Prediction of Capillary Pressure and Relative Permeability Curves using Conventional Pore-scale Displacements and Artificial Neural Networks

By
Siyan Liu

Submitted to the graduate degree program in Department of Chemical & Petroleum Engineering and the Graduate Faculty of the University of Kansas in partial fulfillment of the requirements
for the degree of Master of Science.

Chair: Dr. Amirmasoud Kalantari-Dahaghi

Dr. Laurence Weatherley

Dr. Shahin Negahban

Dr. Russell Ostermann

Dr. Arsalan Zolfaghari

Date Defended: 30 November 2017

The thesis committee for Siyan Liu certifies that this is the approved version of the following thesis:

Prediction of Capillary Pressure and Relative Permeability Curves using Conventional Pore-scale Displacements and Artificial Neural Networks

Chair: Dr. Amirmasoud Kalantari-Dahaghi

Date Approved: 30 November 2017

Table of Contents

List of Figures

List of Tables

Abstract

Traditional network models use simplified pore geometries to simulate multiphase flow using semi-analytical correlation-based approaches. In this work, we aim at improving these models by (I) extending the numerical methodologies to account for pore geometries with convex polygon cross sections and (II) utilizing Artificial Neural Networks (ANN) to predict flow-related properties. Specifically, we simulate fluid displacement sequences during a drainage process in bundles of capillary tubes with randomly generated convex polygon cross-sections. In the beginning, we assume that capillary tubes are fully saturated with water and that they are strongly water-wet. Then, oil is injected to displace water during the primary drainage process. The model calculates threshold capillary pressures for all randomly generated geometries using Mayer-Stowe-Princen (MS-P) method and the minimization of Helmholtz free energy for every pore-scale displacement event. Knowing pore fluid occupancies, we calculate saturations, phase conductances, and two-phase capillary pressure and relative permeability curves. These parameters are then used as input to train an ANN. ANN theories and related applications have been significantly promoted due to the fast increasing performance of computer hardware and inheratively complicated nature of some research areas. Various Artificial Intelligence (AI) applications have been developed specifically for the oil and gas industry such as AI assisted history matching, oil field production and development predictions, and reservoir characterization. The objective of this study is to develop an ANN training and predicting workflow that can be integrated with the conventional pore network modeling techniques. This hybrid model is computationally much faster which is beneficial for large-scale simulations in 3D. It could

also be used to improve prediction of flow-related properties in similar rock types. Specifically, we are interested in the training of ANNs to predict threshold capillary pressures and multi-phase flowrates as a function of cross-sectional shapes and wettabilities given for each capillary tube of the bundle. To do so, we have generated multi-phase flow properties for two large datasets consisting of 40,000 and 60,000 capillary tubes each. The predictive capability of the ANN is gauged by performing some quality control steps including blind test validations. We present the results primarily by demonstrating the calculated errors and deviations for any randomly generated bundles of capillary tubes from the aforementioned dataset. We show that generating high-quality training dataset is critical to improving model's predictive capabilities for a wide range of pore geometries, e.g., shape factors and elongations. Additionally, we demonstrate that feature selection and preprocessing of the input data could significantly impact ANN's predictions. We analyze a wide range of structures for the ANN models. The Multi-layer perceptron (MLP) Neural Network with three hidden layers is adequate for dealing with the complexity and non-linearity of most of our studied cases. This model is approximately an order of magnitude faster than conventional direct calculations using a personal desktop computer with four cores CPU. Such improvement in the speed of calculations becomes extremely important when dealing with larger models, adding more dimensionality, and/or introducing pore connectivity in 3D.

Acknowledgement

I would like to thank my graduate advisors Dr. Amirmasoud Kalantari-Dahaghi for his motivation and support in my research project and thesis, also I would like to thank Dr. Laurence Weatherley for his guidance during my first year at KU, I am grateful to Dr. Arsalan Zolfaghari for his help with my thesis, and I appreciate the time, comments from rest of my thesis committee members Dr. Shahin Negahban and Dr. Russell Ostermann.

I am also grateful to the recommendations and help from the friends in our research group including Dr. Zhi Zhong, Dr. Mohammad Kazemi, Nijat Hakimov, Yuhao Yang and Yudan Li. I would also thank all the lovely friends at KU and Lawrence that give me truly memorable experiences.

I would extend my appreciation to my former advisor Dr. Ping Wang in Chang'an University and Zach Chen from North Carolina for their continuous support and guidance.

At last, I dedicate this thesis to my parents for giving me endless love, encouragement, and support.

1. Introduction and Problem Statement

The first chapter includes an introduction to pore-scale network modeling (PNM) techniques and Artificial Neural Network (ANN) methodology. First, we start with the introduction of calculating threshold capillary pressures and relative permeability curves using PNM techniques. We then shift gears by introducing different aspects of Neural Network models (e.g., pre- and post-processing steps) and demonstrating some of their results with respect to the petroleum engineering.

1. Pore-scale network models

Characterization of multi-phase fluid flow through porous media is particularly important for a wide range of applications related to the industry or academic research settings (Blunt, 2017) Porous media can range from geologic formations to synthetic materials designed for a particular purpose (e.g., gas separation (Sharak, Samimi, Mousavi, & Bozarjamhari, 2014), catalysis, etc). With respect to the geologic formations, the challenge arises because of the existence of a wide range of scales, pore topologies, wettability, and hysteresis. These parameters impact fluid flow behaviors at larger scales of interest. An improved understanding of the underlying physics could help us to develop more efficient and reliable models to not only predict but also classify flow-related properties in a wide range of different porous media.

Traditionally, large-scale reservoir simulations solve Partial Differential Equations (PDE), arising from continuity equations, using various schemes such as Finite Difference Method (FDM), Finite Volume Method (FVM) and Finite Element Method (FEM) (Kuwauchi, Abbaszadeh, Shirakawa, & Yamazaki, 1996). In these formulations, macroscopic flow

properties, such as capillary pressure and relative permeability curves are required to update parameters during large-scale simulations (Zolfaghari Shahrak, 2014). These parameters have to be either measured experimentally or modeled numerically. Experimental measurements usually entail uncertainty. They are also extremely time-consuming (Alizadeh & Piri, 2014; Arshadi, Zolfaghari, Piri, Al-Muntasheri, & Sayed, 2017; Drla, Pope, Sepehrnoorl, & Texas, 1993; Honarpour, Koederitz, & Harvey, 1986; Oak, Baker, & Thomas, 1990).

As a result, physically based modeling techniques are developed for the simulation of pore-scale fluid/fluid displacements, and hence, prediction of the capillary pressure and relative permeability curves. The latest developments on pore network models are listed elsewhere (Aghaei & Piri, 2015; Blunt, 1998; Piri & Blunt, 2005a, 2005b; Ryazanov, van Dijke, & Sorbie, 2009; Arsalan Zolfaghari & Piri, 2017a, 2017b). All of these models have one thing in common, i.e., simplified geometries are used for each pore and throat in the network. The model relies on the interconnection of pores and throats to accurately represent the inherently sophisticated pore space topology in 3D. Building a representative network of pores is not a trivial task, as there is no single algorithm that can be used for the extraction of representative networks from different rock samples (Dong, 2007; Gesho, M., Zolfaghari, A., Piri, M., Pereira, n.d.).

Researchers use a variety of methods to calculate threshold capillary pressures of given displacements in capillary tubes. One of the most commonly used methods is MS-P analysis which is based on the minimization of Helmholtz free energy of the corresponding displacement (Lago & Araujo, 2001; Mason & Morrow, 1984; Mayer & Stowe, 1965; Princen, 1969a, 1969b, 1970) Various approaches were proposed by researchers to

calculate pore fluid configurations for different porous media. Thermodynamically consistent threshold capillary pressure for every possible pore fluid configuration is primarily used by researchers to determine the possible pore fluid occupancies under a wide range of flow conditions (Van Dijke & Sorbie, 2006; A. Zolfaghari & Piri, n.d.-a, n.d.-b).

Most researchers used simple regular geometries in their proposed pore network models. Empirical correlations have been proposed to calculate threshold capillary pressures based on geometrical properties of pore elements (D. Fenwick & Blunt, 1998; D. H. Fenwick & Blunt, 1998; Van Dijke et al., 2007; Van Dijke, Lago, Sorbie, & Araujo, 2004; Van Dijke & Sorbie, 2003). On the other hand, in their recent work, Zolfaghari and Piri (Arsalan Zolfaghari & Piri, 2017a, 2017b) have adopted MS-P method to calculate entry pressures for any piston-like displacements relevant to two- and three-phase flow under mixed-wet wettability conditions. They particularly used irregular triangular, square, and circular cross-sectional shapes and presented the corresponding analytical equations covering a wide range of displacements under capillary dominated flow regimes.

In this work, we extend the application of MS-P analysis to pore cross-sectional shapes of randomly generated convex polygons during drainage. We use the calculated threshold entry capillary pressures to determine pore fluid occupancies and hence, phase conductances in a bundle of parallel capillary tubes. We determine saturations and water and oil relative permeabilities following the calculation of multi-phase fluid conductances for each capillary tube. The detail of such calculations are listed elsewhere (M. H. Hui & Blunt, 2000; Piri, 2003). This model constitutes our base model which is then used to train Artifical Neural Networks (ANN) in the second part of this project. In this work, we

propose two ANN approaches to estimate threshold capillary pressure and relative permeability curves in a bundle of randomly generated capillary tubes with any arbitrary convex polygon cross-sectional shapes.

2. Machine Learning, Neural Network, Deep Learning concepts.

Neural Network originates from the so-called 'perceptron' (Rosenblatt, 1958) which contains 3 layers (output layer, input layer, and one hidden layer), and can approximate simple function. Until the 1980s the Multi-Layer Perceptron (MLP) was proposed(Rumelhart, Hinton, & Williams, 1986) that can somewhat overcome the disadvantages of perceptron, MLP is harness feed forward signal transport scheme and `Sigmoid`, `Tanh`, and so forth. activation functions within neurons to address the nonlinearities, the Back-propagation (BP) algorithm (Werbos, 1990) was used to back propagate the gradient for weights readjustment during training process. Then MLP with more layers was believed to have stronger capabilities to deal with more sophisticated real-life problems (Bengio, 2009). However, more problems were encountered while the network is becoming deeper such as the solutions will be 'trap' with local minima and the preferred global minima is rarely reached (G.E. Hinton and R. Salakhutdinov, 2006a). Another big issue is gradient vanishing during back-propagation when widely used `Sigmoid` activation functions were applied in the deeper network (J. Hochreiter, 1991; S. Hochreiter & Frasconi, 2009). *Hinton et al.* (G.E. Hinton and R. Salakhutdinov, 2006b) improved the network performance regarding the gradient vanishing issue in a seven hidden layers network by using a 'pretraining' technique. Along with some new activation functions such as `Maxout`, `ReLU`, `Leaky ReLU`, and so forth. which can be used to avoid gradient vanishing or explode in the deep neural network (I. J. Goodfellow, Warde-Farley,

Mirza, Courville, & Bengio, 2013; Maas, Hannun, & Ng, 2013). The deep residual network even as deep as more than 100 layers without gradient vanishing issues in image recognition task (He, Zhang, Ren, & Sun, 2016).

The Neural Network techniques we mentioned above perform well in various applications, but the neurons in each layer are connected to all neurons with adjacent layers (called fully connected the network). This would result in a large number of weights or parameters needed to be updated during training, and the network has strong probabilities to be over-fitted and becomes less generalized for various tasks.

Different Neural Network structures and layers with different functionalities are continuously being developed such as Convolutional Neural Network (CNN) (LeCun, Bottou, Bengio, & Haffner, 1998; Name et al., 1998), Recursive Neural Network (RvNN), Recurrent Neural Network (RNN) (Frasconi, Gori, & Sperduti, 1998; Sperduti & Starita, 1997), Gate Recurrent Unit (GRU) (Cho et al., 2014), Long-Short Term Memory Network (LSTM) (S. Hochreiter & Urgen Schmidhuber, 1997), Sequence to sequence learning model(Sutskever, Vinyals, & Le, 2014), Generative Adversarial Network(GAN) (I. Goodfellow et al., 2014), to tackle more complex real-world problems. Recent proposed ‘capsules’ and ‘CapsNet’ concepts by *Hinton et al.* use ‘dynamic routing’ mechanism instead of back-propagation learning scheme. This might completely change the future path of the deep learning community. Along with the new network structures and concepts, various learning enhancement techniques such as `Adam` optimizer (Kingma & Ba, 2015), training batch normalization (Windows et al., 2014), neurons ‘dropout’ (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014) played important role in network generalization, stability, and robustness improvement.

3. Application of Machine Learning, Neural Networks in reservoir engineering and pore network modeling/simulations.

Due to nonlinearity of Neural Network and some "learning" capabilities in terms of regression trend analysis, pattern recognition, feature extraction, multi-class classification and generalization, Artificial Neural Network is becoming more and more popular than ever before in oil and gas industry and research domain. Petroleum engineering or reservoir engineering subsurface problems are extremely complex and non-linear in terms of the rock and fluid interactions. One solution for one scenario is good does not mean it will fit into another scenario even they are similar based on our knowledge and understanding. The fact of Artificial Neural Network will allow the engineer to produce approximate results within acceptable tolerance for a specific problem in oil and gas industry, such as the permeability prediction based on well loggings, hydraulic fracturing performance evaluation, and prediction (Mohaghegh & Ameri, 1995).

Simple Artificial Neural Network driven by back-propagated training scheme was used to predict relative permeabilities from basic rock and fluid properties such as water saturation, porosity, interfacial tension, and so forth. (Al-Alawi, Kalam, & Al-Mukheini, 1996; Edris Joonaki, 2013; Guler, Ertekin, & Grader, 1999) *Hamada et al.* created Neural Network to estimate reservoir characterizations from seismic properties and conventional well log data (Hamada & Elshafei, 2010). *Jreou et al*. applied Neural Network to predict oil production of the entire oil field from basic wells information along with oil and water production/injection history (Jreou, 2012). A systematic workflow which leverages Neural Network to petroleum engineering and a case study was presented for water saturation prediction in Oman from various well logging data, and validated results by doing a comparison with conventional core measurements (Al-Bulushi, King, Blunt, & Kraaijveld,

2012). *Reichhardt et al.* use Neural Network in corporation with Kriging method to predict or improve the input data of conventional reservoir simulator. The porosity, permeabilities are predicted from well logging and seismic data in such way to do sparse data completion for better history matching simulation (Reichhardt & Isaiah, 2013). Some researchers harness Neural Network and its derivatives to replace conventional polynomial approximation or Kriging method which used in simulations (Bruyelle & Guérillot, 2014). *Chaki et al.* proposed Modular Artificial Neural Network (MANN) as a workflow which separates big task into small ones and deals them with the modular network, in the case study, the specific module is responsible to specific well log fraction, final reservoir characteristics are the combination of the separated predictions (Chaki, Verma, Routray, Mohanty, & Jenamani, 2014). *Jamshidian et al.* applied Multi-Layer Perceptron (MLP) network to predict Nuclear Magnetic Resonance (NMR) logging parameters instead of using conventional method. *Shokooh et al.* integrated wavelet (as activation function) and Artificial Neural Network to improve the permeability prediction performance (Shokooh Saljooghi & Hezarkhani, 2015). *Zhong et al*. use support vector machine (SVM) with mixed kernels function (MKF) ANN methods to create relationship between limited conventional well log suites and sparse core data, and predict CO_2-reservoir oil minimum miscibility pressure (Zhong, 2017; Zhong & Carr, 2016), *Gholanlo et al.* proposed a new radial based function neural network to predict water saturation in carbonate reservoir (Gholanlo, Amirpour, & Ahmadi, 2016). With the help of evolutional optimization algorithms, *Ahmadi et al.* combined the algorithms such as GA, PSO and Artificial Neural Network to predict oil-water relative permeability in the reservoir from various rock-fluid properties, e.g., formation type, wettability type, porosity, water saturation (Ahmadi,

Zendehboudi, Dusseault, & Chatzis, 2016). Besides the regression problems, Neural Network with unsupervised learning scheme is proposed to do automated rock image classifications (Shu, McIsaac, Osinski, & Francis, 2017).

In pore scale and pore network modeling as well as simulation related topics, some state-of-the-art techniques and approaches were proposed: *Miao et al.* developed a new way which utilizing Neural Network to estimate the hydraulic conductance from direct CFD process for several thousands of 2D capillary tube cross-section, which save lots of computation time by Neural Network prediction for new tube cross-section in acceptable tolerance (Miao, Gerke, & Sizonenko, 2017a). Applying state-of-the-art generative adversarial neural network to pore network image reconstruction and statistical analysis was proposed by *Mosser et al.*, the implicit realizations generate from the network can somewhat replace the conventional stochastic methods which used in pore network reconstruction (Mosser, Dubrule, & Blunt, 2017). *Wonjin Yun* used and compared Fully Connected Network (FCN) and Convolutional Neural Network (CNN) in micro-level fluid flow characterization in man-made micromodel fabrication and somewhat to predict the potential oil/water configurations in given pore network images (Yun, n.d.). *Rabbani et al.* utilize thin section images analysis to obtain pore network parameters and incorporate with Neural Network to do permeability estimation in carbonates (Rabbani, Assadi, Kharrat, Dashti, & Ayatollahi, 2017).

In this thesis, author is using Neural Network approach to estimate entry capillary pressures and oil-water relative permeabilities directly from capillary geometries such that basic pore network modeling concepts are used to create simple bundle of capillary tubes with randomly generated convex polygon cross-sections, and calculate the properties of

simplified two-phase flows in piston-like primary drainage process, Multi-Layer Perceptron (MLP) neural networks are generated to predict the capillary pressures of each capillary tubes and water/oil relative permeabilities at specific scenario. A systematic workflow includes capillary tubes generating, conventional entry pressure and relative permeabilities calculation method and MLP training and prediction for corresponding properties, results validation and comparison.

2. Methodology

2.1 Model description

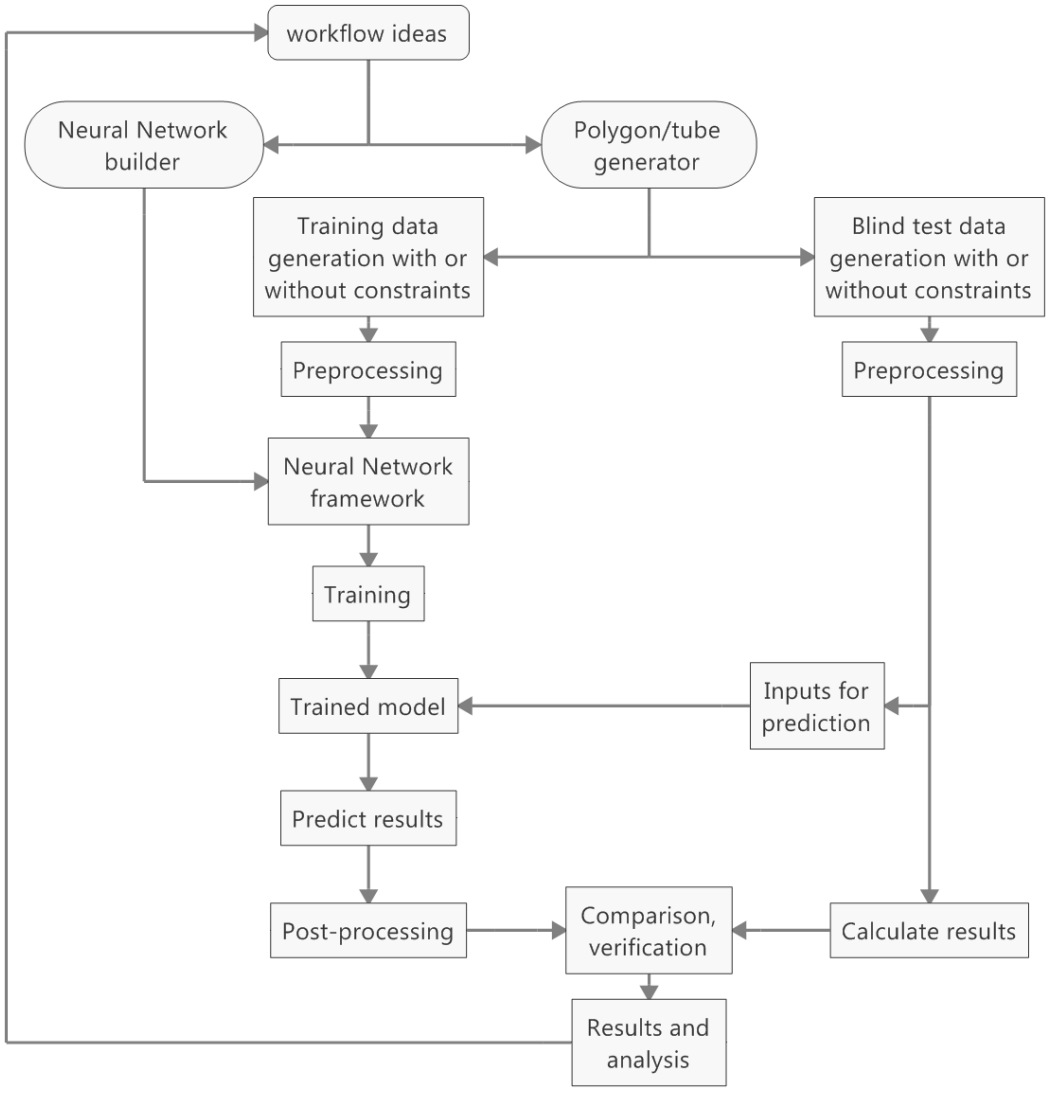

Figure 2.1 Overall workflow for the study

Randomly shaped convex polygons or randomly shaped convex polygons with specific constraints were generated. Basic statistical analysis of the generated polygons plus blind test polygons is conducted to control the generalization of the Neural Network. All tubes are filled with water after the threshold capillary pressure is reached, the Non-wetting oil

will invade the tube and occupy the center area of the tube, wetting phase water will be sitting at the corners from the cross-section view.

The configurations of the oil/water within each one of the capillary tubes are extract by apply thermodynamically consistent threshold capillary pressure calculations (Zolfaghari Shahrak, 2014), then the following water saturation is calculated for each one of the capillary tubes. Relative permeability curves are produced by a systematic screening process through all tubes to solve the dynamic flow rate for oil and water based one previously calculated threshold capillary pressures. Multi-layer perceptron (MLP) Neural Networks are created to train the results produced by direct calculations, then do blind prediction tests verifications to control the quality of the Neural Network. Neural Network structures and hyperparameters are tuned to get a better prediction in fewer computation costs. Also, the input sensitivity analysis is performed to reduce the unnecessary input parameters. Necessary data preprocessing before Neural Network training and postprocessing after prediction are performed based on different approach scenarios.

2.2 Data and model generation and preprocessing (Random Polygons)

2.2.1 Software APP for random polygon generation by using random number generator and some additional algorithms.

To generate a large amount of polygon cross-section areas with or without specific geometrical property constraints in our study, a small polygon generator application is developed to allow us easily to create desired polygons into easy to access data format. See Figure 2.2.

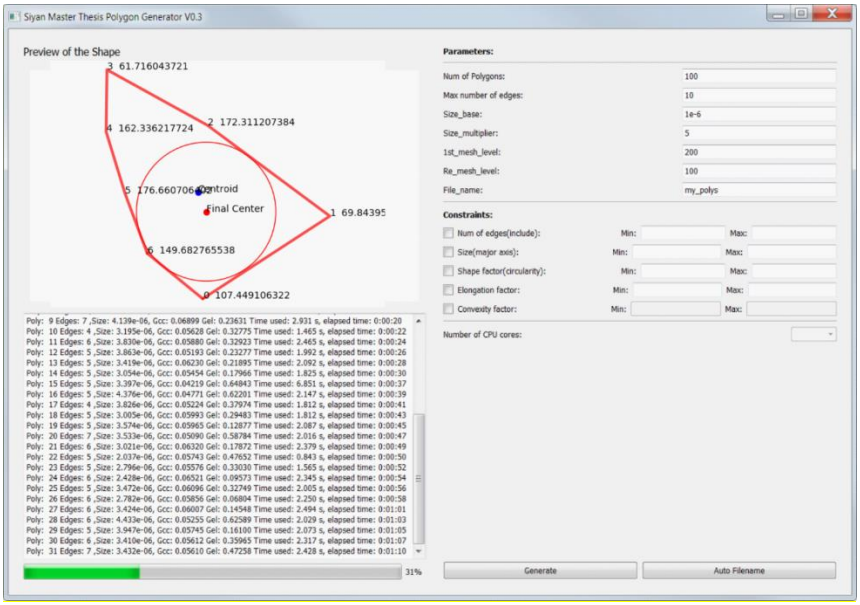

Figure 2.2 Snapshot of the random polygon generator used in this study

The input parameters and descriptions are listed below:

- Num_polys: Total number of polygons to be generated.
- Max_edges: Maximum possible edges/corners of each one of the generated polygons.
- Size_base and size_factor: The two parameters will determine the size of the generated polygons.
- Mesh_level and re-mesh level: These parameters will control the process of finding the biggest inscribed radius inside the individual polygon.
- File _name: After finishing all polygons generating, a structured dataset will be saved with the input file name.
- Constraints for the number of edges: Limit the minimum and the maximum number of edges generated by this application, ranging from 3 (triangle) to an unlimited number.

- Constraints for the size of the polygon: Any polygon generated with the major axis direction size which outside the input range will be deprecated, and re-generate another polygon until the requirement is satisfied and save to file.
- Constraints for shape factor: shape factor which can be called circularity, the valid range is from 0 to 0.7958 (circle).
- Constraints for elongation factor: Defined by (Miao, Gerke, & Sizonenko, 2017b), where W and L represent the width and length of the smallest found circumscribed rectangle, the factor ranging from 0 to 1.
- Constraints for convexity factor which defined as the perimeter of convex hull divide by the perimeter of the polygon (Miao et al., 2017b) is not available here because all of our polygons are convex polygon based on model assumptions, which result in same convexity factor.

All the constraints input can be work individually or combined to provide more flexibility in terms of polygon geometry.

After each one of the polygon candidate that meet all the requirements, the programme will run further to automatically the relate polygon geometrical properties such as the geometrical center of the polygon, all inner angles and corner half angles for individual polygon, area, perimeter. Also, the program will use a hybrid algorithm (see algorithm 1) to find the biggest possible inscribed circle inside the polygon and its center coordinates.

Algorithm 1 Finding largest inscribed circle algorithm

1. Create mesh grid based on the coordinates of corner points of the original polygon
2. Do 'binary erosion' for the mesh grid iteratively till the threshold for specific number/fraction of points left
3. Find the minimum rectangle which covers all points left from the previous step and does mesh grid on the rectangle.
4. Loop all meshed coordinates and calculate the distances between the testing point and all edges of the polygon.
5. Find the candidate point with the maximum value for 'minimum distance to all edges' and return the coordinates of the point and use that specific 'minimum distance to all edges' as the preferred radius.

2.2.2 Generated specific data structure and format for flexible processing and storage.

At this point, a complete single polygon generating process is finished, the program will repeat a certain number of times which is user input previously. Finally, all generated polygons with their calculated properties will be merged into structure part of a flexible data format, at the same time the input parameters will be merged into the unstructured part respectively.

There are two original input datasets generated by using the random polygon generator:

- Training dataset 1: The dataset contains 60,000 random polygons without any constraints. The maximum generating edges was set to 10; size base is 1E-6, size multiplier is 5, the first level of meshing is 200, 2nd level of meshing is 100.
- Training dataset 2: The dataset contained 40,000 polygons in total and merged into 8 parts, each one of those parts has 5000 polygons generated by using same generator parameters as training dataset 1, the 8 section has different constraints in

terms of polygon geometry, shape factor (circularity) belongs to 0 to 0.02, 0.02 to 0.04, 0.04 to 0.06, 0.06 to 0.07958 (circle), and elongation factor belongs to 0 to 0.25, 0.25 to 0.50, 0.50 to 0.75, 0.75 to 1.00 respectively. The reason to create the second dataset is trying to make training dataset more generalized that can provide enough information of those polygons with 'slim' or 'plate' like shape.

Constraints	Training Dataset 1	Training Dataset 2
All random	60,000	5,000
$\boldsymbol{g_{cc}} \in [\mathbf{0}, \mathbf{0.02}]$	0	5,000
$\boldsymbol{g_{cc}} \in [\mathbf{0.02}, \mathbf{0.04}]$	0	5,000
$\boldsymbol{g_{cc}} \in [\mathbf{0.04}, \mathbf{0.06}]$	0	5,000
$\boldsymbol{g_{cc}} \in [\mathbf{0.06}, \mathbf{0.07958}]$	0	5,000
$\boldsymbol{g_{el}} \in [\mathbf{0}, \mathbf{0.25}]$	0	5,000
$\boldsymbol{g_{el}} \in [\mathbf{0.25}, \mathbf{0.50}]$	0	5,000
$\boldsymbol{g_{el}} \in [\mathbf{0.50}, \mathbf{0.75}]$	0	5,000
$\boldsymbol{g_{el}} \in [\mathbf{0.75}, \mathbf{1.00}]$	0	5,000
Total tubes	60,000	40,000

Table 2.1 Summary of the training dataset used in this study

Meanwhile, five different blind verification datasets were generated for blind testing later, they have same generator parameters but using different constraints in terms of shape factor and elongation factor:

- Blind test dataset 1: 3,000 random polygons with shape factor from 0 to 0.04.
- Blind test dataset 2: 3,000 random polygons with shape factor from 0.04 to 0.07958.
- Blind test dataset 3: 3,000 random polygons with elongation factor from 0 to 0.50.
- Blind test dataset 4: 3,000 random polygons with elongation factor from 0.50 to 1.00.
- Blind test dataset 5: 3,000 random polygons without any shape constraints.

Blind test dataset	Constraints	Number of tubes
Blind test 1	$g_{cc} \in [0, 0.04]$	3,000
Blind test 2	$g_{cc} \in [0.04, 0.07958]$	3,000
Blind test 3	$g_{el} \in [0, 0.5]$	3,000
Blind test 4	$g_{el} \in [0.5, 1.0]$	3,000
Blind test 5	All random	3,000

Table 2.2 Summary of the training dataset used in this study

2.3 Datasets generated description and statistical analysis

After the polygon generation processes, several simple statistical analysis and plots were performed to check the data quality and to see if the distribution of the properties is satisfied.

2.3.1 Descriptions and statistical analysis for training dataset

First, we examined the dataset with 60,000 polygons generated without shape constraints and specific size and number of edges constraints.

