv:1807.00652*, 2018.

[227] W. Wang and U. Neumann, "Depth-aware cnn for rgb-d segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 135–150.

[228] C. Couprie, C. Farabet, L. Najman, and Y. LeCun, "Indoor semantic segmentation using depth information," *arXiv preprint arXiv:1301.3572*, 2013.

[229] A. Raj, D. Maturana, and S. Scherer, "Multi-scale convolutional architecture for semantic segmentation," *Robotics Institute, Carnegie Mellon University, Tech. Rep. CMU-RITR-15-21*, 2015.

[230] J. Wang, Z. Wang, D. Tao, S. See, and G. Wang, "Learning common and specific features for rgb-d semantic segmentation with deconvolutional networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 664–679.

[231] X. Qi, R. Liao, J. Jia, S. Fidler, and R. Urtasun, "3d graph neural networks for rgbd semantic segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5199–5208.

[232] Z. Li, Y. Gan, X. Liang, Y. Yu, H. Cheng, and L. Lin, "Lstm-cf: Unifying context modeling and fusion with lstms for rgb-d scene labeling," in *European conference on computer vision*. Springer, 2016, pp. 541–557.

[233] H. Fan, X. Mei, D. Prokhorov, and H. Ling, "Rgb-d scene labeling with multimodal recurrent neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 9–17.

[234] J. Huang and S. You, "Point cloud labeling using 3d convolutional neural network," in *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2016, pp. 2670–2675.

[235] L. Tchapmi, C. Choy, I. Armeni, J. Gwak, and S. Savarese, "Segcloud: Semantic segmentation of 3d point clouds," in *2017 international conference on 3D vision (3DV)*. IEEE, 2017, pp. 537–547.

[236] H.-Y. Meng, L. Gao, Y.-K. Lai, and D. Manocha, "Vv-net: Voxel vae net with group convolutions for point cloud segmentation," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 8500–8508.

[237] F. Liu, S. Li, L. Zhang, C. Zhou, R. Ye, Y. Wang, and J. Lu, "3dcnn-dqn-rnn: A deep reinforcement learning framework for semantic parsing of large-scale 3d point clouds," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5678–5687.

[238] D. Rethage, J. Wald, J. Sturm, N. Navab, and F. Tombari, "Fully-convolutional point networks for large-scale point clouds," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 596–611.

[239] A. Dai and M. Nießner, "3dmv: Joint 3d-multi-view prediction for 3d semantic scene segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 452–468.

[240] G. Riegler, A. Osman Ulusoy, and A. Geiger, "Octnet: Learning deep 3d representations at high resolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3577–3586.

[241] B. Graham, M. Engelcke, and L. Van Der Maaten, "3d semantic segmentation with submanifold sparse convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9224–9232.

[242] L. Yi, W. Zhao, H. Wang, M. Sung, and L. J. Guibas, "Gspn: Generative shape proposal network for 3d instance segmentation in point cloud," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3947–3956.

[243] J. Hou, A. Dai, and M. Nießner, "3d-sis: 3d semantic instance segmentation of rgb-d scans," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4421–4430.

[244] B. Yang, J. Wang, R. Clark, Q. Hu, S. Wang, A. Markham, and N. Trigoni, "Learning object bounding boxes for 3d instance segmentation on point clouds," *Advances in neural information processing systems*, vol. 32, 2019.

[245] W. Wang, R. Yu, Q. Huang, and U. Neumann, "Sgpn: Similarity group proposal network for 3d point cloud instance segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2569–2578.

[246] H. Jiang, F. Yan, J. Cai, J. Zheng, and J. Xiao, "End-to-end 3d point cloud instance segmentation without detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 796–12 805.

[247] C. Liu and Y. Furukawa, "Masc: Multi-scale affinity with sparse convolution for 3d instance segmentation," *arXiv preprint arXiv:1902.04478*, 2019.

[248] J. Lahoud, B. Ghanem, M. Pollefeys, and M. R. Oswald, "3d instance segmentation via multi-task metric learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9256–9266.

[249] L. Jiang, H. Zhao, S. Shi, S. Liu, C.-W. Fu, and J. Jia, "Pointgroup: Dual-set point grouping for 3d instance segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4867–4876.

