School of Computing
FACULTY OF ENGINEERING AND
PHYSICAL SCIENCES

Final Report

Application of Convolutional Neural Networks for the Characterisation of Porous Materials

Hope Brooke

Submitted in accordance with the requirements for the degree of MEng, BSc Computer Science

2024/25

COMP3931 Individual Project

The candidate confirms that the following have been submitted.

Items	Format	Recipient(s) and Date
Final Report	PDF file	Uploaded to Minerva (30/04/2024)
Link to online code repository	URL	Sent to supervisor and assessor (30/04/2024)
Link to Google Drive folder of datasets	URL	Sent to supervisor and assessor (30/04/2024)

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of Student) Hope Brooke

Summary

This project explores the application of Convolutional Neural Networks (CNNs) in predicting the properties of porous materials using micro-tomography images. Porous materials play an important role in various industries due to their unique properties. Traditional characterisation techniques are often time-consuming and expensive, leading to the investigation of machine learning methods as alternatives. CNNs, known for their effectiveness in processing grid-like data, offer a promising alternative for accurate property prediction.

A new compact dataset is derived from the publicly available DeePore dataset, containing six image slices of each material. CNN models are trained on this dataset to predict material properties, with a focus on comparing models trained on three slices against those trained on six slices of data, and exploring the impact of residual connections in the CNN architecture. Results indicate that increasing the channels of input data leads to slower model convergence but reduced bias, while the addition of residual connections has minimal effect on performance. Furthermore, training models on specific property subgroups reveals varying degrees of accuracy improvement, and a workflow is provided to allow for customised model development tailored to specific property requirements.

Acknowledgements

I would like to thank my supervisor, Arash Rabbani, for his guidance and support throughout the duration of this project. His expertise in materials science and deep learning allowed for him to give me insightful direction. I would also like to thank my assessor, Toni Lassila, for his constructive feedback and discussion midway through the project, his guidance on approaching the writing of the report was invaluable.

Finally, I would like to thank my family and friends for their unwavering emotional support and patience. I would not have have been able to do this without them.

Contents

Chapter 1

Introduction and Background Research

1.1 Introduction

Porous materials play an important role in the materials science industry as a result of their unique properties and versatile applications. Their high surface area makes them invaluable for processes such as adsorption, purification, filtration, catalysis, and energy storage [66]. These applications make them indispensable across a range of sectors, including the chemical, automotive, energy, biomedical, and construction industries, and they play a vital role environmental remediation efforts.

The properties of porous materials dictate their potential applications. Accurate characterisation of these properties is essential for maximising their effectiveness in any given application [47]. As traditional characterisation techniques require experimental testing, the integration of computers in the process is common. However, direct pore-scale modelling techniques are costly and time-consuming [80].

To address these challenges, machine learning methods have emerged as promising alternatives, particularly for handling substantial amounts of data [5]. Convolutional Neural Networks (CNNs) are particularly effective in processing image and other grid-like data, as they can effectively capture hierarchical and spatial structures [20].

The aim of this project is to contribute to the field of porous material characterisation through the development of convolutional neural networks. These deep learning models will predict the physical properties of porous materials by analysing their micro-tomography images. The goal is to create a model that can rapidly predict these properties with a higher accuracy than is customary practice in literature. This project aims to contribute to the academic literature in the fields of materials science and machine learning by investigating how various factors, such as the number of input data channels, the number of predicted properties, and the implementation of residual connections, impact the accuracy and speed of the model's predictions. The deliverables of this project include a preprocessed dataset, developed CNN models, and a comprehensive workflow for property prediction, model retraining, and validation.

This chapter provides an overview of porous materials and deep learning, and an evaluation of relevant research in the field. In Chapter 2, the design and training process of the introduced models are outlined and justified, with a particular focus on the implementation of increased input data channels, property grouping, and residual connections. Chapter 3 presents the results of these models, discusses the influence of the outlined factors on the model's performance, and compares the final model chosen with other methods. Finally, Chapter 4 provides a conclusion and a summary of potential avenues for future work.

1.2 Porous Material Characterisation

Porous materials contain spaces within their structures that can vary in size, shape, distribution, and other attributes. The properties of porous materials influence their applications. Yasuda et al. highlights the significance of porous materials as components of filtration membranes, fuel cell electrodes, and catalytic reactor supports [81], while others discuss their importance for composite materials [53], nuclear waste repositories [12] and CO_2 sequestration [17].

1.2.1 Traditional Methods

While the intricacies of materials science lie beyond the scope of this project, understanding the fundamentals of traditional characterisation techniques is vital to appreciate the motivation for introducing machine learning processes. A level of domain knowledge is essential to discern which properties are necessary, what levels of accuracy are desired, and where there are any potential limitations or advantages.

Properties of a porous material, such as permeability, porosity, surface area, and pore size distribution, can be calculated by direct laboratory measurements [63], gas adsorption, mercury injection, electron microscopy, and nuclear magnetic resonance [71][81][83]. However, these approaches come with limitations, for example, mercury intrusion involves high pressures that can damage the pore structure of the material [83]. These techniques not only are time-consuming, but also involve significant costs. They require access to research laboratories, specialised equipment, specific gases or substances, and specific conditions on pressure or heat.

1.2.2 Computational Methods

Scanning electron microscopy (SEM), which scans the surface of a material with a focused beam of electrons, produces high-resolution images of the surface of a material. Although it is useful for surface-level analysis of microscopic images, it cannot capture any material depth and cannot assess structural properties that are not represented in the two-dimensional space. Micro-computed tomography (micro-CT) [22], focused ion beam-scanning electron microscopy (FIB-SEM), and confocal laser scanning microscopy (CLSM) have advanced this field by allowing three-dimensional imaging of porous materials [8][19][71][81]. 3D porous material images are sometimes referred to as 'Digital Rocks' [2].

These advanced technologies enable the use of numerical methods to be used for porous material characterisation, reducing the need for costly experimental calculations [49]. This process involves numerical simulations of a material's pore space, which allows for the calculation of mechanical properties, electrical properties, multi-phase flow characterisation and more [2]. The finite element method (FEM), the finite volume method (FVM), the Lattice-Boltzmann method (LBM) and smoothed particle hydrodynamics (SPH) are conventional numerical approaches, used in computational fluid dynamics (CFD) [4][9][24][63][71][80]. Although LBM is the most common approach to work with pore space images [8][23], it is computationally demanding [61][63][80].

Blunt et al. discuss two main approaches to modeling the pore space of materials: direct modelling and network modelling [8]. The Lattice-Bolzmann method (LBM) can be considered a direct modelling technique, as it directly simulates the behaviour of individual fluid particles within a porous material. Pore network models (PNM) differ from this by representing the pore space as a graph of interconnected pores and throats. While direct pore-scale modelling is more accurate, it's computational costs can make it impractical in some scenarios [30]. In contrast, PNMs are computationally more efficient, but have lower accuracy [50].

1.3 Deep Learning

Deep learning (DL), a subset of machine learning (ML), involves the training of artificial neural networks (ANNs) with multiple layers to extract features from data [43]. Unlike conventional machine learning approaches, deep learning methods learn directly from raw data. ANNs, modelled on the structure of the brain and nervous system, consist of interconnected layers of neurons that are connected to each other by weights [71]. These networks typically include an input layer for receiving raw data, followed by a series of hidden layers responsible for performing various transformations, and an output layer which produces the final prediction(s). Numerous hidden layers allow the network to discern complex patterns within the input data [6].

1.3.1 Convolutional Neural Networks

CNNs are a form of artificial neural network designed for processing grid-like data, such as images or sequences [42][44]. In contrast to conventional ML methods, feature extraction, feature dimensionality reduction and classification are integrated in the CNN process [30].

Convolutional layers serve as the building blocks of CNNs. These layers use filters (kernels) that convolve across input images to produce feature maps. With different filters extracting different features of the input data. These filters applied across the image allow for fewer learnable parameters and preservation of spatial relationships [2]. Pooling layers are often incorporated after convolutional layers to reduce the spatial dimensions of the generated feature maps through downsampling. Activation functions introduce non-linearity to CNNs, to allow for the modelling of complex relationships within the data. Fully connected (FC) layers follow convolutional and pooling layers; extracted features are flattened and then passed through one or more FC layers. They connect every neuron in one layer to every neuron in the next, to map features to output labels.

CNNs ability to capture spatial dependencies and hierarchical features means they excel at image processing tasks, such as image recognition, object detection, and image classification, and exceed conventional ML methods in the computer vision field [43][63]. They demonstrate a wide range of applications, including face detection [45][79], medical image analysis [46] and materials science [15]. Notably, CNNs have demonstrated better image recognition and object detection than humans in certain scenarios [29].

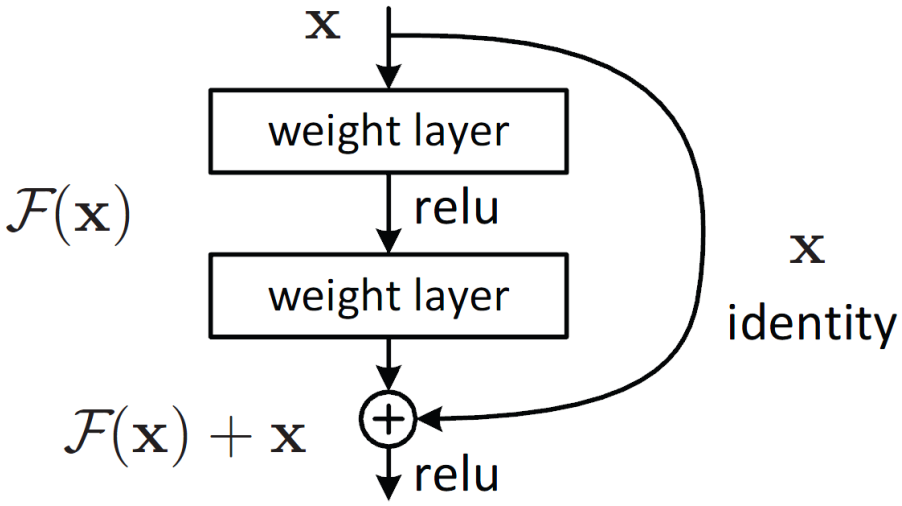

Figure 1.1: Depiction of a residual block [76].

1.3.2 Residual Connections

The residual network (ResNet) is an adaptation of the traditional CNN architecture proposed by He et al. to address the challenges encountered when training deep neural networks [28]. Later, the Deep Residual U-Net (ResUnet) was introduced by Zhang et al. [82], which follows a similar idea, but uses residual units as building blocks and integrates skip connections, also known as residual connections, between them [63].

As the depth of a neural network increases, gradients that are backpropogated during training can become very small, making it hard for layers to learn meaningful representations, this is referred to as the vanishing gradient problem. Residual learning changes the approach of mapping input and output; instead of learning the desired mappings, residual learning aims to learn residual functions, which essentially capture the difference between the desired output and the identity mapping of the input. These skip connections are incorporated into CNN architecture within residual blocks and allow for better flow of information, and therefore fewer parameters are required [28]. A residual block consists of one or more convolutional layers followed by a residual connection that connects the input to the output of the block. This connection allows the input to bypass the convolutional layers within the input block. See Figure 1.1.

While they were initially introduced to help deep CNNs, they have also proven beneficial for shallow CNN architectures, as they can occasionally also suffer from a vanishing gradient. Additionally, added parameters and connections can serve as a form of regularisation, which can help prevent overfitting and improve generalisation of the network. Residual connections can also stabilise the optimisation process and accelerate convergence.

1.4 Machine Learning for Porous Materials

The introduction of machine learning (ML) algorithms that could analyse large, complex data, extract patterns, and make decisions quickly, meant they were quickly adopted for porous

media research due to the time, computation, and specialty cost of numerical simulation methods [69][59]. While there are numerous ML methods well-suited to characterisation, the choice of algorithm depends on the task at hand.

Cawte and Bazylak conducted a study where they trained seven different ML algorithms to predict transport properties of fibrous porous materials [14]. Gradient boosting regression, artificial neural networks (ANNs) and support vector machines (SVMs) were found to have the highest performance, with indistinguishable error rates from each other. On the other hand, a study by Sudakov et al. suggested that CNNs outperformed gradient boosting in both percent error and prediction time, and found that a 3D CNN model is capable of predicting properties at a rate that is 450 times faster than PNM [68]. SVMs are commonly used for classification and regression tasks, and while they have been implemented to characterise porous materials with success [81], they are not always well suited for tasks involving large-scale image datasets, and rely on manually engineered features extracted from the data [16].

Deep learning techniques offer several advantages when applied in this field. Deep learning methods excel in complex pattern recognition, which is fitting considering the intricate structures presented in porous materials. They are also able to learn features without a need for manual engineering, which is both challenging and time consuming. Artificial neural networks (ANNs) have been proposed for use within chemical engineering processes since 1988 [31], and have been used to study relationships between the properties of the porous media and their fluid flow [57][73]. In the field of porous media characterisation, convolutional neural networks are prevalent, and hybrid deep learning models have also shown promise. Meng et al. proposed a 3D CNN-transformer hybrid neural network [50], while Tian et al. proposed a GA-ANN hybrid neural network [71], which both predict permeability with success. Beyond characterisation, deep learning has been successful in segmenting porous materials and digital rocks [40], as well as reconstructing 3D porous materials [10][49][51][75].

1.4.1 Convolutional Neural Network Methods

Convolutional neural networks (CNNs) have been used to predict material properties from various input such as wave propagation [48], contact angles [60], and, most commonly, images [26][78]. In this context, 2D grey-scale images are often used as one-channel input data [2][3][25][26][78], and image slices of 3D material can be used as multiple channels. For instance, Rabbani et al. trained a model on three slices extracted from a 3D microstructure, effectively treating them as RGB channels in an image [58]. The application of CNN property prediction offers significant advantages over traditional computational fluid dynamics simulations, with prediction speeds being several orders of magnitude faster [78].

There is also emerging research on the use of 3D CNNs for property prediction in this field [23][37][63][68][70]. Hong et al. implemented a 3D CNN capable of predicting the permeability of a material quickly and accurately, and outlined the potential to expand the model to predict other properties [30]. However, 3D CNNs are limited due to computational resources required to deal with the large input space and are typically limited to small-scale porous media [50]. To

address this challenge, various approaches have been proposed, such as dividing the domain into smaller pieces [63] or using hierarchical regression to iteratively pass data into finer scale models [64]. Additionally, a 2DCNN-Transformer hybrid model is proposed by Meng et al. to address the limitations outlined above, and demonstrates higher generalisation and accuracy in predicting 3D porous material properties compared to conventional 3D CNN approaches [50].

Residual networks (ResNets) and Residual U-Nets (ResUNets), adaptations of CNNs outlined in Section 1.3.2, have also shown success in characterising both 2D and 3D porous materials [13][63]. Wu et al. observed that the ResNet architecture exhibited marginally better performance compared to the conventional CNN architecture, but the model was discarded due to it's deeper architecture and increased computational demands [77].

Many of these models primarily focus on predicting permeability, occasionally also predicting other properties such as tortuosity, porosity, and wettability. Rabbani et al. developed DeePore, a rapid and comprehensive 2D CNN tailored for characterising a wide range of properties [58]. Although it is a 2D CNN, DeePore takes three slices extracted from 3D binarised micro-tomography images, treating them like RGB channels in an image. It employs a dimensionless approach, allowing for any size input image, with properties scaled to accommodate for this, and can predict properties for both 3D and 2D inputs. It's ability to predict a wide range of properties, coupled with it's lightweight nature, and it's code base that allows for feeding in and visualising data, retraining models, and predicting properties, makes it notably versatile.

1.4.2 Ground Truth Data

In the realm of machine learning, the quality of a dataset is integral to the performance of any model trained with it [36], and establishing a labelled dataset involves multiple steps. Initially, images of porous material are collected and pre-processed. Advancements in micro-CT, as discussed in Section 1.2.2, have facilitated the creation of 3D representations of porous media, and accessing a diverse range of these 3D material images is now feasible through online repositories such as the digital rock portal [54], and the DeePore dataset [55], which is utilised in various similar projects [50][58].

To enhance the size and variation of a dataset, data augmentation methods are commonly applied. Karimpouli and Tahmasebi demonstrated the value of data augmentation by implementing a hybrid pattern- and pixel-based simulation as a porous material data augmentation technique, which increased the coefficient of determination from 0.75 to 0.94 [39]. Other data augmentation techniques include cropping, flipping [32], translating, and reflecting [42]. Additionally, methods such as shrinking and expanding [30], interpolation [58], or adding noise levels at varying percentages [48] have proven effective in augmenting porous media datasets.

Finally, the data set requires labelling to be used for supervised learning. As discussed in Section 1.2.2, various numerical methods can be used to determine the property values of

digital rocks. Typically, the Lattice-Boltzmann method [26][30][63][71] or other computational fluid dynamics algorithms [49][81] are used. Pore network modelling can be implemented as a less computationally intensive alternative, however it's accuracy may not match that of LBM.

This project focuses on the CNN models designed, so a data set that has already undergone augmentation and labelling is used. After exploring options available, the dataset generated by the DeePore workflow [58] is found to be ideal. This dataset contains 17,700 semi-real 3D porous materials generated from applying data augmentation techniques to 60 real porous media. There are 15 single-value properties calculated for each material, as well as four functions and eleven distributions which span over 100 values each. For details on material origin, augmentation, and labeling processes, reference can be made to the original DeePore paper [58]. The ground truth labels are calculated using PNM, and while this modelling methods is not as accurate as alternative direct simulation approaches, it offers large variance and size; an LBM-labelled dataset is unlikely to provide the same level of range due to the limitations of this method outlined in Section 1.2.2.

Chapter 2

Methods

2.1 DeePore Library

This project builds on the DeePore rapid learning workflow [58], a pre-existing machine learning workflow explored in Section 1.4.1; This decision was made for several reasons. The DeePore project generated a ground truth labelled data set of 17,700 semirealistic microstructures of porous materials, making it the most suitable data set available for use in this project. The DeePore code base [56] consists not only of trained ML models and the code used to create them; Beyond loading pre-trained models to predict the properties of materials, it also provides robust functionality for users to retrain models, implement new models, visualise data, and more. These tools are valuable so they are maintained and adapted throughout this project. Building upon this existing code base allows for more time to be spent making adaptations that may improve accuracy and decrease prediction speed, as opposed to creating a new project from the beginning. Working from the same comprehensive dataset, comparisons can be made on the accuracy of any enhancements - using the original DeePore model as a control variable. This will ensure the integrity and repeatability of this project.

To clearly illustrate any changes in accuracy and performance, each stage of adaptations are introduced into the original DeePore workflow [58]. This approach allows us to use the original DeePore chosen model as a benchmark when implementing changes to the dataset, before introducing new models.

It is important to establish that while the software engineering component of this project is built off this code-base [56], this is not a replication of existing work, and any unedited DeePore code is only used for simple 'housekeeping' tasks. In this report, the original library is referred to as DeePore, while this project is referred to as DeePoreRevised.

2.2 Input Data

The generation of a ground truth dataset of labelled porous materials is a substantial process. The DeePore project has a labelled, ground truth data set comprising 17,700 semi-realistic samples of porous materials. The dataset creation process followed a dimensionless approach, allowing any models trained on it to be compatible with materials of varying spatial dimensions [58]. While this allows for flexibility, it requires the scaling of certain properties during prediction This data set is used to guarantee an accurate basis and to ensure that the focus of this project remains on improving the quality of available machine learning models in this field. Despite using the same initial dataset, the following will outline modifications to how these data are pre-processed and prepared.

2.2.1 Slices

The DeePore ground truth dataset is stored as a 13GB file in hierarchical data format (HDF5). While the full dataset contains 3D images, each with the shape 256x256x256, the original DeePore model [58] is trained on a compact dataset to reduce computational costs and training time. In this compact dataset, each material entry contains three slices extracted from the original 3D images of the material. These slices are taken at the midpoint of the image along the x, y, and z axes.

Increasing the amount of input data can lead to improved model accuracy, therefore augmenting the dataset with additional slices presents an opportunity to enhance the performance of the models. While 3D CNNs have demonstrated increased prediction accuracy (see Section 1.4.1), their computational demands reduce efficiency. A key objective of this project is to preserve or enhance speed of prior models. As an alternative, a 2D CNN with increased input channels is implemented. An increase in input data results in longer training times for a model, so while an initial plan was to extract three slices in each direction of the material, the option of two slices in each direction was chosen as favourable due to a more balanced trade-off between increased accuracy and training time. Any increase in accuracy should not come at the expense of a rapid and lightweight nature.