(1) Statistical analysis(plots) for training dataset (40,000 or 60,000 polygons)

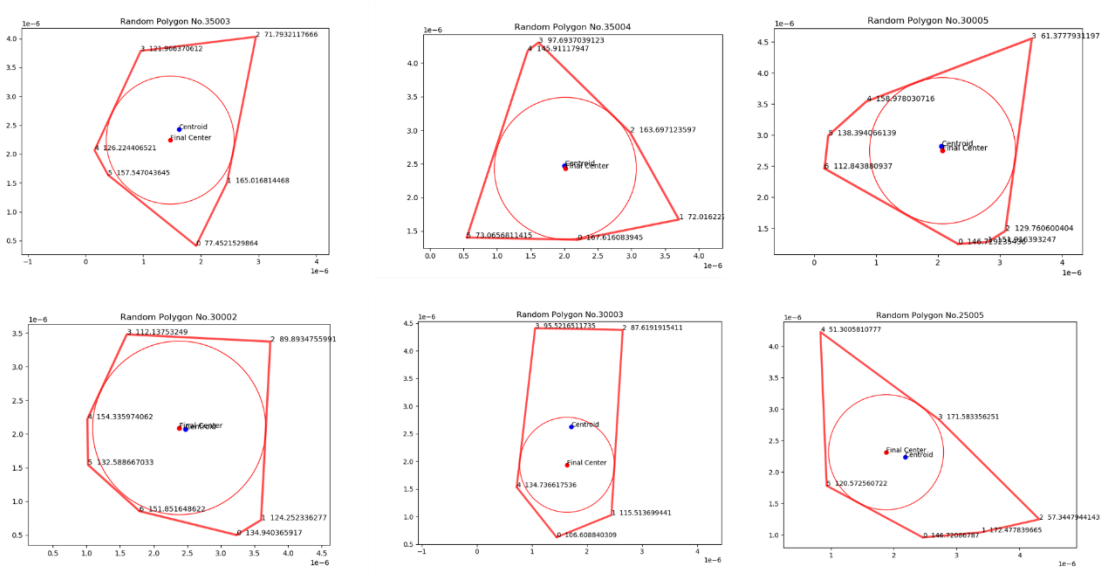

Figure 2.3 Selected polygons from training dataset 1

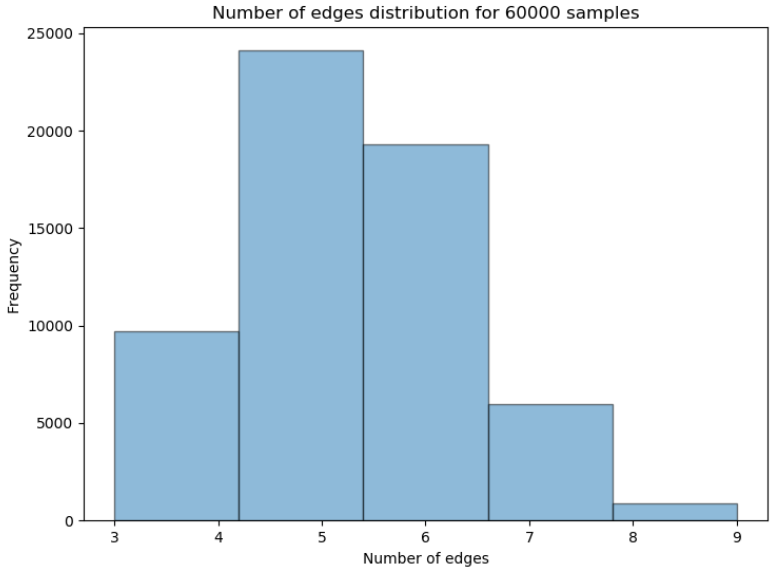

Figure 2.4 Distribution of number of edges of polygons within training dataset 1

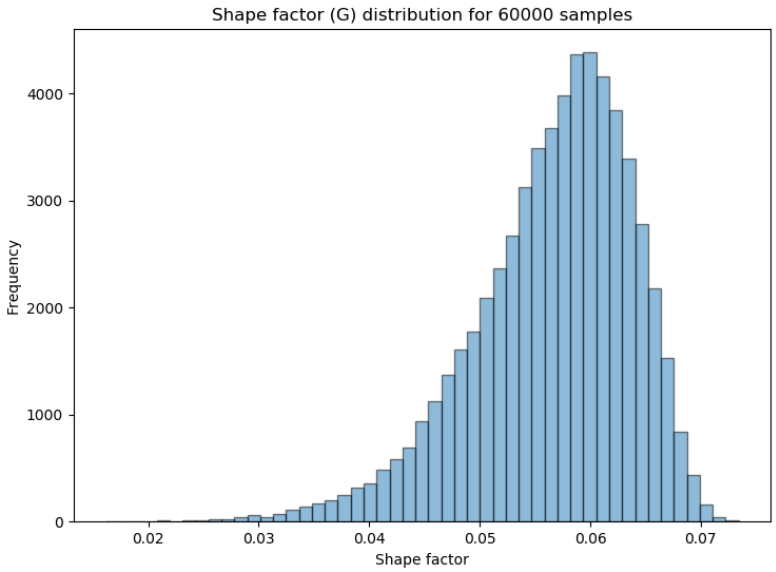

Figure 2.5 Distribution of shape factors (circularity) of polygons within training dataset 1

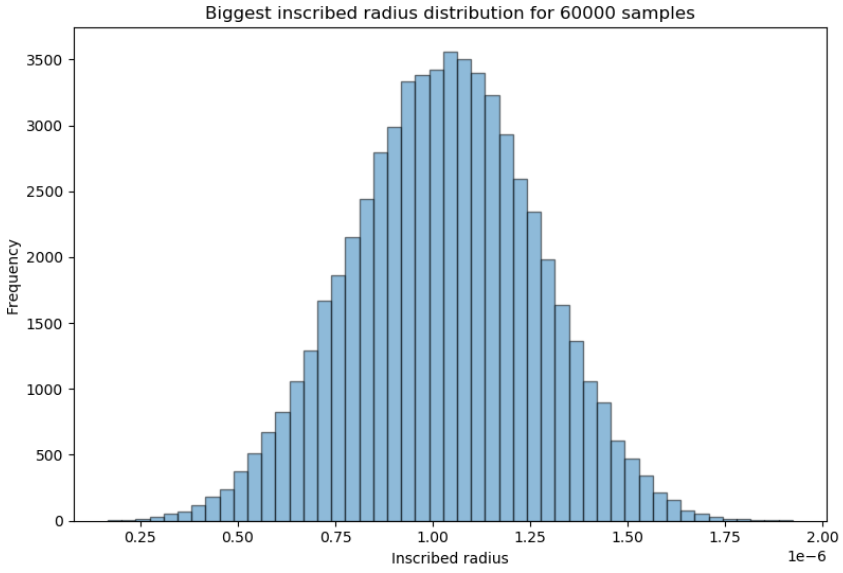

Figure 2.6 Distribution of largest inscribed radius inside polygons within training dataset 1

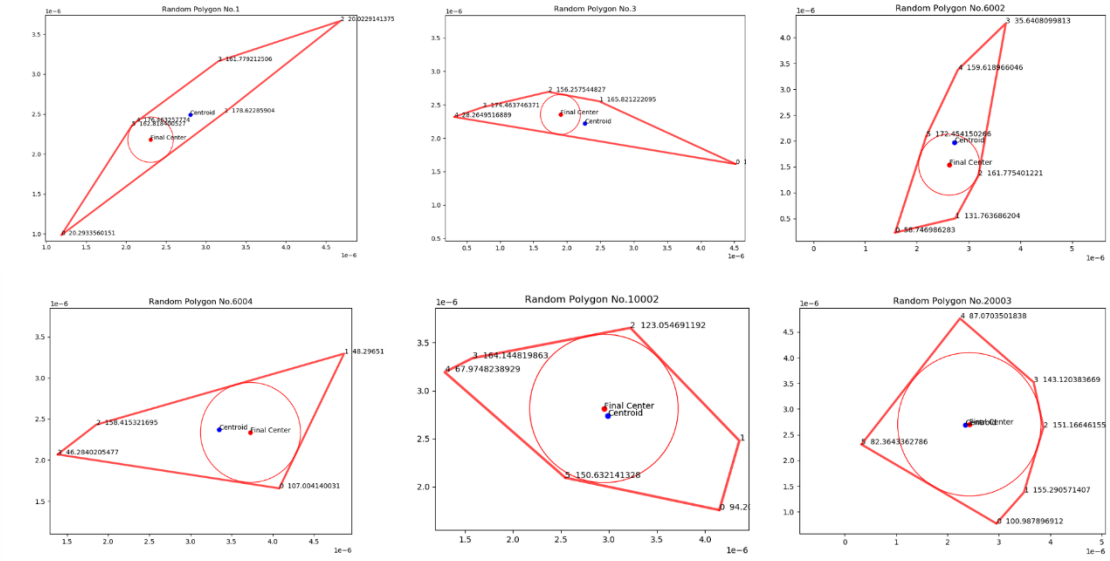

Figure 2.7 Selected polygons from training dataset 2

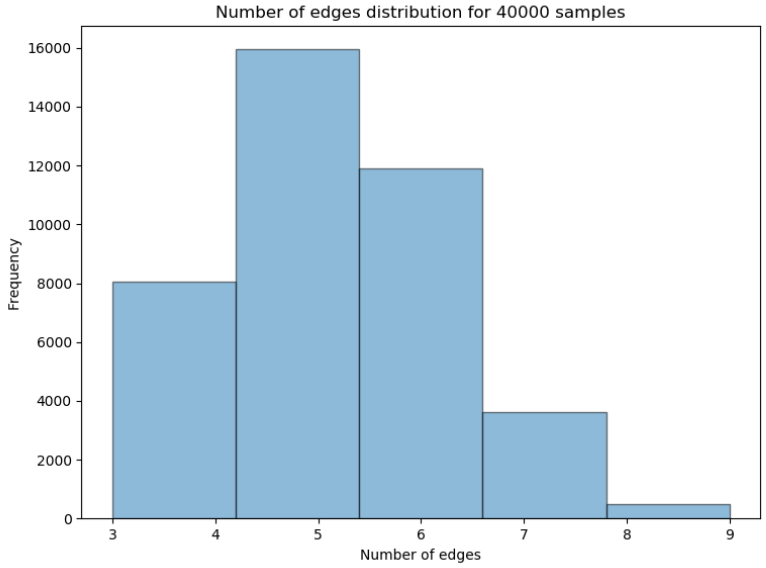

Figure 2.8 Distribution of number of edges of polygons within training dataset 2

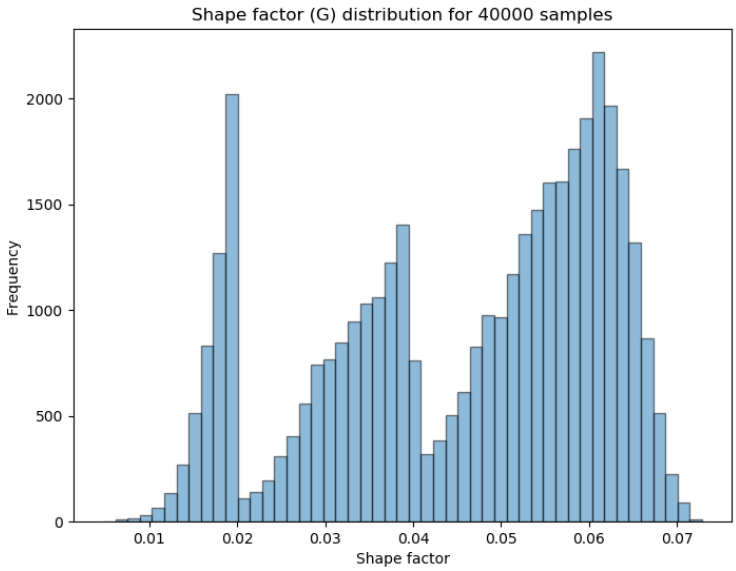

Figure 2.9 Distribution of shape factors (circularity) of polygons within training dataset 2

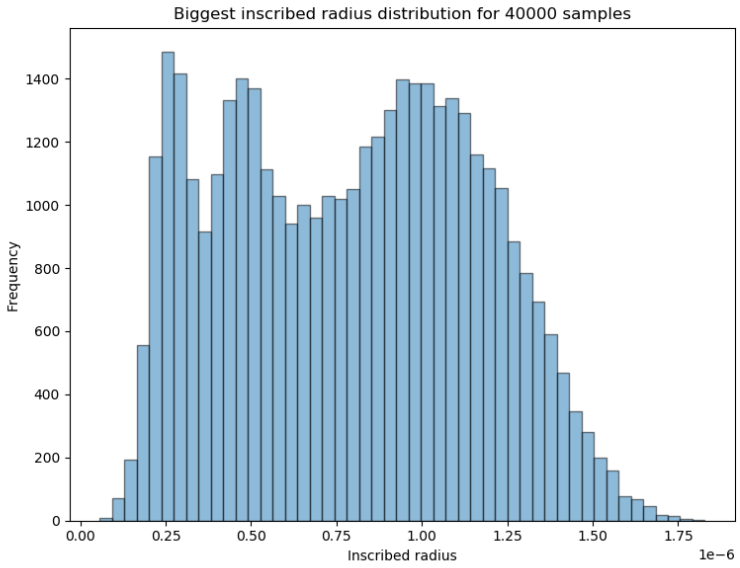

Figure 2.10 Distribution of largest inscribed radius inside polygons within training dataset 2

2.3.2 Descriptions and statistical analysis for blind verification dataset

(2) Statistic analysis(plots) for blind validation dataset (5 validation test datasets)

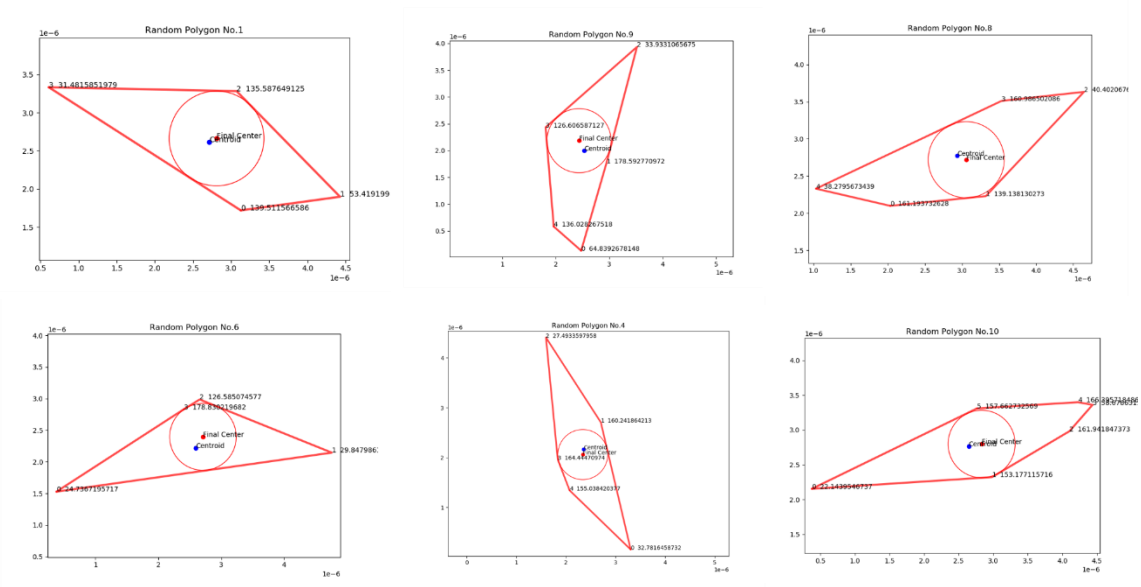

Figure 2.11 Selected polygons from blind verification dataset 1 (shape factor in [0, 0.04])

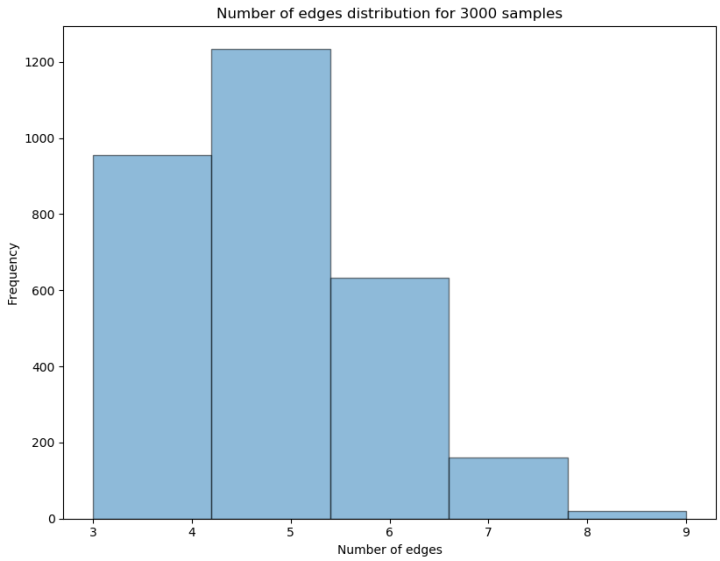

Figure 2.12 Distribution of number of edges of polygons within blind verification dataset 1 (shape factor in [0, 0.04])

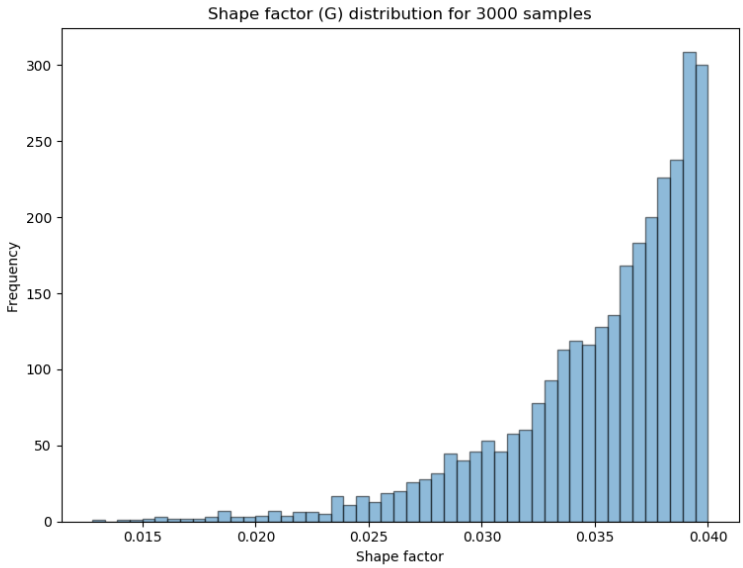

Figure 2.13 Distribution of shape factors (circularity) of polygons within blind verification dataset 1 (shape factor in [0, 0.04])

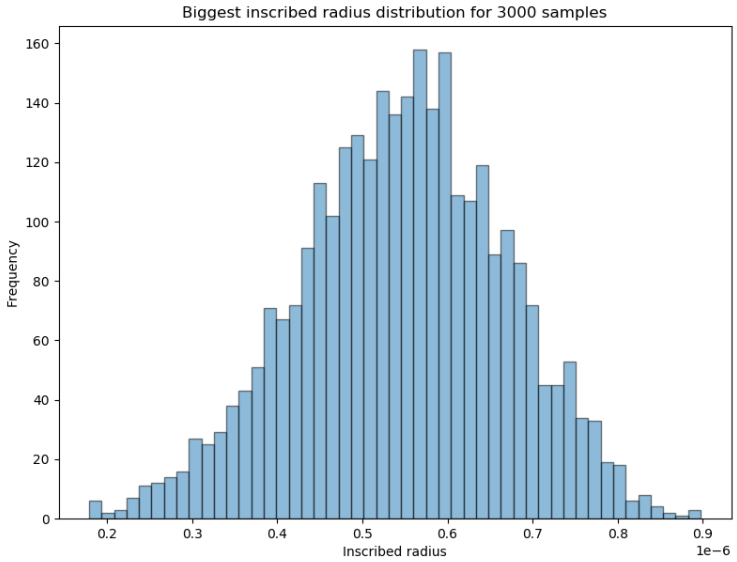

Figure 2.14 Distribution of largest inscribed radius inside polygons within blind verification dataset 1 (shape factor in [0, 0.04])

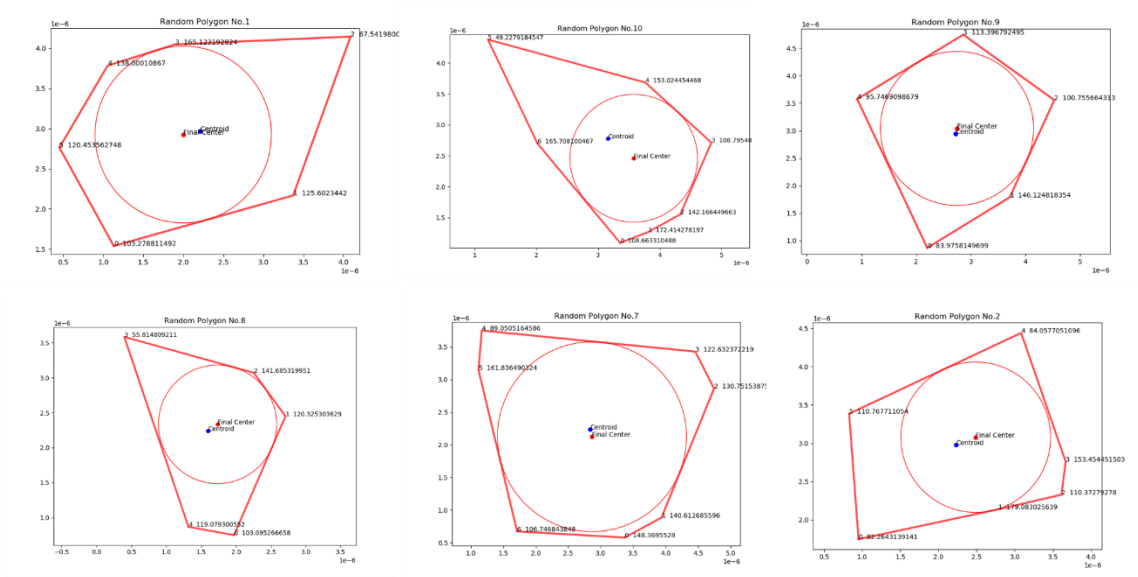

Figure 2.15 Selected polygons from blind verification dataset 2 (shape factor in [0.04, 0.07958])

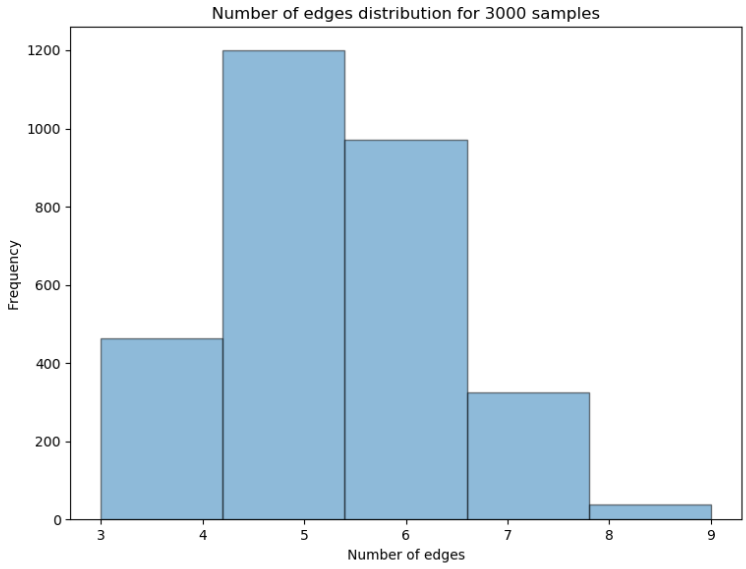

Figure 2.16 Distribution of number of edges of polygons within blind verification dataset 2 (shape factor in [0.04, 0.07958])

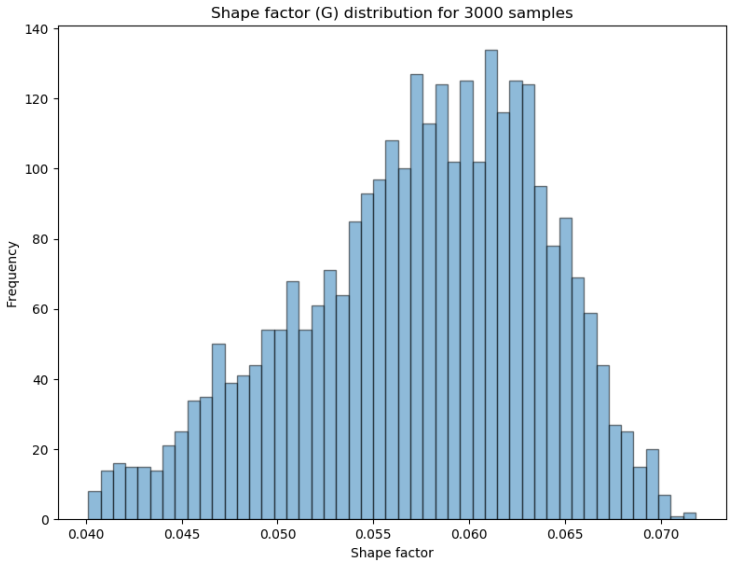

Figure 2.17 Distribution of shape factors (circularity) of polygons within blind verification dataset 2 (shape factor in [004, 0.07958])

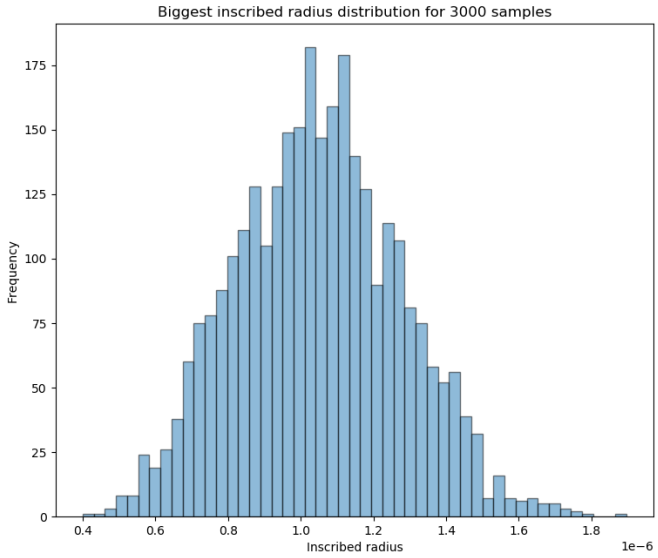

Figure 2.18 Distribution of largest inscribed radius inside polygons within blind verification dataset 2 (shape factor in [0.04, 0.07958])

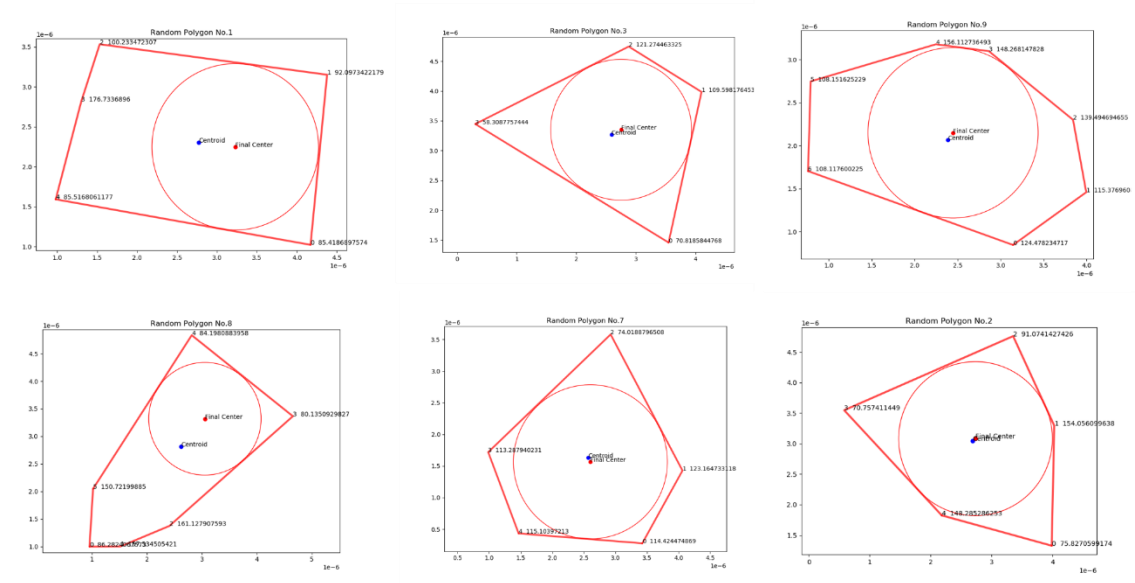

Figure 2.19 Selected polygons from blind verification dataset 3 (elongation factor in [0, 0.50])

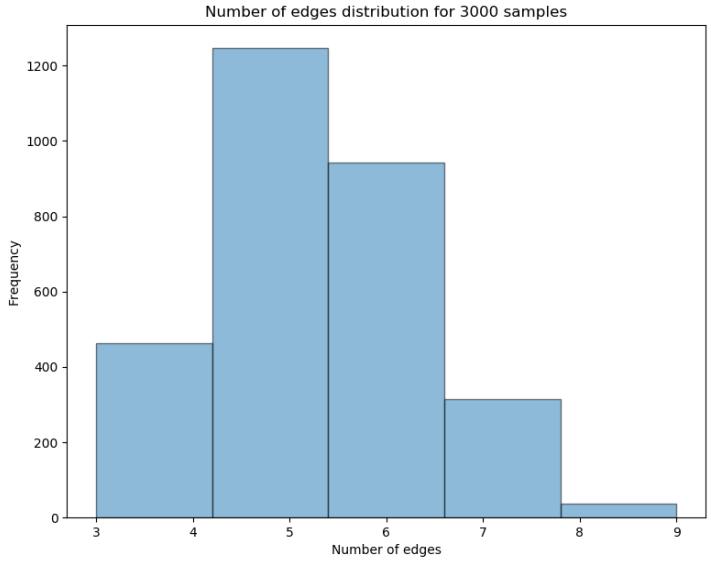

Figure 2.20 Distribution of number of edges of polygons within blind verification dataset 3 (elongation factor in [0, 0.50])

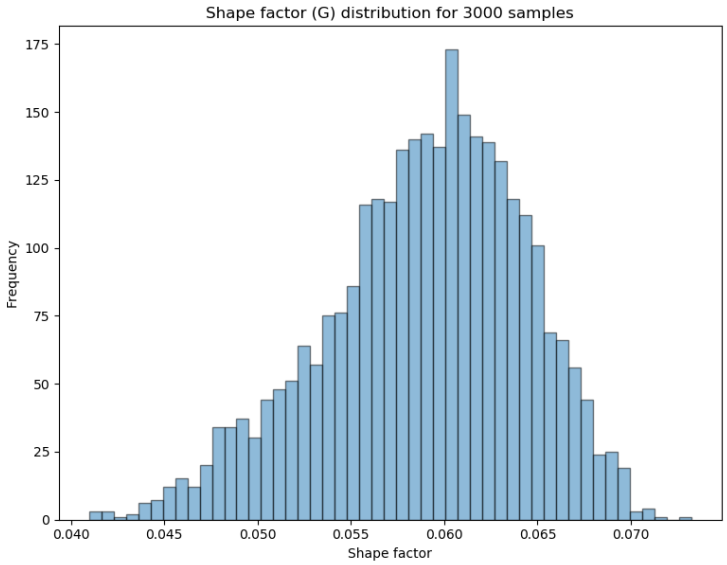

Figure 2.21 Distribution of shape factors (circularity) of polygons within blind verification dataset 3 (elongation factor in [0, 0.50])

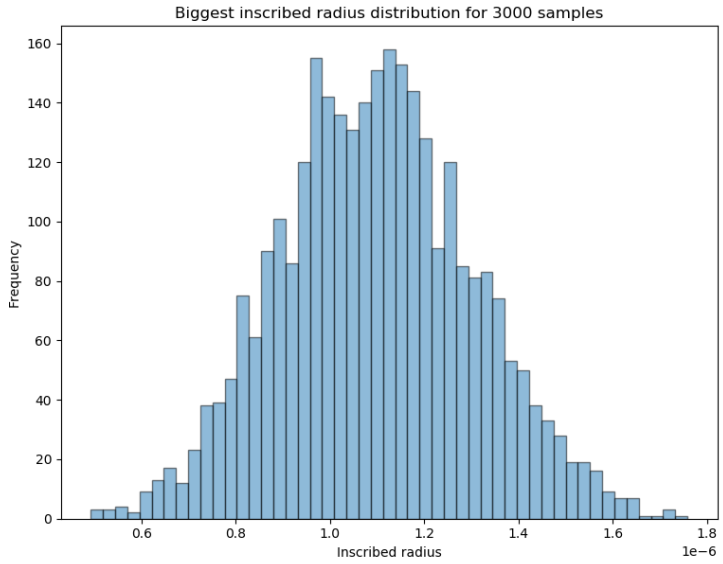

Figure 2.22 Distribution of largest inscribed radius inside polygons within blind verification dataset 3 (elongation factor in [0, 0.50])

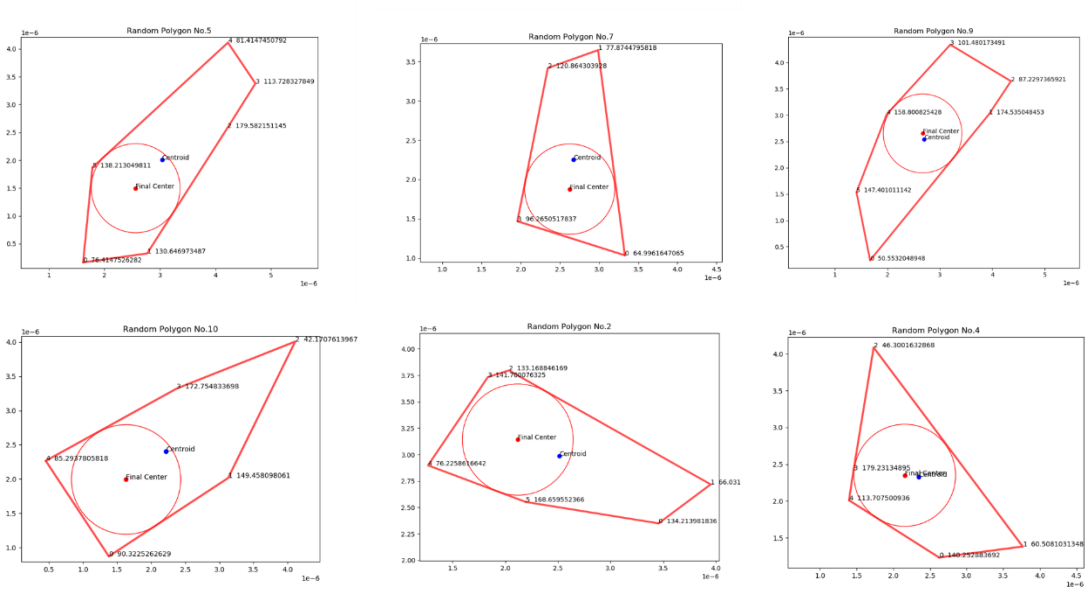

Figure 2.23 Selected polygons from blind verification dataset 4 (elongation factor in [0.50, 1.00])

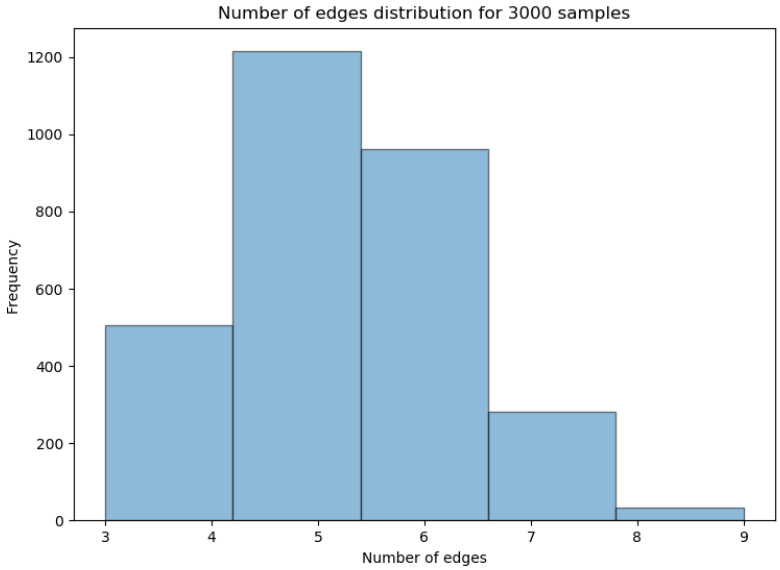

Figure 2.24 Distribution of number of edges of polygons within blind verification dataset 4 (elongation factor in [0.50, 1.00])