[250] C. Elich, F. Engelmann, T. Kontogianni, and B. Leibe, "3d bird's-eye-view instance segmentation," in *German Conference on Pattern Recognition*. Springer, 2019, pp. 48–61.

[251] Q.-H. Pham, T. Nguyen, B.-S. Hua, G. Roig, and S.-K. Yeung, "Jsis3d: Joint semantic-instance segmentation of 3d point clouds with multi-task pointwise networks and multi-value conditional random fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8827–8836.

[252] Z. Liang, M. Yang, H. Li, and C. Wang, "3d instance embedding learning with a structure-aware loss function for point cloud segmentation," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4915–4922, 2020.

[253] L. Han, T. Zheng, L. Xu, and L. Fang, "Occuseg: Occupancy-aware 3d instance segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2940–2949.

[254] E. Kalogerakis, M. Averkiou, S. Maji, and S. Chaudhuri, "3d shape segmentation with projective convolutional networks," in *proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3779–3788.

[255] Z. Wang and F. Lu, "Voxsegnet: Volumetric cnns for semantic part segmentation of 3d shapes," *IEEE transactions on visualization and computer graphics*, vol. 26, no. 9, pp. 2919–2930, 2019.

[256] T. Le and Y. Duan, "Pointgrid: A deep network for 3d shape understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9204–9214.

[257] Y. Song, X. Chen, J. Li, and Q. Zhao, "Embedding 3d geometric features for rigid object part segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 580–588.

[258] H. Xu, M. Dong, and Z. Zhong, "Directionally convolutional networks for 3d shape segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2698–2707.

[259] R. Hanocka, A. Hertz, N. Fish, R. Giryes, S. Fleishman, and D. Cohen-Or, "Meshcnn: a network with an edge," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 1–12, 2019.

[260] F. Yu, K. Liu, Y. Zhang, C. Zhu, and K. Xu, "Partnet: A recursive part decomposition network for fine-grained and hierarchical shape segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9491–9500.

[261] Y. Shen, C. Feng, Y. Yang, and D. Tian, "Mining point cloud local structures by kernel correlation and graph pooling," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4548–4557.

[262] P. Wang, Y. Gan, P. Shui, F. Yu, Y. Zhang, S. Chen, and Z. Sun, "3d shape segmentation via shape fully convolutional networks," *Computers & Graphics*, vol. 76, pp. 182–192, 2018.

[263] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao, "Spidercnn: Deep learning on point sets with parameterized convolutional filters," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 87–102.

[264] N. Verma, E. Boyer, and J. Verbeek, "Feastnet: Feature-steered graph convolutions for 3d shape analysis," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2598–2606.

[265] R. Klokov and V. Lempitsky, "Escape from cells: Deep kd-networks for the recognition of 3d point cloud models," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 863–872.

[266] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong, "O-cnn: Octree-based convolutional neural networks for 3d shape analysis," *ACM Transactions On Graphics (TOG)*, vol. 36, no. 4, pp. 1–11, 2017.

[267] Y. Zhao, T. Birdal, H. Deng, and F. Tombari, "3d point capsule networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1009–1018.

[268] J. Li, B. M. Chen, and G. H. Lee, "So-net: Self-organizing network for point cloud analysis," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9397–9406.

[269] A. Kaestner, E. Lehmann, and M. Stampanoni, "Imaging and image processing in porous media research," *Advances in Water Resources*, vol. 31, no. 9, pp. 1174–1187, 2008.

[270] R. B. Latimer, R. Davidson, and P. Van Riel, "An interpreter's guide to understanding and working with seismic-derived acoustic impedance data," *The leading edge*, vol. 19, no. 3, pp. 242–256, 2000.

[271] N. Siddique, S. Paheding, C. P. Elkin, and V. Devabhaktuni, "U-net and its variants for medical image segmentation: A review of theory and applications," *IEEE Access*, 2021.

[272] R. R. Shamir, Y. Duchin, J. Kim, G. Sapiro, and N. Harel, "Continuous dice coefficient: a method for evaluating probabilistic segmentations," *arXiv preprint arXiv:1906.11031*, 2019.

[273] S. Jha, R. Kumar, I. Priyadarshini, F. Smarandache, H. V. Long *et al.*, "Neutrosophic image segmentation with dice coefficients," *Measurement*, vol. 134, pp. 762–772, 2019.