Slices are extracted at even intervals along each axis, meaning the shape of each material entry in this new compact dataset is 128x128x6. This larger data set requires variations of the model architecture to take advantage of the increased input data. The decision was made to take slices at even intervals across the image of the material to ensure a comprehensive representation of information for each material. Taking the slices at closer intervals may offer the opportunity to preserve relationships demonstrated in the image dimensionality; This is an avenue for future exploration. It is important to note that combining slices from different directions does not account for anisotropic materials, which have different physical properties according to the direction of measurement.

Method

This new compact dataset is created by processing each material entry from the full dataset. Two slices of data are extracted along each axis for each entry. The original DeePore dataset [58] represents material entries as Boolean values to denote pore and void spaces. In the process of generating the compact dataset, distance maps are generated, these provide more information than a simple binary representation, as calculating the euclidean distance between slices captures spatial relationships within the data. To reduce dimensionality, a 2x2 max pooling layer [11] is applied to resize the 256x256 layers to 128x128. To optimise storage, all values are adjusted to fit within the range of the uint8 data type.

As the outcome of this project is to provide functionality beyond a trained model, updates are made to the DeePore codebase [56] to accommodate datasets of varying size, due to the varying number of cuts. This involves developing new features to show material entries, visualise data, calculate Euclidean distances, prepare data, and feed in sample data to the

models. In addition, adaptations are made to the loading, training, testing, and prediction processes to account for the differences in the size and shape of the input data. To ensure clear distinction between data relevant to either dataset, saving and loading names of models and relevant files are updated accordingly.

2.2.2 Reducing Properties

The DeePore dataset [55] consists of 30 properties for each material entry: 15 single value attributes and 15 functions/distributions, represented as a series of 100 values each. This results in each material entry containing 1515 values. The original DeePore model [58] predicts all 30 properties, producing an output shape of 1515x1.

Reducing the number of properties predicted by the model and creating separate models for distinct property groups can increase prediction accuracy. Providing the model with a subset of the original 30 properties simplifies the model's task, and can make it easier for the model to learn meaningful patterns and relationships in the data. Fewer properties will result in a reduction of computational resources and the training time required. This suggests that training separate models for each property could increase accuracy and reduce prediction time for each model, however, this would be a lengthy process and offer limited flexibility across applications.

Property Grouping

There are three primary methods of approaching property grouping: grouping based on physical characteristics (e.g., throat-related properties), domain-specific meaningful groups (e.g., properties required for a specific application), and analysing the data and correlations to organise the groups; These methods may overlap to some extent.

While the main objective is to increase accuracy, it is essential to maintain ease and versatility of use; Therefore, the option to predict all original properties remains available without significantly increasing the required number of models. Considering this balance, the properties are split into two loosely similar-sized groups based on physical characteristics and domain knowledge: Group 1 includes properties related to the physical structure and dimensions of the material's pore network, while Group 2 contains properties related to the composition, connectivity, and mechanical characteristics of the material. Further details on these groups are provided in Table 2.1.

From the original DeePore project [58], a correlation graph of single-value features was generated, shown in Figure 2.1. From analysis of this graph, it is observed that properties in each group tend to exhibit high correlation. This aligns with the concept that properties in meaningful groups are likely to interact. Using highly correlated properties in an ML model can introduce multicollinearity, where two or more predictor variables in a regression model are highly correlated. This can make it difficult for the CNN to estimate the effects of each variable independently and decrease the stability of the model. Techniques such as dimensionality reduction and regularisation can be applied to prevent this.

Group1		Group2	
Name	Data Type	Name	Data Type
Absolute Permeability	Single value	Formation Factor	Single value
Pore Density	Single value	Cementation Factor	Single value
Avg. Throat Radius	Single value	Tortuosity	Single value
Avg. Pore Radius	Single value	Avg. Coordination Number	Single value
Avg. Throat Inscribed Radius	Single value	Specific Surface	Single value
Leverett J-function	Function	Grain Sphericity	Single value
Wetting Rel. Permeability	Function	Avg. Grain Radius	Single value
Non-Wetting Rel. Permeability	Function	Rel. Young Module	Single value
Pore Radius Distrbtn.	Distribution	Two-Point Correlation Func.	Function
Throat Radius Distrbtn.	Distribution	Coordination No. Distrbtn.	Distribution
Throat Length Distrbtn.	Distribution	Pore Sphericity Distrbtn.	Distribution
Pore Inscribed Radius Distrbtn.	Distribution	Grain Sphericity Distrbtn.	Distribution
Throat Inscribed Radius Distrbtn.	Distribution	Grain Radius Distrbtn.	Distribution
Throat Average Distance	Distribution		
Throat Permeability Distrbtn.	Distribution		

Table 2.1: The split of the DeePore data set [55] properties into two related groups.

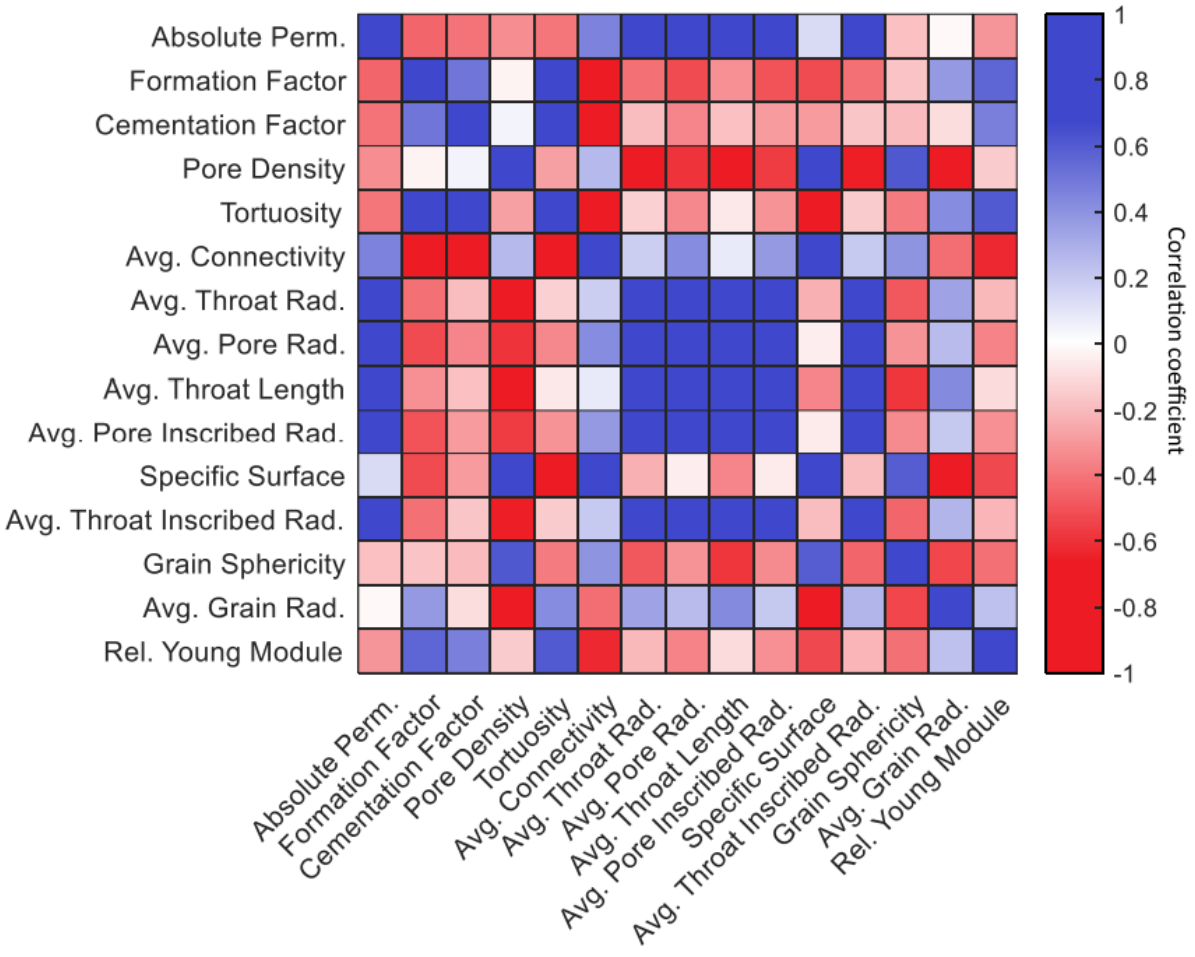

Figure 2.1: A correlation graph of single-value properties [58].

Method

One method of modifying the properties predicted by a model involves adjusting the dataset to include only necessary properties. However, this approach can be resource intensive, requiring significant time and memory space, which may constrain the iterative process of testing and evaluating property groupings. Instead, the same compact datasets can be used by selectively loading and pre-processing data for the specified properties.

Rather than adapting the existing code to account for a fixed set structure of properties, all sections reliant on these assumptions are rewritten to support any given subset of properties from the available 30. This prioritises the versatility and flexibility of the code, making it more convenient for iteratively testing variations, and provides future users of the workflow with the opportunity to train models on property groups of their choosing.

Given that the dataset has an array of length 1515 for these 30 properties, provided properties are mapped to the corresponding dataset indices, and only these values are loaded. The outlier detection and scaling value calculations are adapted to accommodate varying input data structures. The model creation process requires adjustments to the output shape, as well as modifications to the training, testing, and model loading functionalities to account for varying data dimensions. When using a loaded model for prediction, different properties are scaled in different ways, this process is rewritten to account for the dynamic selection of properties.

2.3 Model Architecture

Following the adjustments outlined above, the objective is enhancing model accuracy by making optimisations for the updated input data. Throughout model development, the original DeePore compact dataset [55] is used in parallel to the new dataset to facilitate clear comparisons. Two revised training scenarios are created, with each model trained on the three and six slice dataset separately, resulting in the four new models outlined in Table 2.3. The best performing model architecture is then trained separately on the two property groups.

The original DeePore code base [58] provides nine models, as detailed in Table 2.2. By analysing the architecture and accuracy of these models, as well as other relevant projects in this field, insights can be made regarding the impact that various factors, such as layers, loss functions and optimizers, have on model performance. This analysis allows for informed decision making on the architecture of the proposed new model scenarios.

2.3.1 Input Layer

In Section 2.2.1, the process of extracting material slices from the database, calculating and storing distance maps for each slice, and applying max-pooling for dimensionality reduction is outlined, essentially constituting the first two layers of the CNN models. It is worth noting that using input data with three channels is a common practice for CNNs, as they can be represented as an RGB image, with each channel corresponding to red, green or blue, as

Scenario	Number of Filters	Kernel Sizes	Optimizer	Loss Function	Average R^2
1	6,12,18	$8^2, 4^2, 2^2$	RMSprop	MSE	0.832
2	6,12,18,24	$8^2, 4^2, 2^2, 2^2$	RMSprop	MSE	0.824
3	12,24,36	$3^2, 3^2, 3^2$	RMSprop	MSE	0.885
4	6,12,18	$8^2, 4^2, 2^2$	RMSprop	WMSE	0.775
5	6,12,18,24	$8^2, 4^2, 2^2, 2^2$	RMSprop	WMSE	0.781
6	12,24,36	$3^2, 3^2, 3^2$	RMSprop	WMSE	0.782
7	6,12,18	$8^2, 4^2, 2^2$	Adam	BCE	0.791
8	6,12,18	$3^2, 3^2, 3^2$	Adam	BCE	0.749
9	6,12,18	$3^2, 3^2, 3^2$	Adam	WBCE	0.818

Table 2.2: Original DeePore training scenarios and averageR^2 scores [58].

Model	Input Data	Filters	Kernels	Optimizer	Loss Function
1	3 Slice	6,12,18,24	$8^2, 4^2, 2^2, 2^2$	RMSprop	MSE
2	6 Slice	6,12,18,24	$8^2, 4^2, 2^2, 2^2$	RMSprop	MSE
3	3 Slice	12,24,36	$3^2, 3^2, 3^2$	RMSprop	MSE
4	6 Slice	12,24,36	$3^2, 3^2, 3^2$	RMSprop	MSE

Table 2.3: DeePoreRevised training scenarios.

observed in the original DeePore [58]. However, this is not mandatory for CNN input. For instance, Lahivaara [48] used a single input channel when training a CNN model for the characterisation of porous material properties, while others, such as those using 3D CNNs for similar tasks [30][50][63], demonstrate alternative approaches. In this project, the models are trained on both three slice and six slice input data, so while the 2D image dimensions remain the same for all models, some have three channels and others have six. Despite this increase in channels, the model architecture remains a two-dimensional CNN.

2.3.2 Convolutional Layers

To prevent high model complexity, the number of convolutional layers should not be too high. In the original DeePore study [58], the best performing model contains three convolutional layers. While the four convolutional layer architecture did not increase the accuracy, the generation of a compact dataset, with six slices instead of three, increases the input data of the model. Increasing the number of convolutional layers has the potential to capture more complex relationships in the data [23], therefore, a model with four convolutional layers may be better suited for the larger compact dataset. As a result, one model with three convolutional layers and another with four is implemented.

To maintain variability in the training scenarios, one model implements varied filter sizes, while another implements fixed filter sizes. The choice of fixed filter sizes, specifically $3^2, 3^2, 3^2$, has proven effective in previous studies [2], and in DeePore training scenario 3 [58], this configuration is retained. The four convolutional layer model incorporates variable filter sizes of $8^2, 4^2, 2^2, 2^2$. Variable filter sizes are good at capturing hierarchical patterns within the data across different scales, while fixed filter sizes offer computational efficiency and simplicity. Increasing filter depth can potentially capture more variability and complex relationships in larger input data. However, it also introduces a higher number of parameters, leading to

increased computational requirements. To find a balance between prediction speed and model complexity, filter depths are limited to the larger among those used in the original DeePore training scenarios. Future work could explore the impact of increased filter depths in this model architecture.

Following each convolutional layer, a 2x2 max-pooling filter is applied to down sample the data. While alternative pooling methods exist, max pooling is preferred over average pooling due to it's effectiveness in handling porous media [39]. After all convolutional layers, the data is flattened before being passed to two fully connected layers. The first dense layer uses Rectified Linear Unit (ReLU) [52] as the activation function, while the second layer uses Sigmoid activation function. ReLU is the one of the most commonly used activation functions [62], as it introduces non-linearity without incurring significant computational cost [23][42]. While Sigmoid encounters problems with vanishing gradients, coupling the two activation functions can mitigate these issues. ReLu and Sigmoid are defined as follows:

$$\text{ReLu} : max(0, x) \qquad \text{Sigmoid} : \sigma = \frac{1}{1 + e^{-x}}$$

2.3.3 Residual Connections

Performance degradation due to transformations applied at each layer of a model can hinder a network's ability to learn meaningful connections [28]. Residual connections, introduced in Section 1.3.2, can achieve better generalisation, as well as faster and more stable training. These connections, also known as skip connections, are implemented in both models; Figure 2.2 provides the architecture of the CNN designed with three convolutional blocks and added residual connections.

2.3.4 Output Layer

The full model output consists of an array of 1515 elements. As outlined in Section 1.4.2, the initial 15 values represent single attributes, followed by 4 functions and 11 distributions, each comprising 100 indices. The shape of this output varies accordingly when only a subset of properties is chosen. The four functions are represented in 100 values by segmenting the function curve into 100 equal sections, and recording the corresponding value at that interval. Similarly, the eleven distributions are represented as cumulative probability distribution functions (CDFs), with the range of zero to one divided into 100 segments. For a complete understanding of this format, reference can be made to the original DeePore documentation [58].

2.3.5 Model Training

Based on the findings from the DeePore training scenarios [58], displayed in Table 2.2, it is evident that the choice of loss function significantly influences the model's performance. In scenarios 1, 2 and 3, mean squared error (MSE) is utilised as the loss function. Given that the latter 15 properties consist of 100 output indices each, models trained using MSE understandably excel in predicting functions and distributions compared to single values. To

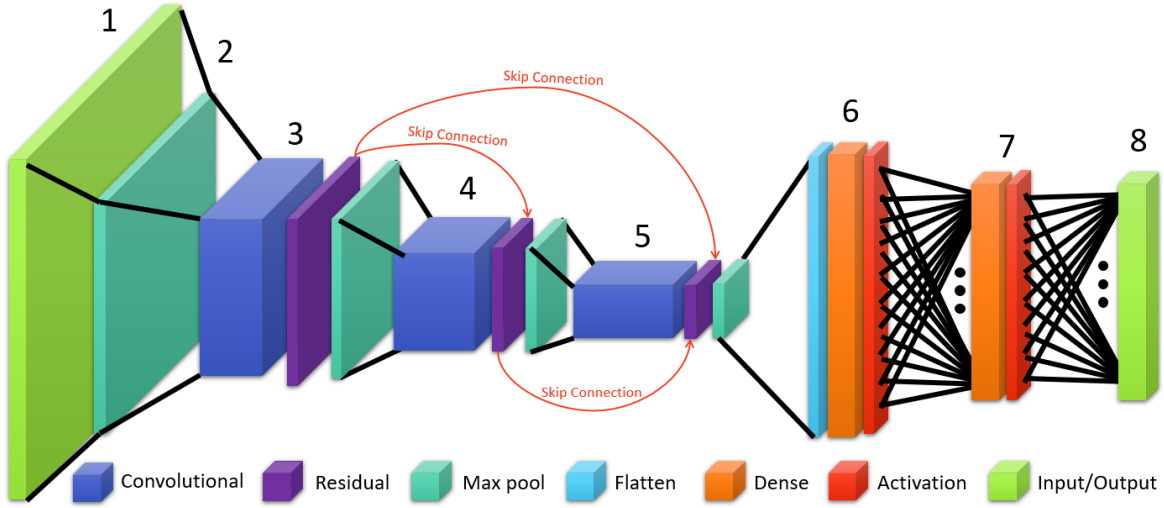

Figure 2.2: Architecture of the 3 convolutional block CNN with residual connections. Arrows labelled 'Skip Connection' denote the identity mapping performed in the network. Diagram is adapted from the given DeePore diagram structure [58].

address the disparity in the impact of single values and range values, weighted mean squared error (WMSE) is used in DeePore scenarios 4, 5 and 6. Regarding single values, models that use WMSE generally outperform models that use MSE, however this slight increase in accuracy comes at the considerable expense of accuracy in predicting functions and distributions.

Binary cross-entropy (BCE) and weighted binary cross-entropy (WBCE) serve as the loss functions in DeePore training scenarios 7, 8 and 9. While BCE demonstrates strong performance in predicting functions and distributions, it exhibits poor performance predicting single-value features. To address this, WBCE is used as the loss function in DeePore training scenario 9 [58], resulting in improved performance; however the average R^2 remains lower compared to scenarios that use MSE. Other loss functions, such as mean absolute error (MAE), Huber loss [33] and root mean squared error (RMSE), have been successfully utilised in similar projects [2][48][50], and offer the advantage of being robust to outliers. MSE is selected as the most appropriate choice due to it's performance in DeePore training scenarios [58], and due to its success in related literature [26][39][60]. The MSE loss function is calculated as follows:

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

Where N is the number of properties predicted, 1515 in the full model and adjusted appropriately for different property groupings, and y_i, $\hat{y}_i$ are the actual and predicted values of the ith property.

Optimizers play a crucial role in CNNs by minimising the loss function. While various optimizers are suitable for regression CNN models, Adam [41] and RMSProp [72] are notably stable and fast. Although Adam is commonly used and has demonstrated success in prior works [2][39][50][63], RMSProp is chosen as the optimizer for both new models as it

outperforms the Adam optimizer in the DeePore training scenarios [58]. Given the limitations of what can be done within the scope of this project, no other optimizers will be tested at this time, but this is an area open for future exploration.

There was originally consideration given to incorporating regularisation techniques into the models to mitigate overfitting. However, to maintain focus on other aspects of this project, such as changes made to the input data and the addition of skip connections, these techniques are not implemented. Options such as L2 regularisation on convolutional layers, batch normalisation [35], or dropout layers [67] could enhance the model's performance by reducing variance and increasing bias. Batch normalisation has been implemented in various works within the field of deep learning and porous materials [26][39][50][60], while dropout layers have occasionally been used in relevant works as well [2][60]. This presents an opportunity for future work.

The batch size used to train a model is limited by GPU memory, which can impact performance [38]. Following some initial testing, a fixed batch size is determined for all models to maintain consistency. Learning rates of $10^{-3}, 10^{-4}, 10^{-5}$ and 10^{-6} are tested on an initial model architecture; as a step size of 10^{-5} is found to perform the best, it is implemented in all further model training. Once the data is processed and outliers are removed, it is partitioned into training, validation, and testing sets using an 80-10-10 split. This allocation allows for effective generalisation, mitigates the risk of overfitting and ensures a substantial portion of the dataset is available for training. Additionally, probability curves of each property are plotted between the training, validation, and testing data to verify that there is an even distribution.