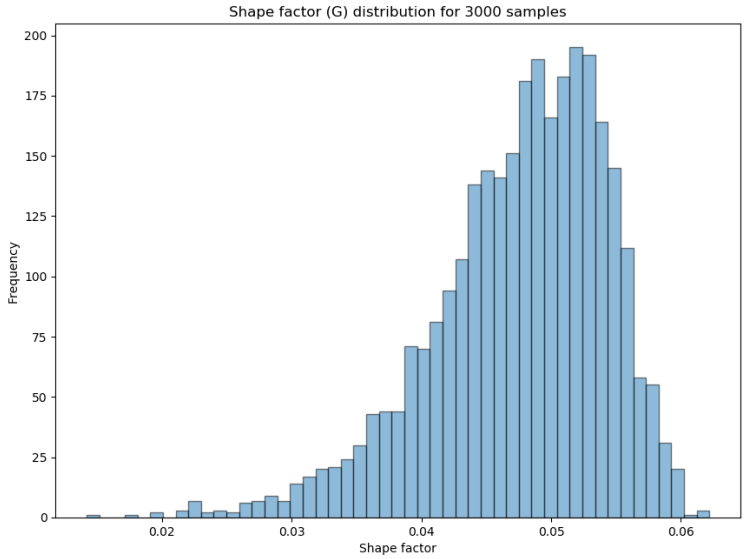

Figure 2.25 Distribution of shape factors (circularity) of polygons within blind verification dataset 4 (elongation factor in [0.50, 1.00])

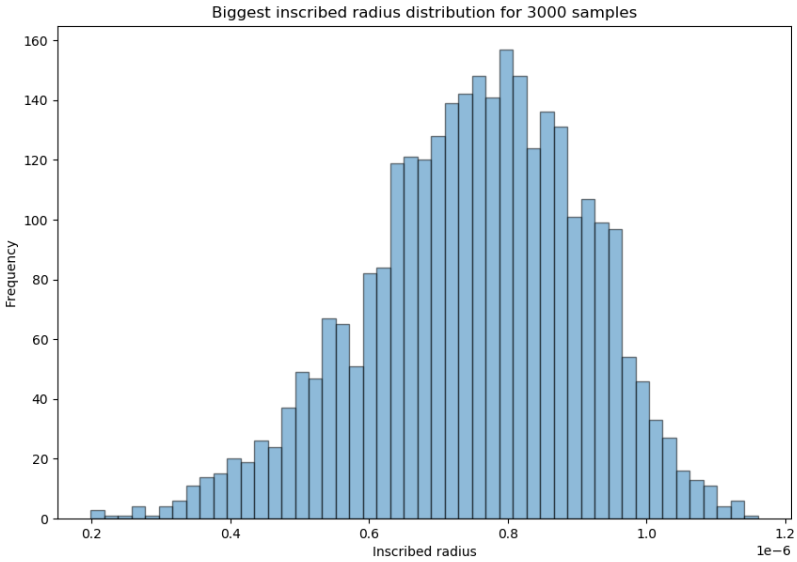

Figure 2.26 Distribution of largest inscribed radius inside polygons within blind verification dataset 4 (elongation factor in [0.50, 1.00])

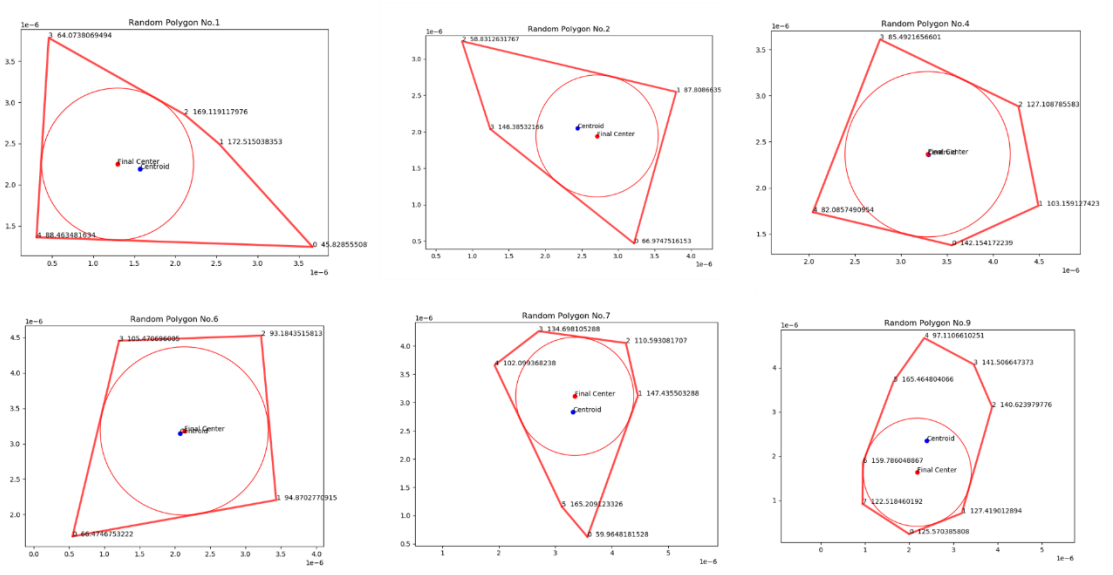

Figure 2.27 Selected polygons from blind verification dataset 5 (All random polygons without any constraints)

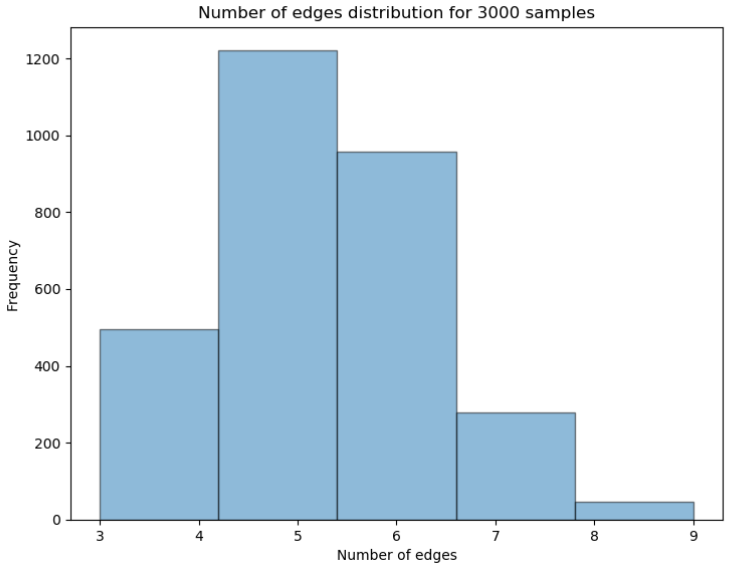

Figure 2.28 Distribution of number of edges of polygons within blind verification dataset 5 (All random polygons without any constraints)

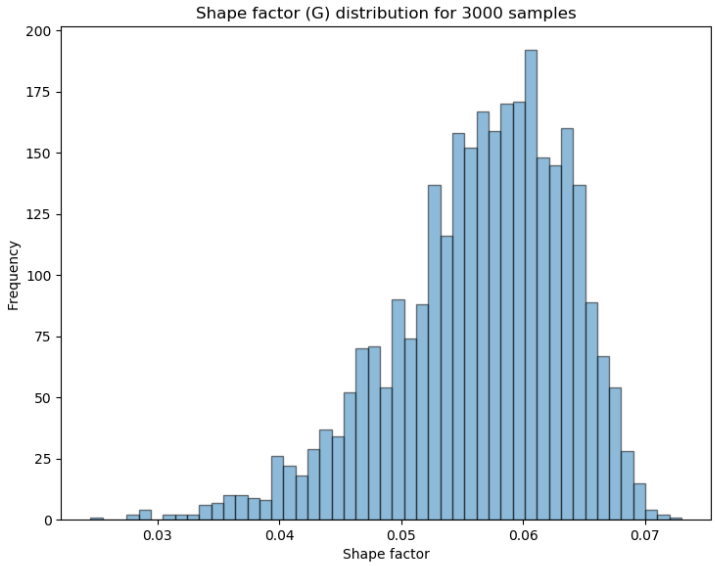

Figure 2.29 Distribution of shape factors (circularity) of polygons within blind verification dataset 5 (All random polygons without any constraints)

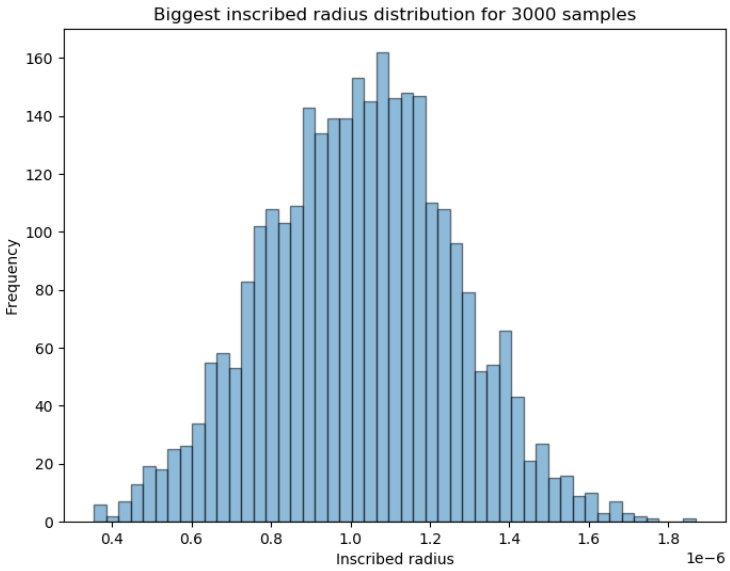

Figure 2.30 Distribution of largest inscribed radius inside polygons within blind verification dataset 5 (All random polygons without any constraints)

2.4 Primary drainage threshold capillary pressure(Pcow) and water saturation calculations

2.4.1 Pressure difference within non-circular cross-section capillary tube

The pressure difference between two fluid interfaces can be obtain by applying Young-Laplace equation (Rowell, 1998):

$$P_1 - P_2 = \sigma_{12}\left(\frac{1}{r_1} + \frac{1}{r_2}\right) \quad (1)$$

Where the P_1 and P_2 are the phase pressure for phase 1 and phase 2, r_1 and r_2 are principal radii of curvature of the contact interface. Two type of fluid-fluid contact interface within angular cross-section capillary tubes instead of forming only one type in circular shaped tubes.

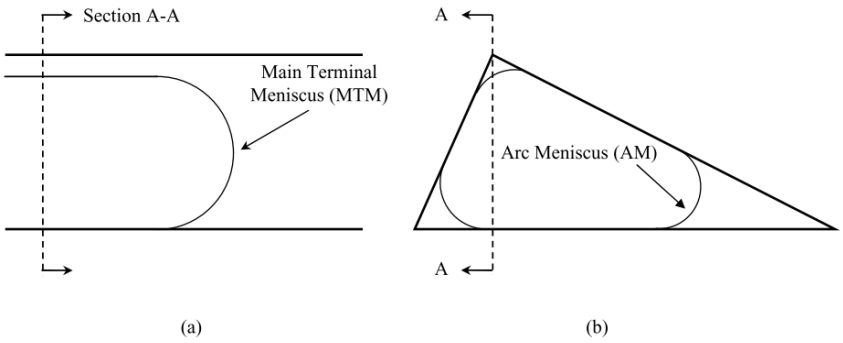

Figure 2.31 Main Terminal Menisci (MTM) (a) and Arc Menisci(AM) (b) (Piri, 2003)

AM is formed at the corner of the angular cross-section of a capillary tube which the wetting phase is sitting at the corners, and non-wetting phase is located in the center of the tube. It is obvious that the curvature of the contact interface which formed AMs are parallel to the tube direction if the capillary tube is straight with same inlet and outlet cross-section

area. In this scenario if the equation (1) is applied with oil and water two-phase flow in the tube, the equation can be re-write as:

$$P_o - P_w = \sigma_{ow}\left(\frac{1}{r_{ow}}\right) \tag{2}$$

In a polygon cross-section two-phase flow system from AM's perspective, the fluid-solid contact length within each corner can be obtained by:

$$b = r_{ow}\frac{\cos(\theta + \alpha)}{\sin(\alpha)} \tag{3}$$

The fluid-fluid contact length can be obtained by:

$$L = 2r_{ow}\left(\frac{\pi}{2} - \theta - \alpha\right) \tag{4}$$

Moreover, the cross-section area for the wetting phase fluid located at the corner is calculated by:

$$A = r_{ow}^2\left[\theta + \alpha - \frac{\pi}{2} + \cos(\theta)\frac{\cos(\theta + \alpha)}{\sin(\alpha)}\right] \tag{5}$$

Where the α is the corner half angle for the specific corner of the polygon cross-section, r_{ow} represent the effective curvature radius of the fluid-fluid contact interface at the corner. is the contact angle between the fluid-fluid interface and the solid inner surface, note that the angle is the one toward the apex of the corner.

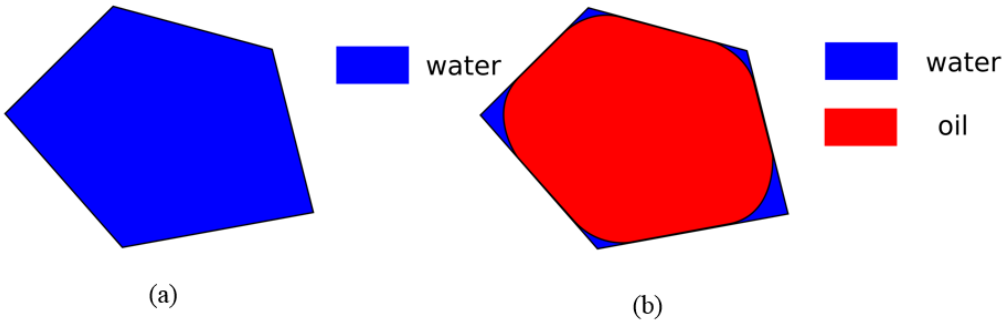

Figure 2.32 Cross-section of a polygon tube with original configuration (a) and the configuration after primary drainage (b), the oil-water contact at corners show the AMs

2.4.2 MS-P method

The MS-P method is short for Mayer-Stowe-Princen method (Mason & Morrow, 1984; Mayer & Stowe, 1965; Princen, 1969a, 1969b, 1970) which described the capillary entry pressure calculation method for capillary dominated fluid flow within an angular cross-section pore/tube. The Helmholtz free energy F is minimized to obtain the threshold pressure for invading fluid phase within a thermodynamically reversible and piston-like only displacement scenario. In such way, the fluid configuration changes for invading process associated threshold capillary pressure can be obtained by minimize the Helmholtz free energy or in other words making the derivative of Helmholtz free energy to zero. For example, an immiscible system which thermodynamic equilibrium is formed, the changing of Helmholtz energy is described as:

$$dF = -\sum_{i=1}^{n} P_i dV_i + \sum_{ij=12,13,..23,24..33..}^{\frac{n!}{2(n-2)!}} \sigma_{ij} dA_{ij} \qquad (6)$$

Where right-hand side of the Eq. (6) represent the changing of Helmholtz free energy contributed by the fluid-fluid and fluid-solid configurations changing before and after

invasion, the left-hand side represent the volume changing of phases during invading process.

2.4.3 Apply MS-P method for 2 phase flow Primary Drainage displacements

For two-phase oil displace water drainage process in this study, Eq. (6) can be re-write as:

$$dF = -P_w dV_w - P_o dV_o + \sigma_{ws} dA_{ws} + \sigma_{os} dA_{os} + \sigma_{ow} dA_{ow} = 0 \tag{7}$$

Combine the equation above and capillary pressure and force balance equations:

$$P_{cow} = P_o - P_w \tag{8}$$

$$\sigma_{os} - \sigma_{ws} = \sigma_{ow} \cos(\theta_{ow}) \tag{9}$$

Then we obtain:

$$dF = -P_{cow} dV_o + \sigma_{ow} \left[dA_{ow} + \cos(\theta_{ow}) dA_{os} \right] = 0 \tag{10}$$

Combine Eq. (10) and Eq. (2) then we have:

$$\frac{-dA_o}{r_{ow}} + dL_{ow} + \cos(\theta_{ow}) dL_{os} = 0 \tag{11}$$

2.4.4 Quadratic equations derivation and solving for capillary pressure

From the Eq. (11) the thermodynamically consistent capillary entry pressure for the piston-like drainage process within random convex polygon cross-section tubes is derived as below:

$$-\left(\frac{A_p - \sum_{i=1}^{n} r_{ow}^2 \left[\theta + \alpha_i - \frac{\pi}{2} + \cos(\theta) \frac{\cos(\theta + \alpha_i)}{\sin(\alpha_i)} \right]}{r_{ow}} \right) + \sum_{i=1}^{n} 2 r_{ow} \left(\frac{\pi}{2} - \theta - \alpha_i \right) + \cos(\theta) \left[P - \sum_{i=1}^{n} 2 r_{ow} \frac{\cos(\theta + \alpha_i)}{\sin(\alpha_i)} \right] = 0 \tag{12}$$

Where A_p represent the total area of the convex polygon cross-section. P represent the perimeter of the polygon cross-section.

After rearranging Eq. (12) we obtain:

$$r_{ow}^2\left[\begin{array}{l} n\left(\theta-\frac{\pi}{2}\right)+\sum_{i=1}^{n}\alpha_i+\sum_{i=1}^{n}\cos(\theta)\frac{\cos(\theta+\alpha_i)}{\sin(\alpha_i)} \\ +2n\left(\frac{\pi}{2}-\theta\right)-2\sum_{i=1}^{n}\alpha_i-2\cos(\theta)\sum_{i=1}^{n}\cos(\theta)\frac{\cos(\theta+\alpha_i)}{\sin(\alpha_i)} \end{array}\right] \\ +r_{ow}\cos(\theta)P-A_p=0 \qquad (13)$$

The effective curvature radius is obtained by solving the quadratic equation for each convex polygon cross-section. Then the corresponding threshold capillary pressure can be calculated by Eq. (13)

2.4.5 Water saturation (Sw) calculations

From the previous section, we derived the equations for threshold capillary pressure, effective curvature radius, and so forth. If specific invading pressure is given for the system, the capillary tubes with entry capillary pressure above the given pressure will not be invaded, and the oil volume is zero and water volume is the volume of the tube. The tubes with entry capillary pressure lower than given pressure will be invaded, and the volume of water sitting at corners can be obtained by multiplying cross-section area and the tube length. The individual cross-section area within particular corner is calculated by Eq. (5), and the total area is a summation of all corner areas, then the rest area which occupied by oil in the center is given by total area minus area occupied by water. Now the volume of

water and oil within each one of the tubes can be calculated individually at given invading pressure, the total system water/oil saturation is easy to get from all individual tubes.

2.5 Primary drainage Kro, Krw calculations

2.5.1 Absolute permeability and conductance

The absolute permeability as the denominator for relative permeability calculation for each fluid flow phase is obtained by using Darcy's Law:

$$K = \frac{\mu_i q_i^{total} L}{A\Delta p} \tag{14}$$

Where the μ_i is the viscosity for fluid I, L is the length, ΔP represent the pressure drop between inlet and outlet along length direction, A is the area of cross-section which perpendicular to the fluid flow direction, K represent the absolute permeability of the system, q_i^{total} represent the total flow rate of fluid i.

The conductance of fluid phase in the tube is calculated before calculating relative permeabilities. The conductance can be obtained by direct solve Navier-Stokes equations numerically or use the empirical correlations from numerical solutions, *Miao et al.* use Neural Network to predict hydraulic conductance from the Computation Fluid Dynamics (CFD) software COMSOL results. In this thesis, we are not focusing on conductance estimation, simple correlations and interpolations methods are used in our study. There are three conductance calculation equations for triangular, square and circular cross-section (Oren, Bakke, & Arntzen, 1998; Patzek & Silin, 2001; Piri, 2003)

$$g_{tri} = \frac{0.6 G_{tri} A_{tri}^2}{\mu} \tag{15}$$

$$g_{sqr} = \frac{0.5623 G_{sqr} A_{sqr}^2}{\mu} \tag{16}$$

$$g_{cir} = \frac{0.5 G_{cir} A_{cir}^2}{\mu} \tag{17}$$

Where G is the shape factor of the cross-section, A is the area of the cross-section, μ is the viscosity of the fluid. The conductance factor here for convex polygons are obtained by interpolation of the three geometries mentioned above. The actual single-phase flow rate from the inlet to outlet within an individual tube is calculated by:

$$\mathrm{q} = \frac{\mathrm{g}}{L} \Delta P \tag{18}$$

Where g is the conductance, L is the length of the tube and ΔP is the pressure difference between inlet and outlet of the capillary tube.

2.5.2 Relative permeability calculations (including sw_area)

The relative permeabilities for the specific fluid phase in a multi-phase flow system are obtained by dividing the phase flow rate in the system by the total flow rate when there is only this fluid flow in the system:

$$Kr_x = \frac{q_x^{total}}{q_x^{total\_single\_phase}} \tag{19}$$

From the last section, we calculated the single-phase flow rate, and in multi-phase flow, conductance for each fluid phase can be obtained by various methods, the equation proposed by *Hui & Blunt* (M.-H. Hui & Blunt, 2000a, 2000b) is used in our study:

$$\mathrm{g}_{\mathrm{corner}} = \frac{A_{corner}^2 (1 - \sin\alpha)^2 (\varphi_2 \cos\theta - \varphi_1) \varphi_3^2}{12\mu \sin^2\alpha (1 - \varphi_3)^2 (\varphi_2 + f\varphi_1)^2} \tag{20}$$

$$\varphi_1 = (\frac{\pi}{2} - \alpha - \theta) \tag{21}$$

$$\varphi_2 = \cot\alpha\cos\theta - \sin\theta \tag{22}$$

$$\varphi_3 = (\frac{\pi}{2} - \alpha)\tan\alpha \tag{23}$$

Where A_{corner} is the area of the corner occupied by water, α is corner half angle, μ is viscosity of the fluid sitting at corners (here is water), f will be zero if free boundary condition is considered between fluid-fluid contact interface, if no-flow boundary is considered here then f is set to 1. In our calculation, we consider no-flow boundary at interface.

Then the total conductance of water sitting at corners are obtained by summation of the conductance calculate by Eq. (24):

$$g_{corner\_total} = \sum_{i=1}^{n} g_{corner}^{i} \tag{24}$$

The conductance of oil which sitting in the center of the tube is calculated by the equation from Eq. (15, 16, 17) with the interpolated factor value.

Then the flow rate for each phase within an individual capillary tube can be calculated by Eq. (18) with given test pressure, relative permeability values for certain testing pressure are solved. When a series of testing pressures are given, the relative permeability curves will be obtained by a series invading test and properties calculation process.

2.6 Neural Network training/prediction development

2.6.1 Overview of Neural Network approach

In our study, various Multi-Layer Perceptron (MLP) Neural Networks are created by utilizing `Keras(Chollet & others, 2015)`., which is running on top of open sourced

Google `Tensorflow` (Abadi et al., 2016) Neural Network framework, it provides powerful and flexible foundations and APIs for researchers and industry users based on creating a computation graph.

Several customized pre-processing and post-processing modules are written for feature selection, data normalization, training visualization, model prediction and blind verification, prediction and calculation result comparison and plotting, and so forth.

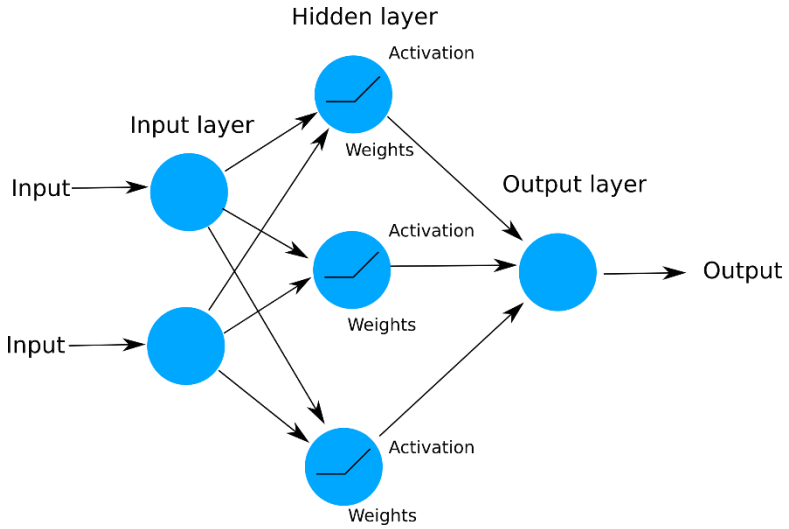

Figure 2.33 Schematic plot for Multi-Layer Perceptron (MLP) network

Two type of Neural Network training and prediction patterns are built for our purpose:

- Approach 1: Build Neural Network just for threshold capillary pressure prediction based on the input data which related to tube cross-section geometrical properties.

- Approach 2: Trying to build Neural Network and predict threshold capillary pressures and water, oil relative permeabilities simultaneously without building two separate models. The first approach is relatively easy to do because the threshold capillary pressures are strongly related to the geometry of the cross-section areas

from the derivation in previous chapters, all bundle of parallel capillary tubes is independent of each other in terms of entry capillary pressure and invasion status. In such cases, simple machine learning or Neural Network is capable of capturing the correlations between input parameters and output threshold capillary pressures. However, relative permeabilities are generated from a series invasion tests based on a series testing pressures. Different testing pressure will result in a different number of capillary tubes that is invaded, and with increasing testing pressure, the water-oil configurations for those tubes have already been invaded will slightly change, so the real-time conductances are dynamically changing then led to changing flow rate and relative permeability values. Testing pressures must be one of the input features, but it is not 'directly' correlate with most of the inputs, such as the geometrical properties of the polygons, also the number of testing pressure does not necessarily have the same shape with other inputs, i.e., they are not compatible with each other as the same input dataset.

The first approach is straightforward and only for threshold capillary pressure estimation, after some Neural Network experiments, we obtained satisfactory results, but not capable of tackling the relative permeability problems. In this thesis, we are mainly focused on the second approach and present a potential way to reach the point.

2.6.2 Training data formulation for Pcow, Kro, Krw prediction

(1) Ideas

For approach 1, there's no need to reconstruct the original training dataset because of the features and properties of each one of the capillary tubes are independent of others.

For approach 2, the overall idea is to 'discretize' the original datasets in terms of the testing pressures for the system. In order to force Neural Network to learn the non-linear behaviors among inputs and outputs, input dataset is modified to standard Neural Network input format that each one of the polygon properties data line will be copied and repeat then correspond to every assigned testing pressure, and then calculate the P_{cow}, K_{ro} and K_{rw}, water-saturated area (S_w_area), water-saturated area fraction (S_w_frac) for each one of the 'tube, test pressure combination'. The reconstructed training dataset could be huge and direct calculation for that huge dataset is extremely computation expensive, so that we use random sampling method to extract certain number of polygons from original dataset and calculate all 'discretized.' properties as part of training data. Perform sampling at satisfactory times, e.g. extract 100 tubes from 5000 original tubes and calculate related properties then combine all results as final training dataset after doing 1000 times of sampling. Properties calculation on small number of capillary tubes is much faster than in larger dataset, so each one of the 'sampling-capillary pressure calculation -relative permeability related parameters calculation' iteration cost few seconds on my desktop computer, huge training dataset is constructed in small amount of time which might be enough in terms of individual capillary tube coverage and 'tube - test pressure combination' coverage.

Note that the reason for creating two intermediate predicting variables S_w_area and S_w_frac is that the water saturated area within one tube is a very small number but water saturation (fraction) is between 0 and 1, the neural network performance will be evaluate when dealing with values in different order of magnitude.

(2) Procedures and results

The strategy and procedure for sampling in this study are randomly select 100 capillary tubes from all capillary tubes, and calculate the corresponding threshold capillary pressures. After finding the minimum and maximum threshold capillary pressures within the selected samples, a 100 evenly separated testing pressure array will be created based on the two values. Relative permeability related properties are calculated based on the 100 selected capillary tubes and the 100 testing pressures, then save all results into a specific data format. The program will randomly select another 100 tubes and repeat all process until 1000 iterations reached. All results will be combined into one dataset as input for Neural Network training process. A dedicated application for such preprocessing and some post-processing is developed to preprocess training/blind test datasets, see Figure 2.34.

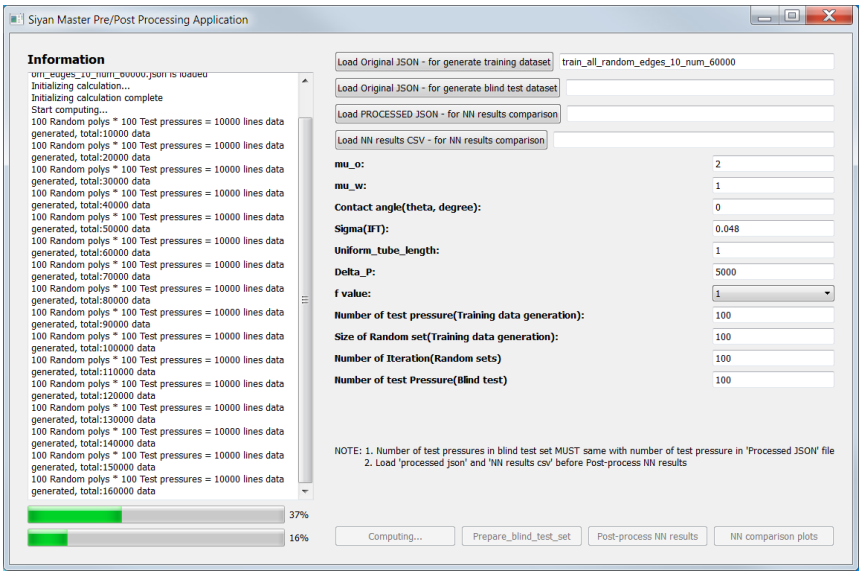

Figure 2.34 Snapshot of software tool for preprocessing and postprocessing

2.6.3 NN input features selection

For approach 1, the output feature would be only P_{cow} (threshold capillary pressure), and the potential input parameters would be combination of geometrical properties of polygon cross-section area, namely, area (A), perimeter (P), number of edges/corners (n), largest inscribed radius (R_{in}), shape factor (g_cc), elongation factor (g_el).

For approach 2, besides all the input features mentioned in approach 1, the generated P_{test} data array should be adding into the training dataset as well, also according to the ideas of approach 2 in previous section, 4 outputs were assigned to the training system: threshold capillary pressure $Pcow$, the oil flow rate for individual capillary tube at a specific testing pressure q_o, the water flow rate for individual capillary tube at specific testing pressure q_w, and the actual water saturated cross-section area within individual capillary tube at specific testing pressure $S_{w\_}area$ or $S_{w\_}frac$. In such case P_{cow} and the rest 3 'discretized' parameters can be combined and feed into Neural Network.