[274] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," in *2016 fourth international conference on 3D vision (3DV)*. IEEE, 2016, pp. 565–571.

[275] R. Zhao, B. Qian, X. Zhang, Y. Li, R. Wei, Y. Liu, and Y. Pan, "Rethinking dice loss for medical image segmentation," in *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2020, pp. 851–860.

[276] I. N. Bankman and S. Morcovescu, "Handbook of medical imaging. processing and analysis," 2002.

[277] W. Fellenius, "Calculation of the stability of earth dams trans. 2nd congr," *Large Dams*, vol. 1, 1936.

[278] A. W. Bishop, "The use of the slip circle in the stability analysis of slopes," *Geotechnique*, vol. 5, no. 1, pp. 7–17, 1955.

[279] J. Lowe, "Stability of earth dams upon drawdown," in *Proc. 1st Pan American Conference on Soil Mechanics and Foundation Engineering, Mexico City, 1960*, 1960.

[280] N. Janbu, "Slope stability computation, soil mechanics and foundation engineering report," *The Technical University of Norway, Trondheim*, 1968.

[281] N. Morgenstern and V. E. Price, "The analysis of the stability of general slip surfaces," *Geotechnique*, vol. 15, no. 1, pp. 79–93, 1965.

[282] E. Spencer, "A method of analysis of the stability of embankments assuming parallel inter-slice forces," *Geotechnique*, vol. 17, no. 1, pp. 11–26, 1967.

[283] D. W. Taylor, "Stability of earth slopes," *J. Boston Soc. Civil Engineers*, vol. 24, no. 3, pp. 197–247, 1937.

[284] J. H. Hunter and R. L. Schuster, "Chart solutions for analysis of earth slopes," *Highway Research Record*, no. 345, 1971.

[285] B. F. Cousins, "Stability charts for simple earth slopes," *Journal of the Geotechnical Engineering Division*, vol. 104, no. 2, pp. 267–279, 1978.

[286] C. A. Coulomb, "Essai sur une application des regles de maximis et minimis a quelques problemes de statique relatifs a l'architecture (essay on maximums and minimums of rules to some static problems relating to architecture)," 1973.

[287] E. Hoek, "Estimating mohr-coulomb friction and cohesion values from the hoek-brown failure criterion," in *International Journal of Rock Mechanics and Mining Sciences & Geomechanics Abstracts*, vol. 27, no. 3. Pergamon, 1990, pp. 227–229.

[288] D. C. Drucker and W. Prager, "Soil mechanics and plastic analysis or limit design," *Quarterly of applied mathematics*, vol. 10, no. 2, pp. 157–165, 1952.

[289] P. V. Lade and J. M. Duncan, "Elastoplastic stress-strain theory for cohesionless soil," *Journal of the Geotechnical Engineering Division*, vol. 101, no. 10, pp. 1037–1053, 1975.

[290] P. V. Lade and H. M. Musante, "Three-dimensional behavior of remolded clay," *Journal of the Geotechnical Engineering Division*, vol. 104, no. 2, pp. 193–209, 1978.

[291] I. Smith and R. Hobbs, "Finite element analysis of centrifuged and built-up slopes," *Geotechnique*, vol. 24, no. 4, pp. 531–559, 1974.

[292] O. Zeinkiewicz, C. Humpheson, and R. Lewis, "Associated and non-associated visco-plasticity in soils mechanics," *Journal of Geotechnique*, vol. 25, no. 5, pp. 671–689, 1975.

[293] D. Griffiths and P. Lane, "Slope stability analysis by finite elements," *Geotechnique*, vol. 49, no. 3, pp. 387–403, 1999.

[294] D. Potts, G. Dounias, and P. Vaughan, "Finite element analysis of progressive failure of carsington embankment," in *Selected papers on geotechnical engineering by PR Vaughan*. Thomas Telford Publishing, 2009, pp. 212–234.

[295] T. Matsui and K.-C. San, "Finite element slope stability analysis by shear strength reduction technique," *Soils and foundations*, vol. 32, no. 1, pp. 59–70, 1992.

[296] B. Jeremić, "Finite element methods for 3d slope stability analysis," in *Slope Stability 2000*, 2000, pp. 224–238.