2.4 Model Evaluation

MSE serves as a performance metric during model training. The best performing DeePore model will be trained with the hyper-parameters and model architecture described [58], and this model will serve as a baseline for comparison. The original DeePore models are unique in their breadth of properties predicted, lightweight model architecture, and rapid prediction speed, and these new models designed attempt to maintain these qualities. Considering this, the model will not be evaluated against other research that sacrifices prediction speed, and range of predictions, for the sake of increased prediction accuracy.

Reference/estimated scatter plots are generated for all single-value properties to visualise model accuracy, and investigate predictions made on the test set. Probability distributions of the reference and estimated single value features are plotted to ensure that the range of values are represented. For functions and distributions, which are more complex to evaluate, line charts are plotted for a randomly selected material entry from the test set to compare estimated and reference values for each property.

2.5 Project Management

This project can be divided into four main stages.

1. **Slice Increase:** This stage involves creating a new compact dataset, and adjusting functionality to account for this.
2. **Property reduction:** This stage involves retrieving subsets of the original 30 properties, and adjusting functionality to account for this.
3. **Model Optimisation:** This stage involves creating models that are designed and optimised to accommodate the modifications made to the input data and testing different combinations of model architecture and data set modifications.
4. **Final Testing and Evaluation:** This stage involves thorough testing and evaluation of all modified models and making final adjustments to the chosen model.

While the first two stages are independent of each other, the latter two are order dependent. It is crucial that the modifications made are completed to a sufficient standard of quality before moving on to the next stage.

These distinct stages allow the project's development cycle to be effectively divided into sprints, which are an element of the Scrum project management framework [65]. To ensure thorough exploration of different modifications, combinations, and performances, and ensure the robustness and accuracy of any results, the steps within each sprint are executed iteratively. The implementation of four two-week sprints encourages agile practices, such as iterative development, refinement, and flexibility.

2.5.1 Version Control

GitHub is used for version control, hosting the project repository, and managing code revisions. At the beginning of each stage, a new branch is created for any work specific to that stage, before it is merged and closed on completion of the sprint. Separate milestones are set up for each sprint, with issues opened and assigned to the relevant milestone for any tasks, enhancements, and bugs that need addressing. The use of issues and milestones allows for effective project management by providing a platform for tracking progress, organising workload, and prioritising tasks.

2.6 Development Tools

This project is developed in Python, using the Keras framework [18] to interface with the TensorFlow platform [1] for model implementation. Although PyTorch is an appropriate alternative, TensorFlow is the most common platform used in projects like this [2][48][49], and this choice aligns with the original DeePore codebase [58].

Code libraries such as NumPy [27], Matplotlib [34], and SciPy [74] are used for data processing and visualisation tasks. H5py [21], a Python package that provides an interface to the HDF5 library, is used to manipulate the data, and read and write to the HDF5 dataset.

As discussed in Section 2.5.1, the code repository is managed using GitHub; The datasets are stored on Google Drive due to GitHub file size limits. Google Colaboratory ('Colab') [7] serves as the development environment for all functionality and model training. Colab was chosen over Jupyter Notebook due to its seamless integration with Google Drive and GitHub. A Colab Pro subscription is used to prevent premature termination of code and mitigate issues that arise from Colab's fluctuating usage limits.

Chapter 3

Results

This chapter provides an overview of the new dataset created and the developed DeePoreRevised workflow, followed by a comprehensive discussion of all models introduced.

3.1 Dataset Modifications

As described and justified in Chapter 2, the initial approach to improve precision involves expanding the number of slices of each material fed into the model, from one in each direction to two in each direction. The development of the new compact dataset results in a dataset size of 1.26GB. While the data set itself cannot be hosted on the GitHub repository due to file size constraints, the code used to create it is provided. To minimise storage required, feature maps are stored as uint8.

3.2 DeePoreRevised

As this project is building upon the DeePore library [58], it is crucial that the usability, modularity, and flexibility of the original code-base is maintained. Instead of hardcoding separate function routes for modifications, changes are integrated to ensure that users of the workflow can interact with the project in a consistent manner.

To distinguish the code developed within the scope of this project, all code within this library is clearly labelled as new, revised, or original. Any changes and additions are accompanied by comments describing the revision made and the reasoning behind it. This duplicate library is named DeePoreRevised, and the original DeePore library [58] is contained within the GitHub repository to allow comparisons. Beyond the implementation and training of models, there is extensive adaptation to functionality. To ensure simplicity of use, parameters specifying the slice number and the properties to predict are passed to functions. This project's primary focus lies in designing and training models, but the workflow is also an important component that allows for use beyond the scope of this project; a demo is provided in the repository to facilitate future applications.

3.3 Model Performance

After some initial testing, a batch size of 100 is determined to provide the best trade-off between efficiency and performance. As described in Section 2.3.5, the data set is partitioned into training, validation, and testing sets with an 80-10-10 split. To ensure that a diverse range of values are fairly and evenly distributed in each subset, reference can be made to the probability distribution graph illustrated in Figure 3.1.

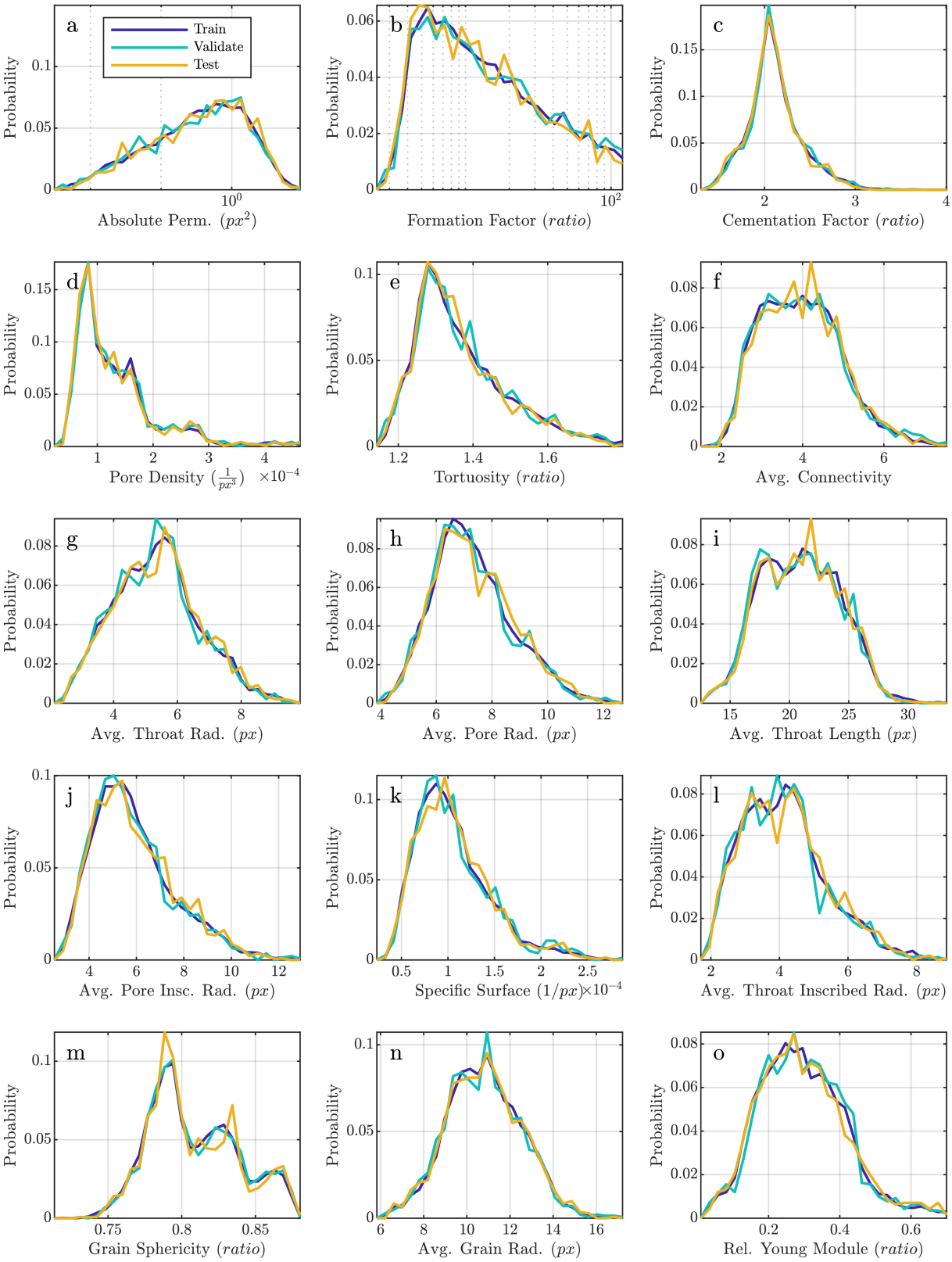

Figure 3.1: Single-value property probability distribution for training, validating, and testing data [58].

Model	Slices	Epochs	Time/Epoch	Time/Step	Total Time	Test MSE	Time/Predict
DP3	3	100	239s	2s	6.64 hrs	0.0047	0.22ms
DPR1	3	100	232s	2s	6.45 hrs	0.0049	0.28ms
DPR1	6	145	427s	3.2s	17.2 hrs	0.0053	0.38ms
DPR2	3	100	259s	2s	7.23 hrs	0.0048	0.24ms
DPR2	6	145	426s	3.2s	17.1 hrs	0.0050	0.36ms

Table 3.1: Training times for DeePore scenario 3 [58], and the DeePoreRevised scenarios. All models are trained on the T4 GPU.

The models trained on the three-slice dataset are trained for 100 epochs, as no significant improvement is observed after that. Models trained on the six-slice dataset exhibit a slower reduction in loss, and do not reach a plateau as quickly. Given that improvements are still notable at 100 epochs, training is extended to 145 epochs for these models, where any decline in loss has slowed down considerably. The MSE loss of these models is still decreasing at this point, but time constraints limit further training. Extending the training epochs for the six-slice models is an avenue for future exploration.

Model training is performed on the GPU architecture provided by Google Colab, specifically the Nvidia Tesla T4 GPU. Refer to Table 3.1 for an overview of epochs, training time, final testing MSE, and prediction time for the four created models alongside the original DeePore model [58]. All models trained on the three-slice dataset take approximately 4 minutes per epoch. This training time is notable; in DeePore [58], the model introduced only takes 45 seconds per epoch using an Nvidia GeForce GTX 1660 Ti. However, when the same model is trained on Google Colab's T4 GPU, it extends to 4 minutes per epoch. This is likely due to differences in GPU architecture, memory bandwidth, or other overheads unique to Google Colab, such as resource sharing and network latency. Training models on the six-slice dataset take approximately 7 minutes per epoch, which is expected due to increased data processing requirements. The average prediction time for a sample is determined by calculating the time of prediction on the test set and dividing it by the number of samples. Although none of the developed models provide faster predictions than the DeePore model [58], they are still rapid. Prediction speed is longer for models trained on the six-slice dataset which is to be expected. Final testing MSE is presented in Table 3.1 and training and validation MSE are illustrated in Figure 3.2.

Three-slice Input Data

In comparison to the original DeePore training scenario 3 (DP3) [58], DeePoreRevised training scenario 2 (DPR2) performs similarly when trained on the three-slice dataset. Although DPR2 takes marginally longer for training and validation MSE to reach a plateau, it achieves a similar MSE. The main distinction between their architectures is the inclusion of residual connections in DPR2. While residual connections help mitigate vanishing gradient problems, they require additional operations, potentially resulting in the slower convergence of this introduced model. This trend is also reflected in the final testing MSE, where DPR2 displays a loss that is only 10^{-4} higher than DP3.

DeePoreRevised training scenario 1 (DPR1) demonstrates lower accuracy compared to the above models concerning training and validation loss on the three-slice data set. It takes longer to reach a plateau, and does so with a higher MSE. This aligns with the discussion presented in Section 2.3, where models with three convolutional layers perform better than those with four when trained on the three-channel porous material data format. Despite the higher training and validation loss, the final testing MSE is only 10^{-4} higher than DPR2.

Six-slice Input Data

On the six-slice dataset, DPR1 appears to reach a plateau sooner; however, DPR2 ultimately exhibits lower training and validation loss. This suggests that a three convolutional layer architecture remains better suited to the input data, despite the increase in input dimensions. DPR1 demonstrates less stable validation MSE, characterised by sharp 'jumps' as depicted in Figure 3.2. The final testing MSE is lower for DPR2 than DPR1 when trained on the six-slice input data, which is consistent with the observed training and validation trends.

Overall

In general, the models trained on the three-slice dataset exhibit lower MSE compared to those trained on the six-slice data. The six-slice models take longer to converge, and on analysis of the results, it seems as though these architectures may benefit from an increased learning rate. The initial tuning of learning rate was enacted on a model trained on the smaller dataset, and the optimal value found was implemented as consistent across all models. Future work should be done to optimise the learning rate for the increase in input data.

Residual connections are seen to have little impact on the performance of this model, likely because the model is not deep enough to benefit from the increased flow of information. The suitability of residual connections on deeper architectures compared to shallower architectures is highlighted by the fact that DeePoreRevised scenario 1 (DPR1) displays the shortest training time on the three-slice dataset, despite having more layers than the other architectures.

3.4 Final Model

While models trained on the three-slice dataset exhibit lower MSE, further analysis of the test set predictions show unexpected results, DeePoreRevised scenario 2 trained on the six-slice dataset is better able to capture the complexity of the data. This is discussed and explored in more detail in the following section. For clarity, all future references to 'the final model' are in reference to this specific training scenario.

3.4.1 Evaluation

The following evaluation process is done for all models designed, however results are only displayed for the final model. Please refer to the the relevant appendices to see the plotted graphs of other models.

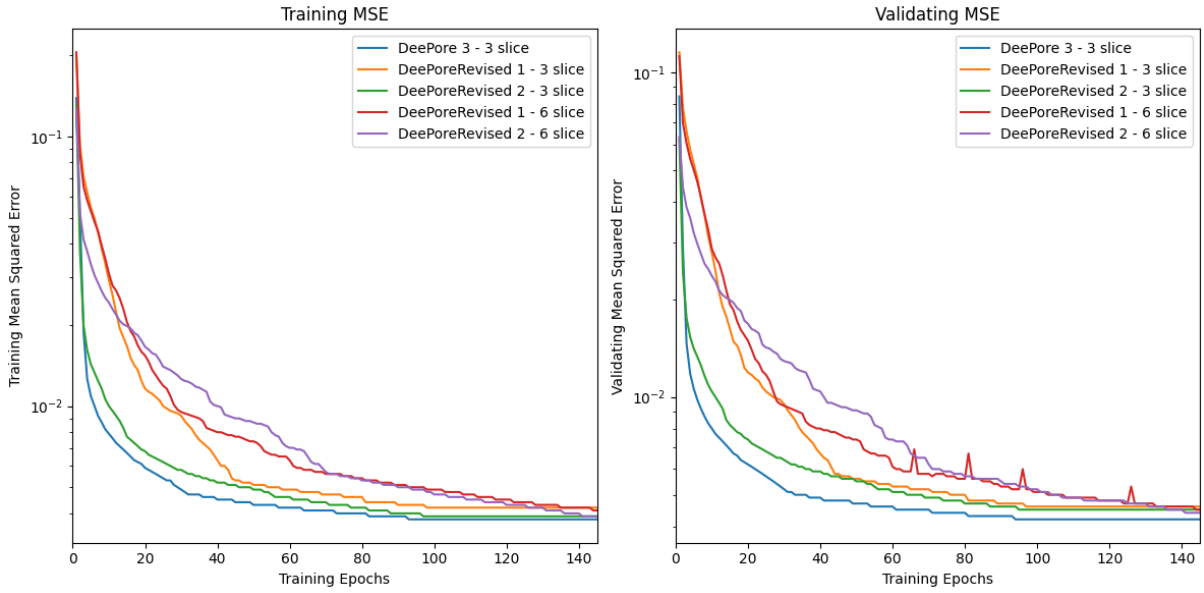

Figure 3.2: Training and validation MSE for DeePoreRevised scenarios 1-4 and DeePore scenario 3 [58]. As the three-slice models are only trained for 100 epochs, a straight horizontal line is plotted from final MSE at epoch 100.

To investigate the performance of the models created, probability distributions for the test set's reference and estimated single-values are plotted to ensure that the model accurately represents the distribution of data (Appendix C). While all models manage to capture the general trend, models trained on three slices of data are not able to capture the range, and tend to over predict the mean. The final model displays the most accurate distribution match (see Figure 3.4). This suggests that other models, especially those trained on three slices, are unable to capture variability in the dataset and have a higher degree of bias towards the mean. Although the final model performs the best in this regard, there are areas for improvement. For instance, specific surface values are consistently predicted to be below 0.2, pore density values are underestimated, and mean values tend to be over-predicted for pore-to-throat aspect ratio.

Additionally, the relationship between reference and estimated values in the test set is depicted for single-value properties (Appendix D). These scatter plots prove the assumption that while models trained on three-slice data have less variance, they have much more bias. The final model is the only one that doesn't display predictions that tend towards the horizontal mean (see Figure 3.5). The accuracy of the final model's predictions vary across different properties, with some predicted more accurately than others. However, it is evident that the model is capable of effectively capturing the complexity and variability of single-value properties for a material.

To illustrate an example of the prediction accuracy for the functions and distributions, predicted values from a random material in the test set are plotted (Appendix E). While this representation does not speak to the accuracy of the prediction of these values as a whole, and cannot be used to generalise the accuracy of the model, it offers insight into the model's capabilities. The final model oscillates heavily, and there is evidently room for improvement,

but it shows success in effectively capturing the trend of the reference values (see Figure 3.6). Despite the limitations, this final model still seems to perform better than others at capturing the trend.

All of these results suggest that while the final model exhibited higher MSE, it is much more capable at dealing with the variance in the data, and is a more generalisable model. It seems as though the models trained on the reduced input data exhibit higher bias and are oversimplifying the relationship between the input features and the predictor variables. The models trained on larger input data dimensions are likely to be more complex, and this complexity may allow it to capture more of the underlying variability in the data.

Deciding between a model with lower MSE and one that better captures the trends of the data depends on various factors, it is a trade-off between prediction accuracy and model interpretibility. In situations where prediction errors carry significant consequences, a model with lower MSE is preferable - models with with lower MSE, on average, predict values that are closer to the true values in the dataset. However, as this is an academic project, higher interpretibility is favoured.

3.4.2 Property Group Training

This final model is trained separately on the property groups defined in Section 2.2.2 to evaluate how different groupings and predictor sizes affect it's performance. Consistency is maintained with a training duration of 145 epochs, as other six-slice models have had. Figure 3.3 provides a visualisation of the training and validation MSE for both property grouped models, as well as the final model trained on the full set of properties. The final test MSE values for Group 1 and Group 2 are 0.0056 and 0.0039, respectively. These outcomes align closely with the trends of their training and validation MSE. The model trained on Group 2 exhibits superior performance in training, validation and testing compared to the full model. Conversely, the model trained on Group 1 shows inferior performance across all metrics.

The model trained on Group 2 outperforms others, as expected, due to it's reduced predictor values. The output shape of the model has a shape of 508, as opposed to the 1515 output of the full model. This aligns with the initial assumption that reducing predictor values enables the model to learn the desired properties more effectively. The final testing MSE of 0.0039 is a good indicator of it's efficacy, it is the only model developed within the scope of this project that exhibits a lower MSE than the original DeePore model [58]. This model still shows reductions in MSE towards the end of training epochs, but they are capped at 145 for consistency. It is viable that prediction accuracy may increase with additional epochs, and this is an area for future work.