2.6.4 Neural Network structure creation and activation functions

For approach 1, we believe simple Neural Network structure is good enough to catch the correlations between input and output, but to validate our speculation, 3 Neural Network model are generated, one base mode, one with wider hidden layer, one with wider and deeper hidden layers:

- Model A1-T1-H [5]: Single hidden layer with five neurons, Training dataset No.1 is input for this model. (Figure 2.35)

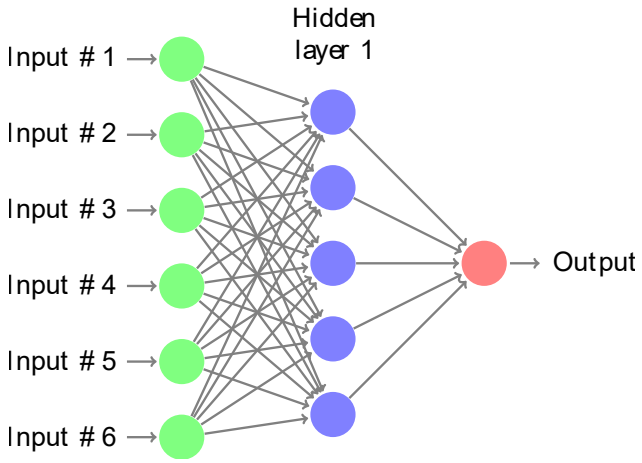

Figure 2.35 Network structure for one hidden layer with 5 neurons

- Model A1-T2-H [5]: Single hidden layer with 5 neurons, Training dataset No.2 is input for this model.
- Model A1-T1-H [10]: Single hidden layers with 10 neurons (Figure 2.36)

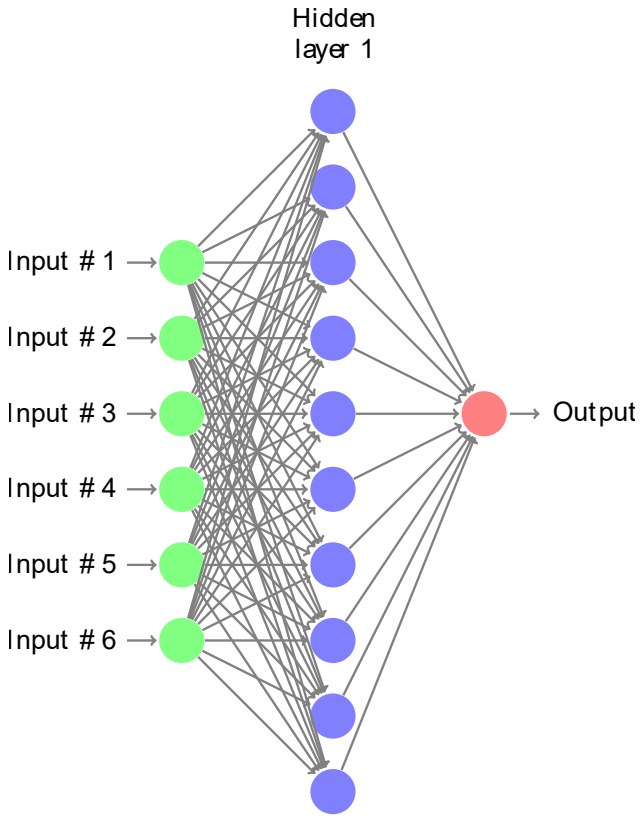

Figure 2.36 Network structure for one hidden layer with 10 neurons

- Model A1-T1-H [5, 5] Two hidden layers with 5 neurons for each one of them (Figure 2.37)

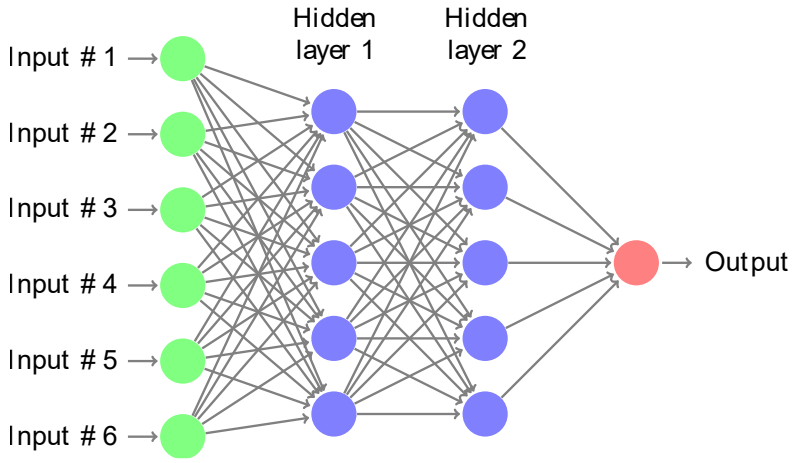

Figure 2.37 Network structure for two hidden layers and 5 neurons in each layer

- Model A1-T1-H [10,10] Two hidden layers with 10 neurons for each one of them (Figure 2.38)

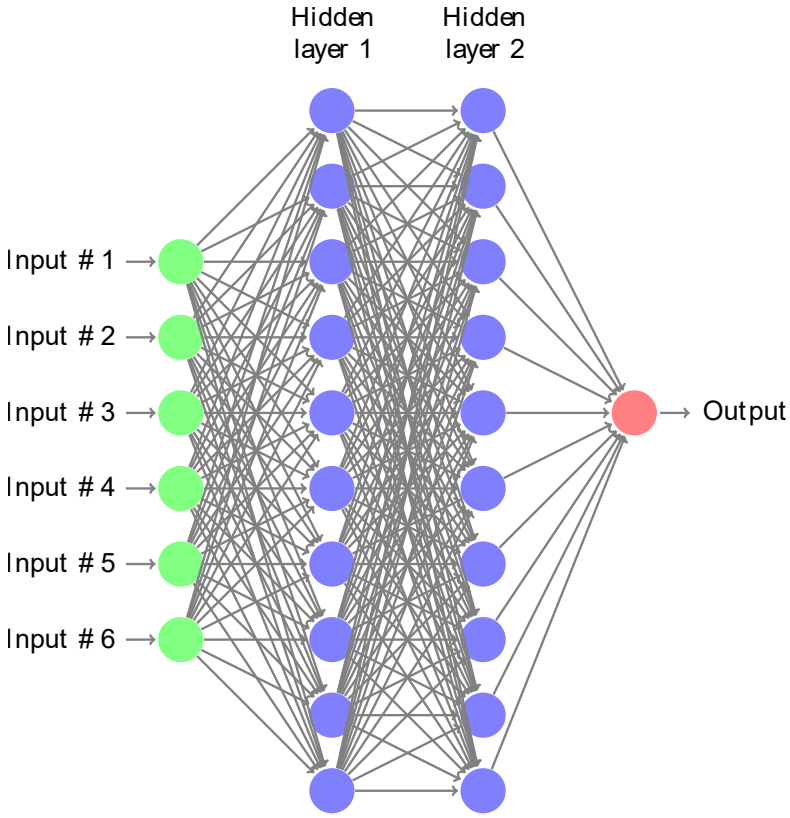

Figure 2.38 Network structure for two hidden layers and 10 neurons in each layer

These 3 models are using all 6 input parameters as training data, a separate input data sensitive analysis is in 2.6.9.

For approach 2, more complexity is adding into the system, and training dataset is significantly larger than the one in approach 1. We presume that more hidden layers with more neurons would produce better results. After some trial and errors, the results showed that the model with single hidden layer with fewer neurons still performs well (see the results in 2.6.9 sensitivity analysis). We present a network structure that produces better results with 3 hidden layers and 200, 100, 50 neurons respectively. Also in approach 2 we tried two schemes for S_w training and prediction and created two models, which would result in 2 different post-processing approaches:

- Model A2-T1-H [200,100,50]-Sw_area

  In this model, besides the P_{cow}, q_o, q_w as output parameters, the actual water saturated area within individual capillary tube at specific testing pressure (S_w_area) was used as last output parameter during training and prediction process. Training dataset No.1 is input for this model.

- Model A2-T2-H [200,100,50]-Sw_area

  Change training input to training dataset 2, rest of them are same with Model A2-T1-H [200,100,50]-Sw_area

- Model A2-T1-H [200,100,50]-Sw_frac

  In this model, besides the P_{cow}, q_o, q_w as output parameters, the water saturated area fraction (S_w_frac) was used as last output parameter during training and prediction process. Training dataset No.1 is input for this model.

- Model A2-T2-H [200,100,50]-Sw_frac

  Change training input to training dataset 2, rest of them are same with Model A2-T1-H [200,100,50]-Sw_frac

In these two models, all 6 input parameters previously mentioned plus P_{test} were used in training Neural Network, sensitivity analysis in terms of input parameters for approach 2 see section 2.6.9.

The Neural Network structure for approach 2 is showing in Figure 2.38 (Due to space limitation, plot created with double hidden layers with 10 neurons in each one of the layers as the demonstration)

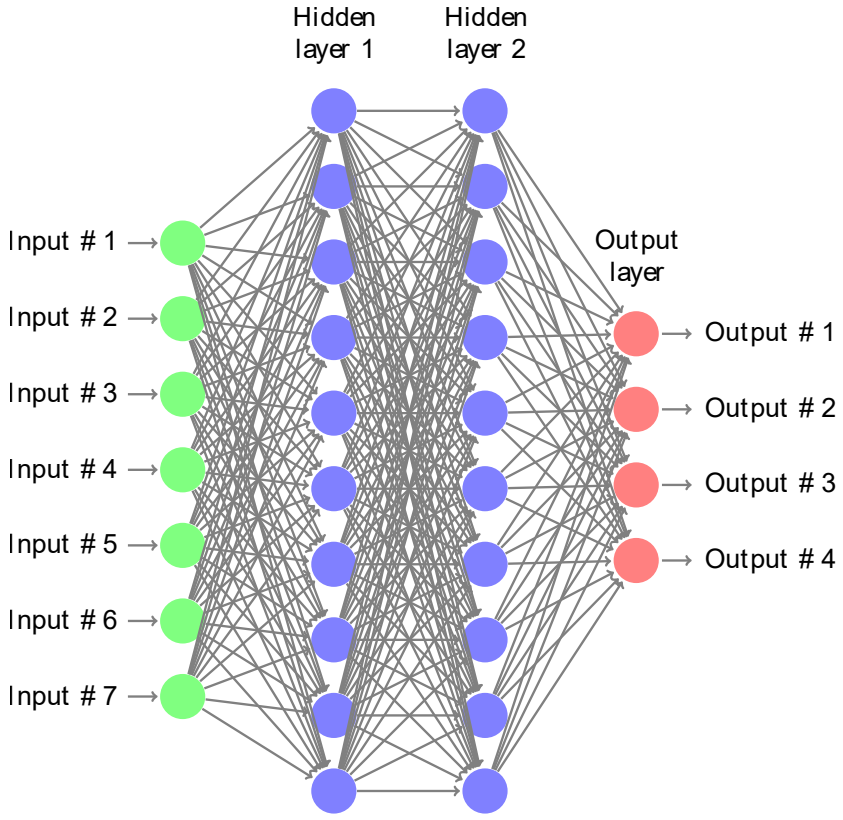

Figure 2.39 Demonstration of network structure for approach 2

Another important setup needs to be done during Neural Network structure creation is to assign activation functions within each one of the layers except first input layer. Each one of the activation functions resides within in one neuron would do ‘input receiving - signal processing - forward result to next neuron’ working cycle during the training process.

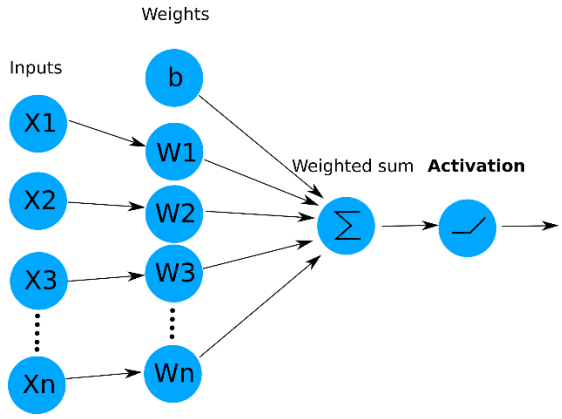

Figure 2.40 Simple schematic plot for activation function

After multiplications and summations of inputs and weights, the corresponding activation function would behave like a ‘scaling’ function that re-scale the output values before forwarding information to neurons in next layer. Through this way, the non-linearities are introduced to the system which could be used to do approximation for any non-linear functions or non-linear system.

Without activation functions, the feedforward process would become the results of linear functions, in other words, the outputs of each layer are a linear function of inputs from the previous layer, thereafter the inputs and outputs would of the Neural Network form the results of linear combinations which has poor performance for solving non-linear problems. For a typical regression problem in our study, there are three most popular activation functions used in Neural Network training:

- Sigmoid: `Sigmoid` activation function (Han & Moraga, 1995) is widely used activation function for quite a long period. See Eq. (25) and Figure 2.40, the function would 're-scale' input values into the range of 0 and 1. Sigmoid has been popularly used before but rarely used in today's Neural Network structure because of the following disadvantages:

  (1) Gradient vanishing or exploding could be easily happened for Neural Network with more than 2 layers by using the `Sigmoid` function, so it worked well in previous one hidden layer network but failed in today's deep neural network.

  (2) Non-zero-centered (output zero when the input is locating in the center of the range) is not suitable for maintaining training stability; the gradient updating process might suffering oscillations issues.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{25}$$

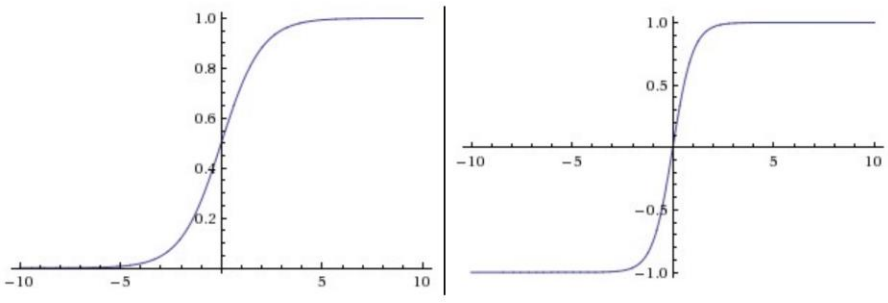

Figure 2.41 Sigmoid and Tanh activation function

- Tanh: `Tanh` activation function can be described as a 're-scaled' version of the `Sigmoid` function, see Eq. (26) and Figure 2.40, the results from `Tanh` is scale to range from -1 to 1, the advantage of Tanh from Sigmoid is Tanh

is a zero-centered function which would produce more desirable non-linearity than Sigmoid (Li, n.d.; Weisstein, n.d.)

$$\mathrm{Tanh}(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{26}$$

- ReLU: Rectified Linear Unit (`Relu`) is one of the most popular activation functions used in Neural Networking training due to the simplicity, fast calculation scheme and stabilities (Glorot, Bordes, & Bengio, 2011; Maas et al., 2013). Compared to Tanh and Sigmoid function mentioned above, `Relu` functions has several advantages:

  (1) From the equation and function graph, Relu can be implemented by fast checking the threshold at zero, $f(x) = \max(0, x)$, zero will be produced when the input less than zero and 1 will be output when input is greater than 0. Without any relative computation expensive operations such as exponential operations in Sigmoid and Tanh functions, the overall computation would be much faster.

  (2) Relu function would result in much faster convergence speed regarding the gradient descent based optimizations, 6 times speedup compared to Tanh activation function was reported based on SGD optimizer (Krizhevsky, Sutskever, & Hinton, 2012).

  But one should be careful with controlling the learning rate when choose Relu as activation functions because the specific neuron might suffering 'deactivate' or 'die' in some scenarios, such as when the neuron and Relu function are dealing with a large gradient which due to high learning rate, the unit may not be activated again then led to negative impact on the entire

Neural Network performance, so a small or moderate learning rate would produce better quality network even there is some learning speed decrease.

$$f(x) = \begin{cases} 0, when\ x < 0 \\ x, when\ \ x \geq 0 \end{cases} \tag{27}$$

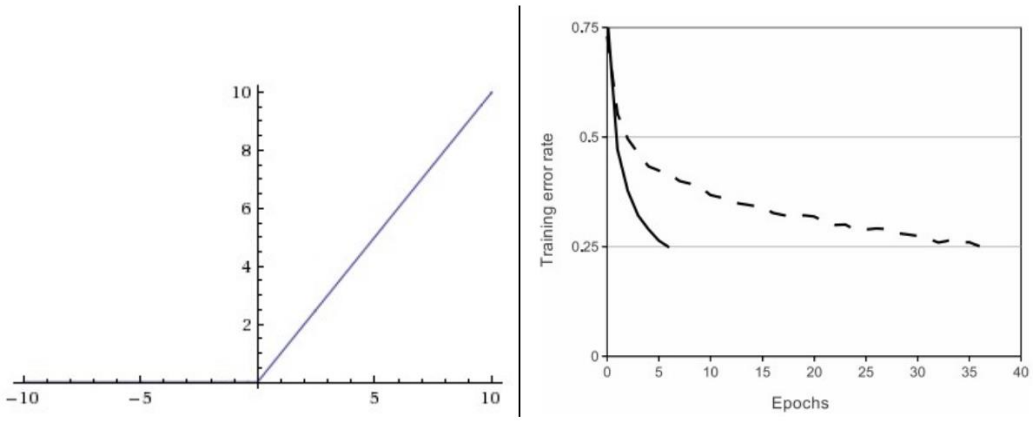

Figure 2.42 Rectified Linear Unit (ReLU) activation function and the training convergence comparison between ReLU and Tanh unit which shows 6x improvement (Krizhevsky et al., 2012; Li, n.d.)

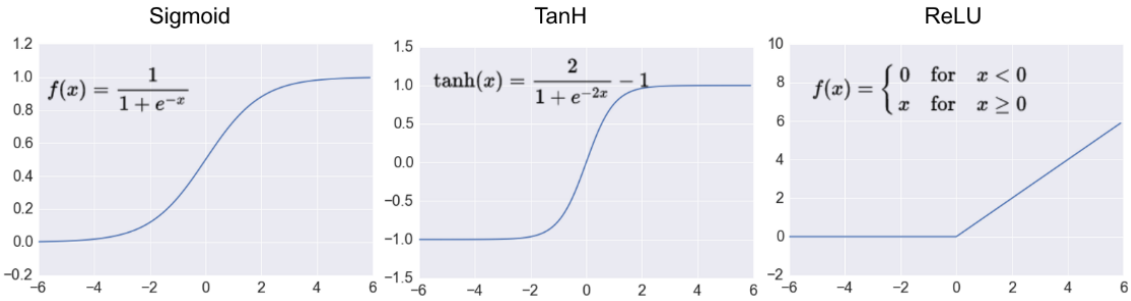

Figure 2.43 Comparison for Sigmoid, Tanh and ReLU activation function (Moujahid, 2016)

Relu activation function is selected in all hidden layers within each one of the Neural Network models created.

2.6.5 Importing data preprocessing

Preprocessing input data is necessary before feeding them into Neural Network. Multi-layer Perceptron Neural Network will adjust the weights by optimizing the gradients within each training iteration to obtain better fitting results.

Before training, all the weights will be randomly initialized before training process. If the training input data is larger or even larger than several orders of magnitude compare to initial weights, also the weights and gradients calculations which involve lots of multiplication operations, the values would possibly decrese to a very small number, even computer is not capable of dealing with (then will be zero) when lots of weights are small number less than 1. On the other hand, many values greater than 1 present and been multiplied multiple times would result in a huge number, in such cases the Neural Network will encounter instability issues which described as gradients vanishing and/or gradients exploding (Nielsen, 2015).

For those reasons mentioned above, the original input data usually been standardized or transform into a specific range, e.g., 0 to 1. In this thesis, a useful tool `StandardScaler, MinMaxScaler` in `scikit-learn` Python data mining, analysis, and machine learning package are used to preprocess all input

data before training (Pedregosa & Varoquaux, 2011). Note that the keeping the scaler parameters or scaler instance is useful for inverse transforming after Neural Network prediction to obtain the actual predicting values.

2.6.6 Hyperparameter selections, NN training, and validation.

Several hyperparameters need to be set before training:

- **Number of Epochs:** Maximum number of training iterations, training will stop when the number is reached. For our study, 10 epochs are good enough.
- **Batch size:** Number of input data samples that are passing through basic training cycle and updating gradients, in other words, partition big training task into small parts, which would result in less memory using and faster training process in most cases. 512 is used in this study.
- **Loss function:** Defined to describe the errors for the current state of NN which will be incorporated with the optimizer to improve the NN performance during each training iteration. Many loss functions can be used in this task, most popular loss function for classification problem is Cross Entropy Loss (Goodfellow, Ian, Bengio, Yoshua, Courville, 2016), we use another popular loss function which mostly used in regression problem: Mean Squared Error or MSE loss function, $MSE = \frac{1}{n}\sum_{i=1}^{n}(predict_i - actual_i)^2$
- **Optimizer:** Various gradient descent based optimization algorithms can be applied to Neural Network training process (Ruder, 2016). The purpose of using optimizer is to find a set of optimal weights that minimize the errors (Described as loss function) within each training iteration. Widely used optimizer such as Stochastic gradient descent (`SGD`), `RMSprop`, `Adagrad`, Adaptive Moment Estimation (`Adam`) (Kingma & Ba, 2015), `Adadelta`, and so forth., could be applied to various Neural Network scenarios. `Adam`

optimizer is select for our study with default optimizer parameters from original article (Kingma & Ba, 2015),

- **Early stopping criteria:** The training process will stop when specific criteria are reached, such as training R-Square score or testing R-Square scores, or some other customized stopping criteria can be defined to control the early stopping behavior. Testing R-Square score 0.995 is set as an input in this study.

- **Test fraction:** Fraction value ranging from 0 to 1 which determine the splitting fraction of training and internal validation data, 0.2 is used in our study which means the program will randomly select 20% of input data as validation and rest of 80% data as training data.

- **Data scaling:** Option to do preprocessing for input data or not, see detailed descriptions in section 2.6.5. Input data was scaled to range from 0 to 1 in this study.
- **Computing device options:** Option for select computation device if the platform equipped with single or multiple Graphics Processing Unit (GPU), the computation expensive matrix multiplication can be transfer to GPU for massively parallel computing, also the number of CPU threads can be select for training if CPU only mode is selected.

The Neural Network will start training after setting up all hyperparameters. Several customized `callbacks` functions will be activated during each training iteration

to monitor the entire training process such as writing customized logs, visualization and calculate R-Square scores based on each training-prediction cycle. The training model including all weights and Neural Network configurations will be saved automatically or manually as long as the training is complete.

2.6.7 Post-processing for model prediction results

Before doing blind tests, the dataset should also use same software application used in section 2.6.2 to preprocess input data but does not need to do sampling thousand times, just use the same 'discretize' scheme to run only once to reconstruct the input format which would be exactly same as training inputs. Meanwhile, the application will do all direct calculations to solve threshold capillary pressures for each one of the tubes and K_{ro}, K_{rw} values for each one of the tubes at a series assigned testing pressures to prepare the calculated results for comparison later.

The '`MinMaxScaler`' instance with previously saved scaler parameters is loaded to normalize input data before feed into well trained Neural Network to make predictions. After preprocessing, the program will load previously saved model configuration file, Neural Network weights file and some other customized log file to construct a new computation graph then compile with same hyperparameters previously used. Now the compiled model can predict blind test results by feed preprocessed blind test input data.

- Prediction for Approach 1: P_{cow} only:

In approach 1 one set of input polygon geometrical properties is correlated to one P_{cow}, the prediction is independent with each other, so the only step need to be done after Neural Network prediction is inverse transform the results to actual value scale, then the comparison between predict values and calculate values can be easily perform.

- Prediction for Approach 2: P_{cow}, q_o, q_w, $S_{w\_}area$ or $S_{w\_}frac$:

  In approach 2 the P_{cow} can be predicting directly as approach 1 does, for q_o and q_w, a series of summation calculations based on data grouped by each one of the testing pressure are performed to produce a list of summed oil/water flow rate at specific testing pressure. If the model train and predict $S_{w\_}area$, the overall water saturated area for entire system can be calculated by do summations for predict $S_{w\_}area$ based on data grouped by each one of the testing data directly; if $S_{w\_}frac$ is selected to participate in training and predicting, the predict fraction array need to multiply the polygon areas array to obtain the actual water saturated area list, then do summation based on data grouped by each testing pressure.

  Finally, the predicted S_w with respect to each testing pressure can be obtain by divide summation of all polygon areas.

Before calculating the relative permeabilities, the fixed single oil/water flow rate values need to be calculated based on equation (28), (29) for each one of the individual tubes then do the summation.

$$q_{o_{total}} = \sum_{i=1}^{n} \frac{g_o^i}{L^i} * \Delta P \tag{28}$$

$$q_{w_{total}} = \sum_{i=1}^{n} \frac{g_w^i}{L^i} * \Delta P \tag{29}$$

Where g_o^i and g_w^i are calculate by:

$$g_o^i = \frac{g_{factor}^i * g_{cc}^i * area^2}{\mu_o} \tag{30}$$

$$g_w^i = \frac{g_{factor}^i * g_{cc}^i * area^2}{\mu_w} \tag{31}$$

To obtain g_{factor}^i, some researchers use Computation Fluid Dynamics (CFD) method with commercial software such as COMSOL to get the values from direct solving the fluid flow equations (Miao et al., 2017b), also they tried to use Neural Network to predict hydraulic conductance. The main focusing of our study is trying to use Neural Network to predict P_{cow}, K_{ro} and K_{rw} directly, so we jump over the conductance predicting and just perform simple interpolation to get the coefficients based on data proposed by another researcher (Piri, 2003).

Then we will obtain the predict K_{ro}, K_{rw} lists by:

$$pred_{K_{ro}^i} = \frac{pred_{q_{oP_i}}}{q_{o_{total}}} \tag{32}$$

$$pred_{K_{rw}^i} = \frac{pred_{q_{wP_i}}}{q_{w_{total}}} \tag{33}$$

The Neural Network prediction quality will be examined after finishing all postprocessing workflow, the coefficient of determination (R^2) (Miles, 2014) is calculated by 1 minus residual sum of square over total sum of square, see Eq. (34)

$$R^2 = 1 - \frac{\sum_{i=0}^{n}(y_i - f_i)^2}{\sum_{i=0}^{n}(y_i - \bar{y})^2} \tag{34}$$

Where $\bar{y}$ is the mean of the predicted data (Eq. (35)),

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{35}$$

combined with actual and prediction data cross-plot are used for quality control, also the final prediction and calculate results comparison plots for P_{cow}, K_{ro}, K_{rw} would be the useful verification.

2.7 Neural Network structure analysis and adjustment

Although some researchers proposed the Neural Network with one hidden layer can perform well, and can predict non-linear behaviors with arbitrary precision (Krose & Smagt, 1996). However, with the development of deep learning theory (LeCun, Bengio, & Hinton, 2015) and related applications which becoming more and more popular in many areas, wider and deeper network structure would be more representative for complex tasks. However, we would like to reduce the size of the network to save computation resources and avoid over-fitting if the prediction quality is still satisfied. The structure analysis is performed by changing different network structures to train and test the corresponding Neural Network for the same dataset and evaluate the performance. Same training

parameters and hyperparameters are used for the training and predicting tests; the only changes are network structures (Number of hidden layers and number of neurons). Since the training dataset 2 is more generalized than dataset 1 (see results from 3.1), training dataset 2 is select for network structure test for approach 1 and approach 2.

2.8 Sensitivity analysis for input features

The Artificial Neural Network or some other statistical model may have problems when dealing with a large number of input parameters or inputs. Not only we concern about the huge meaningless dataset will cost more computation time, but also there might have some close related input parameters in the dataset which might cause redundancy and over-fitting issues (May, Dandy, & Maier, 2011). We have a limited number of inputs in the study so that basic trial and error approaches are performed to address the impact on the network performance

For approach 1, one hidden layer with 5 neurons MLP network is used for all models with different combinations of input parameters. For approach 2, one hidden layer with 10 neurons MLP network is used for all testing models with a designed combination of input parameters. The training and validation accuracy scores would be the Neural Network performance indicator.

We conducted the analysis based on keep output parameter(s) fixed and changing the input parameters, and evaluate the performance of different scenarios either for approach 1 or approach 2, see results in section 3.2.

2.9 Process speed analysis for conventional method and Neural Network approach

The direct Multi-phase fluid flow simulation in pore-scale is time-consuming, the computation expensive process still needs high computation resource even in simplified pore-network modeling and simulation approaches, especially when more complex pore geometries, connectivity, and heterogeneity are introduced into the system.

We compared the time used for conventional calculations and our Neural Network predictions for models with various size. The results in section 3.4 shows the benefits of our Neural Network approach

3. Results

3.1 Results for $Pcow, Kro, Krw$ validation of blind tests

3.1.1 P_{cow} blind tests for Approach 1: (5 neurons single layer)

(1) Model from training dataset 1(60,000 random polygons)

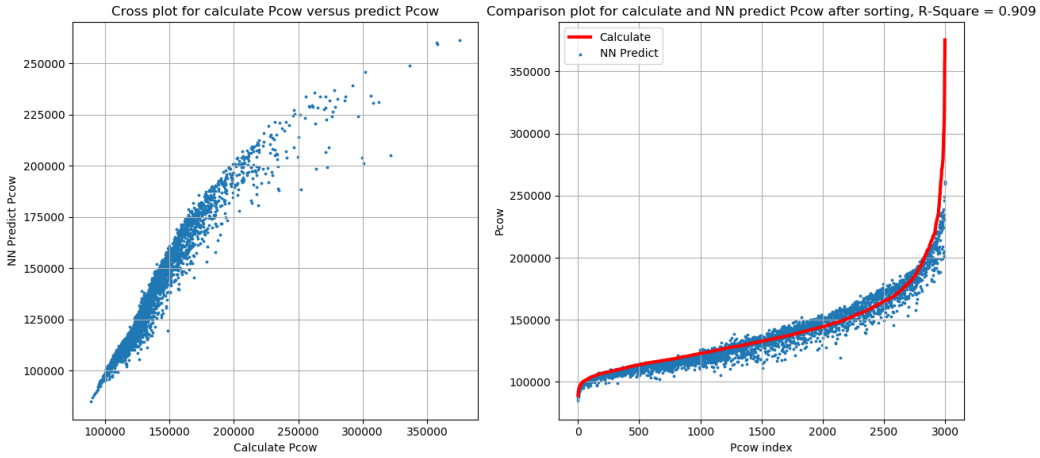

Figure 3.1 Cross-plot and comparison plot for test data with shape factor from 0 to 0.04 for training dataset 1 by approach 1

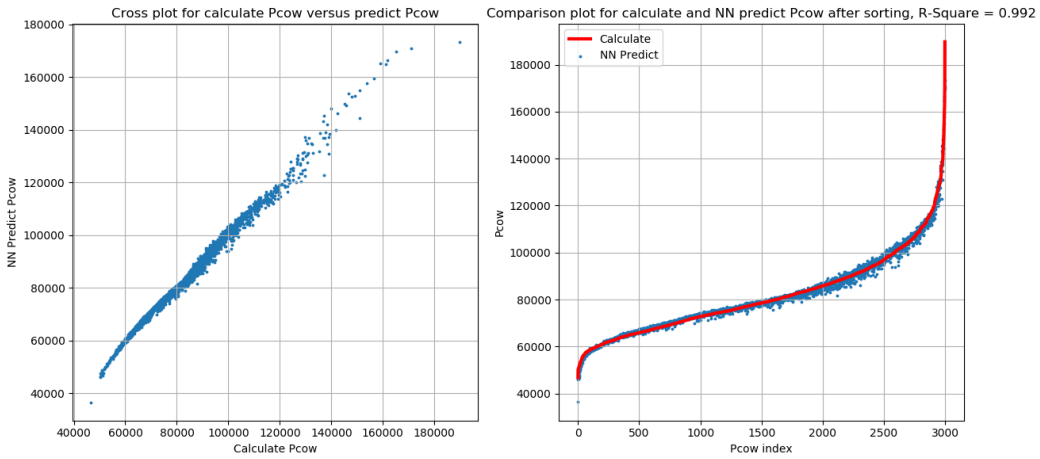

Figure 3.2 Cross-plot and comparison plot for test data with shape factor from 0.04 to 0.07958 for training dataset 1 by approach 1

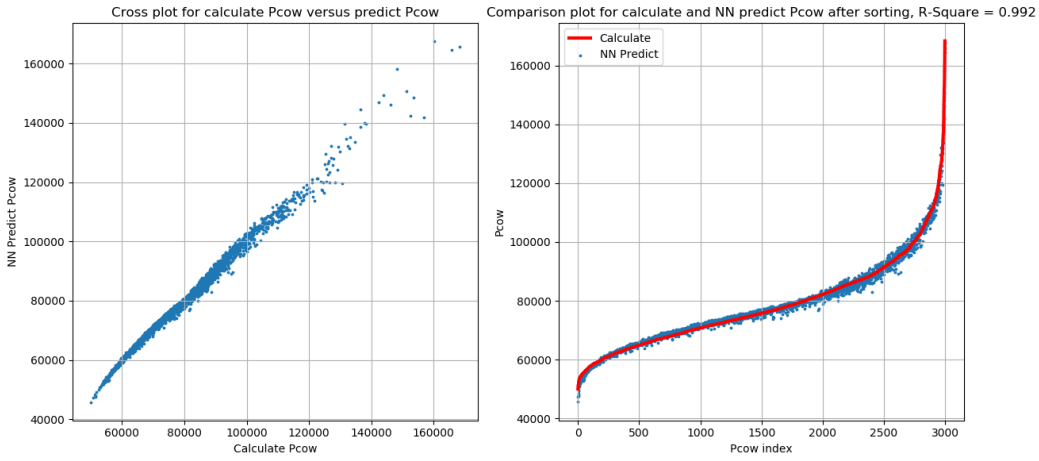

Figure 3.3 Cross-plot and comparison plot for test data with elongation factor from 0 to 0.5 for training dataset 1 by approach 1

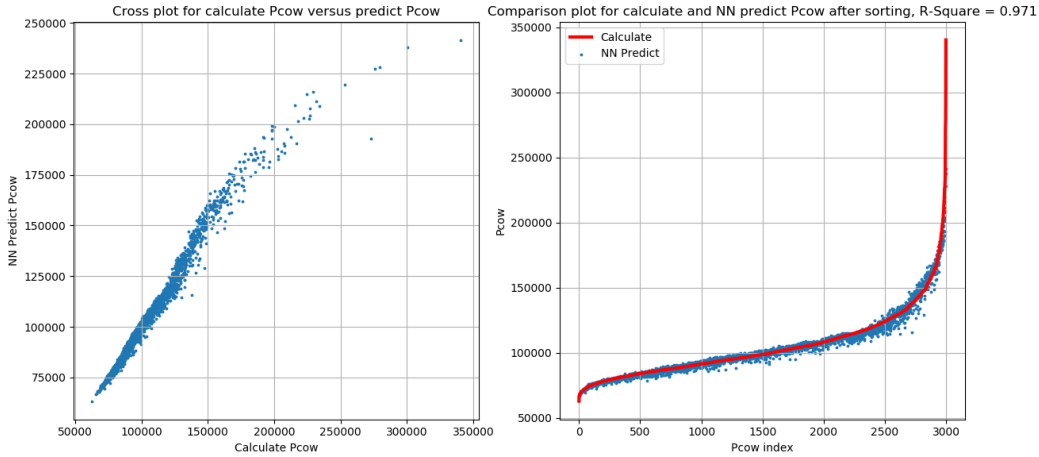

Figure 3.4 Cross-plot and comparison plot for test data with elongation factor from 0.5 to 1 for training dataset 1 by approach 1

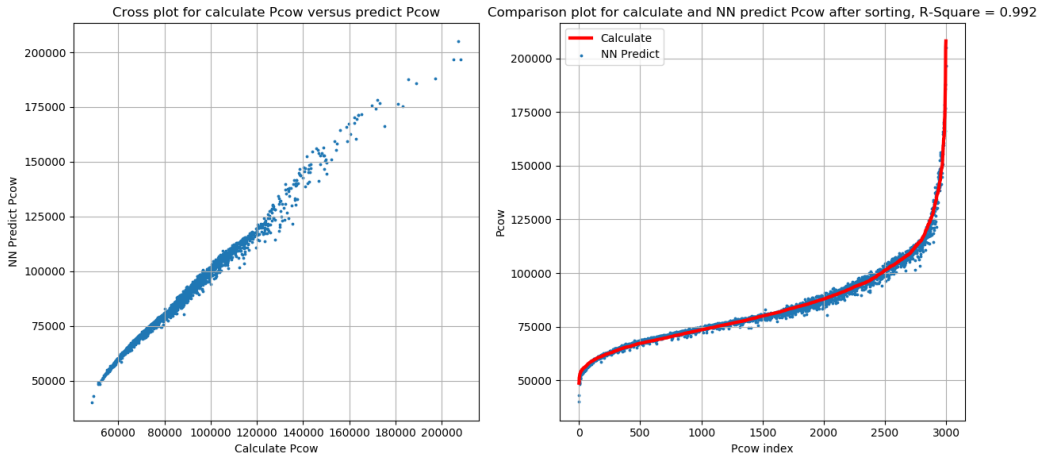

Figure 3.5 Cross-plot and comparison plot for test data with 3,000 random polygons for training dataset 1 by approach 1

As figures show above, most of the results from this model produced relatively good results in terms of either R-Square score (0.971-0.992) or the comparison plots, the overall correlation trends between input geometrical properties and P_{cow} are learned, even there are some oscillations within the prediction scatter points. The results from test dataset 1 with smaller shape factor (0 - 0.04) has more oscillated scatter points and relative lower R-Square score (0.909), also obvious deviation from prediction and actual at high P_{cow} section can be found. One of the reasonable explanation is that the training dataset 1 contains all random polygons and most of them has the shape factor around center of the range (around 0.04), less number of polygons with 'non-circular' shape will lead to less representative of the trained Neural Network.