[297] P. Lane and D. Griffiths, "Assessment of stability of slopes under drawdown conditions," *Journal of geotechnical and geoenvironmental engineering*, vol. 126, no. 5, pp. 443–450, 2000.

[298] J. Lechman and D. Griffiths, "Analysis of the progression of failure of earth slopes by finite elements," in *Slope Stability 2000*, 2000, pp. 250–265.

[299] A. Sainak, "Application of three-dimensional finite element method in parametric and geometric studies of slope stability analysis," in *Advances in geotechnical engineering: The Skempton conference: Proceedings of a three day conference on advances in geotechnical engineering, organised by the Institution of Civil Engineers and held at the Royal Geographical Society, London, UK, on 29–31 March 2004*. Thomas Telford Publishing, 2004, pp. 933–942.

[300] L. Zhang, J. Zhang, L. Zhang, and W. H. Tang, "Stability analysis of rainfall-induced slope failure: a review," *Proceedings of the Institution of Civil Engineers-Geotechnical Engineering*, vol. 164, no. 5, pp. 299–316, 2011.

[301] D. Griffiths and R. Marquez, "Three-dimensional slope stability analysis by elasto-plastic finite elements," *Geotechnique*, vol. 57, no. 6, pp. 537–546, 2007.

[302] X. Li, "Finite element analysis of slope stability using a nonlinear failure criterion," *Computers and Geotechnics*, vol. 34, no. 3, pp. 127–136, 2007.

[303] P. J. Schmid, "Dynamic mode decomposition of numerical and experimental data," *J Fluid Mech*, vol. 656, 2010.

[304] J. Tu, C. Rowley, D. Luchtenburg, S. Brunton, and J. Kutz, "On dynamic mode decomposition: theory and applications arxiv preprint arxiv:13120041," 2013. [Online]. Available: http://arxiv.org/abs/13120041

[305] Z. Bai, E. Kaiser, J. L. Proctor, J. N. Kutz, and S. L. Brunton, "Dynamic mode decomposition for compressive system identification," *AIAA J*, vol. 58, 2020.

[306] M. Liu, L. Tan, and S. Cao, "Dynamic mode decomposition of cavitating flow around ale 15 hydrofoil," *Renew Energy*, vol. 139, 2019.

[307] C. Pan, J. Wang, and M. Sun, "Dynamics of an unsteady stagnation vortical flow via dynamic mode decomposition analysis," *Exp Fluids*, vol. 58, 2017.

[308] D. F. Gomez, F. D. Lagor, P. B. Kirk, A. H. Lind, A. R. Jones, and D. A. Paley, "Data-driven estimation of the unsteady flowfield near an actuated airfoil," *J Guid Control Dyn*, vol. 42, 2019.

[309] B. W. Brunton, L. A. Johnson, J. G. Ojemann, and J. N. Kutz, "Extracting spatial-temporal coherent patterns in large-scale neural recordings using dynamic mode decomposition," *J Neurosci Methods*, vol. 258, 2016.

[310] G. Berkooz, P. Holmes, and J. L. Lumley, "The proper orthogonal decomposition in the analysis of turbulent flows," *Annual Review of Fluid Mechanics*, vol. 25, 1993.

[311] J. Du, J. Zhu, Z. Luo, and I. Navon, "An optimizing finite difference scheme based on proper orthogonal decomposition for cvd equations," *Int J Numer Methods Biomed Eng*, vol. 27, 2011.

[312] Z. Luo, J. Chen, J. Zhu, R. Wang, and I. Navon, "An optimizing reduced order fds for the tropical pacific ocean reduced gravity model," *Int J Numer Methods Fluids*, vol. 55, 2007.

[313] Z. Luo, J. Zhu, R. Wang, and I. M. Navon, "Proper orthogonal decomposition approach and error estimation of mixed finite element methods for the tropical pacific ocean reduced gravity model," *Comput Methods Appl Mech Eng*, vol. 196, 2007.

[314] Z. D. Luo, W. a. n. g. RW, and J. Zhu, "Finite difference scheme based on proper orthogonal decomposition for the nonstationary navier-stokes equations," *Sci China Ser A Math*, vol. 50, 2007.

[315] Y. Cao, J. Zhu, Z. Luo, and I. M. Navon, "Reduced-order modeling of the upper tropical pacific ocean model using proper orthogonal decomposition," *Comput Math Appl*, vol. 52, 2006.