The performance of the model trained on Group 1 properties raises questions. While it is logical that it wouldn't perform as well as the model trained on Group 2, due to it's larger predictor space (1007 output indices compared to the 508 in Group 2), it's unexpected that it performs worse than the model trained on all 30 properties (1515 output indices). There are

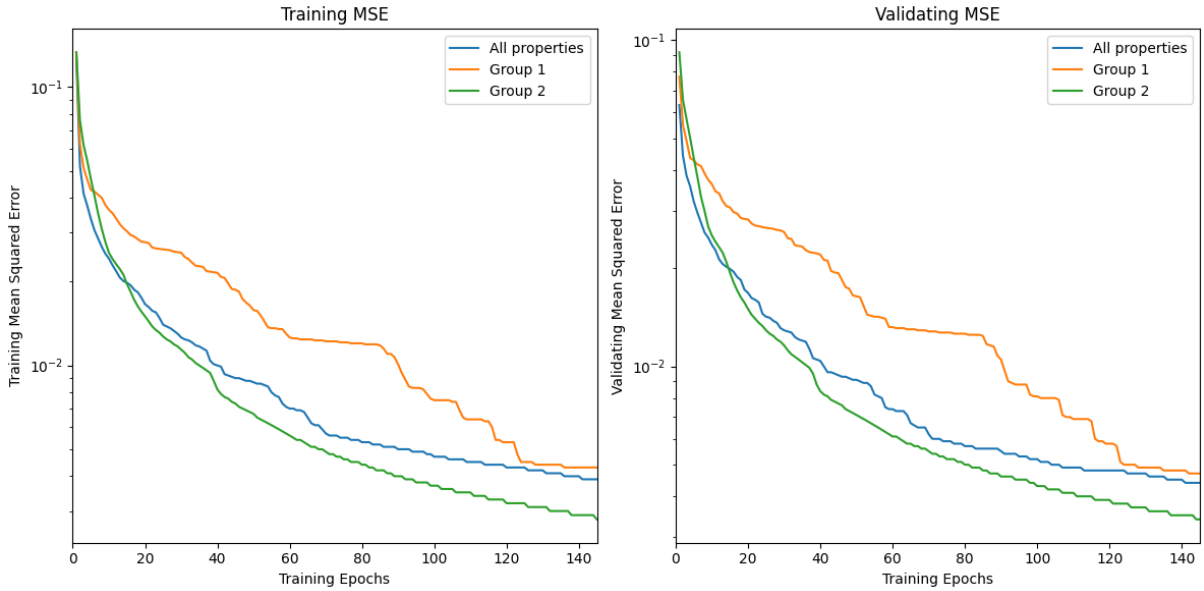

Figure 3.3: Training and validation MSE for the full final model, and for the final model trained separately on different property groupings.

several factors that could explain this. Firstly, the combination of properties could have led to multicolinearity issues in the model, discussed in Section 2.2.2. Additionally, the full model has varying accuracies for different properties, and the division of groups may have inadvertently resulted in Group 1 containing properties with lower prediction accuracy. The plotted MSE line for Group 1 is also noteworthy as, unlike the relatively consistent curves observed in all other training scenarios, it exhibits slower convergence and step-like reductions. Factors like learning rate and batch size can affect the rate of MSE reduction, however these remain consistent over all models. Instead, outliers or variability in the training data could be contributing to these fluctuations in the MSE curve. Further investigation into how outliers are handled for different properties may provide insight into this irregularity. In summary, the reduction of property predictor values shows significant promise, but further investigation into the influence of these groupings is necessary.

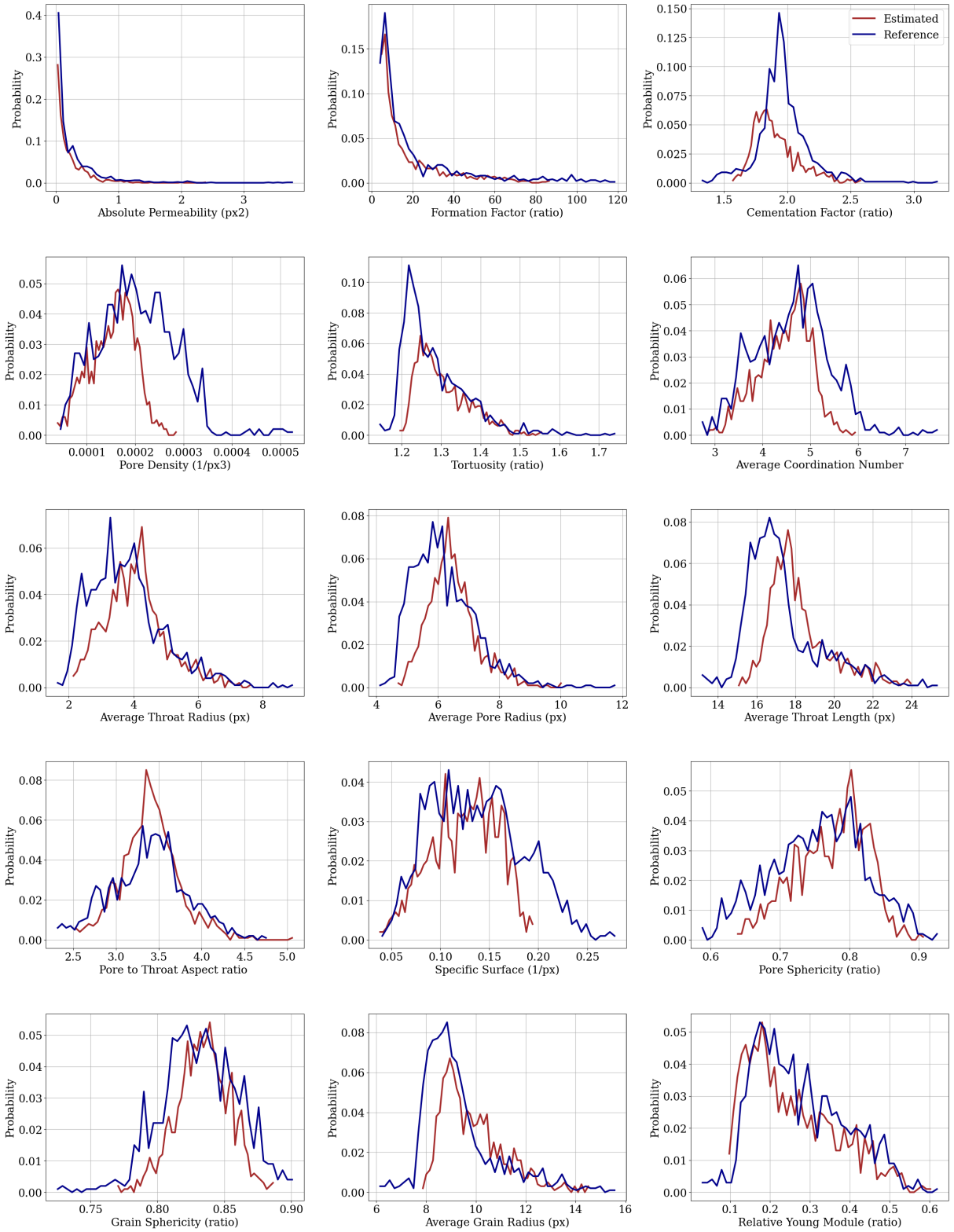

Figure 3.4: DeePoreRevised scenario 2 trained on six-slice data - Reference vs. estimate probability distribution for single-value properties.

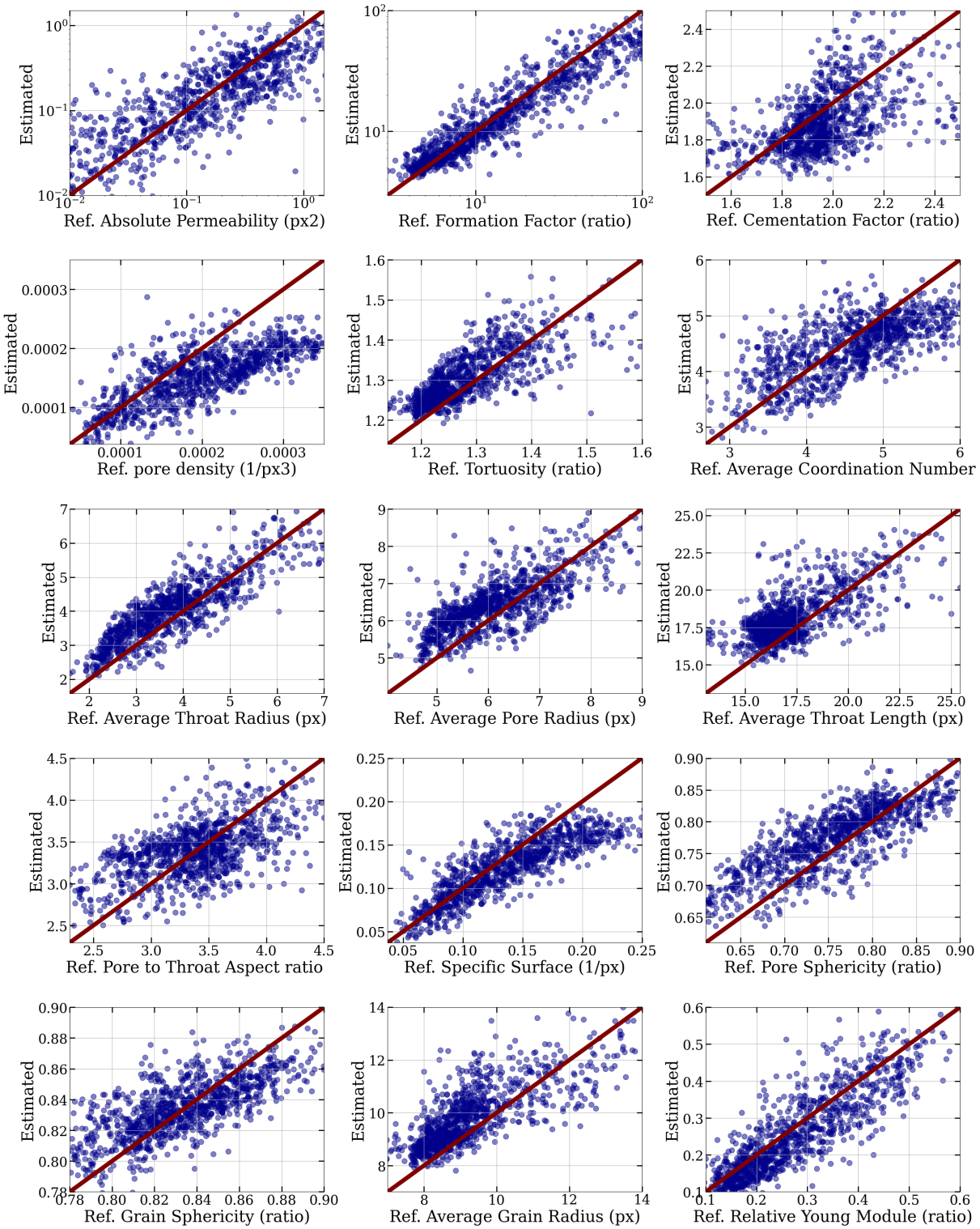

Figure 3.5: DeePoreRevised scenario 2 trained on six-slice data - Reference vs. estimate scatter plot for single-value properties.

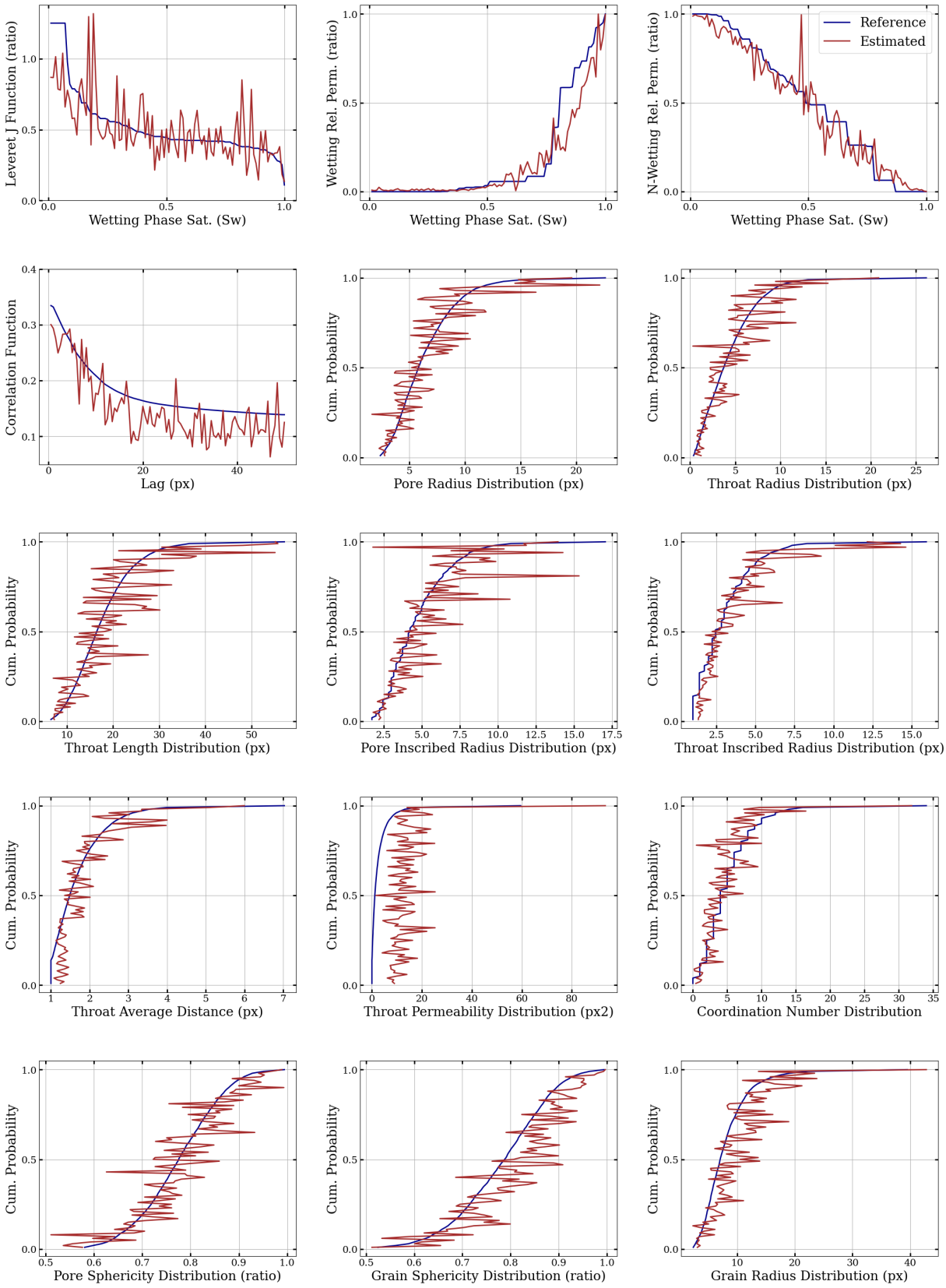

Figure 3.6: DeePoreRevised scenario 2 trained on six-slice data - Reference vs. estimate line chart plotted for functions/distributions from one random material sample from the test set.

Chapter 4

Discussion

4.1 Conclusions

A new compact data set is derived from the original DeePore dataset [58], containing six slices of each material. CNN models are developed, and the performance of models trained on three slices and six slices of data are compared. It is found that doubling the channels of input data increases prediction time, as expected, but does not increase accuracy. It leads to slower convergence of models, higher prediction errors, and increases training and prediction time. However, models with larger input data dimensions are able to capture more complex relationships, as they maintain variability in the model and don't suffer from the bias present in models trained on input data with fewer channels.

This project also investigate the impact of incorporating residual connections to the CNN architecture. Ultimately, the addition of these residual connections does not significantly affect training time or performance. While they are effective in reducing the training time of the deeper model, the computational overhead of connections and operations appears to negate any other advantages to accuracy or prediction speed.

Finally, this project examines the influence of property predictor variations on the performance of models. While it is not consistent across all variations, training the models on subgroups of properties has the potential to increase model accuracy significantly. This is the only approach that yields better results than the DeePore model used as a benchmark.

The outcome of this project is a new compact dataset, three trained models (one that predicts all original properties and two others predicting subsets), and an operational workflow for future applications. Not only can this workflow be used to rapidly predict material properties using the pretrained models, but it also offers the capability for users to train models on any selected subgroup of properties from the original comprehensive set, allowing for the creation of efficient and accurate models tailored to specific applications or requirements.

4.2 Ideas for future work

Due to the time and resource limitations of this project, there are many avenues that have potential but are not explored in depth. These include:

- The training of models on different property combinations to further investigate how property grouping affects accuracy.
- Retraining the final model with different learning rates to find an optimal choice for the increased channels of input data.

- Extracting slices from the 3D material at closer intervals to provide a representation of the relationships between the pores and voids.
- Exploring different optimizers or conducting an exhaustive hyper-parameter grid search, as suggested by Alqahtani et al. [2], and implemented by Wu et al. to determine the number of channels and neurons per layer [78].
- Creating a deeper residual network that takes advantage of skip connections, so that more complex features can be captured without significant performance degradation.
- Increasing filter depths for capturing more complexity in the larger input data.
- Implementing techniques that increase bias and decrease variance for models trained on the six-slice data, such as regularisation techniques. Prior research has shown success with methods such as dropout [39], batch normalisation [25], and self normalisation modules [26].
- Developing a model architecture that can characterise properties of anisotropic materials effectively.
- Extending the project scope to 3D CNNs by using the full dataset of material images.

These all present promising directions for future work that would contribute to the field of deep learning for the characterisation of porous material.

References

[1] M. Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous distributed systems.

[2] N. Alqahtani, F. Alzubaidi, R. T. Armstrong, P. Swietojanski, and P. Mostaghimi. Machine learning for predicting properties of porous media from 2d x-ray images. *Journal of Petroleum Science and Engineering*, 184:106514, 2020.

[3] N. Alqahtani, R. T. Armstrong, and P. Mostaghimi. Deep learning convolutional neural networks to predict porous media properties. In *Deep Learning Convolutional Neural Networks to Predict Porous Media Properties*. OnePetro, 2018.

[4] H. Andrä, N. Combaret, J. Dvorkin, E. Glatt, J. Han, M. Kabel, Y. Keehm, F. Krzikalla, M. Lee, C. Madonna, M. Marsh, T. Mukerji, E. H. Saenger, R. Sain, N. Saxena, S. Ricker, A. Wiegmann, and X. Zhan. Digital rock physics benchmarks—part II: Computing effective properties. *Computers & Geosciences*, 50:33–43, 2013.

[5] E. Avalos-Gauna, Y. Zhao, L. Palafox, and P. Ortiz-Monasterio-Martínez. Porous metal properties analysis: A machine learning approach. *JOM*, 73(7):2039–2049, 2021.

[6] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends® in Machine Learning*, 2(1):1–127, 2009.

[7] E. Bisong. Google colaboratory. In E. Bisong, editor, *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*, pages 59–64. Apress, 2019.

[8] M. J. Blunt, B. Bijeljic, H. Dong, O. Gharbi, S. Iglauer, P. Mostaghimi, A. Paluszny, and C. Pentland. Pore-scale imaging and modelling. *Advances in Water Resources*, 51:197–216, 2013.

[9] A. T. Borujeni, N. M. Lane, K. Thompson, and M. Tyagi. Effects of image resolution and numerical resolution on computed permeability of consolidated packing using LB and FEM pore-scale simulations. *Computers & Fluids*, 88:753–763, 2013.

[10] R. Bostanabad, Y. Zhang, X. Li, T. Kearney, L. C. Brinson, D. W. Apley, W. K. Liu, and W. Chen. Computational microstructure characterization and reconstruction: Review of the state-of-the-art techniques. *Progress in Materials Science*, 95:1–41, 2018.

[11] Y.-L. Boureau, J. Ponce, and Y. LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 111–118. Omnipress, 2010.

[12] A. Bourgeat, M. Jurak, and F. Smaï. Two-phase, partially miscible flow and transport modeling in porous media; application to gas migration in a nuclear waste repository. *Computational Geosciences*, 13(1):29–42, 2009.

[13] R. Cang, H. Li, H. Yao, Y. Jiao, and Y. Ren. Improving direct physical properties prediction of heterogeneous materials from imaging data via convolutional neural network and a morphology-aware generative model. *Computational Materials Science*, 150:212–221, 2018.

[14] T. Cawte and A. Bazylak. Accurately predicting transport properties of porous fibrous materials by machine learning methods. *Electrochemical Science Advances*, 3(1):e2100185, 2023.

[15] A. Cecen, H. Dai, Y. C. Yabansu, S. R. Kalidindi, and L. Song. Material structure-property linkages using three-dimensional convolutional neural networks. *Acta Materialia*, 146:76–84, 2018.

[16] J. Cervantes, X. Li, W. Yu, and K. Li. Support vector machine classification for large data sets via minimum enclosing ball clustering. *Neurocomputing*, 71(4):611–619, 2008.

[17] Y. Chen, Y. Li, A. J. Valocchi, and K. T. Christensen. Lattice boltzmann simulations of liquid CO2 displacing water in a 2d heterogeneous micromodel at reservoir pressure conditions. *Journal of Contaminant Hydrology*, 212:14–27, 2018.

[18] F. Chollet et al. Keras, 2015.

[19] S.-Y. Chung, J.-S. Kim, D. Stephan, and T.-S. Han. Overview of the use of micro-computed tomography (micro-CT) to investigate the relation between the material characteristics and properties of cement-based materials. *Construction and Building Materials*, 229:116843, 2019.