(2) Model from training dataset 2

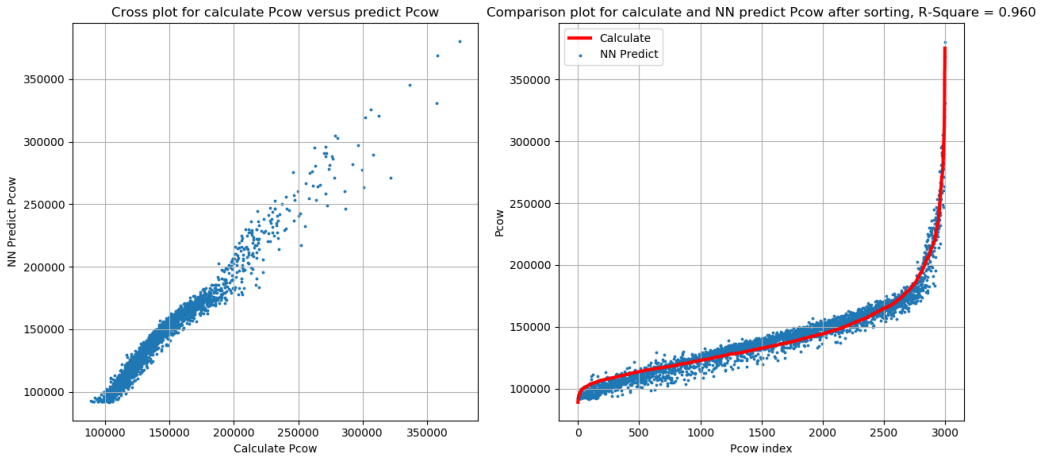

Figure 3.6 Cross-plot and comparison plot for test data with shape factor from 0 to 0.04 for training dataset 2 by approach 1

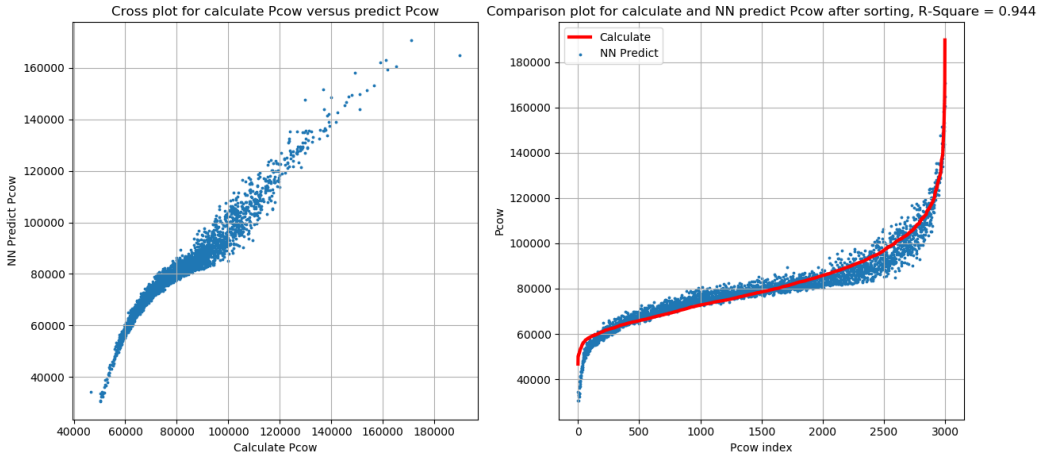

Figure 3.7 Cross-plot and comparison plot for test data with shape factor from 0.04 to 0.07958 for training dataset 2 by approach 1

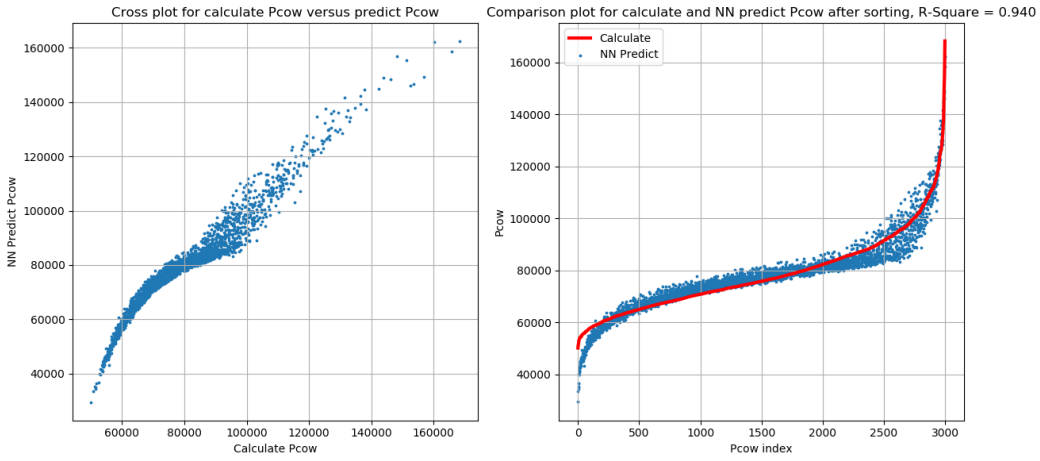

Figure 3.8 Cross-plot and comparison plot for test data with elongation factor from 0 to 0.5 for training dataset 2 by approach 1

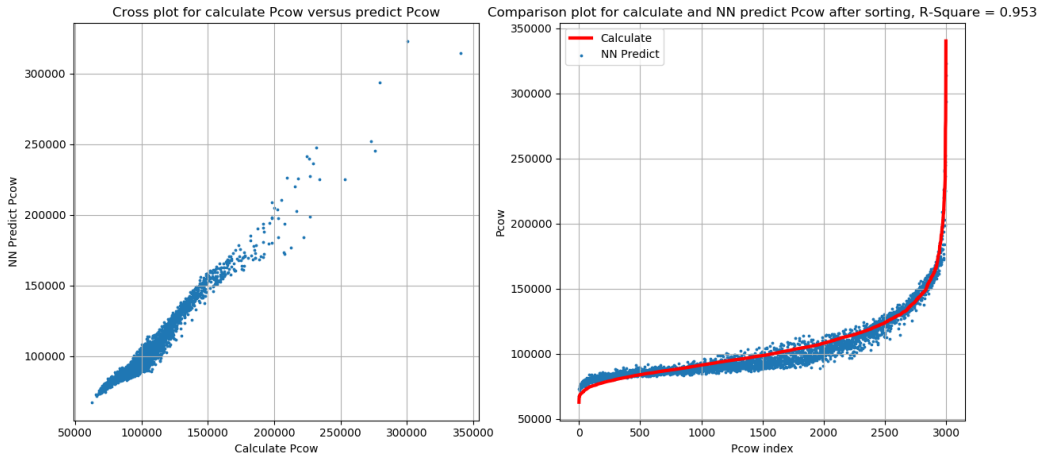

Figure 3.9 Cross-plot and comparison plot for test data with elongation factor from 0.5 to 1 for training dataset 2 by approach 1

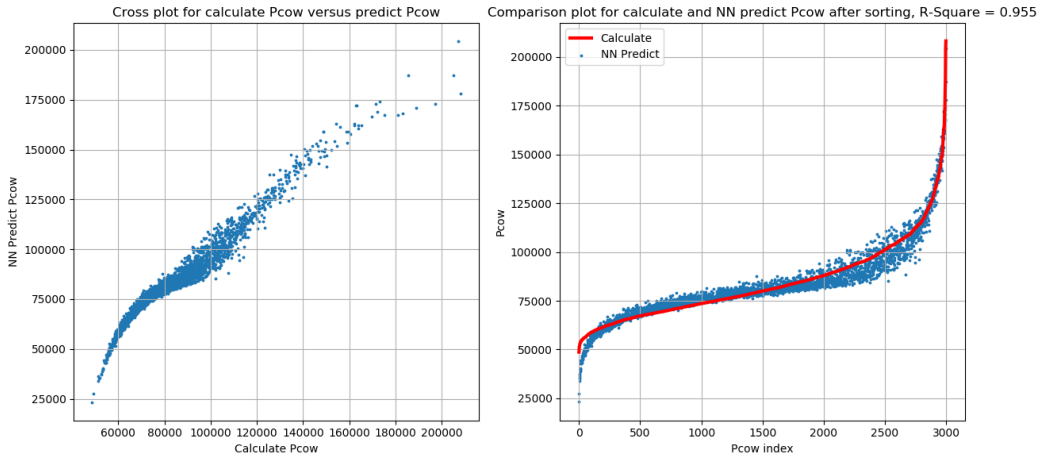

Figure 3.10 Cross-plot and comparison plot for test data with 3,000 random polygons for training dataset 2 by approach 1

The results from test dataset 1 is improved somewhat confirmed the assumption from previous blind test for test dataset 1, but the rest of the blind verifications appeared to be slightly degradation of predicting quality which is believed due to the 'non-smoothness' properties distribution of training dataset 2 (see Figure 2.8-2.9)

3.1.2 P_{cow}, K_{ro}, K_{rw} blind tests for Approach 2: (3 hidden layers, [200,100,50])

(1) Model from training dataset 1(60,000 random polygons)

(a) Output parameter include S_w_area

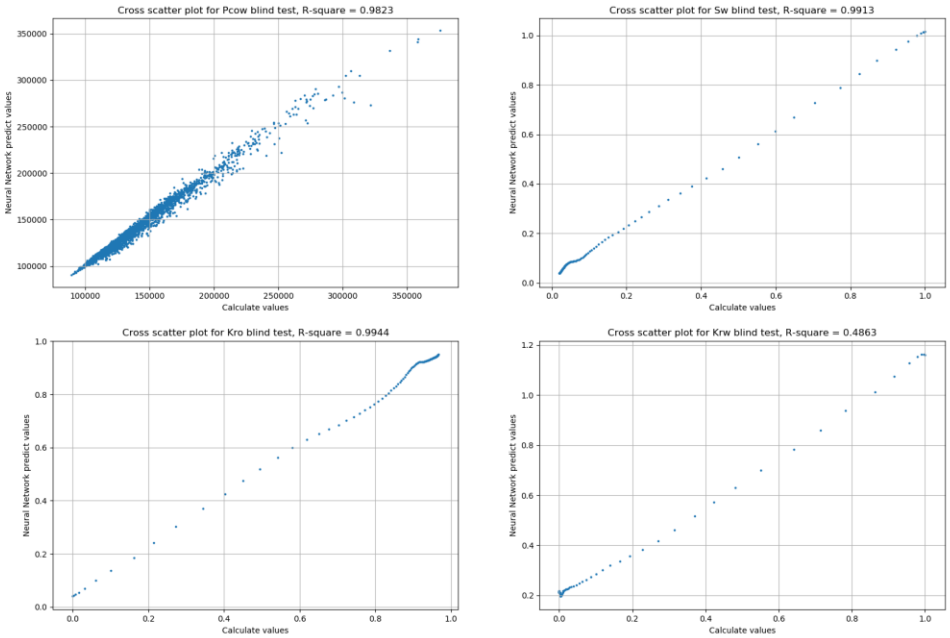

Figure 3.11 Pcow, Sw, Kro, Krw cross-plot for test data with shape factor from 0 to 0.04 for training dataset 1 by approach 2 (Sw_area in output)

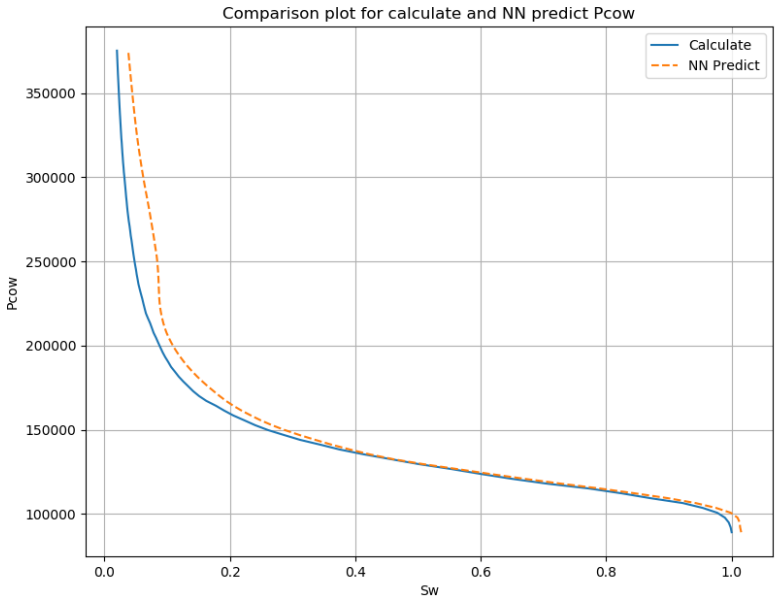

Figure 3.12 Pcow comparison plot for test data with shape factor from 0 to 0.04 for training dataset 1 by approach 2 (Sw_area in output)

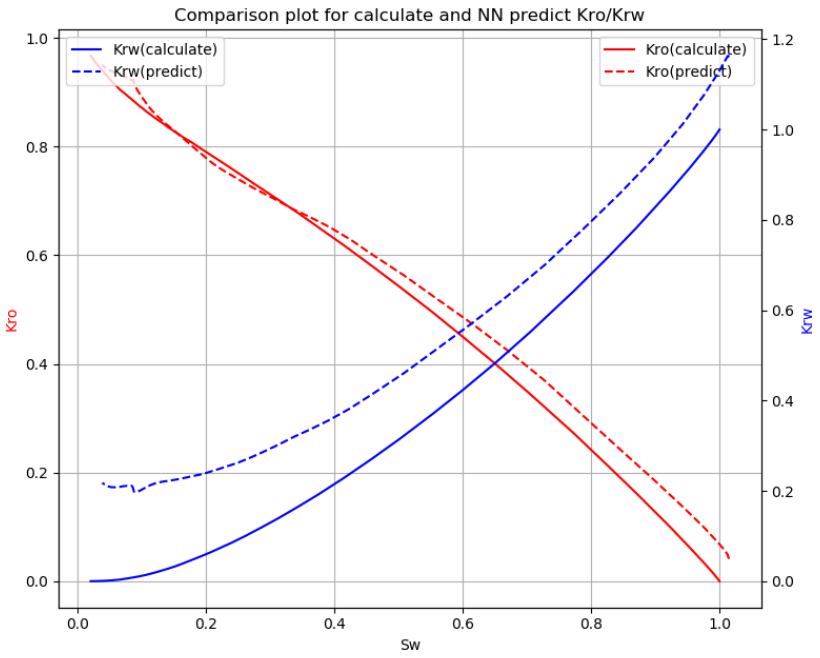

Figure 3.13 Kro/Krw comparison plot for test data with shape factor from 0 to 0.04 for training dataset 1 by approach 2 (Sw_area in output)

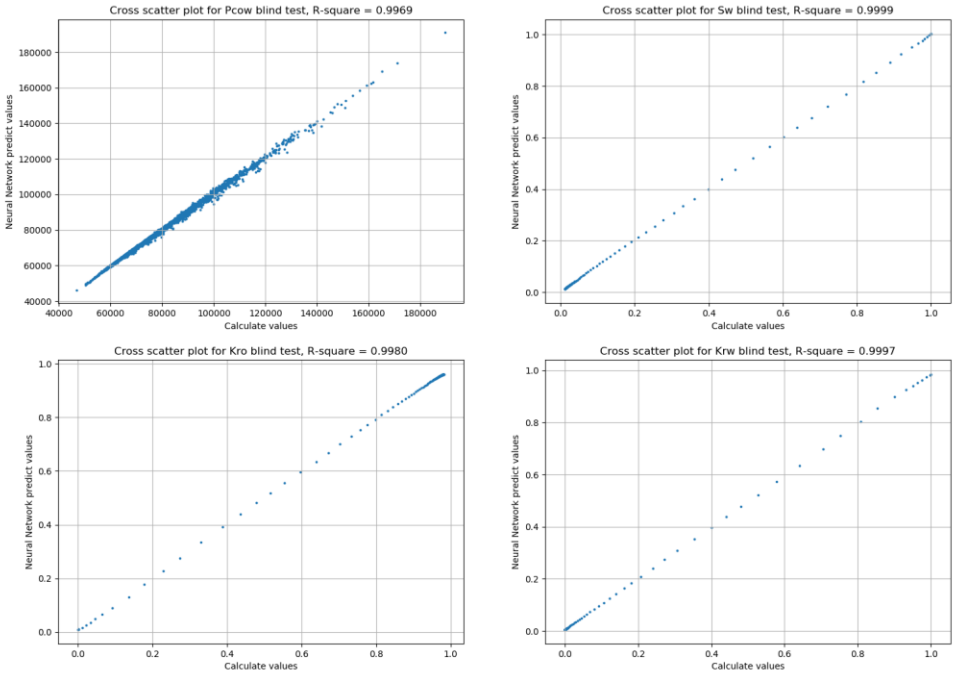

Figure 3.14 Pcow, Sw, Kro, Krw cross-plot for test data with shape factor from 0.04 to 0.07958 for training dataset 1 by approach 2 (Sw_area in output)

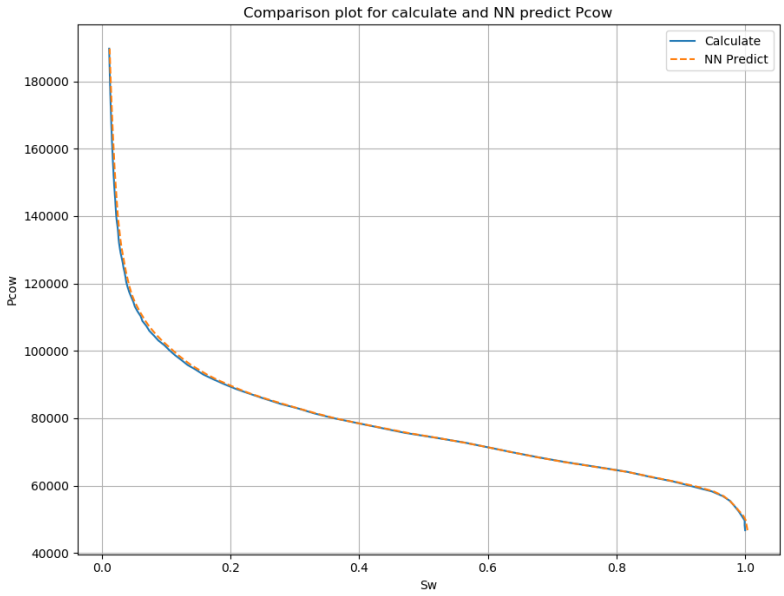

Figure 3.15 Pcow comparison plot for test data with shape factor from 0.04 to 0.07958 for training dataset 1 by approach 2 (Sw_area in output)

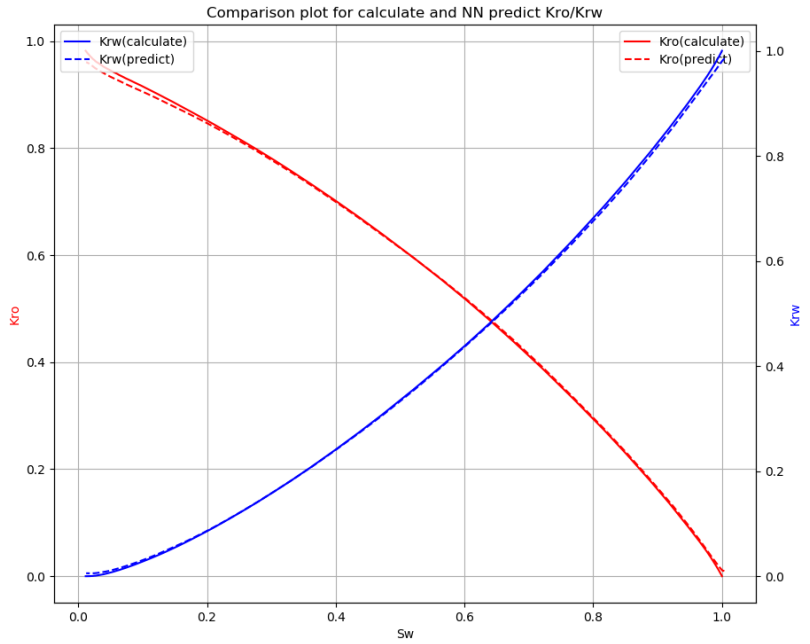

Figure 3.16 Kro/Krw comparison plot for test data with shape factor from 0.04 to 0.07958 for training dataset 1 by approach 2 (Sw_area in output)

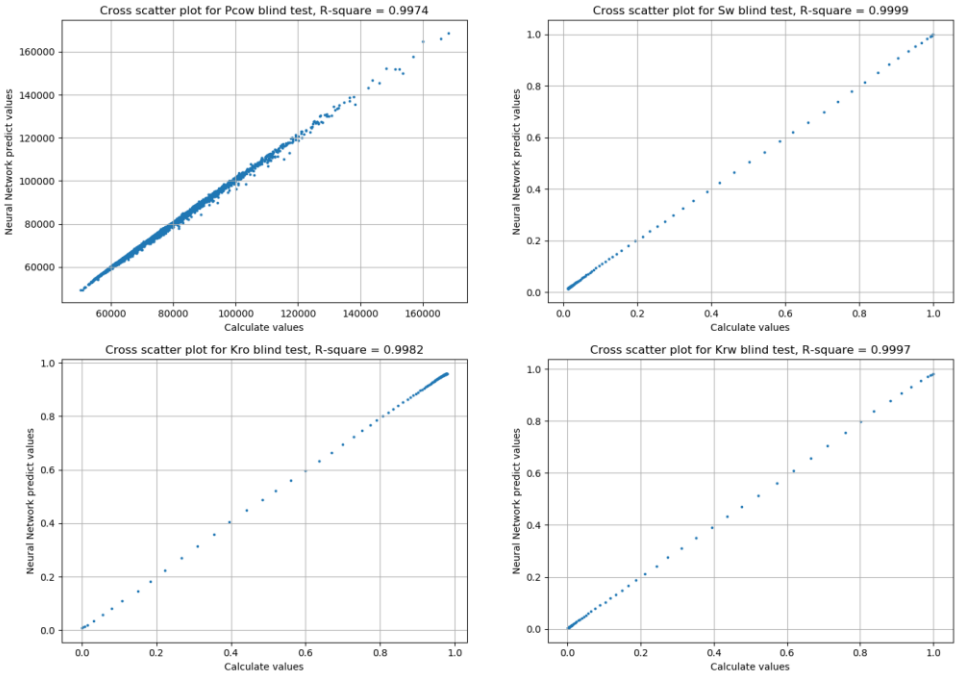

Figure 3.17 Pcow, Sw, Kro, Krw cross-plot for test data with elongation factor from 0 to 0.5 for training dataset 1 by approach 2 (Sw_area in output)

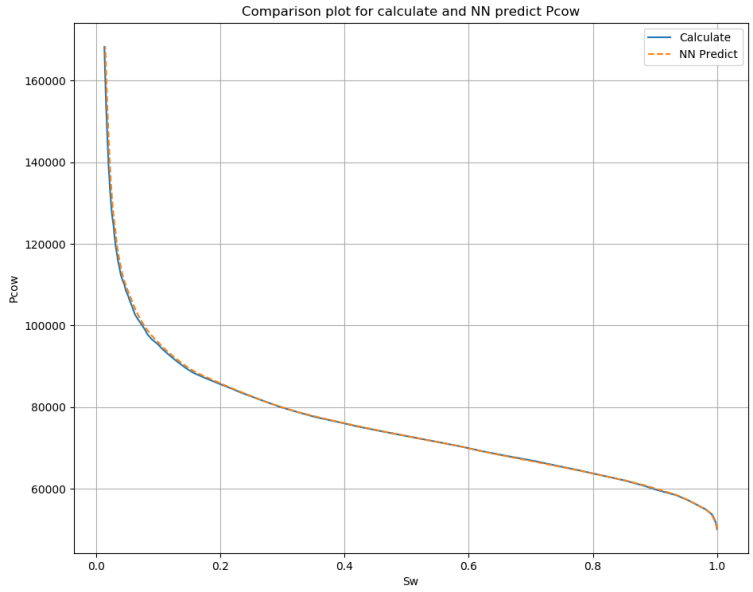

Figure 3.18 Pcow comparison plot for test data with elongation factor from 0 to 0.5 for training dataset 1 by approach 2 (Sw_area in output)

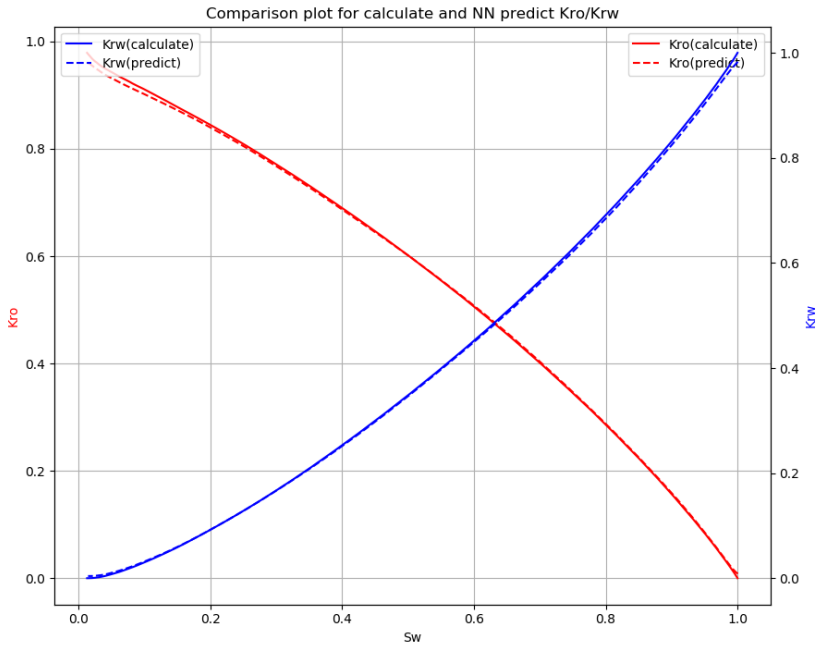

Figure 3.19 Kro/Krw comparison plot for test data with elongation factor from 0 to 0.5 for training dataset 1 by approach 2 (Sw_area in output)

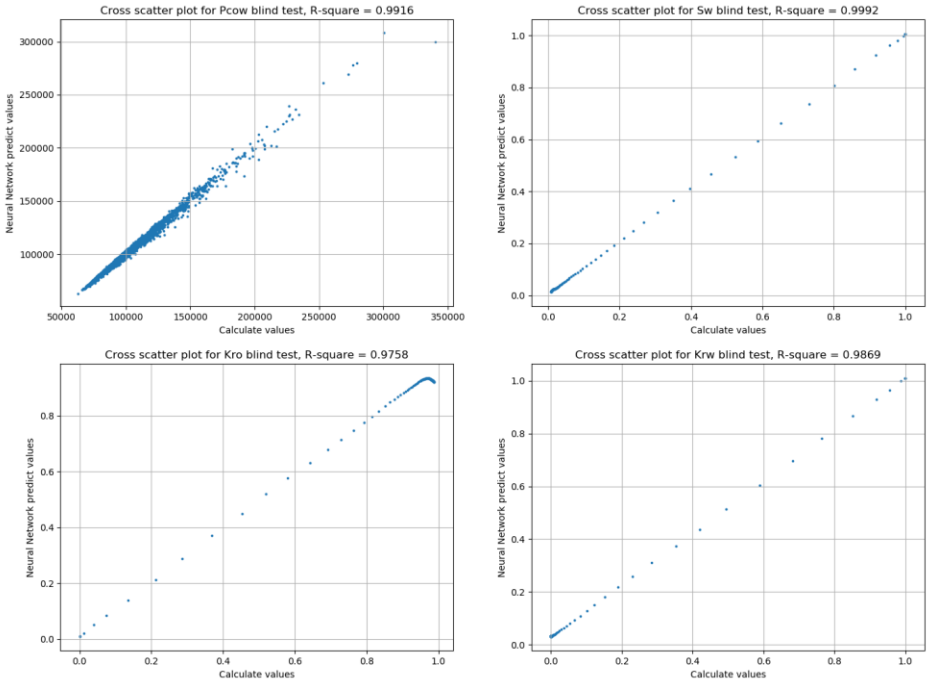

Figure 3.20 Pcow, Sw, Kro, Krw cross-plot for test data with elongation factor from 0.5 to 1 for training dataset 1 by approach 2 (Sw_area in output)

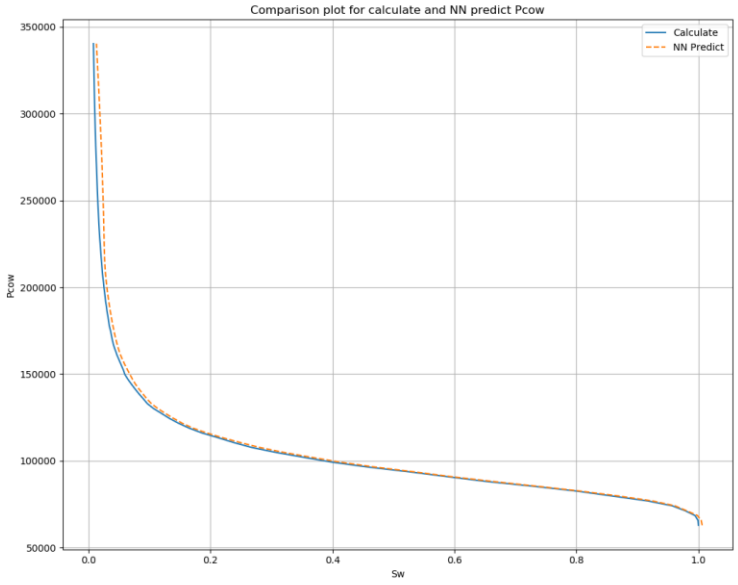

Figure 3.21 Pcow comparison plot for test data with elongation factor from 0.5 to 1 for training dataset 1 by approach 2 (Sw_area in output)