[316] Y. Cao, J. Zhu, I. M. Navon, and Z. Luo, "A reduced-order approach to four-dimensional variational data assimilation using proper orthogonal decomposition," *Int J Numer Methods Fluids*, vol. 53, 2007.

[317] J. N. Kani and A. H. Elsheikh, "Reduced-order modeling of subsurface multi-phase flow models using deep residual recurrent neural networks," *Transp Porous Media*, vol. 126, 2019.

[318] A. Mohan and D. Gaitonde, "A deep learning based approach to reduced order modeling for turbulent flow control using lstm neural networks. arxiv preprint arxiv:180409269," 2018. [Online]. Available: http://arxiv.org/abs/180409269

[319] O. San, R. Maulik, and M. Ahmed, "An artificial neural network framework for reduced order modeling of transient flows," *Commun Nonlinear Sci Numer Simul*, vol. 77, 2019.

[320] S. Fresca, A. Manzoni, L. Dedè, and A. Quarteroni, "Deep learning-based reduced order models in cardiac electrophysiology," *PLoS One*, vol. 15, 2020.

[321] R. Halder, M. Damodaran, and B. Khoo, "Deep learning based reduced order model for airfoil-gust and aeroelastic interaction," *AIAA J*, vol. 58, 2020.

[322] T. Daniel, F. Casenave, N. Akkari, and D. Ryckelynck, "Model order reduction assisted by deep neural networks (rom-net)," *Adv Model Simul Eng Sci*, vol. 7, 2020.

[323] M. Wang, S. W. Cheung, W. T. Leung, E. T. Chung, Y. Efendiev, and M. Wheeler, "Reduced-order deep learning for flow dynamics. the interplay between deep learning and model reduction," *J Comput Phys*, vol. 401, 2020.

[324] A. Gu, C. Gulcehre, T. Paine, M. Hoffman, and R. Pascanu, "Improving the gating mechanism of recurrent neural networks. in: International conference on machine learning, pmlr," pp. pp 3800–3809, 2020.

[325] K. Terzaghi, "&die berechnung der durchlassigkeitszi! er des tones aus dem verlauf der hydrodynamischen spannungsersceinungen', originally published in 1923 and reprinted in from [1]heory to practice in soil mechanics," 1960.

[326] A. Verruijt, "Theory and problems of poroelasticity," *Delft University of Technology*, vol. 71, 2013.

[327] K. Terzaghi and R. B. Peck, "Soil mechanics," *Engineering Practice. John Wiley and Sons, Inc., New York*, 1948.

[328] H. Mounir, A. Nizar, L. Borhen, A. Benamara, and D. Deneux, "Fem simulation based on cad model simplification: a comparison study between the hybrid method and the technique using a removing details," in *Design and modeling of mechanical systems*. Springer, 2013, pp. 587–596.

[329] R. Jain, R. Kasturi, B. G. Schunck *et al.*, *Machine vision*. McGraw-hill New York, 1995, vol. 5.

[330] C. J. Permann, D. R. Gaston, D. Andrš, R. W. Carlsen, F. Kong, A. D. Lindsay, J. M. Miller, J. W. Peterson, A. E. Slaughter, R. H. Stogner, and R. C. Martineau, "MOOSE: Enabling massively parallel multiphysics simulation," *SoftwareX*, vol. 11, p. 100430, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2352711019302973

[331] B. D. Ripley, *Pattern recognition and neural networks*. Cambridge university press, 2007.

[332] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. , Z. Wang, N. Trigoni, and A. Markham, "Randla-net: Efficient semantic segmentation of large-scale point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 108–11 117.

[333] H. Eivazi, H. Veisi, M. H. Naderi, and V. Esfahanian, "Deep neural networks for nonlinear model order reduction of unsteady flows," *Phys Fluids*, vol. 32, 2020.

[334] C. Tong, *Psuade user's manual*. Livermore: Lawrence Livermore National Laboratory, 2005.

[335] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[336] Y. Bengio, "On the challenge of learning complex functions," *Progress in Brain Research*, vol. 165, pp. 521–534, 2007.

[337] G. Mavko, T. Mukerji, and J. Dvorkin, *The rock physics handbook*. Cambridge university press, 2020.