[20] S. Coleman, D. Kerr, and Y. Zhang. Image sensing and processing with convolutional neural networks. *Sensors*, 22(10):3612, 2022.

[21] A. Collette. Python and HDF5.

[22] J. C. Elliott and S. D. Dover. X-ray microtomography. *Journal of Microscopy*, 126(2):211–213, 1982.

[23] M. Elmorsy, W. El-Dakhakhni, and B. Zhao. Generalizable permeability prediction of digital porous media via a novel multi-scale 3d convolutional neural network. *Water Resources Research*, 58(3):e2021WR031454, 2022.

[24] A. Eshghinejadfard, L. Daróczy, G. Janiga, and D. Thévenin. Calculation of the permeability in porous media using the lattice boltzmann method. *International Journal of Heat and Fluid Flow*, 62:93–103, 2016.

[25] K. M. Graczyk and M. Matyka. Predicting porosity, permeability, and tortuosity of porous media from images by deep learning. *Scientific Reports*, 10(1):21488, 2020.

[26] K. M. Graczyk, D. Strzelczyk, and M. Matyka. Deep learning for diffusion in porous media. *Scientific Reports*, 13(1):9769, 2023.

[27] C. Harris et al. Array programming with NumPy. *Nature*, 585(7825):357–362, 2020.

[28] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition.

[29] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015. ISSN: 2380-7504.

[30] J. Hong and J. Liu. Rapid estimation of permeability from digital rock using 3d convolutional neural network. *Computational Geosciences*, 24(4):1523–1539, 2020.

[31] J. C. Hoskins and D. M. Himmelblau. Artificial neural network models of knowledge representation in chemical engineering. *Computers & Chemical Engineering*, 12(9):881–890, 1988.

[32] A. G. Howard. Some improvements on deep convolutional neural network based image classification.

[33] P. J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964.

[34] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[35] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift.

[36] A. Jain, H. Patel, L. Nagalapatti, N. Gupta, S. Mehta, S. Guttula, S. Mujumdar, S. Afzal, R. Sharma Mittal, and V. Munigala. Overview and importance of data quality for machine learning tasks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, pages 3561–3562. Association for Computing Machinery, 2020.

[37] S. Kamrava, P. Tahmasebi, and M. Sahimi. Linking morphology of porous media to their macroscopic permeability by deep learning. *Transport in Porous Media*, 131(2):427–448, 2020.

[38] I. Kandel and M. Castelli. The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT Express*, 6(4):312–315, 2020.

[39] S. Karimpouli and P. Tahmasebi. Image-based velocity estimation of rock using convolutional neural networks. *Neural Networks*, 111:89–97, 2019.

[40] S. Karimpouli and P. Tahmasebi. Segmentation of digital rock images using deep convolutional autoencoder networks. *Computers & Geosciences*, 126:142–150, 2019.

[41] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization.

[42] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[43] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[44] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[45] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua. A convolutional neural network cascade for face detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5325–5334, 2015.

[46] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. W. M. van der Laak, B. van Ginneken, and C. I. Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42:60–88, 2017.

[47] P. S. Liu and G. F. Chen. Chapter nine - characterization methods: Basic factors. In P. S. Liu and G. F. Chen, editors, *Porous Materials*, pages 411–492. Butterworth-Heinemann, 2014.

[48] T. Lähivaara, L. Kärkkäinen, J. M. J. Huttunen, and J. S. Hesthaven. Deep convolutional neural networks for estimating porous material parameters with ultrasound tomography. *The Journal of the Acoustical Society of America*, 143(2):1148, 2018.

[49] Y. Matsuda, S. Ookawara, T. Yasuda, S. Yoshikawa, and H. Matsumoto. Framework for discovering porous materials: Structural hybridization and bayesian optimization of conditional generative adversarial network. *Digital Chemical Engineering*, 5:100058, 2022.

[50] Y. Meng, J. Jiang, J. Wu, and D. Wang. Transformer-based deep learning models for predicting permeability of porous media. *Advances in Water Resources*, 179:104520, 2023.

[51] L. Mosser, O. Dubrule, and M. J. Blunt. Stochastic reconstruction of an oolitic limestone by generative adversarial networks. *Transport in Porous Media*, 125(1):81–103, 2018.

[52] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814. Omnipress, 2010.

[53] S. R. Pride and J. G. Berryman. Linear dynamics of double-porosity dual-permeability materials. i. governing equations and acoustic attenuation. *Physical Review E*, 68(3):036603, 2003.

[54] M. Prodanovic, M. Esteva, and M. Hanlon. Digital rocks portal. *Digital Rocks Portal: a repository for porous media images*, 2015.

[55] A. Rabbani. DeePore dataset.

[56] A. Rabbani. Deepore codebase, 2020.

[57] A. Rabbani and M. Babaei. Hybrid pore-network and lattice-boltzmann permeability modelling accelerated by machine learning. *Advances in Water Resources*, 126:116–128, 2019.

[58] A. Rabbani, M. Babaei, R. Shams, Y. D. Wang, and T. Chung. DeePore: A deep learning workflow for rapid and comprehensive characterization of porous materials. *Advances in Water Resources*, 146:103787, 2020.

[59] A. Rabbani, A. M. Fernando, R. Shams, A. Singh, P. Mostaghimi, and M. Babaei. Review of data science trends and issues in porous media research with a focus on image-based techniques. *Water Resources Research*, 57(10):e2020WR029472, 2021.

[60] A. Rabbani, C. Sun, M. Babaei, V. J. Niasar, R. T. Armstrong, and P. Mostaghimi. DeepAngle: Fast calculation of contact angles in tomography images using deep learning.

[61] A. Q. Raeini, J. Yang, I. Bondino, T. Bultreys, M. J. Blunt, and B. Bijeljic. Validating the generalized pore network model using micro-CT images of two-phase flow. *Transport in Porous Media*, 130(2):405–424, 2019.

[62] P. Ramachandran, B. Zoph, and Q. V. Le. Searching for activation functions.

[63] J. E. Santos, D. Xu, H. Jo, C. J. Landry, M. Prodanović, and M. J. Pyrcz. PoreFlow-net: A 3d convolutional neural network to predict fluid flow through porous media. *Advances in Water Resources*, 138:103539, 2020.

[64] J. E. Santos, Y. Yin, H. Jo, W. Pan, Q. Kang, H. S. Viswanathan, M. Prodanović, M. J. Pyrcz, and N. Lubbers. Computationally efficient multiscale neural networks applied to fluid flow in complex 3d porous media. *Transport in Porous Media*, 140(1):241–272, 2021.

[65] K. Schwaber and J. Sutherland. The scrum guide. In K. Schwaber and J. Sutherland, editors, *The Scrum Guide*, pages 133–152. Wiley, 2012.

[66] A. G. Slater and A. I. Cooper. Function-led design of new porous materials. *Science*, 348(6238):aaa8075, 2015.

[67] 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*, 15(1):1929–1958, 2014.

[68] O. Sudakov, E. Burnaev, and D. Koroteev. Driving digital rock towards machine learning: Predicting permeability with gradient boosting and deep neural networks. *Computers & Geosciences*, 127:91–98, 2019.

[69] P. Tahmasebi, S. Kamrava, T. Bai, and M. Sahimi. Machine learning in geo- and environmental sciences: From small to large scale. *Advances in Water Resources*, 142:103619, 2020.

[70] J. Tian, C. Qi, Y. Sun, and Z. M. Yaseen. Surrogate permeability modelling of low-permeable rocks using convolutional neural networks. *Computer Methods in Applied Mechanics and Engineering*, 366:113103, 2020.

[71] J. Tian, C. Qi, Y. Sun, Z. M. Yaseen, and B. T. Pham. Permeability prediction of porous media using a combination of computational fluid dynamics and hybrid machine learning methods. *Engineering with Computers*, 37(4):3455–3471, 2021.

[72] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop, coursera: Neural networks for machine learning. *University of Toronto, Technical Report*, 6, 2012.

[73] J. H. van der Linden, G. A. Narsilio, and A. Tordesillas. Machine learning framework for analysis of transport through complex networks in porous, granular media: A focus on permeability. *Physical Review E*, 94(2), 2016.

[74] P. Virtanen et al. SciPy 1.0: fundamental algorithms for scientific computing in python. *Nature Methods*, 17(3):261–272, 2020.

[75] Y. Wang, C. H. Arns, S. S. Rahman, and J.-Y. Arns. Porous structure reconstruction using convolutional neural networks. *Mathematical Geosciences*, 50(7):781–799, 2018.

[76] W. Wong. What is residual connection?

[77] H. Wu, W.-Z. Fang, Q. Kang, W.-Q. Tao, and R. Qiao. Predicting effective diffusivity of porous media from images by deep learning. *Scientific Reports*, 9(1):20387, 2019.

[78] J. Wu, X. Yin, and H. Xiao. Seeing permeability from images: fast prediction with convolutional neural networks. *Science Bulletin*, 63(18), 2018.

[79] Y. Wu, J. Li, Y. Kong, and Y. Fu. Deep convolutional neural network with independent softmax for large scale face recognition. In *Proceedings of the 24th ACM international conference on Multimedia*, MM '16, pages 1063–1067. Association for Computing Machinery, 2016.

[80] X. Yang, Y. Mehmani, W. A. Perkins, A. Pasquali, M. Schönherr, K. Kim, M. Perego, M. L. Parks, N. Trask, M. T. Balhoff, M. C. Richmond, M. Geier, M. Krafczyk, L.-S. Luo, A. M. Tartakovsky, and T. D. Scheibe. Intercomparison of 3d pore-scale flow and solute transport simulation methods. *Advances in Water Resources*, 95:176–189, 2016.

[81] T. Yasuda, S. Ookawara, S. Yoshikawa, and H. Matsumoto. Machine learning and data-driven characterization framework for porous materials: Permeability prediction and channeling defect detection. *Chemical Engineering Journal*, 420:130069, 2021.

[82] Z. Zhang, Q. Liu, and Y. Wang. Road extraction by deep residual u-net. *IEEE Geoscience and Remote Sensing Letters*, 15(5):749–753, 2018.

[83] S. Zhou, W. Sheng, Z. Wang, W. Yao, H. Huang, Y. Wei, and R. Li. Quick image analysis of concrete pore structure based on deep learning. *Construction and Building Materials*, 208:144–157, 2019.

Appendix A

Self-appraisal

A.1 Critical self-evaluation

I am proud of the outcome of this project, both in terms of the developed code and the created report. All outlined areas of exploration were investigated, and I am confident that I have executed the project to the best of my ability within the allocated time frame.

At the beginning of this project, I had no background in materials science, which meant there was a significant learning curve. While the project primarily centered around computer science, it was necessary for me to build a strong foundational knowledge base on the field of porous materials, their applications, and characterisation techniques, to understand the motive for the outlined solution, and to understand the relevant academic papers. Extensive research was done to bridge this gap, and it took more time than I anticipated. Fortunately, prior experience with online courses on CNNs and TensorFlow meant that there was some familiarity with the technical aspects.

I started programming two weeks behind schedule, as I felt that I wanted a comprehensive understanding of all relevant research before commencing. However, it became evident that ML work consists of a lot of experimentation and iterative refinement through trial and error, so I eventually started working with the plan to continue researching in parallel. This was definitely the right decision, as I found myself engrossed in the software engineering element, and managed to complete it in the time I had allocated for it, despite starting two weeks late. The training of the models, however, took a lot longer than expected. After spending a long time making iterative adjustments to hyper-parameters to optimise performance, I eventually had to make the choice to leave it how it was and move onto the next stage.

Following advice from my assessor, I adopted the approach of concurrently documenting methods and results as code iterations progressed. This proved invaluable, allowing me to clearly track the evolving project outcomes, and ensure that I would not have too much to write at the end. I left the literature review until the end, and while I had comprehensive written notes, annotated papers, and tracked references using Zotero software, synthesising all the information was more time-consuming than I envisaged.

A.2 Personal reflection and lessons learned

Undertaking a project outside of my comfort zone has proven to be a rewarding journey. It not only sparked an interest in a new field, but also allowed for personal growth. It taught me that with enough determination and perseverance, I can become well versed in concepts or areas that I have no prior experience with. I also found great satisfaction in applying new technology to address real-world problems beyond the field of computing.

This is the first time I have worked on something of this scale and I had very high expectations of myself. As I really enjoy the course, am doing a masters in computer science, and am considering a future career in research, I put a lot of pressure on myself to excel. I found myself spending too long on background research, and wanted my report to be at the standard of published academic reports; While I was ultimately proud of it, I had to acknowledge that this is the first serious research project I have undertaken and it is inevitable that it will not be on par with the quality of published reports by academics who have lots more experience.

My ambition also meant that I was hoping for ground-breaking results, and while they were intriguing, they fell short of my initial expectations and initially evoked a sense of disappointment. I have come to recognise that this is the nature of academic research, and all new findings are valuable, regardless of whether they have significant impact, and I take pride in the dedication and diligence I have invested in this project. The enjoyment and passion I found throughout background research, software development, and report writing has only furthered my interest in pursuing research in future, a lot of these skills are refined through practice and I am excited to apply the lessons I have learnt here to any future work academic work I produce.

A.3 Legal, social, ethical and professional issues

A.3.1 Legal issues

The use of 3D micro-tomography images from the publicly available DeePore dataset ensures compliance with intellectual property regulations. No data privacy or security concerns arise as the dataset does not involve sensitive information.

A.3.2 Social issues

This project does not have any immediate social impact, and is confined to exploration within the academic field.

A.3.3 Ethical issues

Given the absence of human subjects and sensitive data in the dataset, ethical considerations regarding data usage are not applicable.

This project holds promise for further development and potential integration into materials science systems. However, the possibility of inaccurate material property predictions could pose risks to the systems utilising these materials. Therefore, it is imperative that the performance of these models is accurately represented. Additionally, it is essential to emphasise that while the models have been proven to provide precise predictions, users should exercise discretion in their interpretation, and acknowledge the potential for errors.

A.3.4 Professional issues

Research integrity has been upheld throughout the project to ensure adherence to professional standards and all external materials have been cited.

Appendix B

External Material

External materials used and modified:

- The **DeePore Dataset** [55] is used to extract slices from and create a new compact data set, the original compact dataset is also used for comparison. They are clearly labelled *DeePore_Compact_Data.h5* and *DeePoreRevised_Compact_Data.h5*.
- The **DeePore Library** [56] is duplicated and modified, the DeePoreRevised library clearly outlines in comments which segments are original DeePore, which are modified DeePore, and which are newly implemented. The original DeePore library is contained within the repository to facilitate comparisons.

References are made to the DeePore paper [58], the DeePore data [55], and the DeePore repository [56] throughout the report.