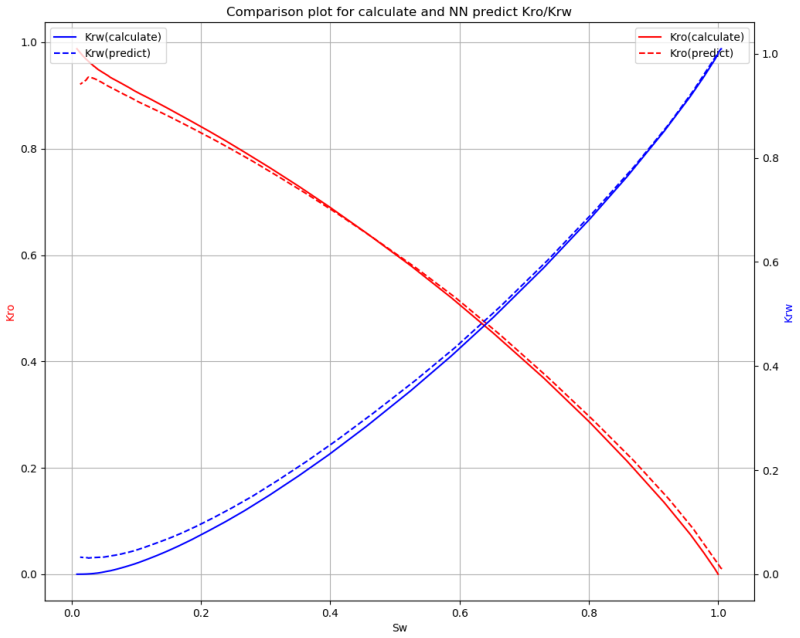

Figure 3.22 Kro/Krw comparison plot for test data with elongation factor from 0.5 to 1 for training dataset 1 by approach 2 (Sw_area in output)

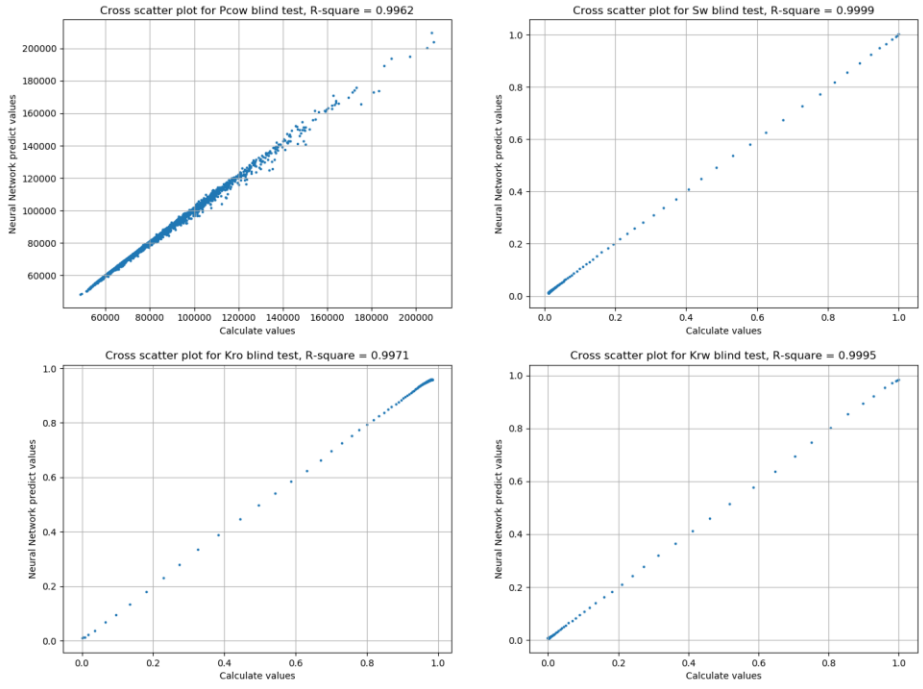

Figure 3.23 Pcow, Sw, Kro, Krw cross-plot for test data with all random polygons for training dataset 1 by approach 2 (Sw_area in output)

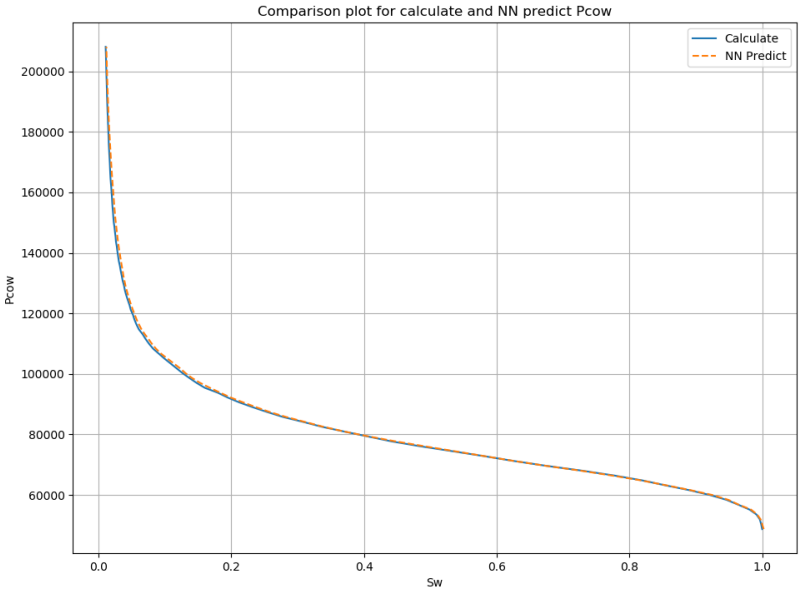

Figure 3.24 Pcow comparison plot for test data with all random polygons for training dataset 1 by approach 2 (Sw_area in output)

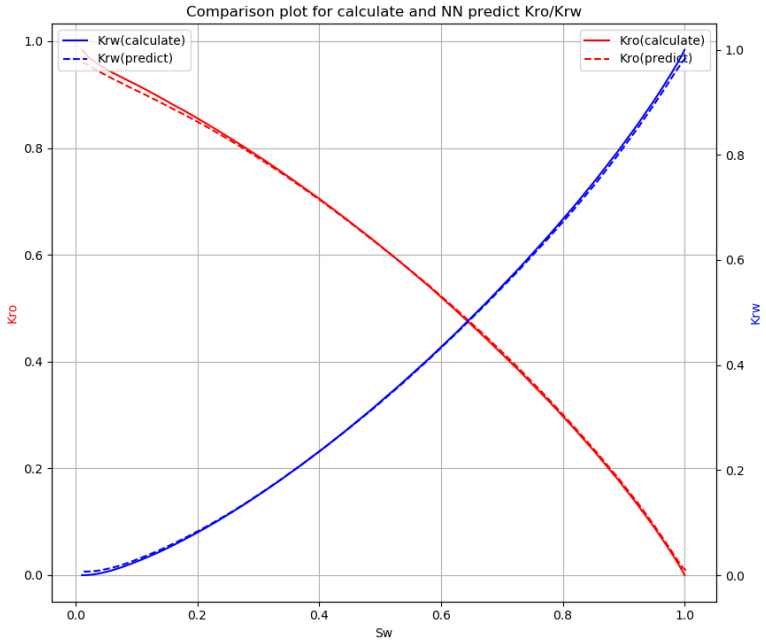

Figure 3.25 Kro/Krw comparison plot for test data with all random polygons for training dataset 1 by approach 2 (Sw_area in output)

Five sets of plots above represent the blind verification results by model from training dataset 1 in approach 2 with S_w_area as one of the output parameter. Similarly, relative bad results can be found in results from test dataset 1, low R-Square score for K_{rw} (0.4863), good in R-Square score for P_{cow} but obvious deviation in comparison plot (Figure 3.12), the predict relative permeability curves are not acceptable as well (Figure 3.13).

Also for the results from test dataset 4 (elongation factor belongs to (0.5,1)) shows the slightly worse than rest of the test data prediction (Figure 3.21 and Figure 3.22) can also due to the training dataset is not contain enough polygons with 'more elongated' polygons. Notably, the predictions of P_{cow} in approach 2 are much better than in approach 1 in terms of the final comparison plots

(b) Output parameter include S_w_frac

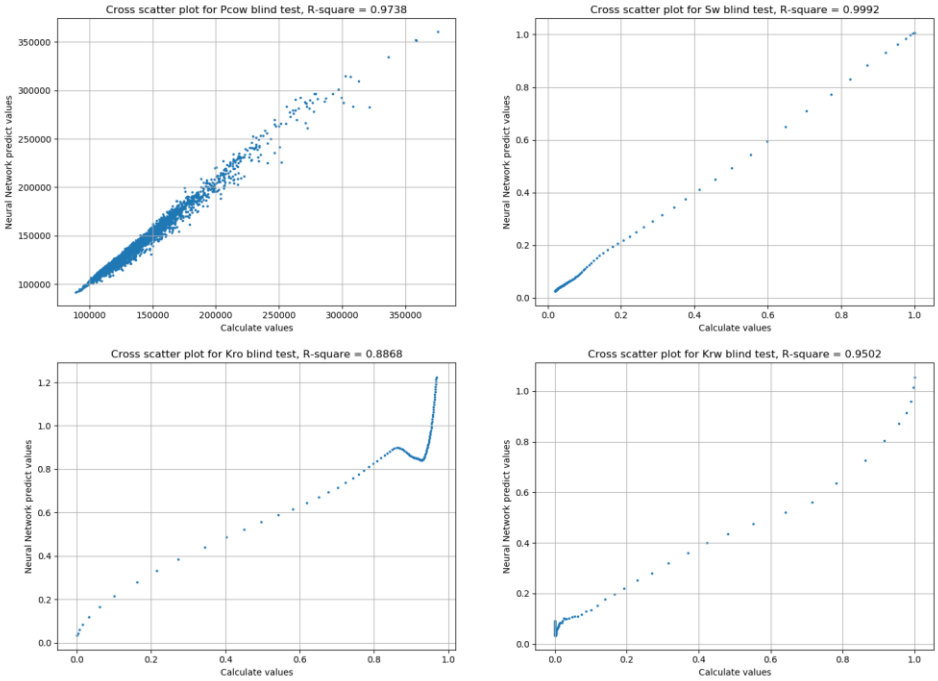

Figure 3.26 Pcow, Sw, Kro, Krw cross-plot for test data with shape factor from 0 to 0.04 for training dataset 1 by approach 2 (Sw_frac in output)

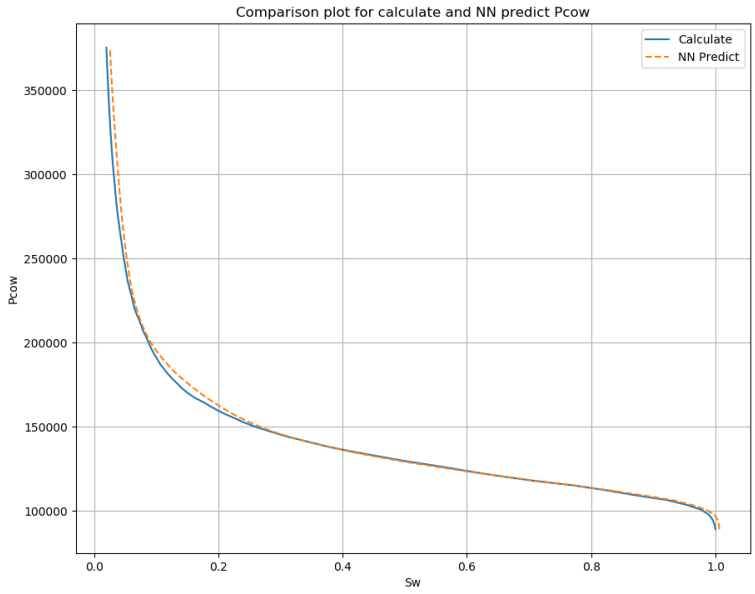

Figure 3.27 Pcow comparison plot for test data with shape factor from 0 to 0.04 for training dataset 1 by approach 2 (Sw_frac in output)

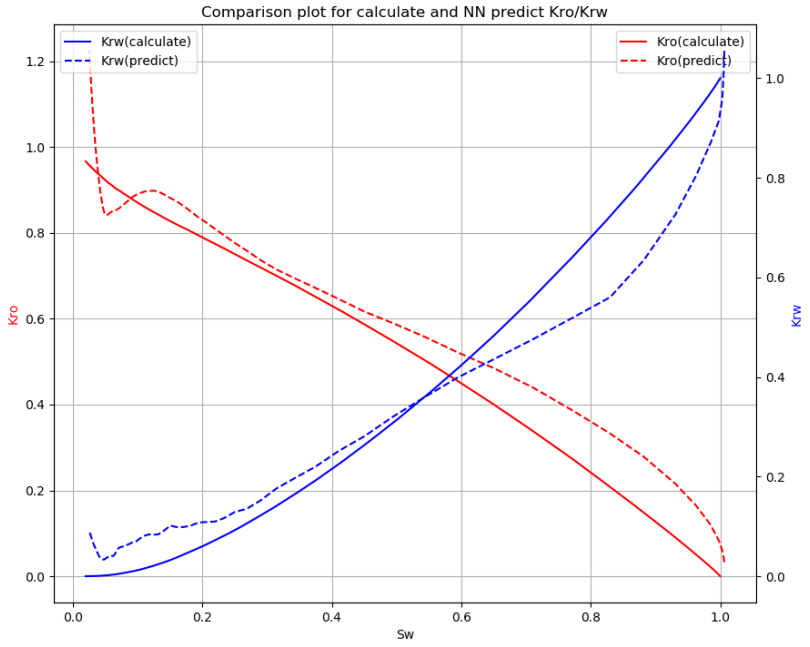

Figure 3.28 Kro/Krw comparison plot for test data with shape factor from 0 to 0.04 for training dataset 1 by approach 2 (Sw_frac in output)

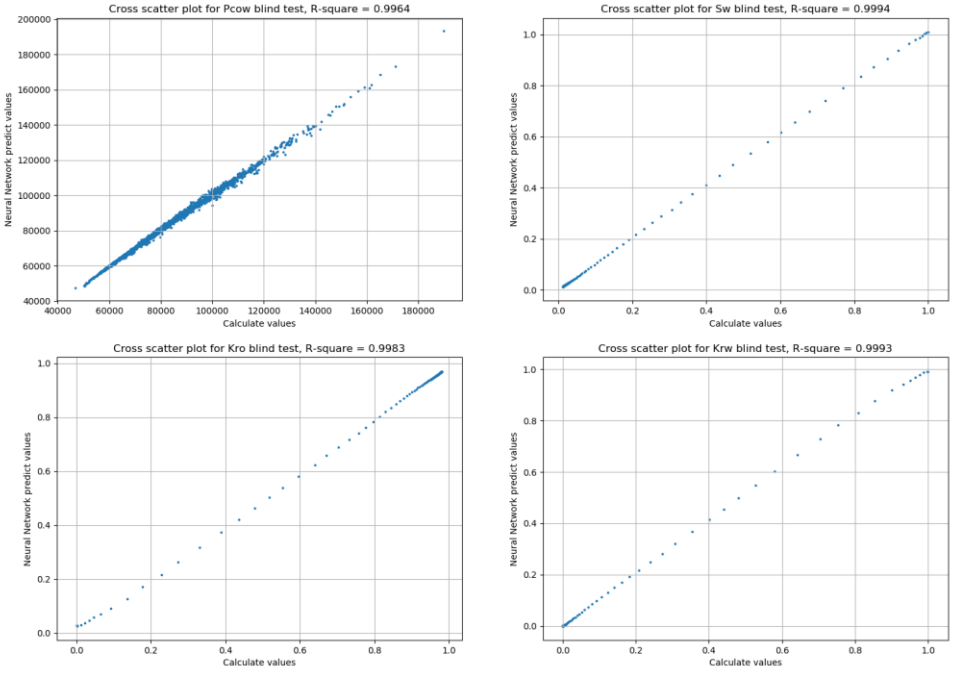

Figure 3.29 Pcow, Sw, Kro, Krw cross-plot for test data with shape factor from 0.04 to 0.07958 for training dataset 1 by approach 2 (Sw_frac in output)

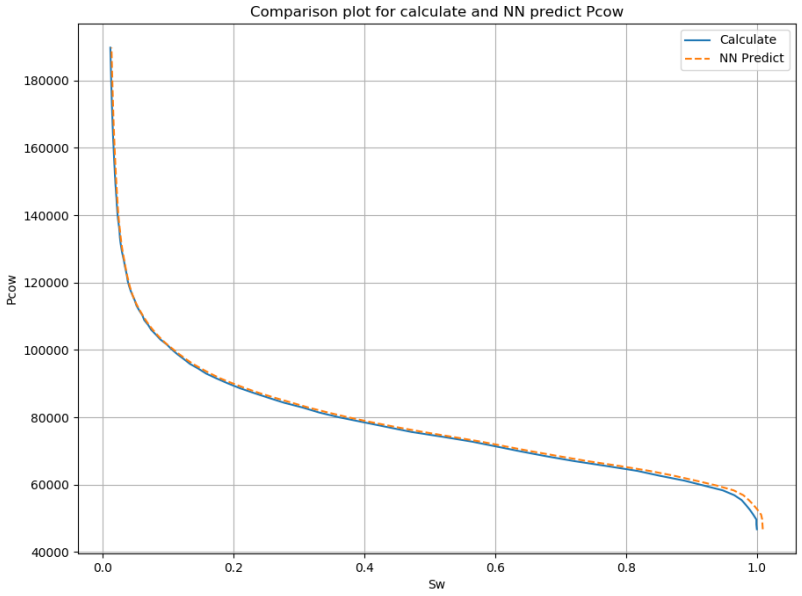

Figure 3.30 Pcow comparison plot for test data with shape factor from 0.04 to 0.07958 for training dataset 1 by approach 2 (Sw_frac in output)

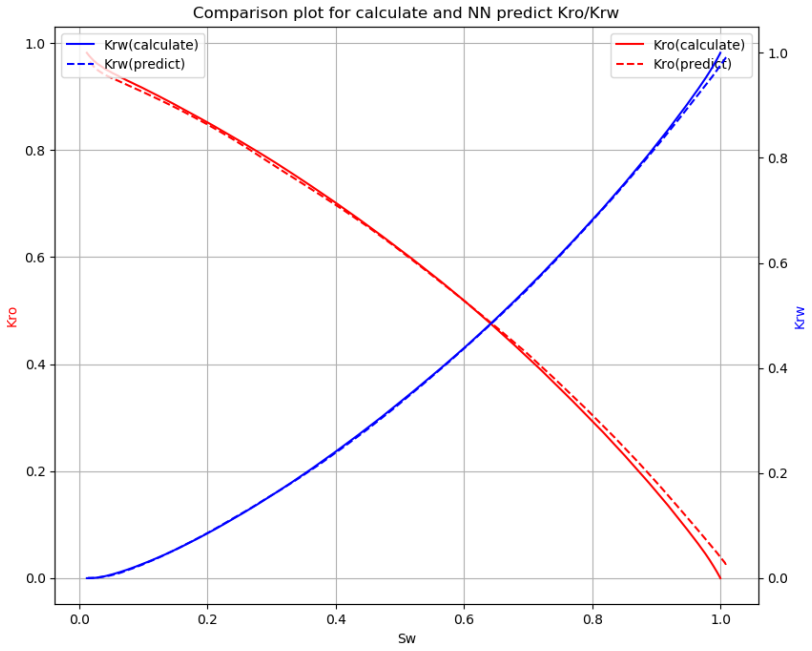

Figure 3.31 Kro/Krw comparison plot for test data with shape factor from 0.04 to 0.07958 for training dataset 1 by approach 2 (Sw_frac in output)

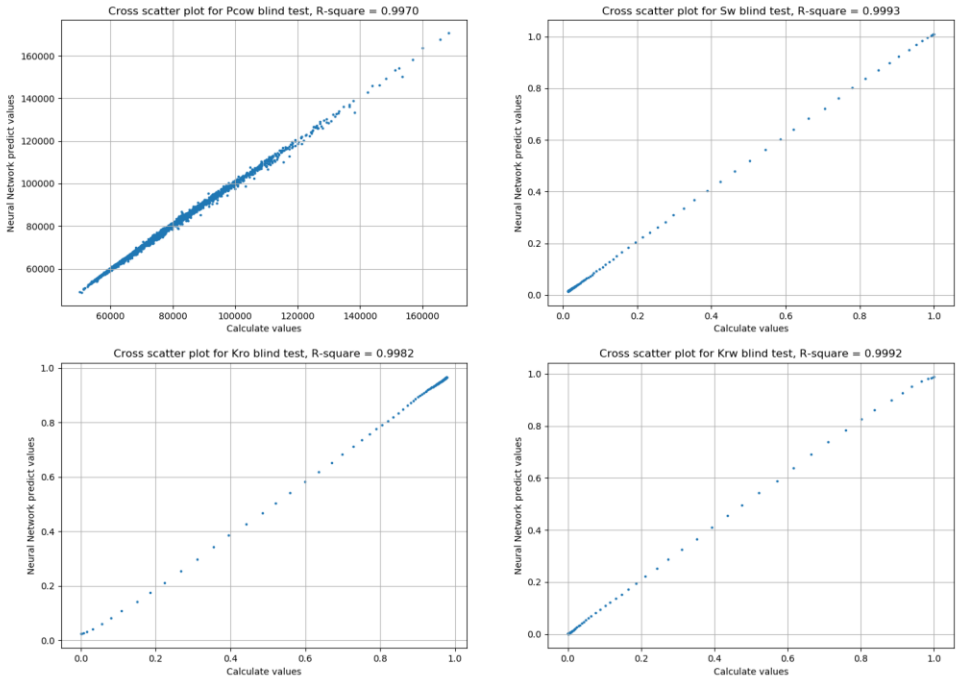

Figure 3.32 Pcow, Sw, Kro, Krw cross-plot for test data with elongation factor from 0 to 0.5 for training dataset 1 by approach 2 (Sw_frac in output)

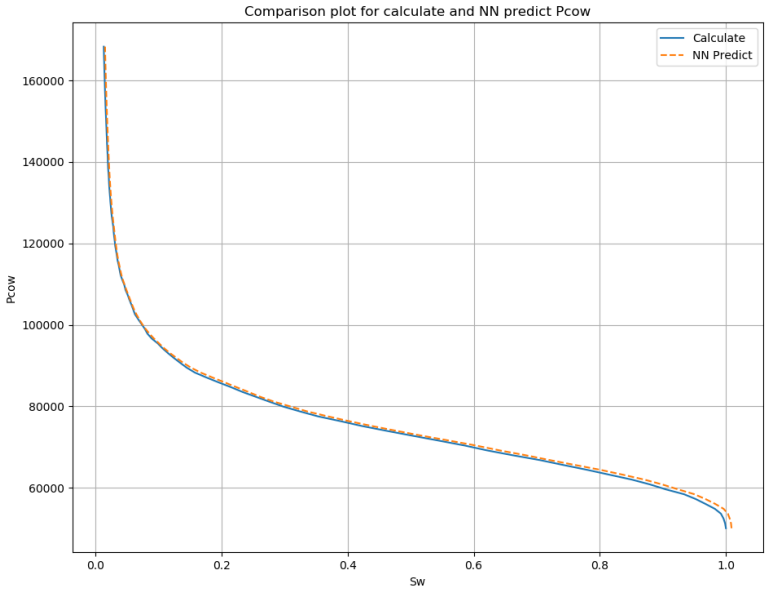

Figure 3.33 Pcow comparison plot for test data with elongation factor from 0 to 0.5 for training dataset 1 by approach 2 (Sw_frac in output)

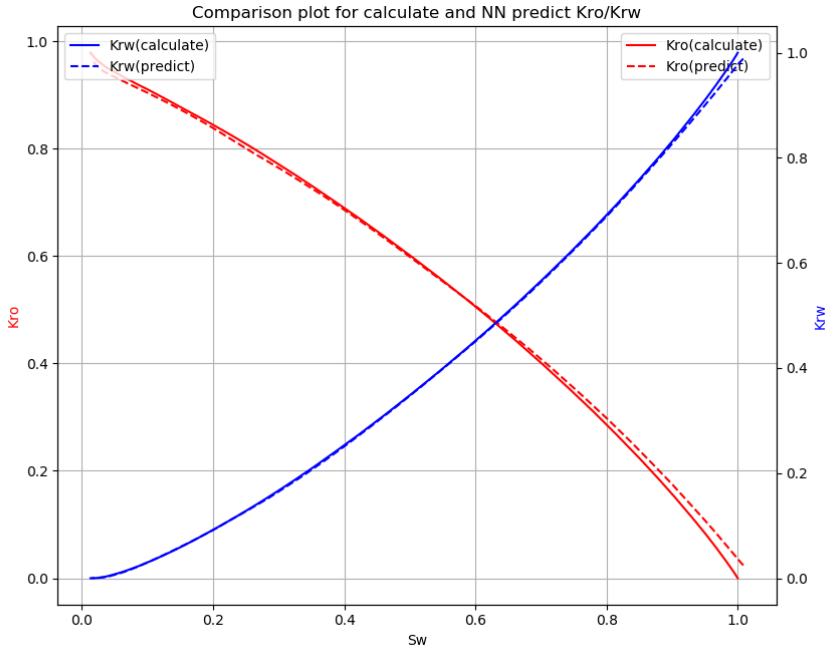

Figure 3.34 Kro/Krw comparison plot for test data with elongation factor from 0 to 0.5 for training dataset 1 by approach 2 (Sw_frac in output)

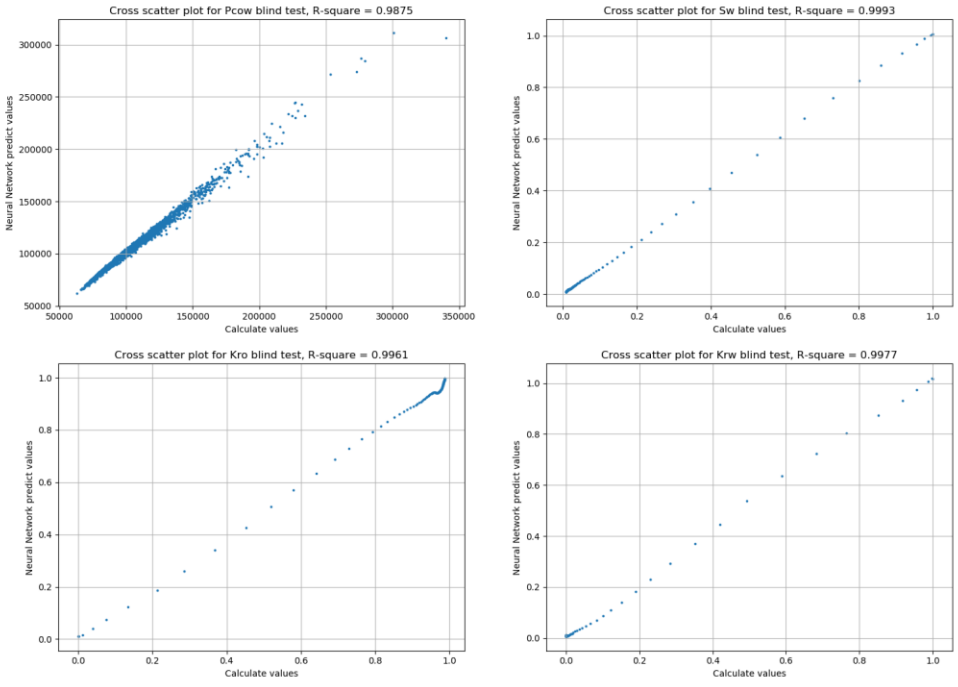

Figure 3.35 Pcow, Sw, Kro, Krw cross-plot for test data with elongation factor from 0.5 to 1 for training dataset 1 by approach 2 (Sw_frac in output)

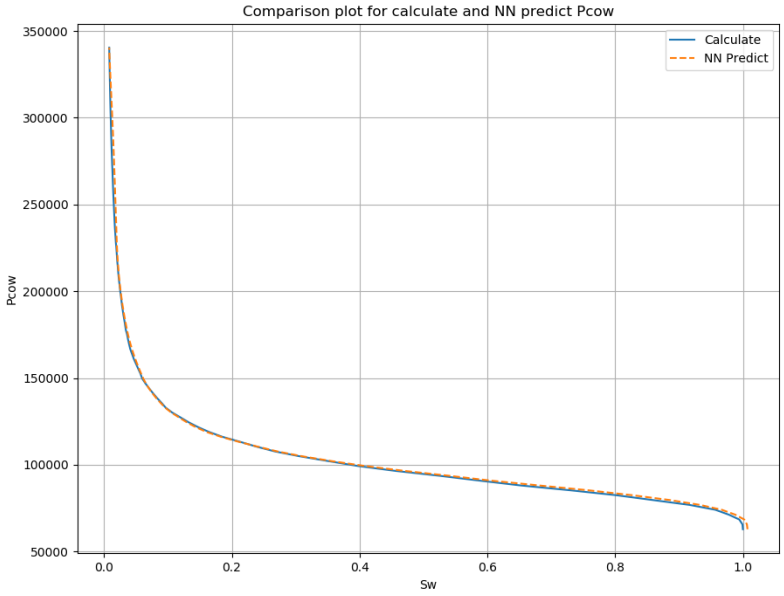

Figure 3.36 Pcow comparison plot for test data with elongation factor from 0.5 to 1 for training dataset 1 by approach 2 (Sw_frac in output)

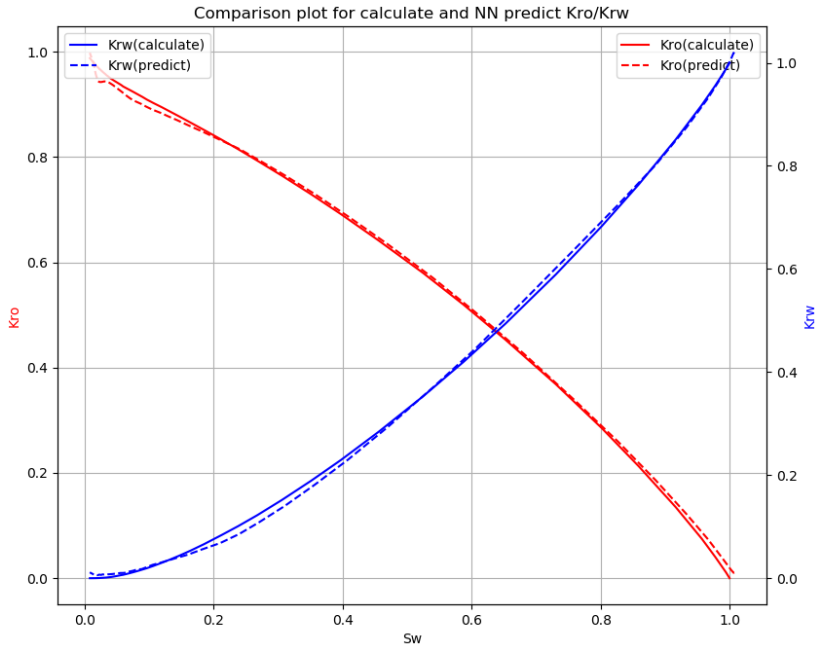

Figure 3.37 Kro/Krw comparison plot for test data with elongation factor from 0.5 to 1 for training dataset 1 by approach 2 (Sw_frac in output)

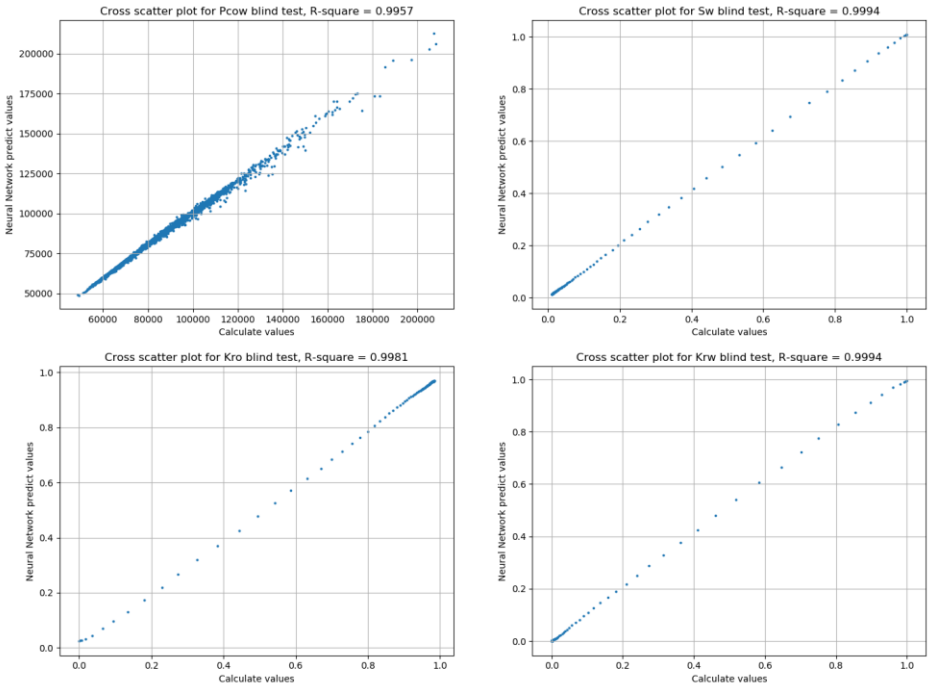

Figure 3.38 Pcow, Sw, Kro, Krw cross-plot for test data with all random polygons for training dataset 1 by approach 2 (Sw_frac in output)

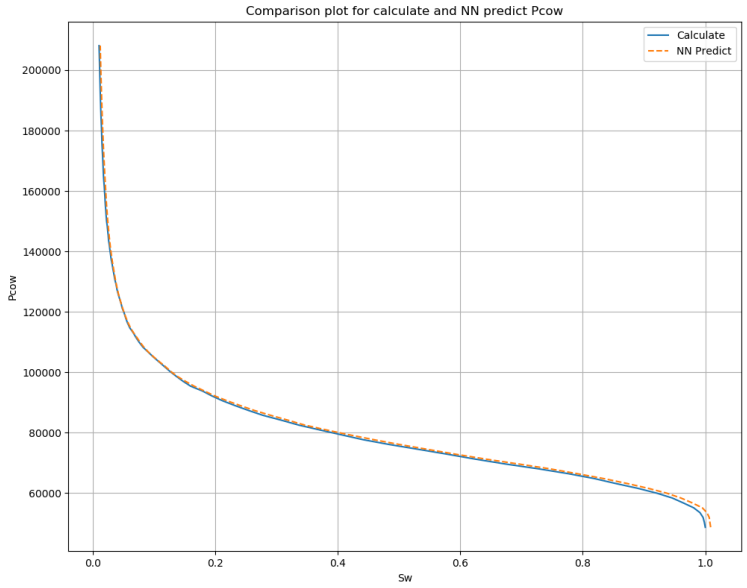

Figure 3.39 Pcow comparison plot for test data with all random polygons for training dataset 1 by approach 2 (Sw_frac in output)

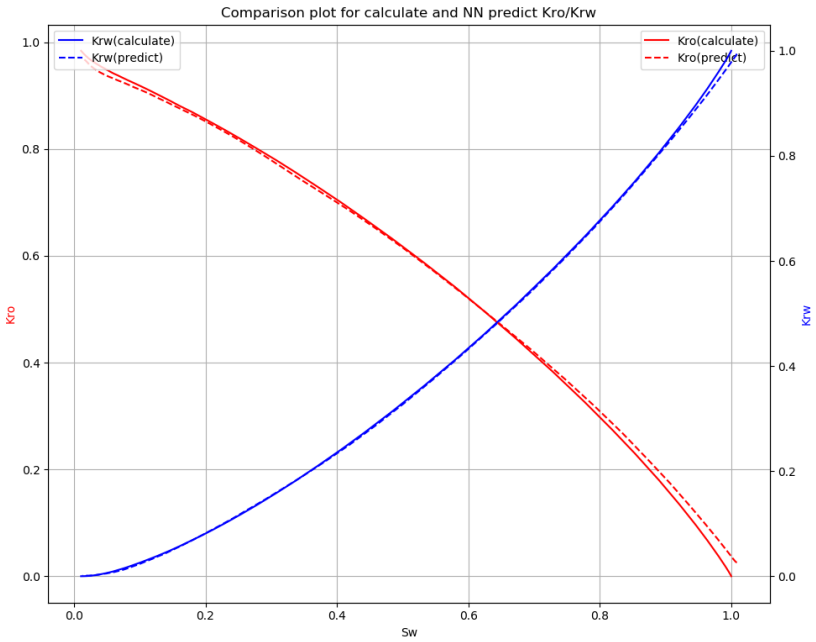

Figure 3.40 Kro/Krw comparison plot for test data with all random polygons for training dataset 1 by approach 2 (Sw_frac in output)

By test using S_w_frac as one of the output parameter instead of train and predict Sw_area, we found that the results from this approach is worse than S_w_area approach especially in predicting S_w, K_{ro}, K_{rw}.