List of Publications

Journal(s)

- **Kumar, A.**, Hu, R., Walsh, S.D.C. (2022), 'Development of Reduced Order Hydro-mechanical Models of Fractured Media', *Rock Mechanics and Rock Engineering* **55(1)**, 235-248

- **Kumar, A.**, Shrivastava, R.K., Singh, K.H. (2020), 'Bayesian Inference of Material Properties in Disordered Media using Sound Characteristics', *Europhysics Letters* **129(2)**, 24001

- Tripathy, A., **Kumar, A.**, Srinivasan, V., Singh, K.H., Singh T.N. (2019), 'Fractal Analysis and Spatial Disposition of Porosity in Major Indian Gas Shales using Low-Pressure Nitrogen Adsorption and Advanced Image Segmentation', *Journal of Natural Gas Science and Engineering* **72**, 103009

- Sarkar, P., **Kumar, A.**, Singh K.H., Ghosh, R., Singh T.N. (2018), 'Pore System, Microstructure and Porosity Characterization of Gondwana Shale of Eastern India using Laboratory Experiment and Watershed Image Segmentation Algorithm', *Marine and Petroleum Geology* **94**, 246-260

- Gautam, P.K., Dwivedi, R., Kumar, A. **Kumar, A.**, Verma, A.K., Singh, K.H., Singh, T.N. (2020), 'Damage Characteristics of Jalore Granitic Rocks after Thermal Cycling Effect for Nuclear Waste Repository', *Rock Mechanics and Rock Engineering* **54**, 235-254

Proceedings / Book Chapter(s)

- **Kumar, A.**, Shrivastava, R.K., Singh, K.H., (2018), 'Bayesian Modelling for Determining Material Properties', *International Conference on Recent Innovations in Electrical, Electronics & Communication Engineering (ICRIEECE)*, 1557-1563

- Singh, K.H., **Kumar, A.**, Pandit, S., Soni, A., (2020), 'Partitioning of Porosity for Carbonate Reservoirs using Differential Effective Medium Models', *Petro-physics and Rock Physics of Carbonate Reservoirs* **1**, 129-145

- Sharma, M., Singh, K.H., Pandit, S., **Kumar, A.**, Soni, A., (2020), 'Petrophysical Modelling of Carbonate Reservoir from Bombay Offshore Basin', *Petro-physics and Rock Physics of Carbonate Reservoirs* **1**, 55-73

Articles Communicated

- Hu, R., **Kumar, A.**, Yellishetty, M., Walsh, S.D.C. (2022), 'A Bootstrap Strategy to Train, Validate and Test Reduced Order Models of Coupled Geomechanical Processes', *Computers and Geosciences*

- **Kumar, A.**, Singh, K.H., Yellishetty, M., Singh, T.N., 2022, 'A Deep Learning Approach to Pore Network Inference in Sedimentary Rock Core Samples', *Journal of Petroleum Science and Engineering*

Articles Under Preparation

- **Kumar, A.**, Yellishetty, M., Singh, K.H., Singh, T.N., 2022, 'Development of a Reduced Order Model for fast Slope Stability Simulation', *Computers and Geosciences*

- **Kumar, A.**, Yellishetty, M., Singh, K.H., Singh, T.N., 2022, 'Performing quick Full-Order pseudo Simulations within Heterogeneous Media', *Rock Mechanics and Rock Engineering*

- **Kumar, A.**, Singh, K.H., Yellishetty, M., Singh, T.N., 2022, 'Towards the Density Estimation of complex Geomechanical Processes', *Information Sciences*

Curriculum Vitae

The author obtained his Integrated M.Tech in Geophysical Technology degree (5yr.) from the Department of Earth Sciences at Indian Institute of Technology Roorkee, Roorkee, in 2013. He then received his dedicated Master of Technology degree in 2015 from the Department of Earth Sciences at Indian Institute of Technology Bombay, Mumbai, specialising in Petroleum Geosciences. After that, he completed PhD programme at IITB-Monash Research Academy, Mumbai, in 2022. He has a research interest in applied machine learning and statistics pertaining to Geoscientific problems. He has published some of his research works in reputed international journals and conference proceedings. He has also mentored three B.Tech students to pursue summer projects in machine learning during his doctoral studies.

ORCiD ID @ 0000-0001-8917-0775