Appendix C

Single Value Probability Distribution Graphs

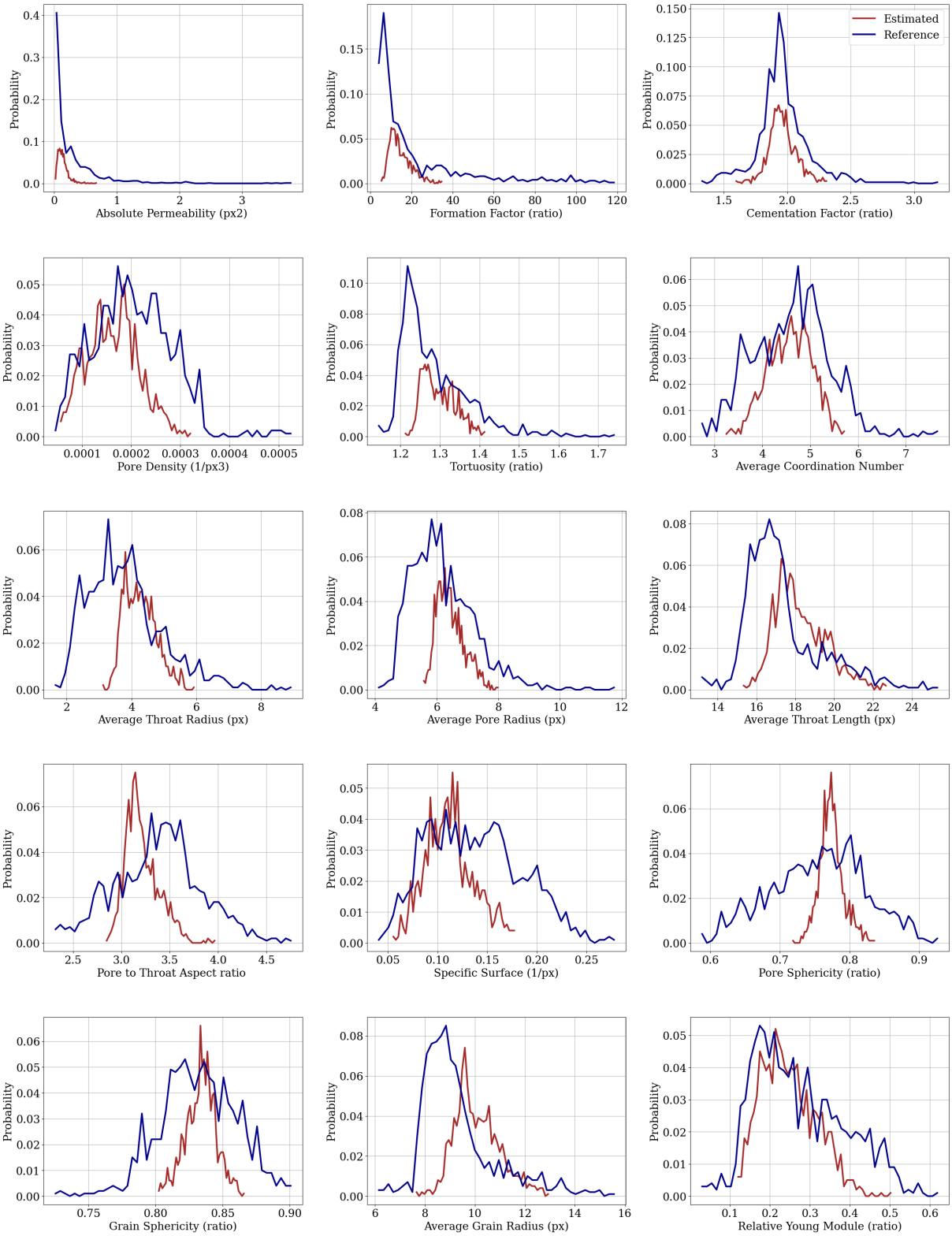

Figure C.1: DeePore3, 3 slices

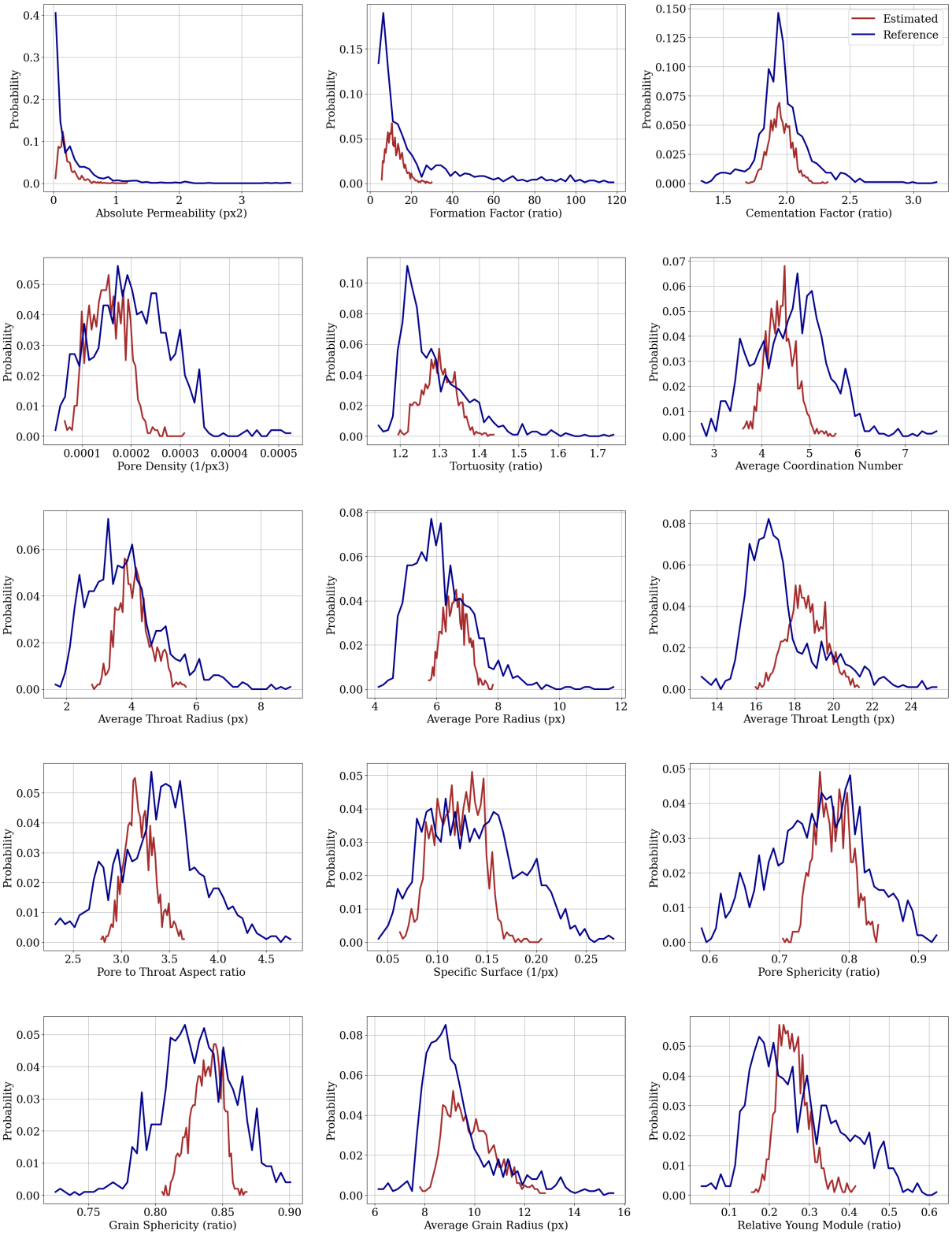

Figure C.2: DeePoreRevised1, 3 slices

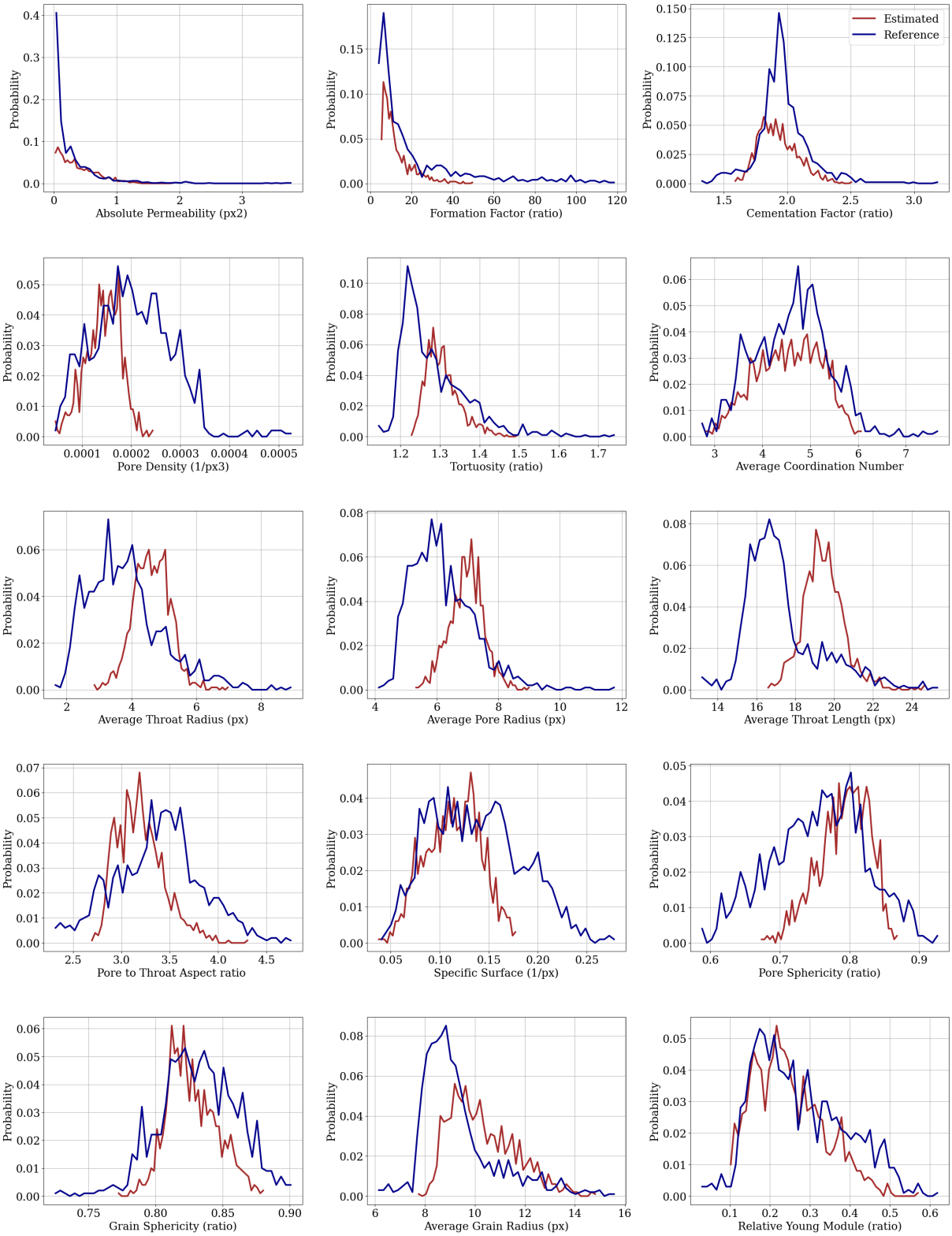

Figure C.3: DeePoreRevised1, 6 slices

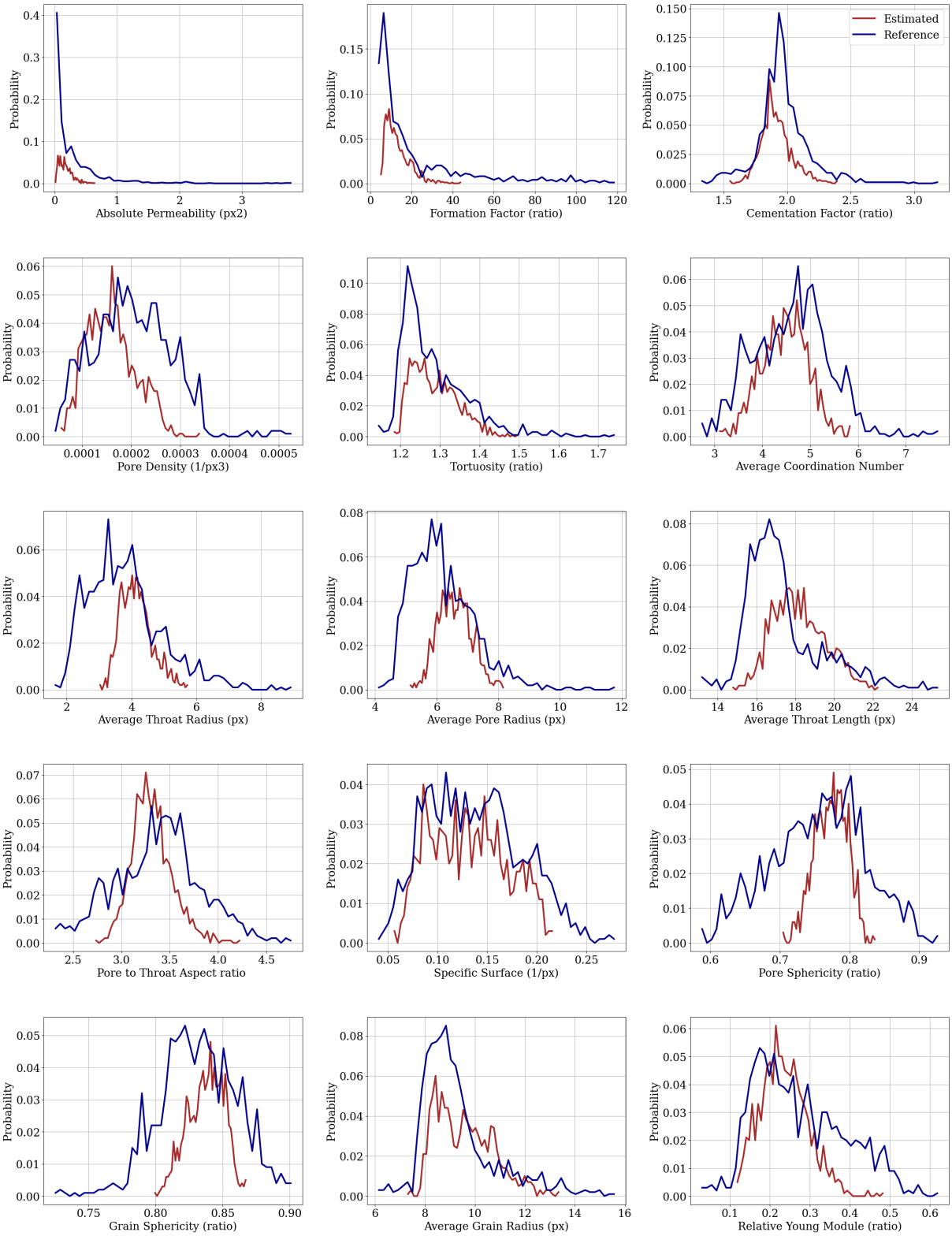

Figure C.4: DeePoreRevised2, 3 slices

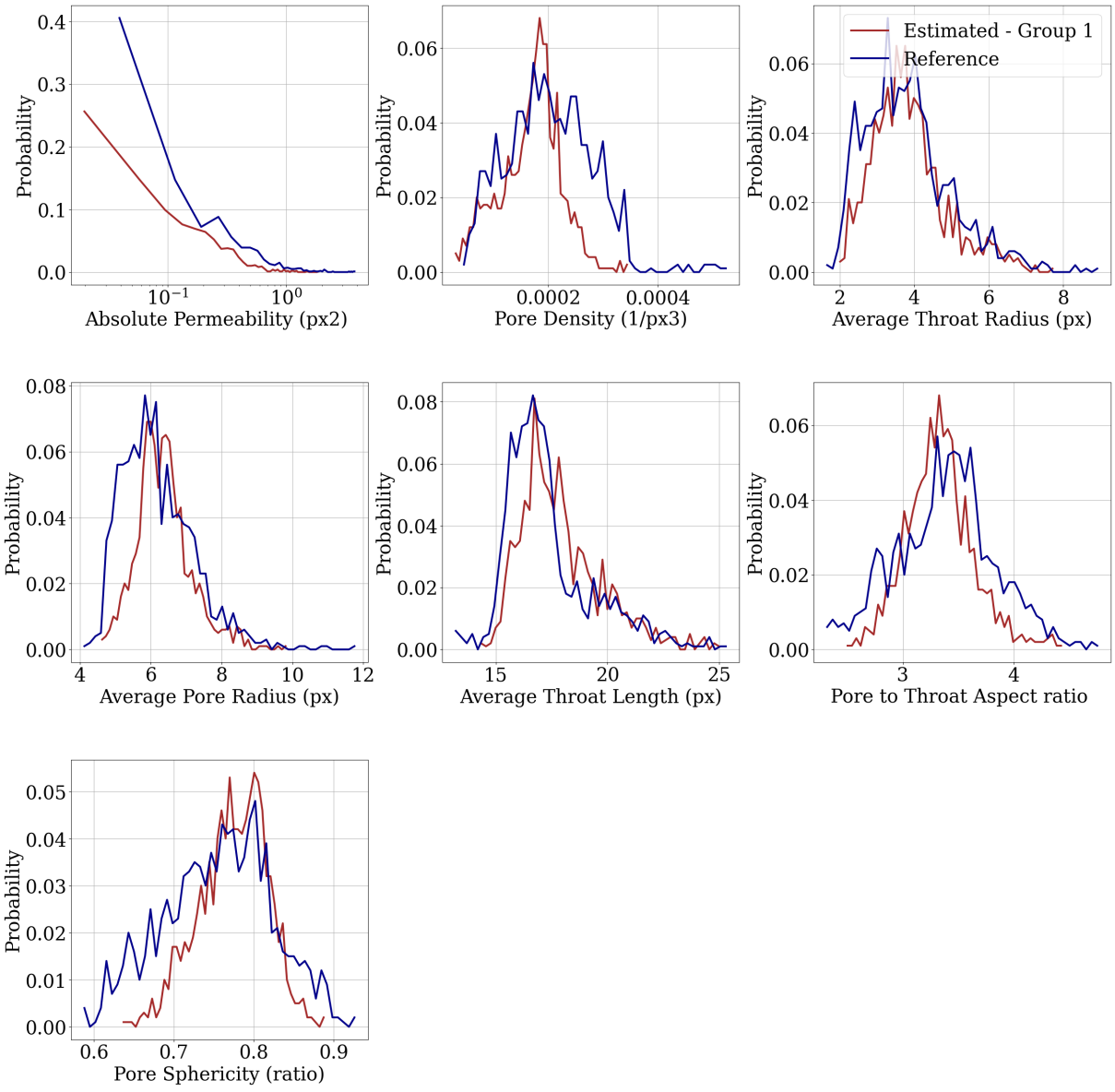

Figure C.5: DeePoreRevised2, 6 slices, Group 1 properties

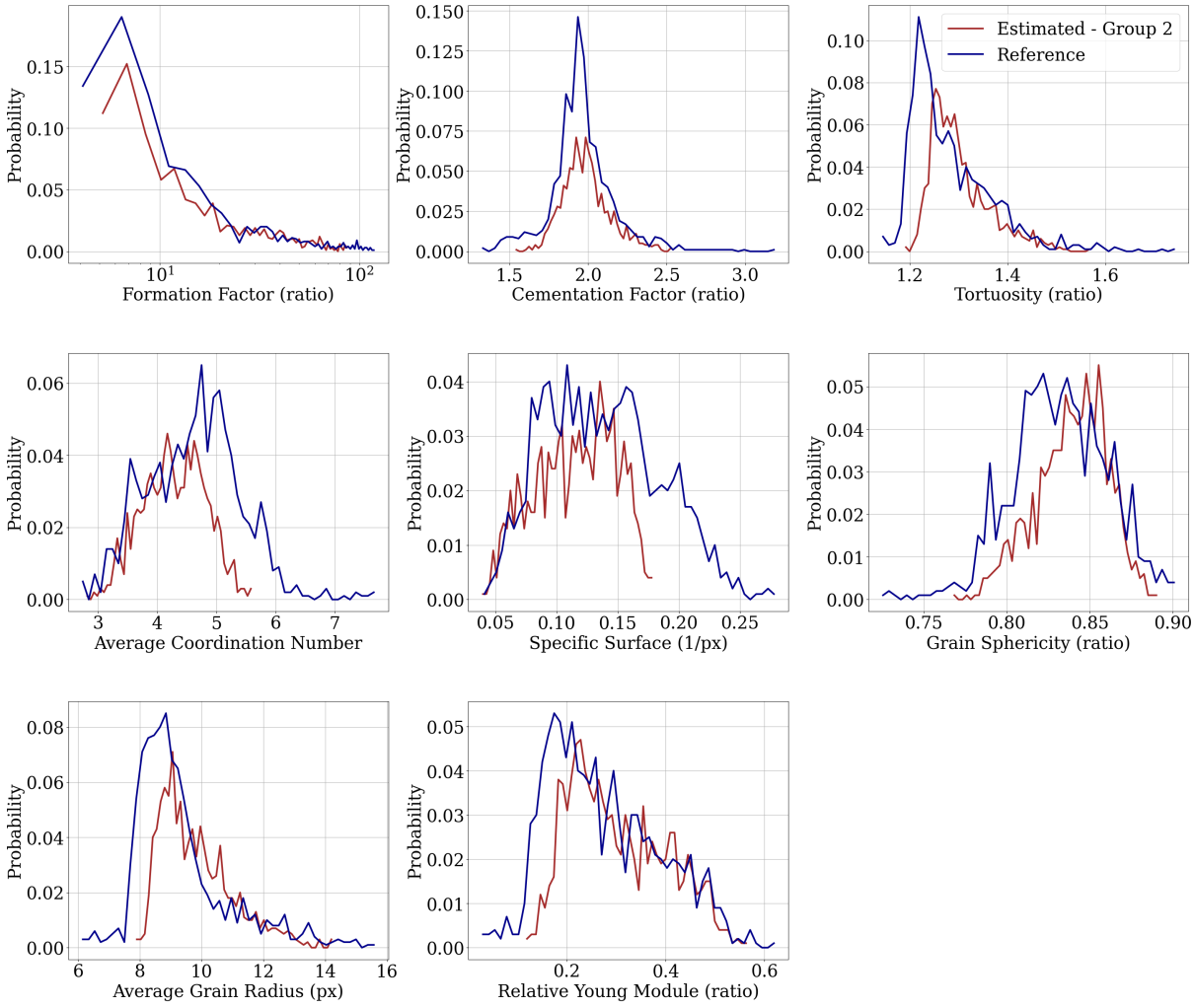

Figure C.6: DeePoreRevised2, 6 slices, Group 2 properties