(2) Model from training dataset 2

(i) Output parameter include S_w_area

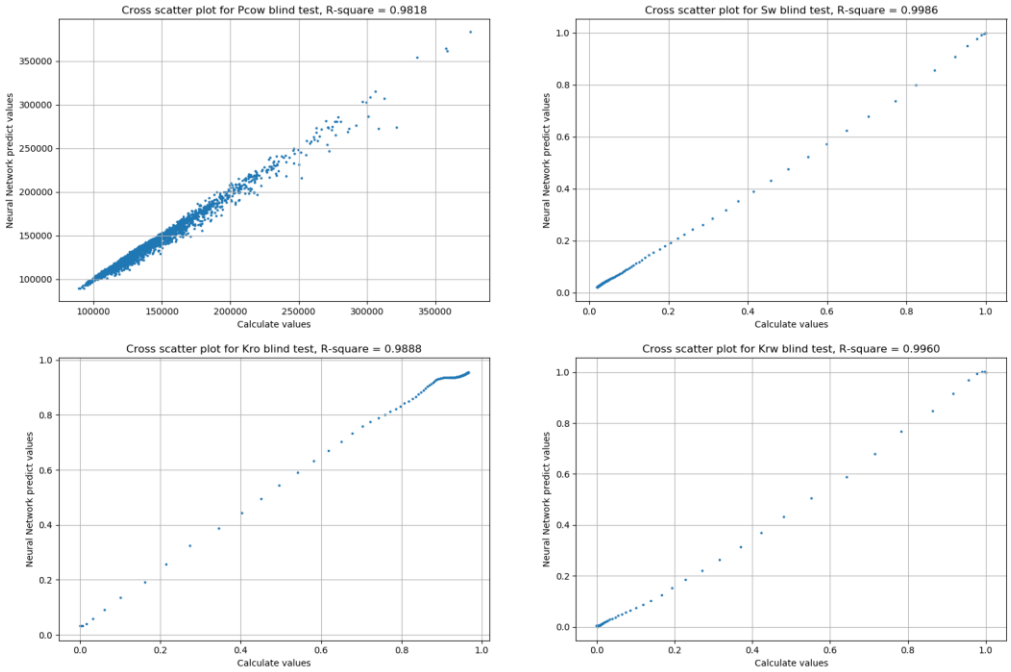

Figure 3.41 Pcow, Sw, Kro, Krw cross-plot for test data with shape factor from 0 to 0.04 for training dataset 2 by approach 2 (Sw_area in output)

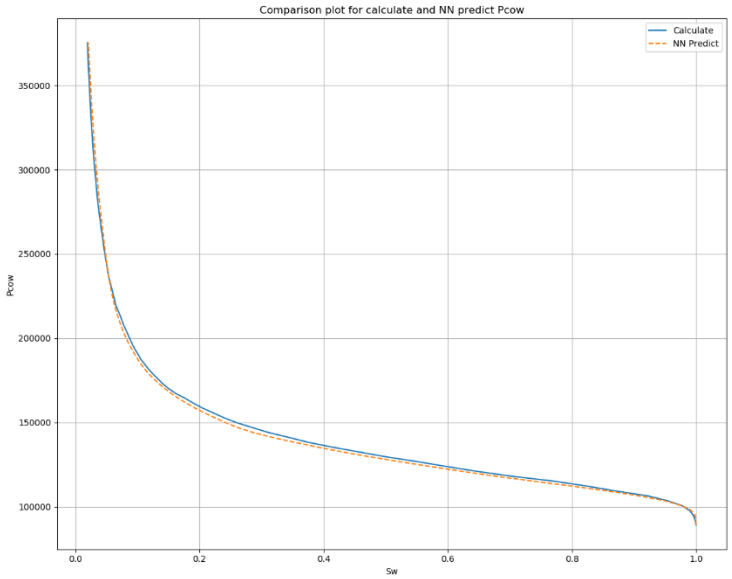

Figure 3.42 Pcow comparison plot for test data with shape factor from 0 to 0.04 for training dataset 2 by approach 2 (Sw_area in output)

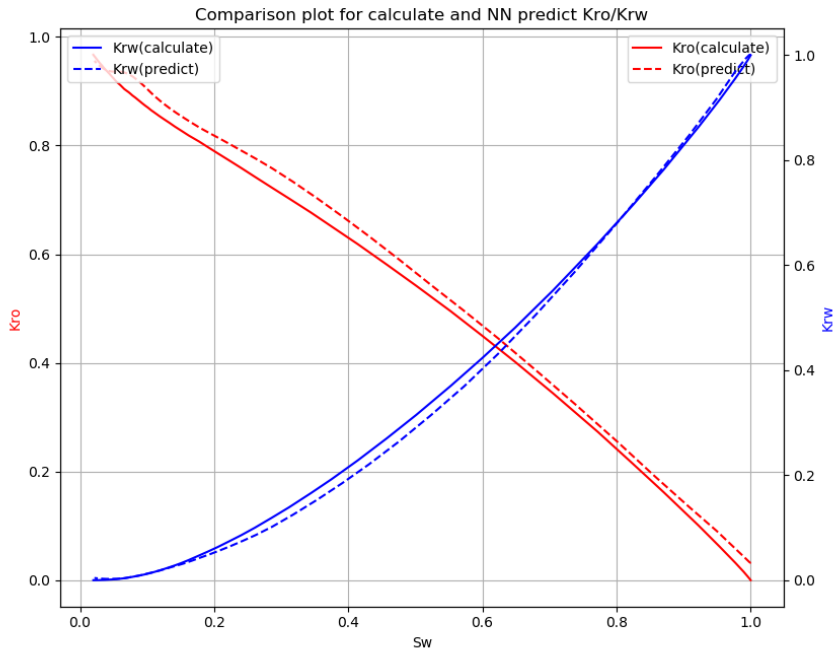

Figure 3.43 Kro/Krw comparison plot for test data with shape factor from 0 to 0.04 for training dataset 2 by approach 2 (Sw_area in output)

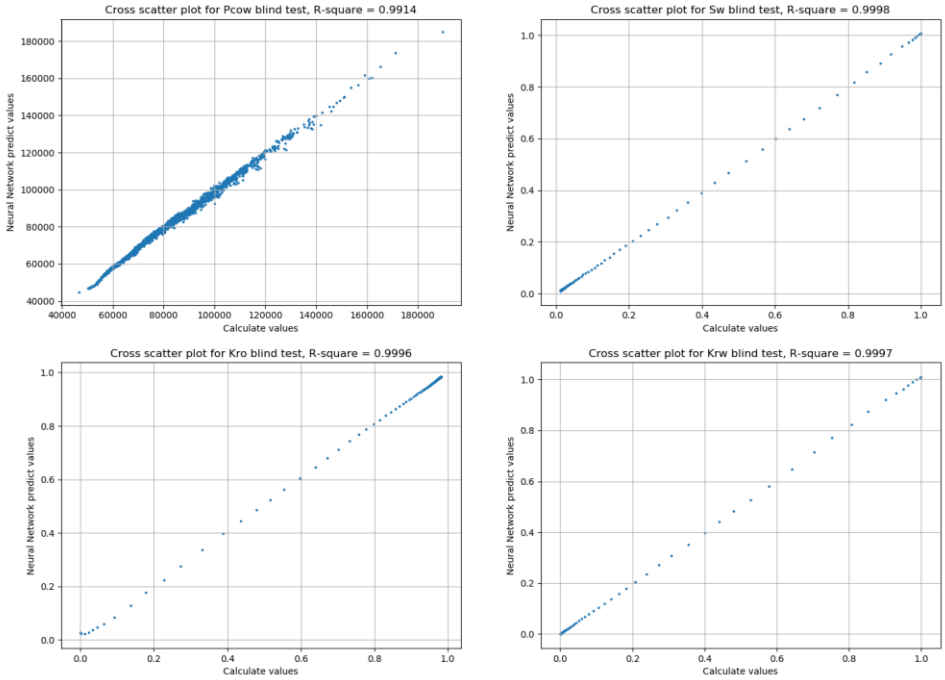

Figure 3.44 Pcow, Sw, Kro, Krw cross-plot for test data with shape factor from 0.04 to 0.07958 for training dataset 2 by approach 2 (Sw_area in output)

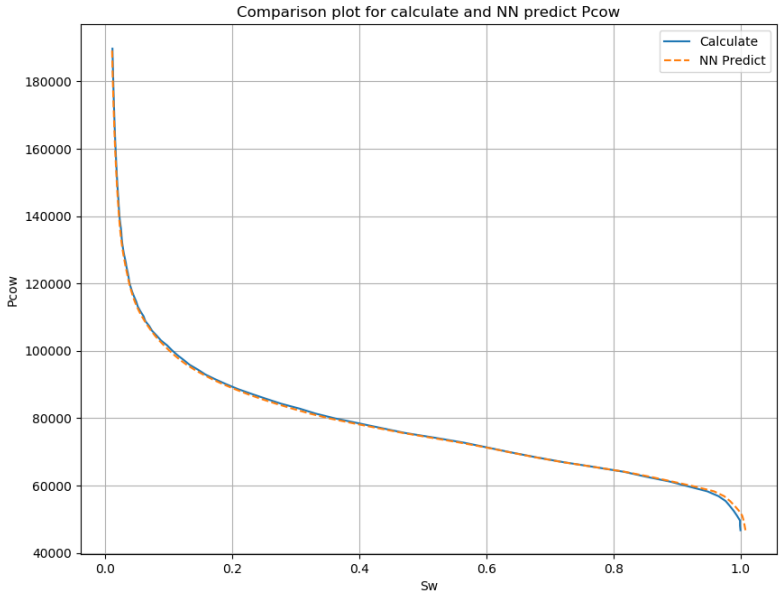

Figure 3.45 Pcow comparison plot for test data with shape factor from 0.04 to 0.07958 for training dataset 2 by approach 2 (Sw_area in output)

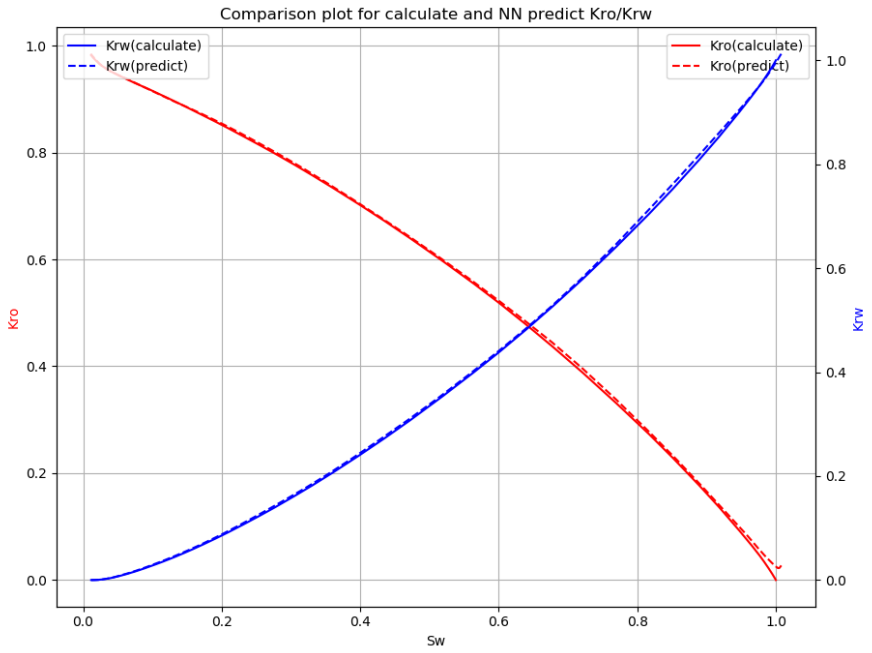

Figure 3.46 Kro/Krw comparison plot for test data with shape factor from 0.04 to 0.07958 for training dataset 2 by approach 2 (Sw_area in output)

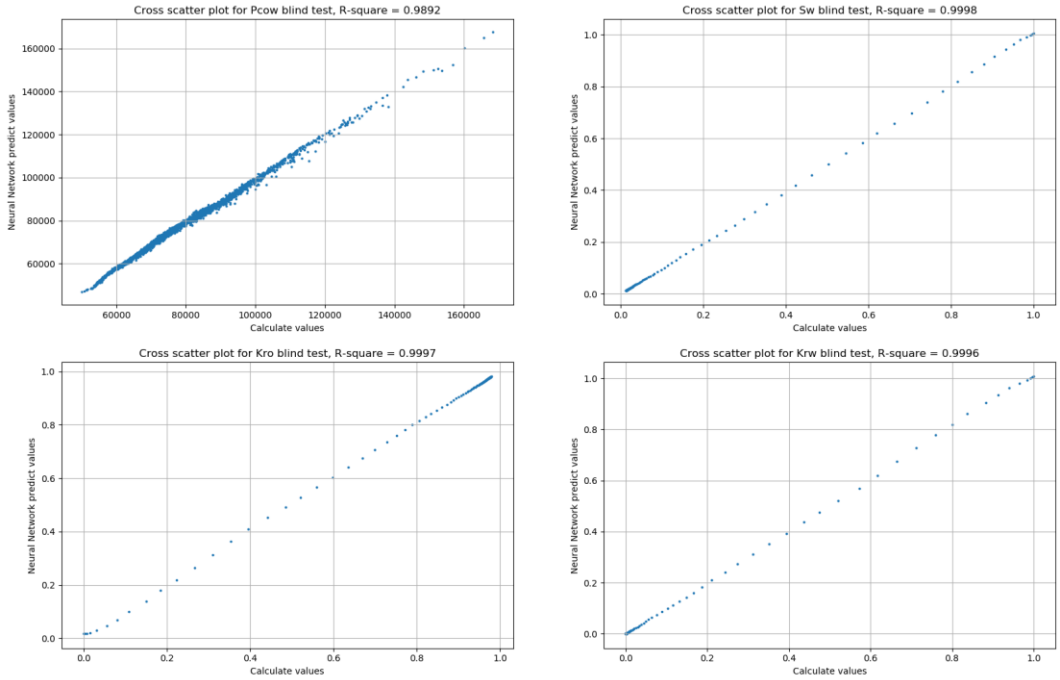

Figure 3.47 Pcow, Sw, Kro, Krw cross-plot for test data with elongation factor from 0 to 0.5 for training dataset 2 by approach 2 (Sw_area in output)

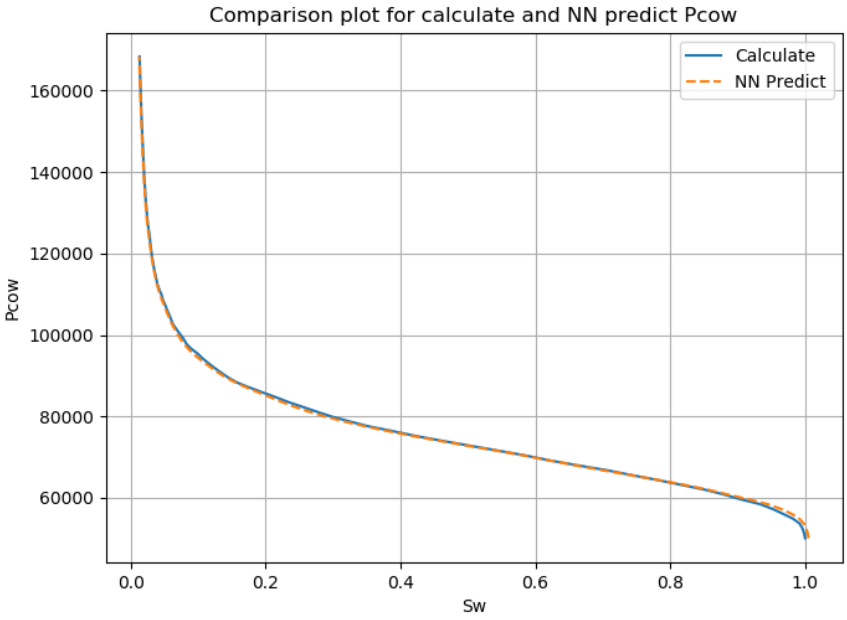

Figure 3.48 Pcow comparison plot for test data with elongation factor from 0 to 0.5 for training dataset 2 by approach 2 (Sw_area in output)

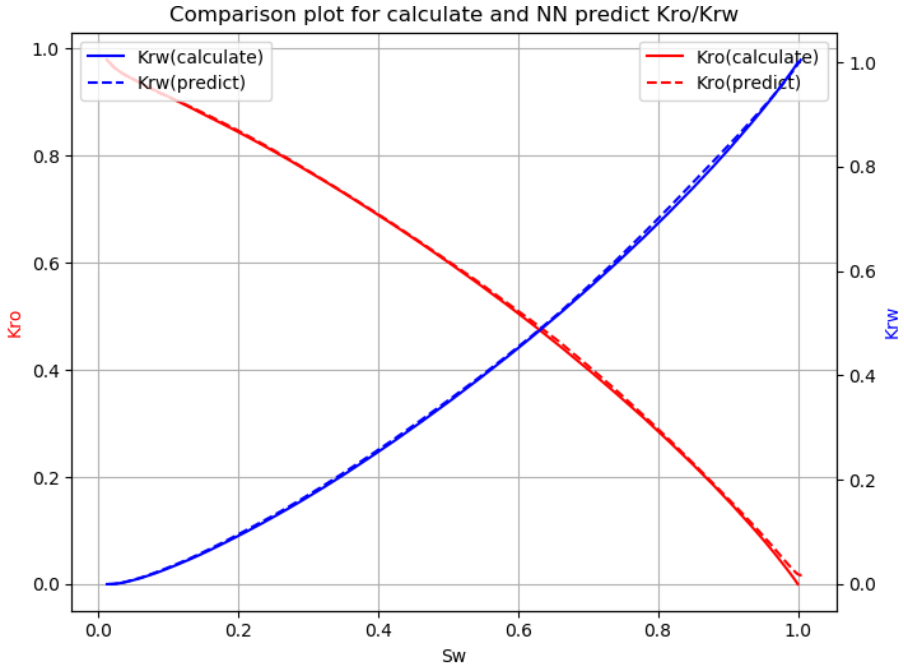

Figure 3.49 Kro/Krw comparison plot for test data with elongation factor from 0 to 0.5 for training dataset 2 by approach 2 (Sw_area in output)

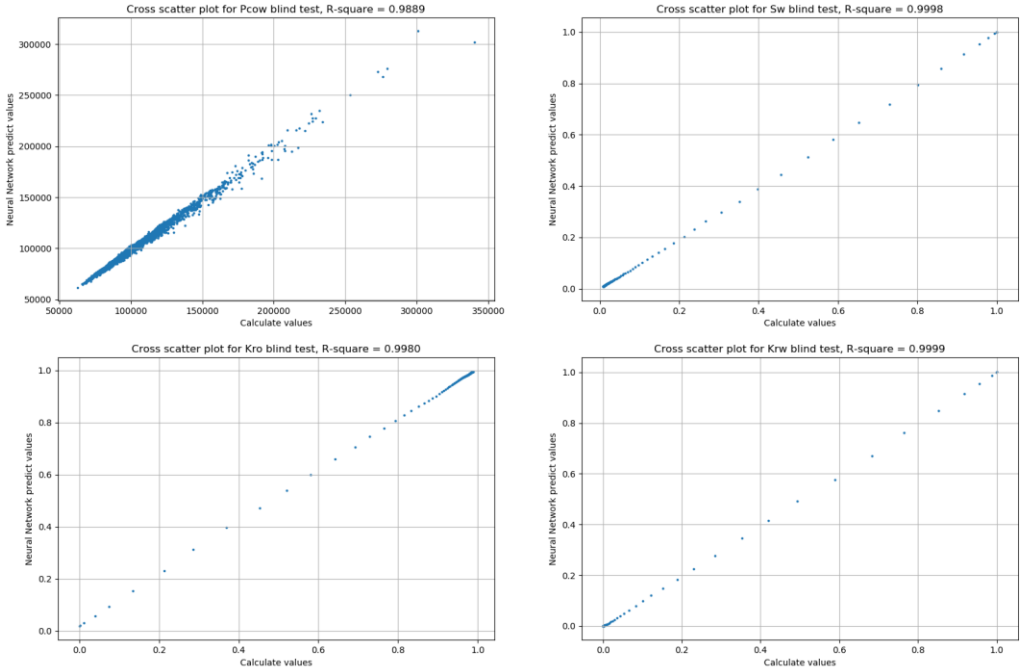

Figure 3.50 Pcow, Sw, Kro, Krw cross-plot for test data with elongation factor from 0.5 to 1 for training dataset 2 by approach 2 (Sw_area in output)

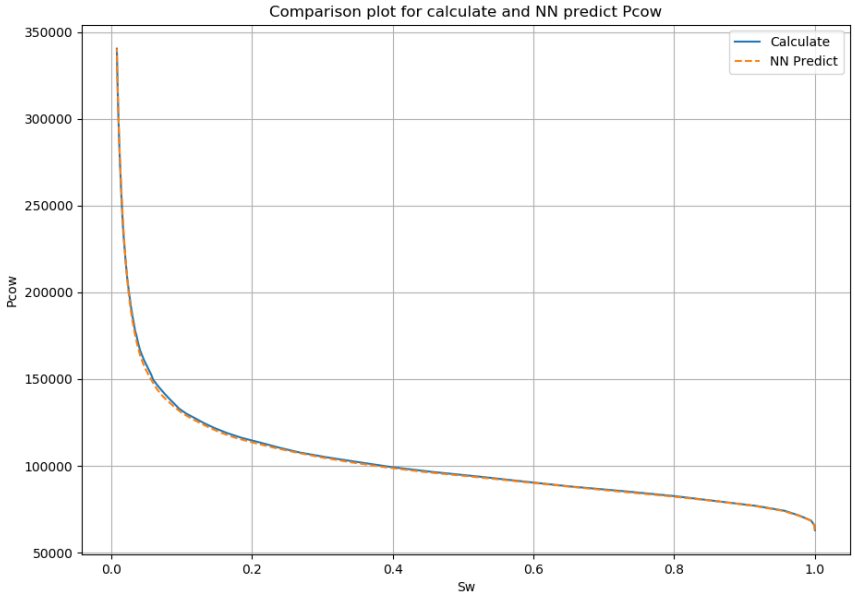

Figure 3.51 Pcow comparison plot for test data with elongation factor from 0.5 to 1 for training dataset 2 by approach 2 (Sw_area in output)

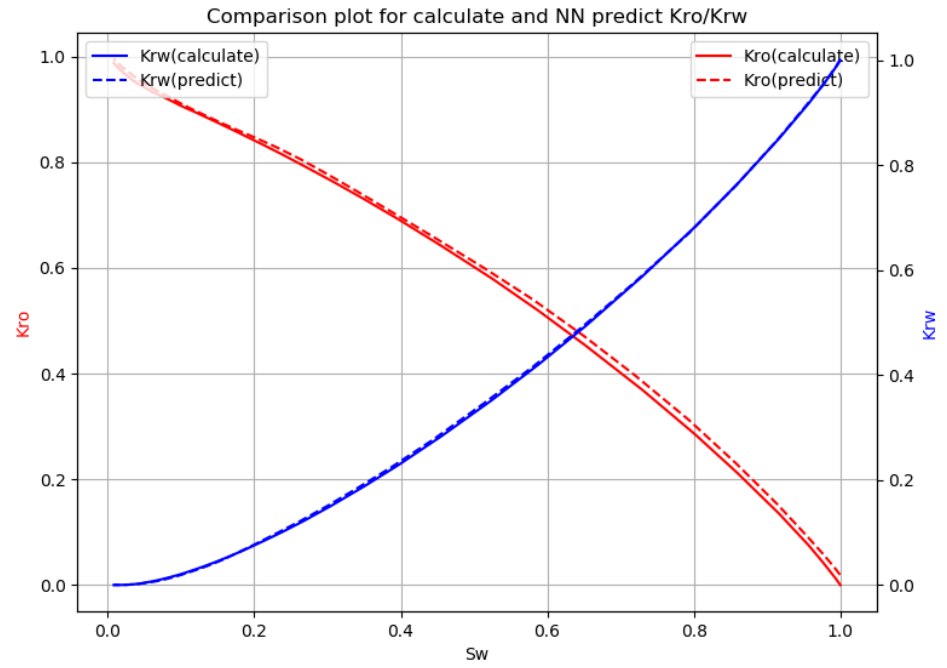

Figure 3.52 Kro/Krw comparison plot for test data with elongation factor from 0.5 to 1 for training dataset 2 by approach 2 (Sw_area in output)

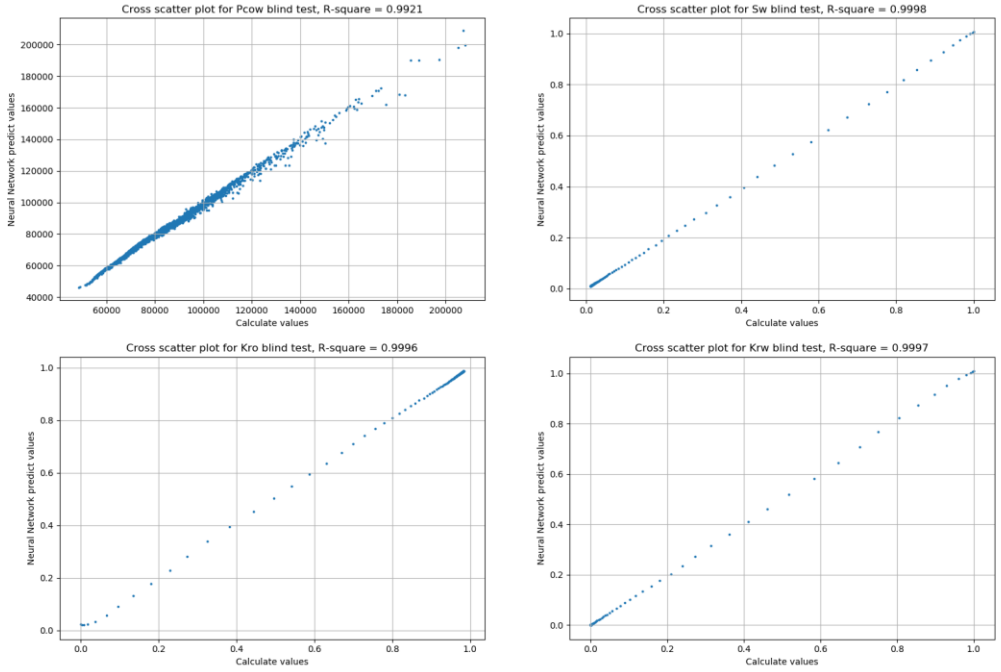

Figure 3.53 Pcow, Sw, Kro, Krw cross-plot for test data with all random polygons for training dataset 2 by approach 2 (Sw_area in output)

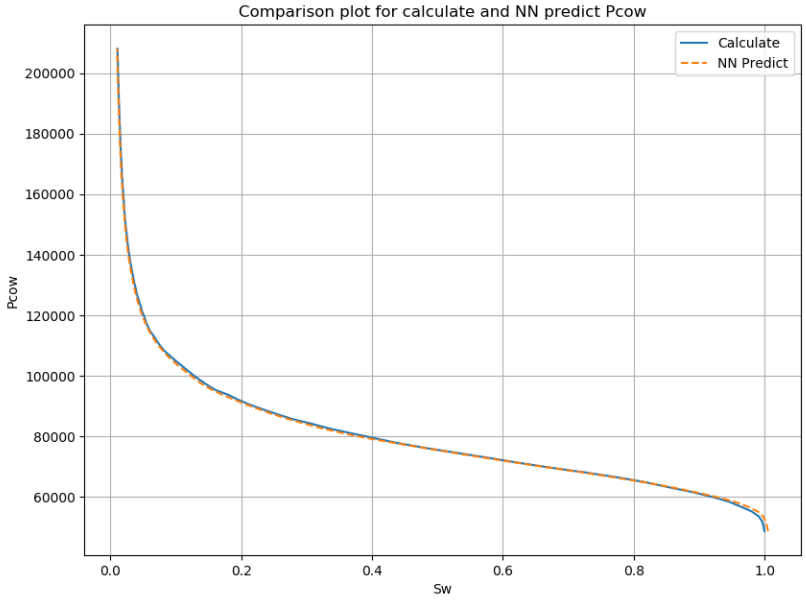

Figure 3.54 Pcow comparison plot for test data with all random polygons for training dataset 2 by approach 2 (Sw_area in output)

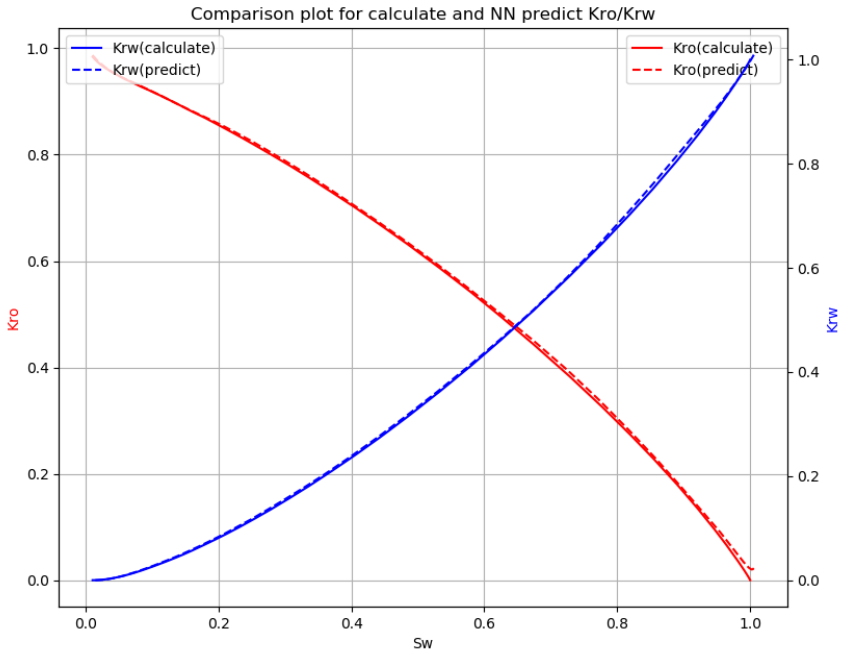

Figure 3.55 Kro/Krw comparison plot for test data with all random polygons for training dataset 2 by approach 2 (Sw_area in output)

The Neural Network model from training dataset 2 performed better the model from training dataset 1 as illustrated by the figures above. The P_{cow} predictions are desirable for all blind testing datasets. Results from blind test dataset 2, 3, 5 are showing nice match in terms of the S_w, K_{ro} and K_{rw} prediction, for test dataset 1 and 4 which represent the samples with shape factor from 0 to 0.04 and elongation factor from 0.5 to 1.0 respectively, the latter one is acceptable and former one is not perfectly match but at the principle trend is catches by the model.