Appendix D

Single Value Reference/Estimated Scatter Plots

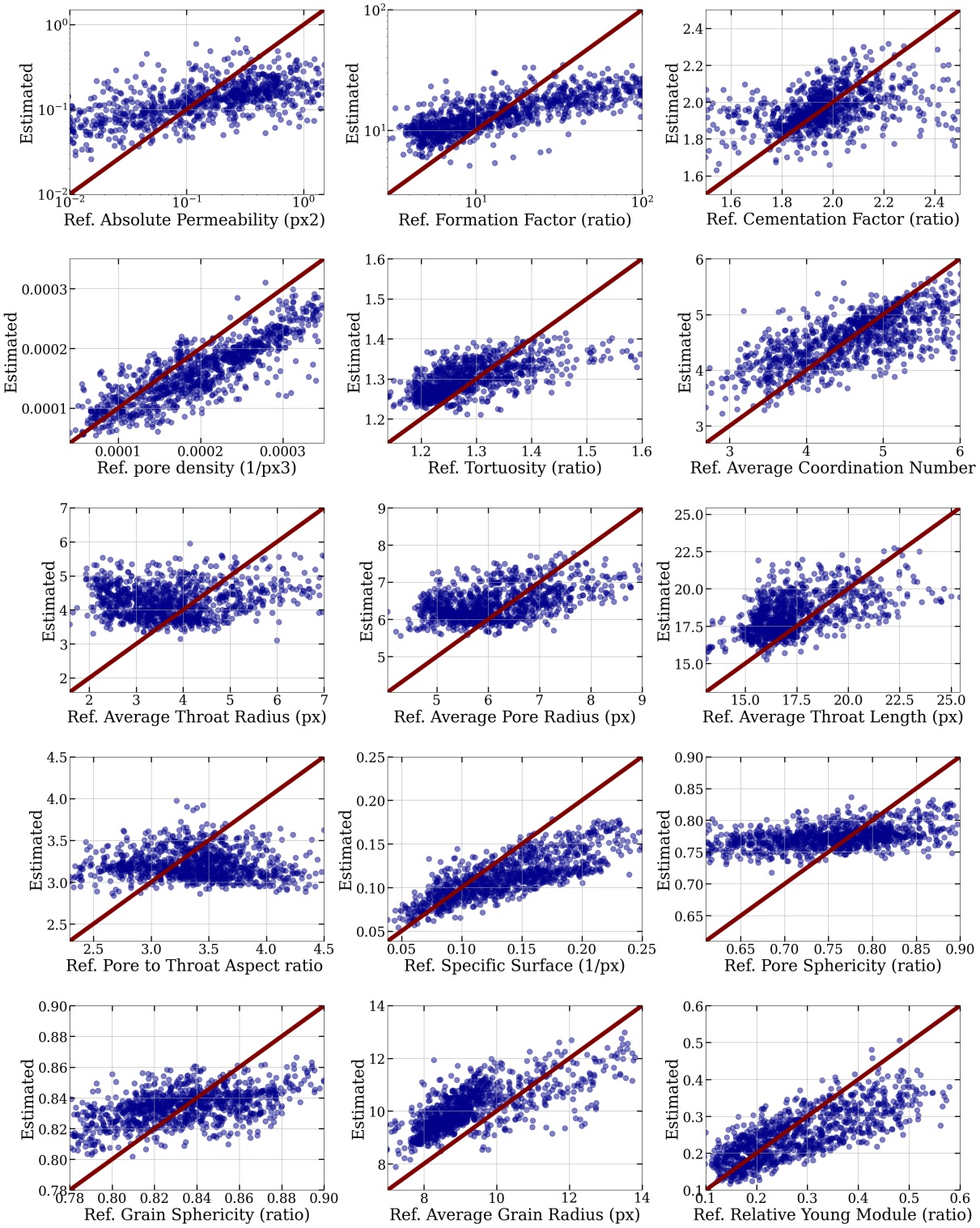

Figure D.1: DeePore3, 3 slices

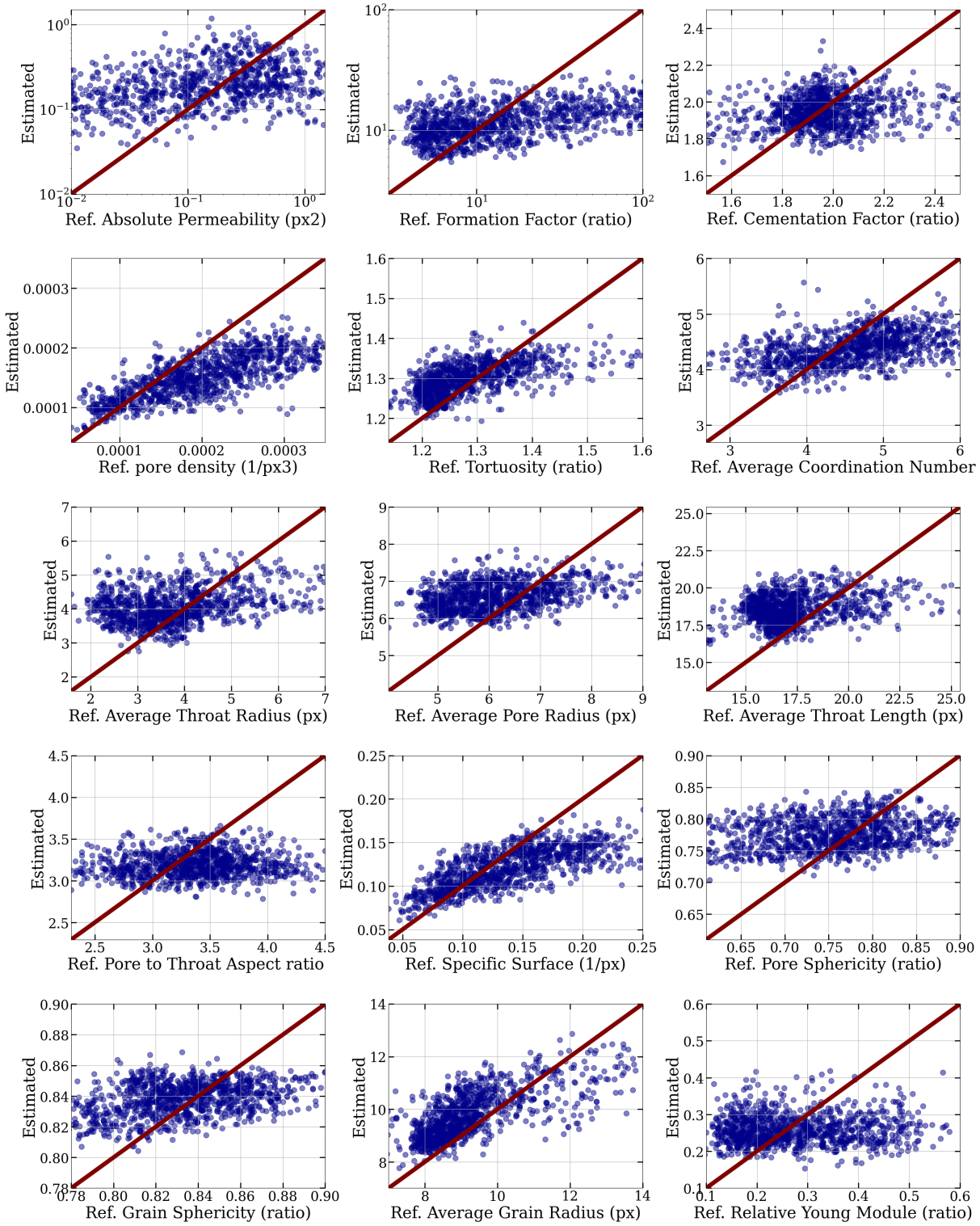

Figure D.2: DeePoreRevised1, 3 slices

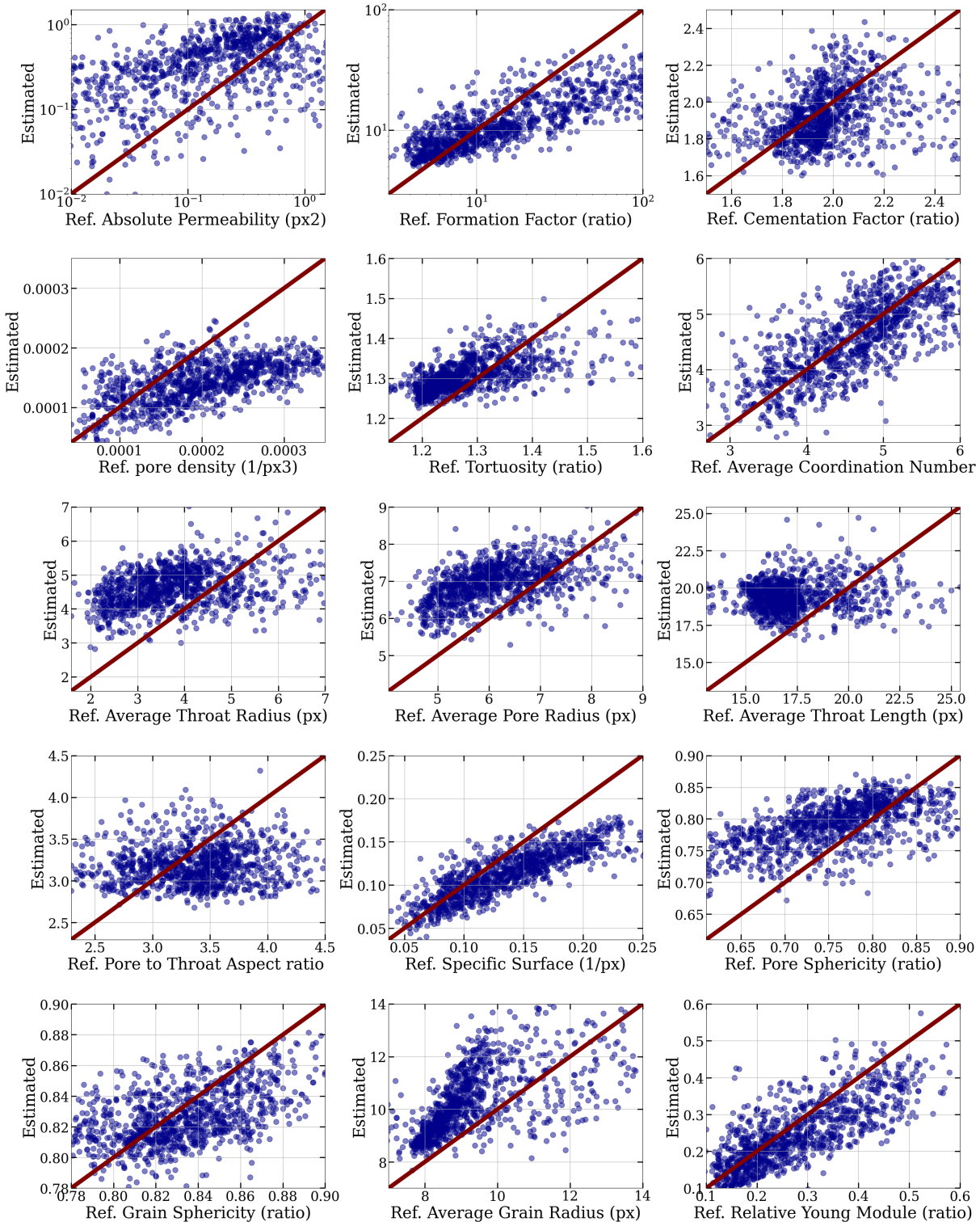

Figure D.3: DeePoreRevised1, 6 slices

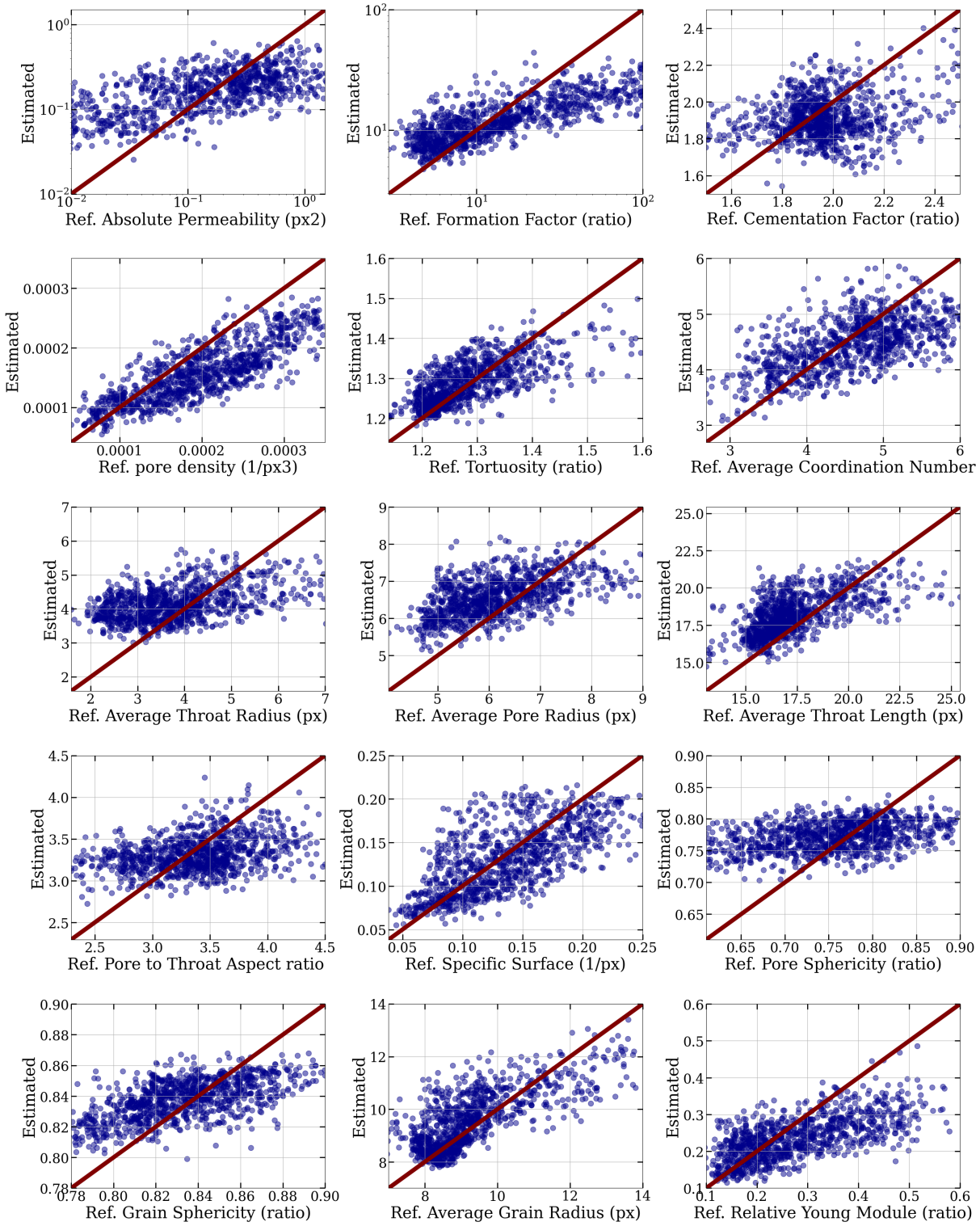

Figure D.4: DeePoreRevised2, 3 slices

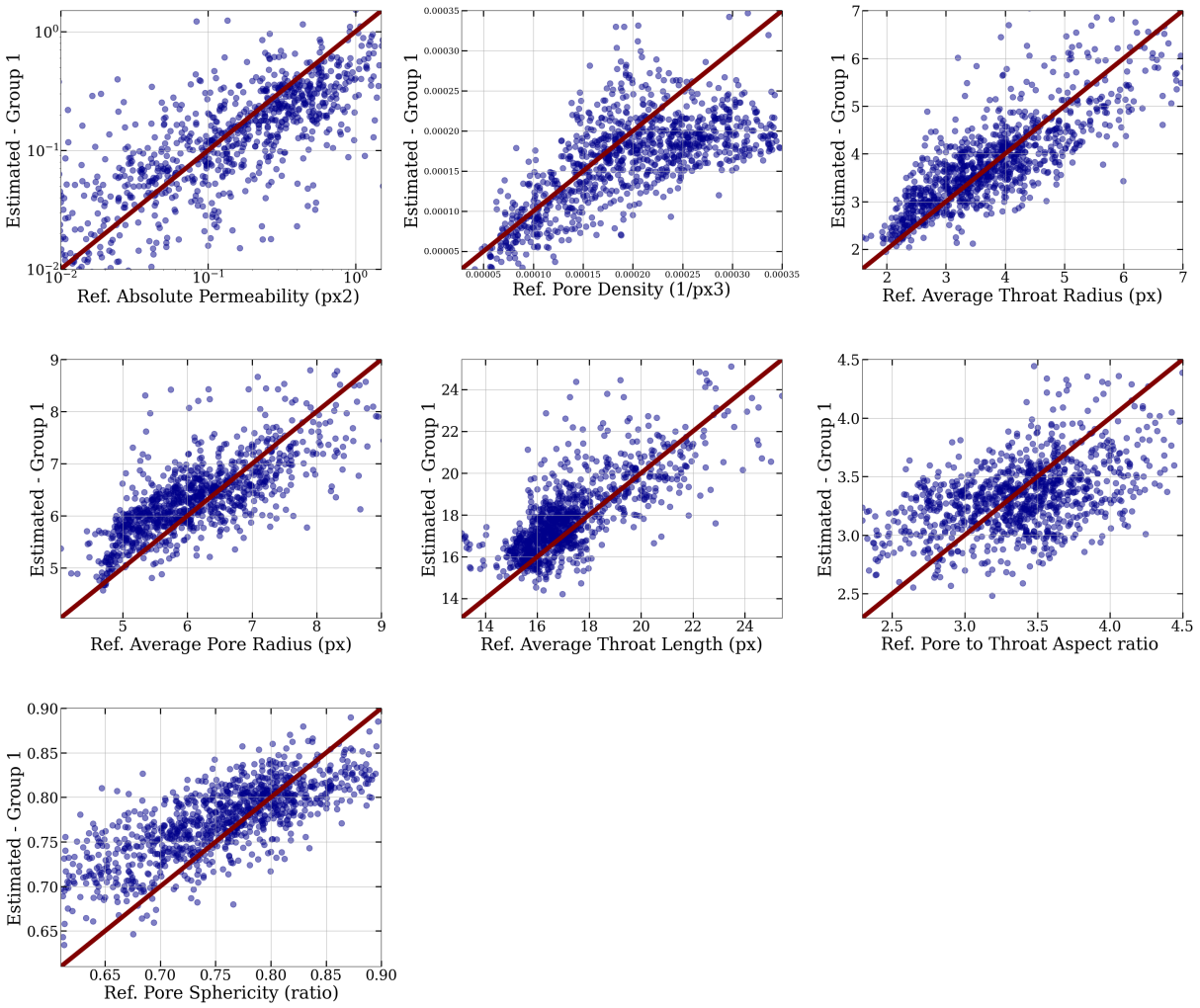

Figure D.5: DeePoreRevised2, 6 slices, Group 1 properties

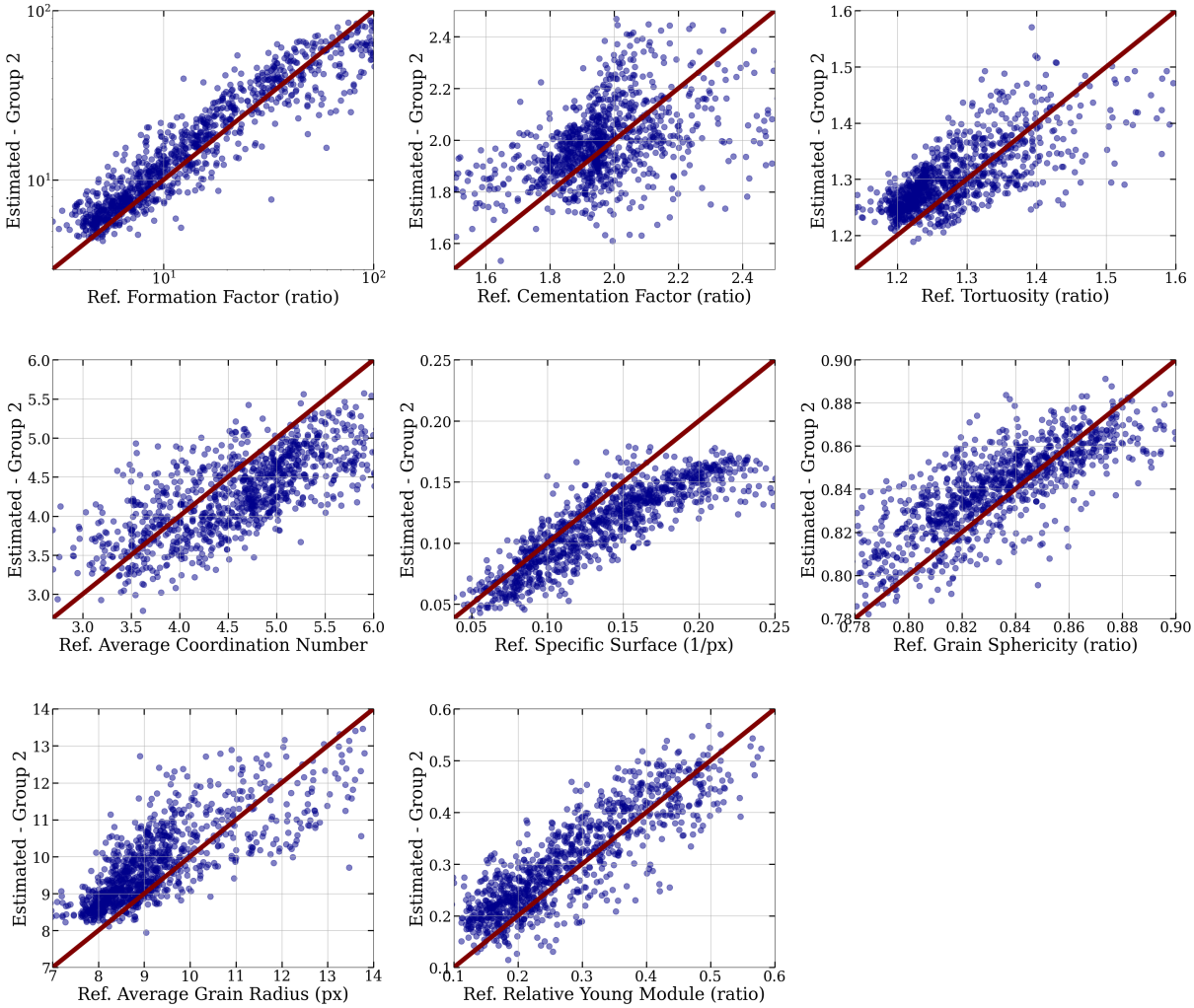

Figure D.6: DeePoreRevised2, 6 slices, Group 2 properties

Appendix E

Functions and Distributions Graphs

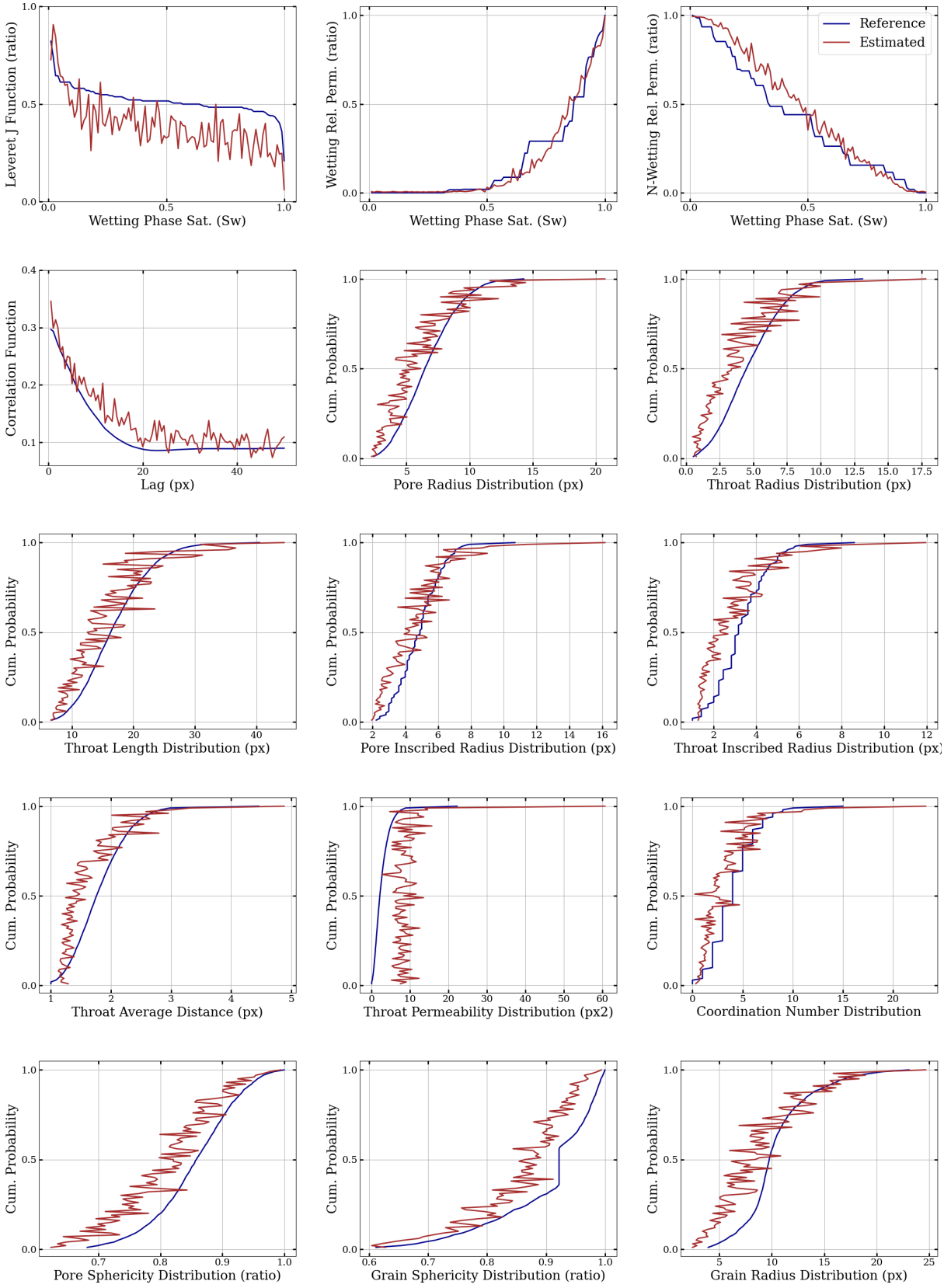

Figure E.1: DeePore3, 3 slices

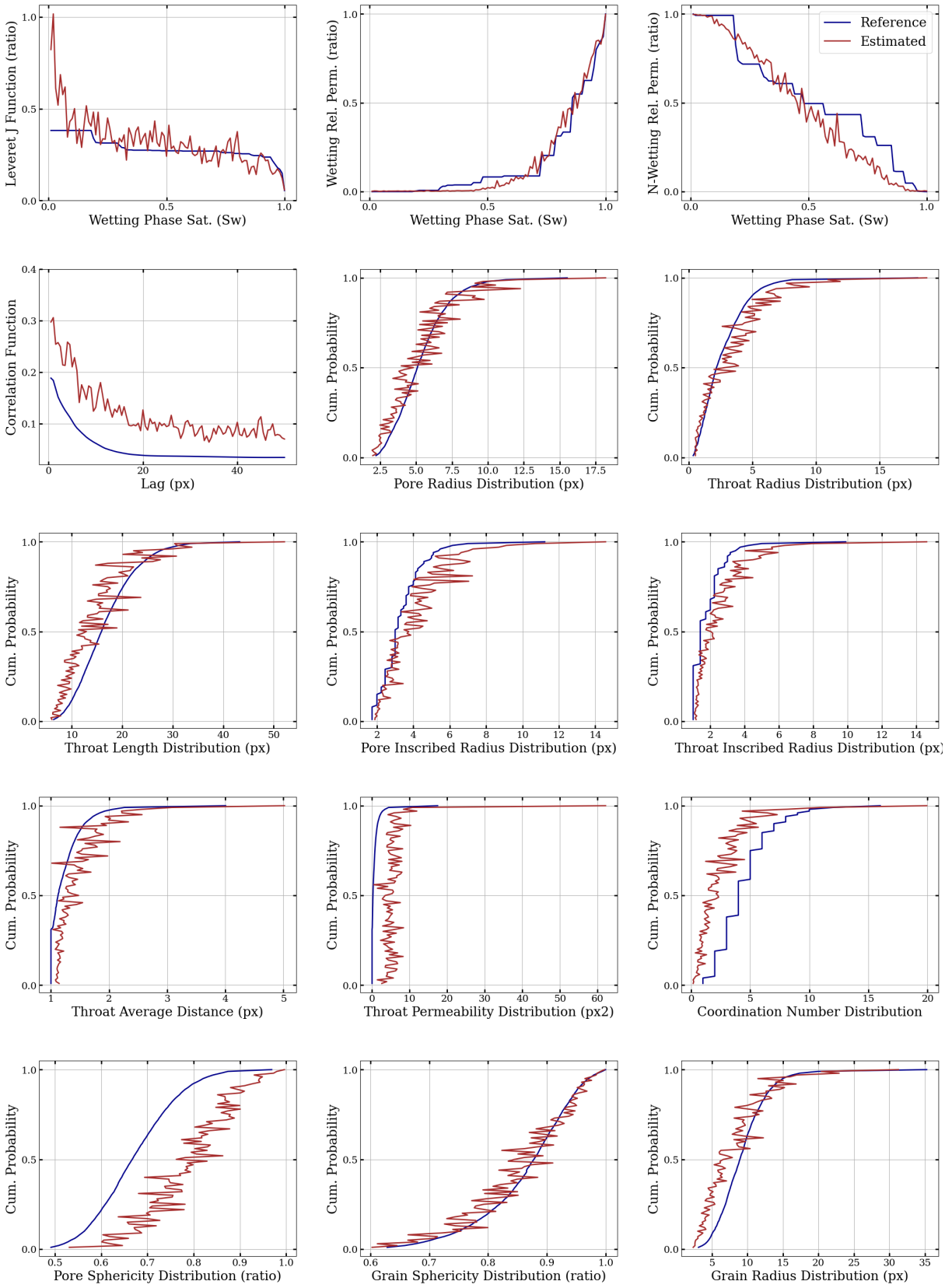

Figure E.2: DeePoreRevised1, 3 slices

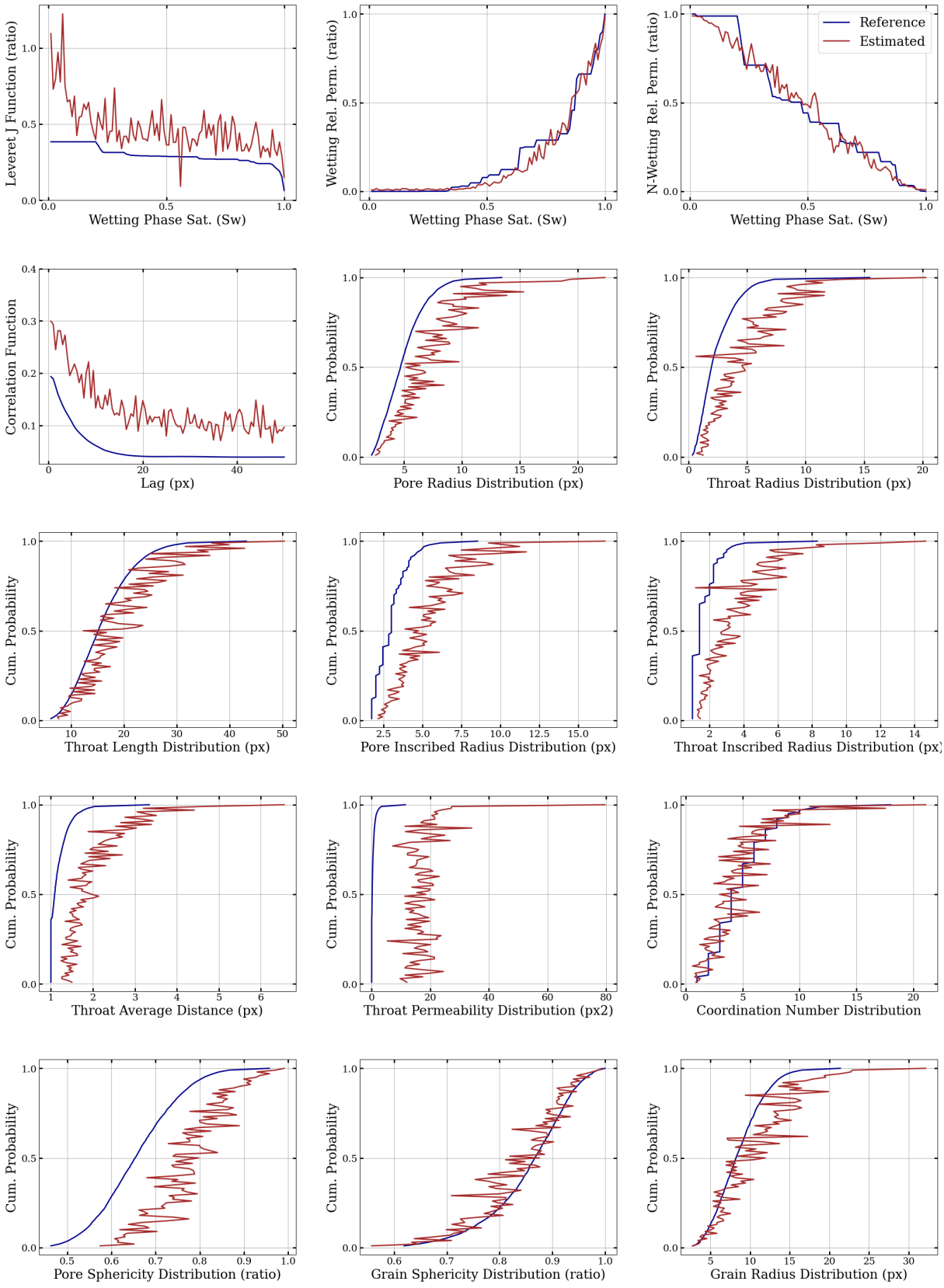

Figure E.3: DeePoreRevised1, 6 slices

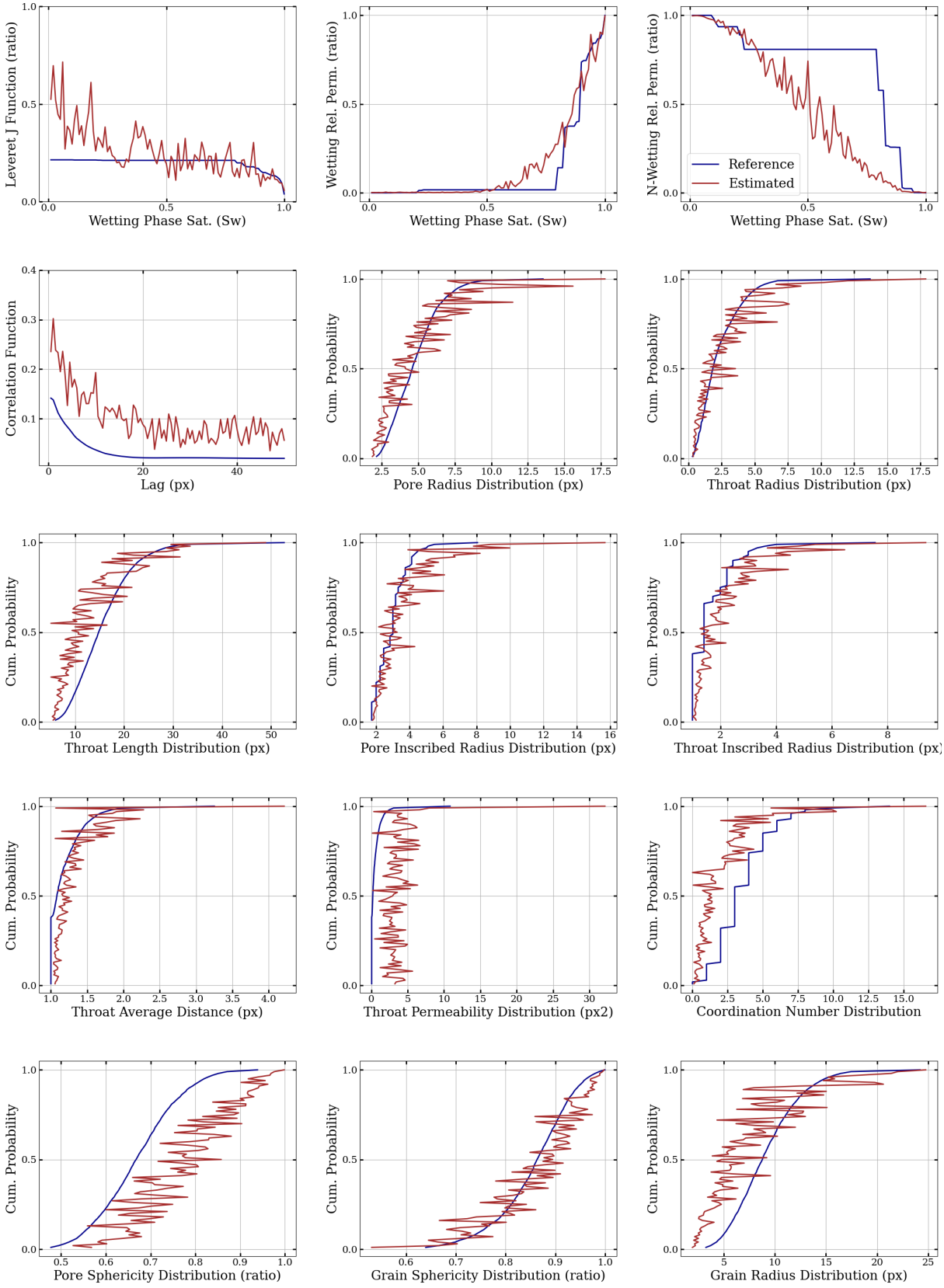

Figure E.4: DeePoreRevised2, 3 slices

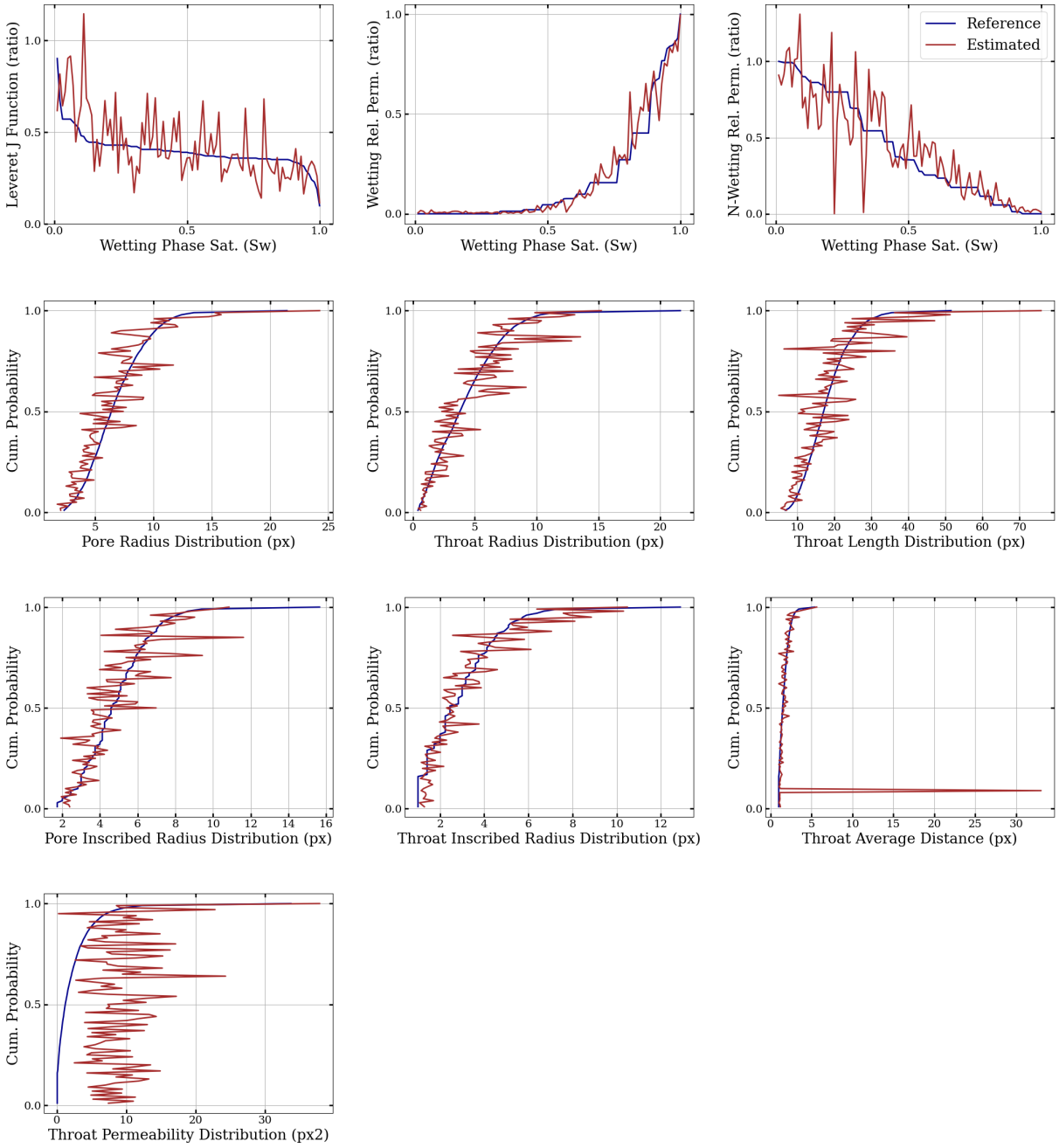

Figure E.5: DeePoreRevised2, 6 slices, Group 1 properties

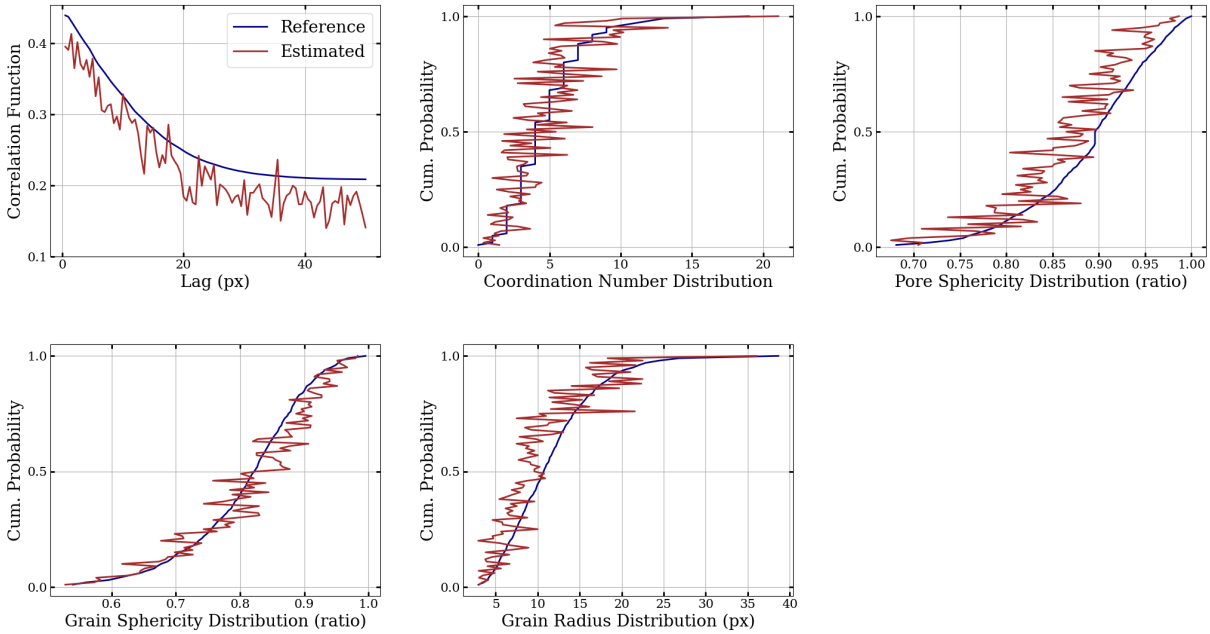

Figure E.6: DeePoreRevised2, 6 slices, Group 2 properties