(a) Output parameter include S_w_frac

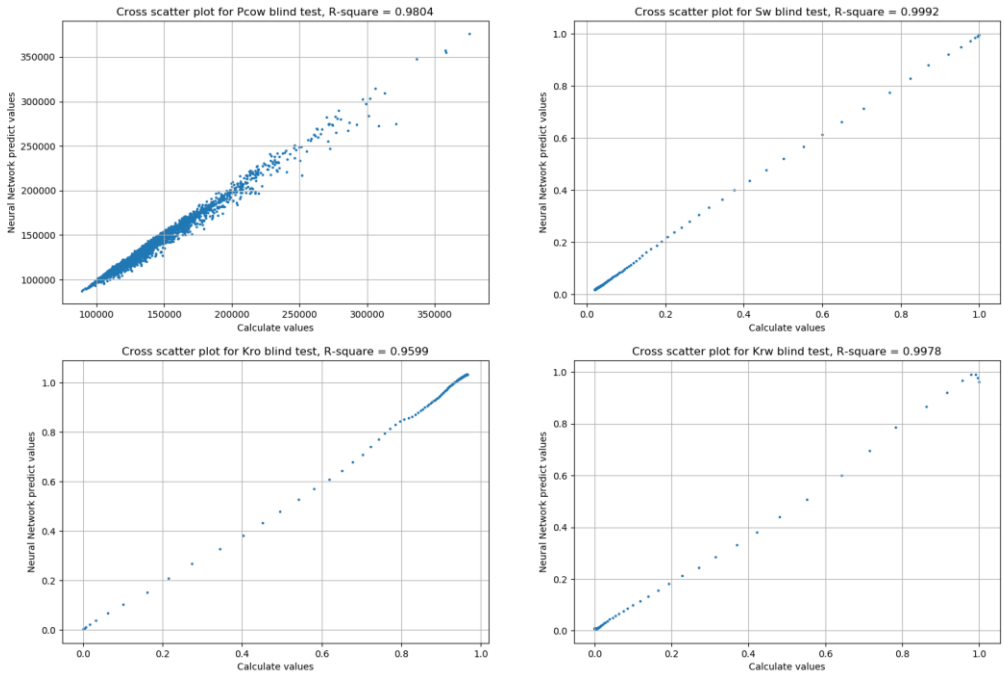

Figure 3.56 Pcow, Sw, Kro, Krw cross-plot for test data with shape factor from 0 to 0.04 for training dataset 2 by approach 2 (Sw_frac in output)

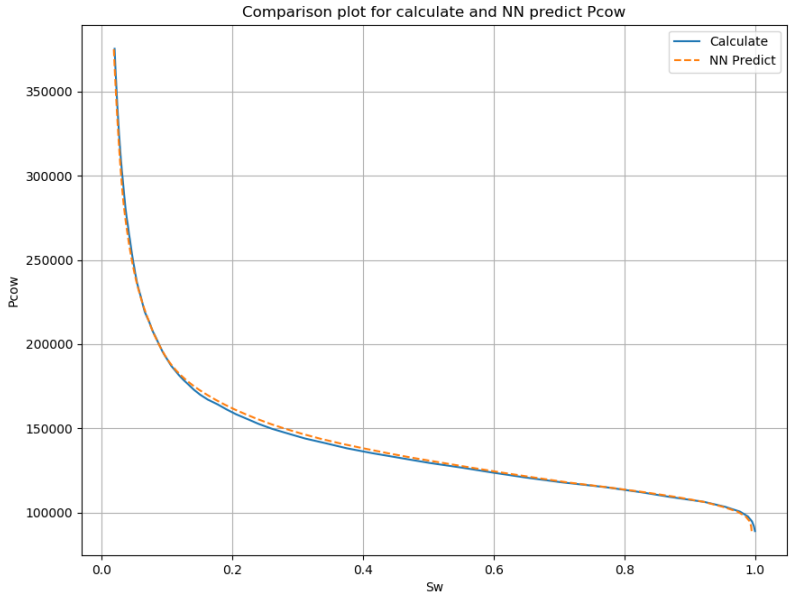

Figure 3.57 Pcow comparison plot for test data with shape factor from 0 to 0.04 for training dataset 2 by approach 2 (Sw_frac in output)

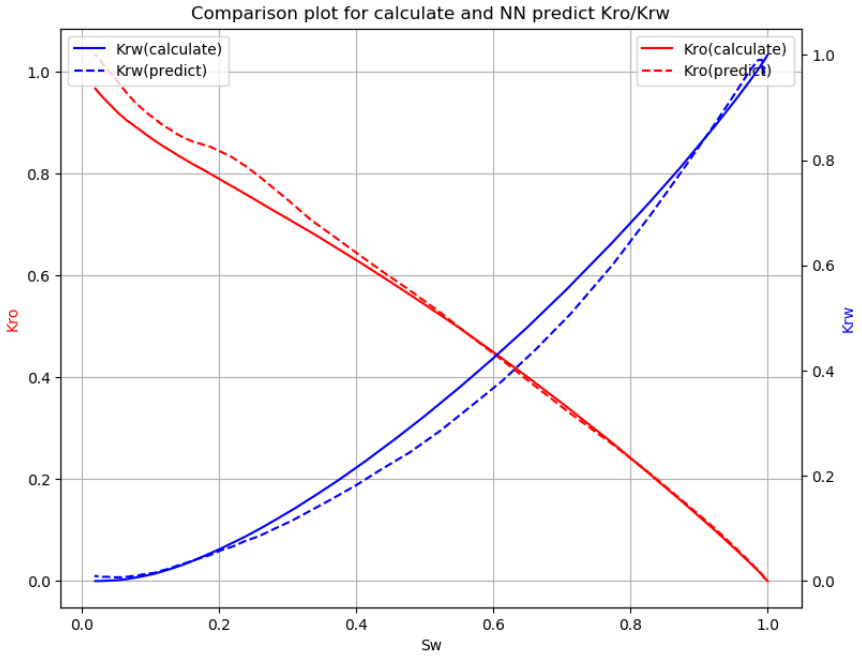

Figure 3.58 Kro/Krw comparison plot for test data with shape factor from 0 to 0.04 for training dataset 2 by approach 2 (Sw_frac in output)

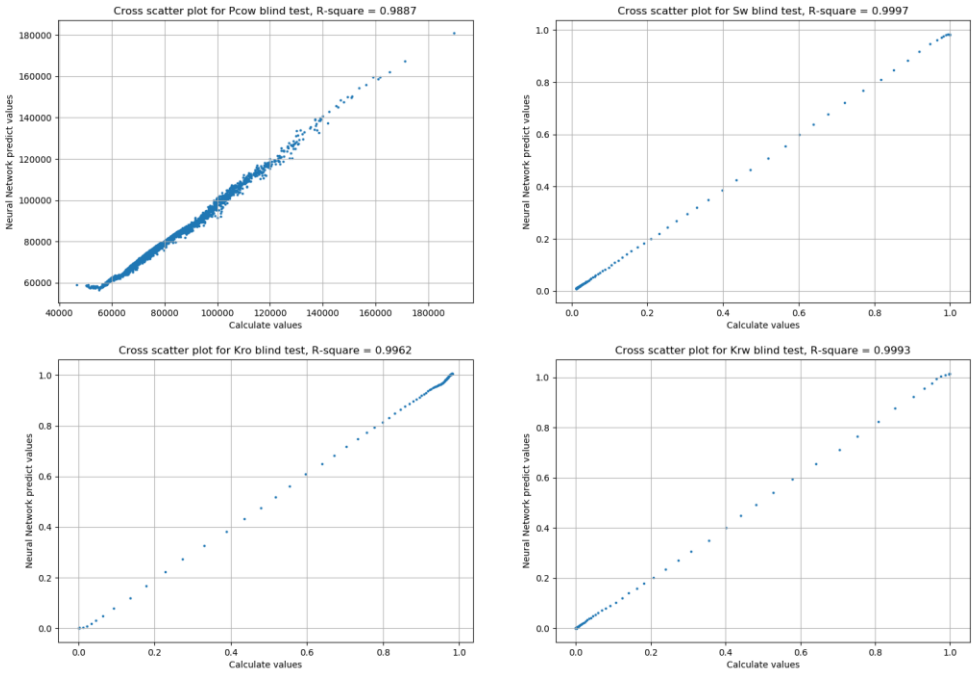

Figure 3.59 Pcow, Sw, Kro, Krw cross-plot for test data with shape factor from 0.04 to 0.07958 for training dataset 2 by approach 2 (Sw_frac in output)

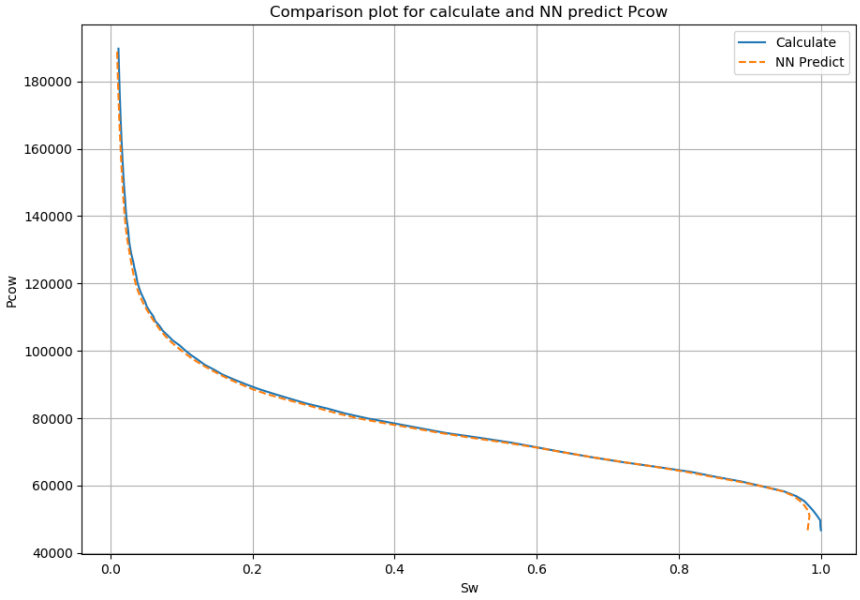

Figure 3.60 Pcow comparison plot for test data with shape factor from 0.04 to 0.07958 for training dataset 2 by approach 2 (Sw_frac in output)

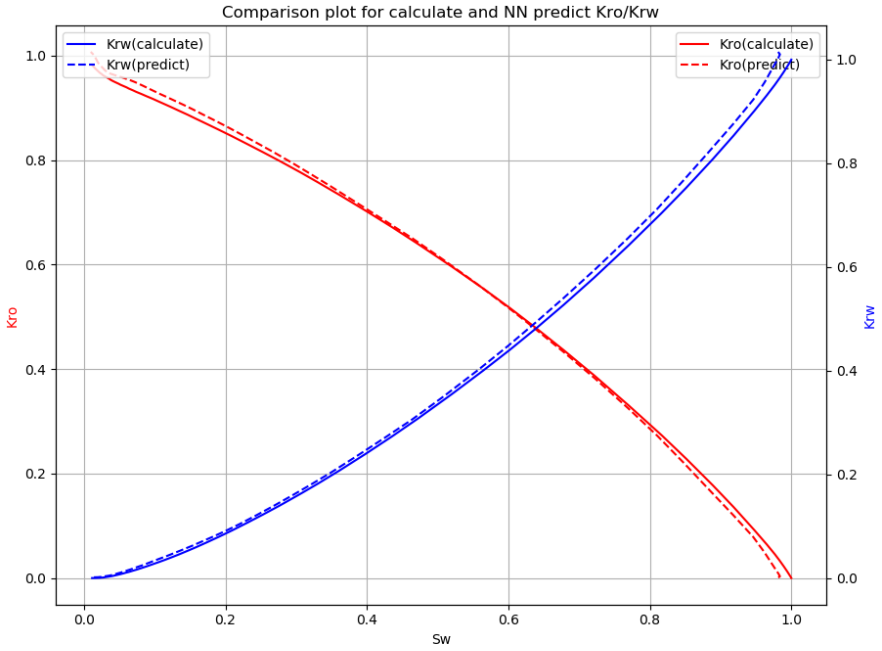

Figure 3.61 Kro/Krw comparison plot for test data with shape factor from 0.04 to 0.07958 for training dataset 2 by approach 2 (Sw_frac in output)

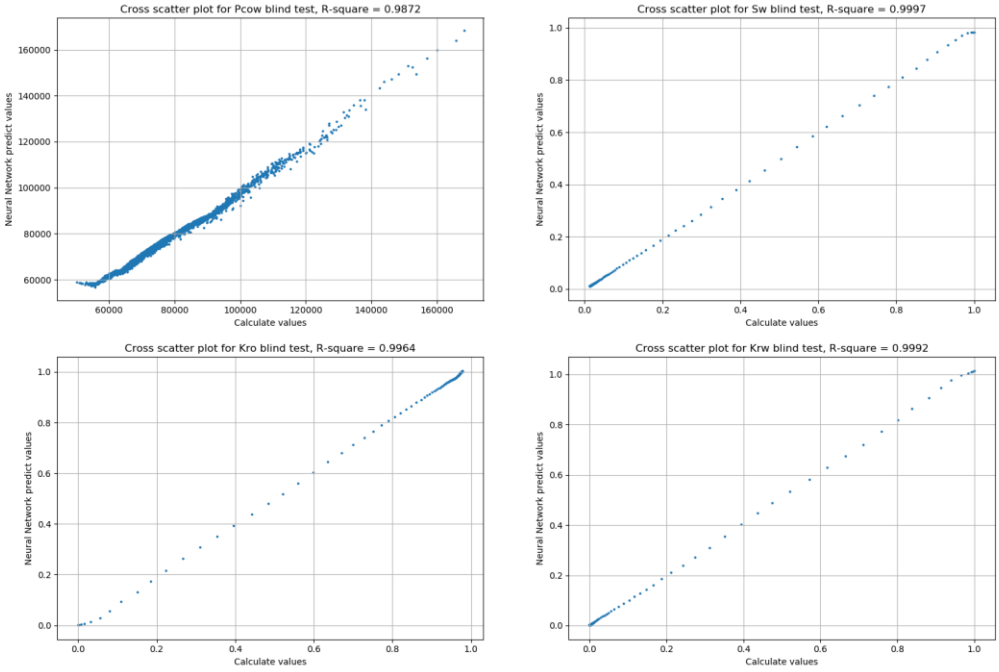

Figure 3.62 Pcow, Sw, Kro, Krw cross-plot for test data with elongation factor from 0 to 0.5 for training dataset 2 by approach 2 (Sw_frac in output)

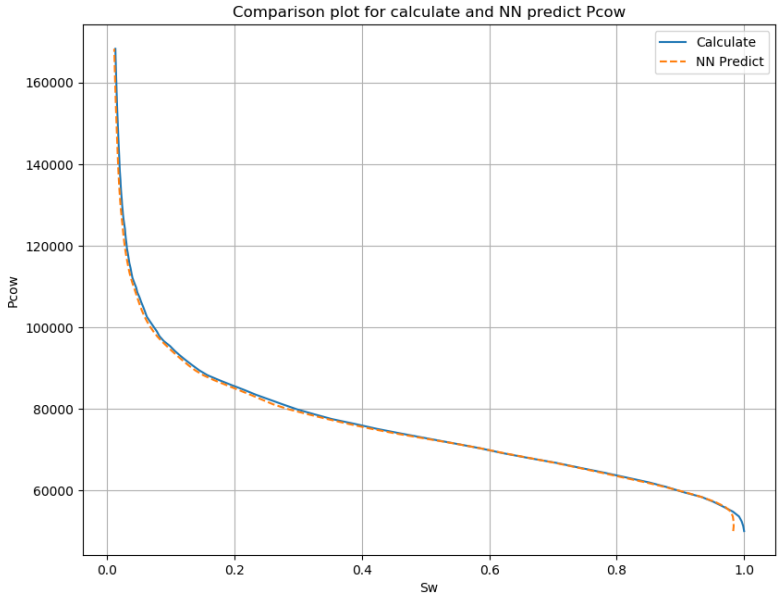

Figure 3.63 Pcow comparison plot for test data with elongation factor from 0 to 0.5 for training dataset 2 by approach 2 (Sw_frac in output)

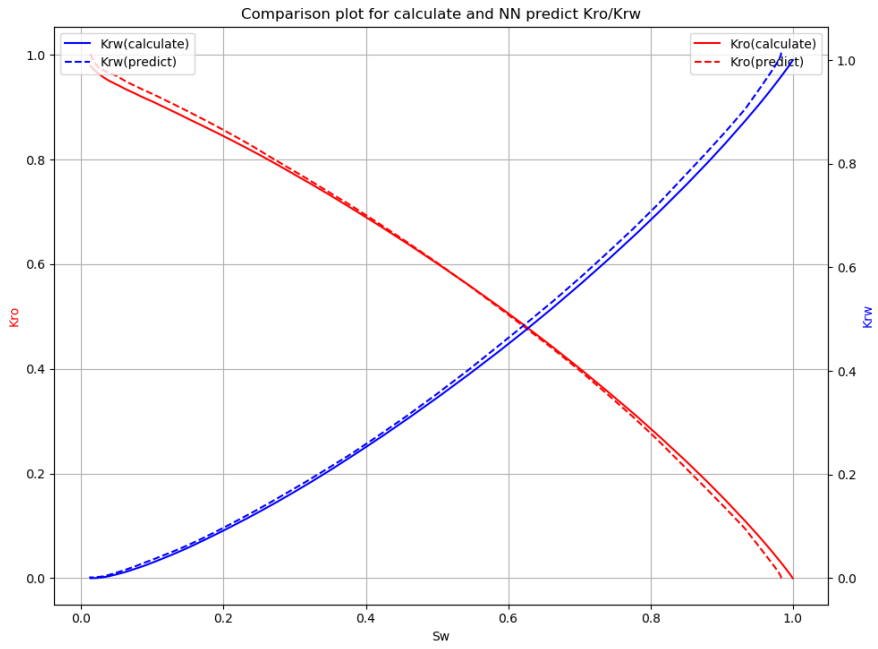

Figure 3.64 Kro/Krw comparison plot for test data with elongation factor from 0 to 0.5 for training dataset 2 by approach 2 (Sw_frac in output)

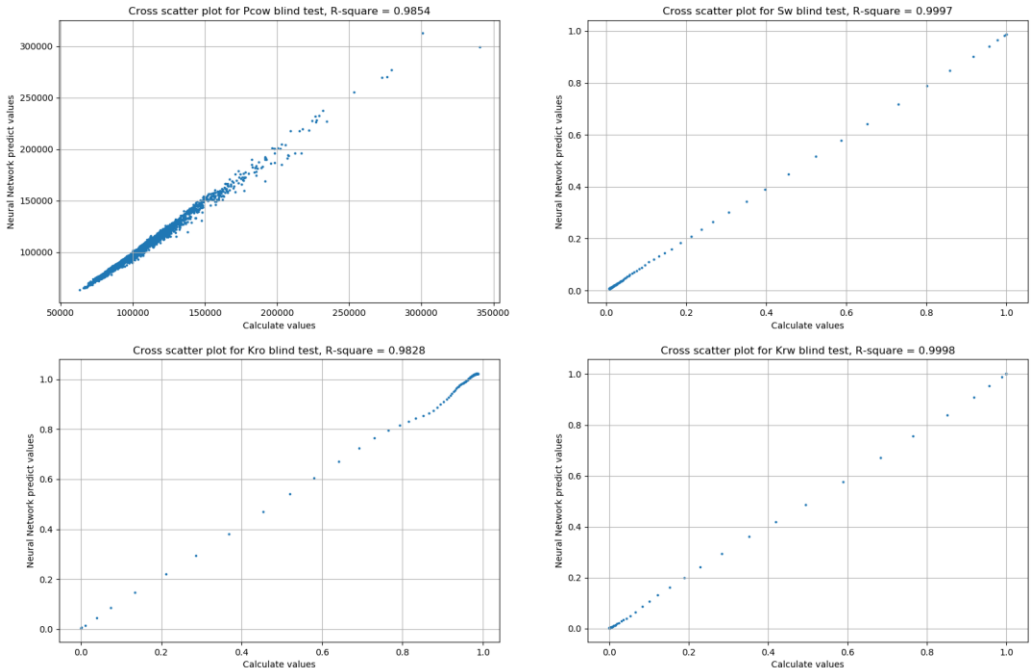

Figure 3.65 Pcow, Sw, Kro, Krw cross-plot for test data with elongation factor from 0.5 to 1 for training dataset 2 by approach 2 (Sw_frac in output)

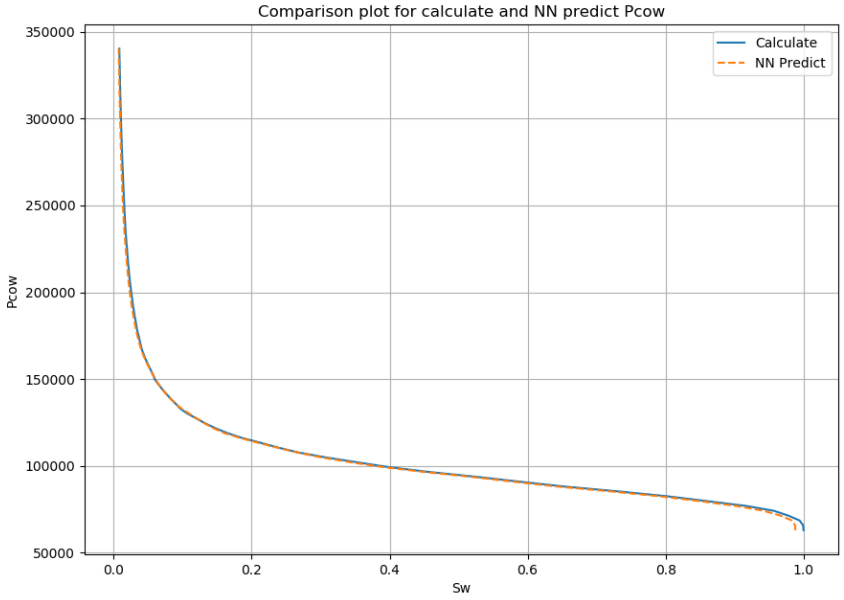

Figure 3.66 Pcow comparison plot for test data with elongation factor from 0.5 to 1 for training dataset 2 by approach 2 (Sw_frac in output)

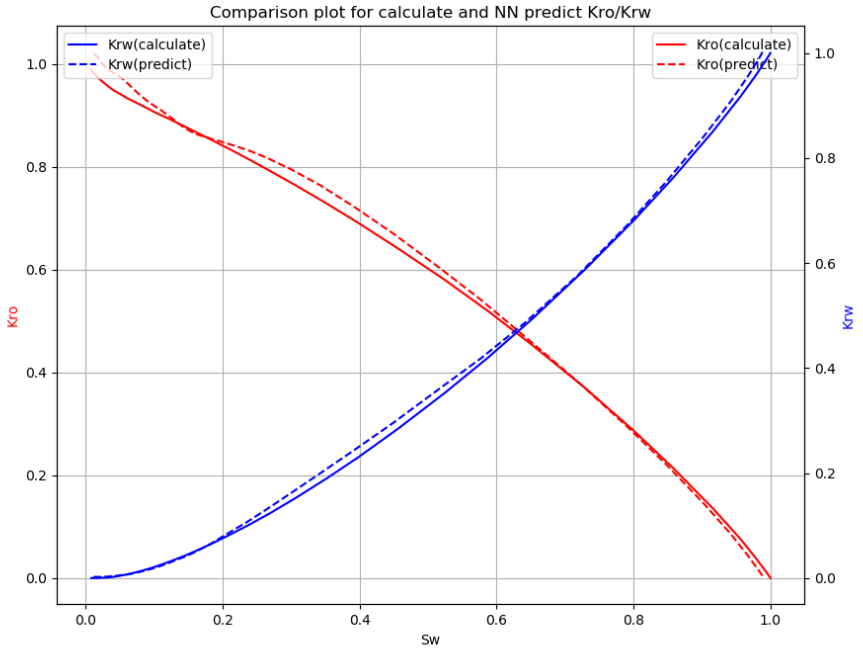

Figure 3.67 Kro/Krw comparison plot for test data with elongation factor from 0.5 to 1 for training dataset 2 by approach 2 (Sw_frac in output)

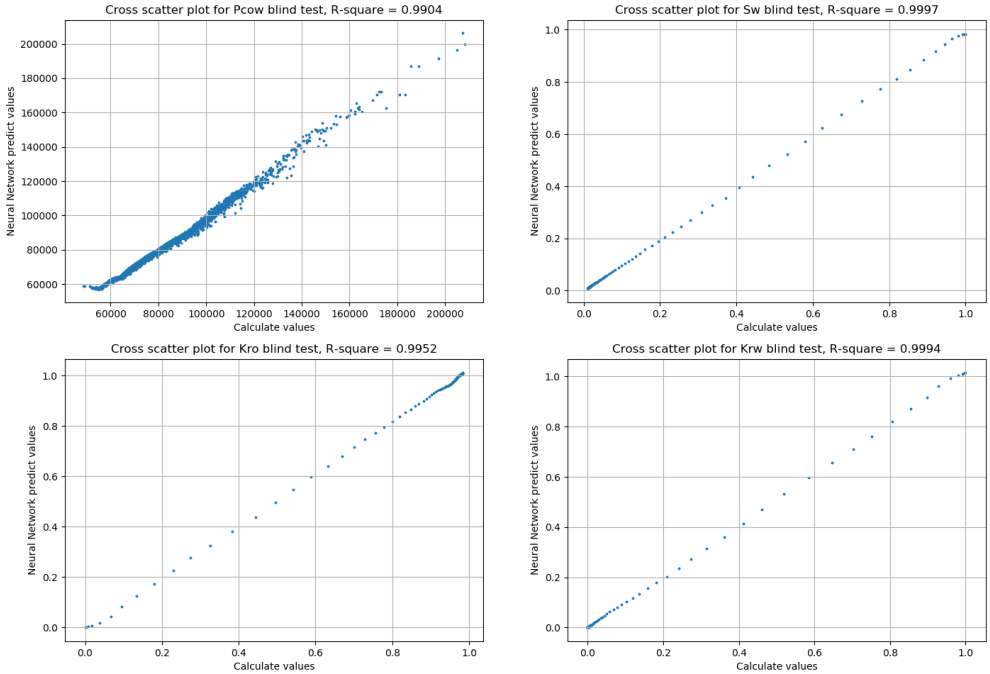

Figure 3.68 Pcow, Sw, Kro, Krw cross-plot for test data with all random polygons for training dataset 2 by approach 2 (Sw_frac in output)

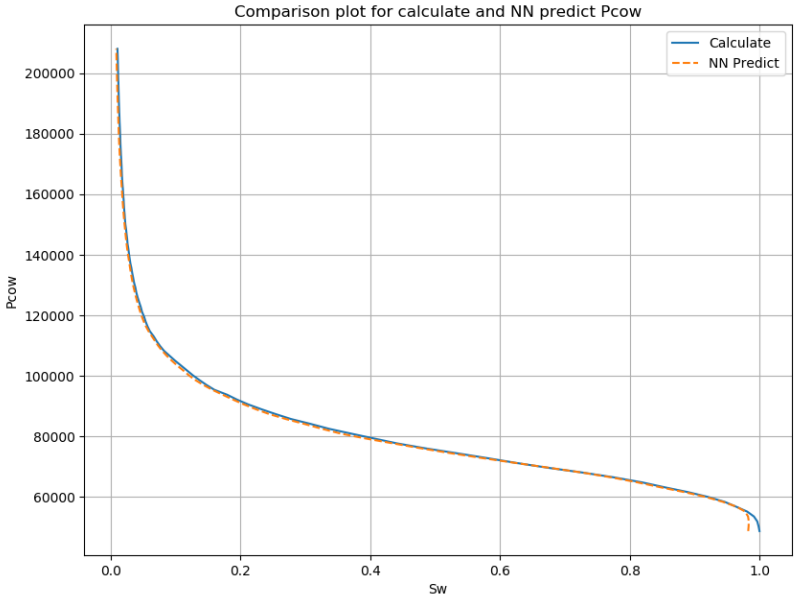

Figure 3.69 Pcow comparison plot for test data with all random polygons for training dataset 2 by approach 2 (Sw_frac in output)

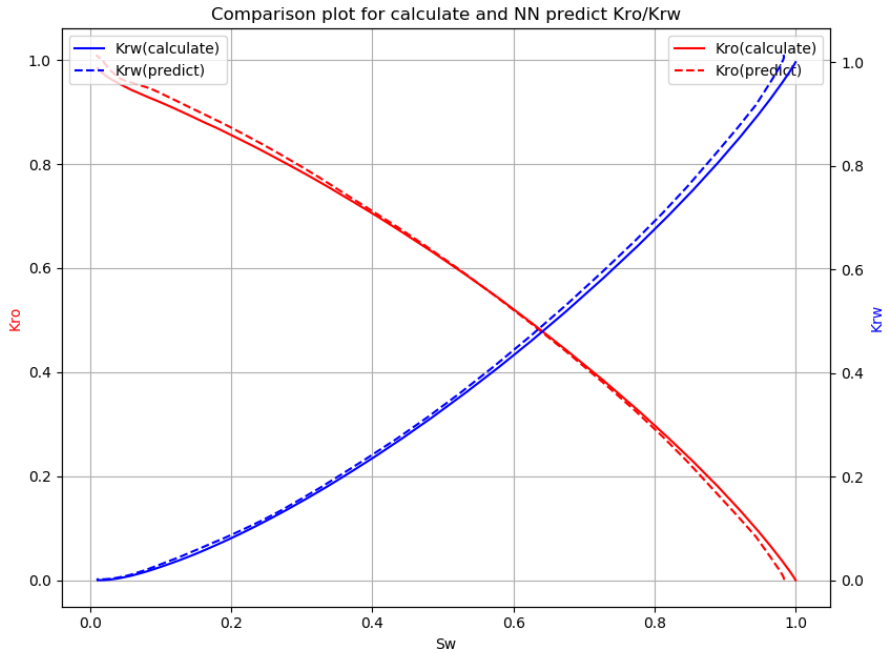

Figure 3.70 Kro/Krw comparison plot for test data with all random polygons for training dataset 2 by approach 2 (Sw_frac in output)

Compared to the model from training dataset 2 by approach 2 and using S_w_area as one of the output to calculate final water saturation and relative permeabilities, the results from model using S_w_frac shows no advantages in terms of S_w, K_{ro} and K_{rw} predictions as all blind test results show different degrees of deviation. So, the utilizing S_w_frac as one of the outputs in approach 2 is deprecated at this point.

Also, the Neural Network trained by training dataset 2 is more capable of predicting not only P_{cow} in approach 1, but also P_{cow}, S_w, K_{ro} and K_{rw} in approach 2. The following Neural Network structure analysis and input parameter sensitivity analysis will base on the model from training dataset 2 in both approaches.

3.2 Results of Neural Network structure analysis and adjustment

3.2.1 For approach 1 with training dataset 2

Five Neural Network structures were tested by using same other parameters and training dataset. In specific, the structures we tested include: simple single layer 5 neurons, wider single layer 10 neurons, deeper double layers with 5 neurons for each one of the layer, wider and deeper double layers with 10 neurons for each one of the layer, and the larger 3 layers network with [200, 100, 50] neurons.

From the overall accuracy versus training epochs for different network structure plot, as we can see the final accuracy is slightly increased when the network is becoming wider and deeper. Larger network will produce relatively high accuracy during initial training iterations, but when some point is reached the results can be a little bit oscillated. The network with smaller size is producing relative stable training accuracy the also final

accuracy is acceptable such as the network with double layers and 10 neurons for each one of the layer.

In approach 1 with training dataset 2, the following 5 MLP network structures were tested:

- Simple one hidden layer with 5 neurons [5]
- Wider: One hidden layer with 10 neurons [10]
- Deeper: Two hidden layers with 5 neurons in each layer [5, 5]
- Wider and deeper: Two hidden layers with 10 neurons in each layer [10, 10]
- Larger network (Previous training network) [200, 100, 50]

The training and validation results show in Figure 3.71, as we can see from the plot, the performance of network with only one hidden layer is good even with only 5 neurons, and reached 0.9809 R-Square at 10 epochs, wider network with 10 neurons is converging slightly faster and the final accuracy is slightly better than the one with 5 neurons. Comparison of the network [5] and [5, 5] shows deeper network is performing better than single layer network and [5, 5] is more stable than the single layer with 10 neurons. The largest network [200, 100, 50] has the fastest converging speed and is reaching best validation accuracy in only 2-3 epochs and starting oscillating. The network [10, 10] is showing almost identical performance with largest network and more stable and smooth converging steps. So, for approach 1 the MLP network with two hidden layers and 10 neurons each are good enough for P_cow estimation.

Network Structure	Input parameters	Best train R^2	Best validation R^2
[5]	All	0.9810	0.9809
[10]	All	0.9856	0.9862
[5, 5]	All	0.9861	0.9863
[10, 10]	All	0.9897	0.9899
[200, 100, 50]	All	0.9885	0.9921

Table 3.1 A summary of the best training and validation R-Square score for different network structures in approach 1 with training dataset 2

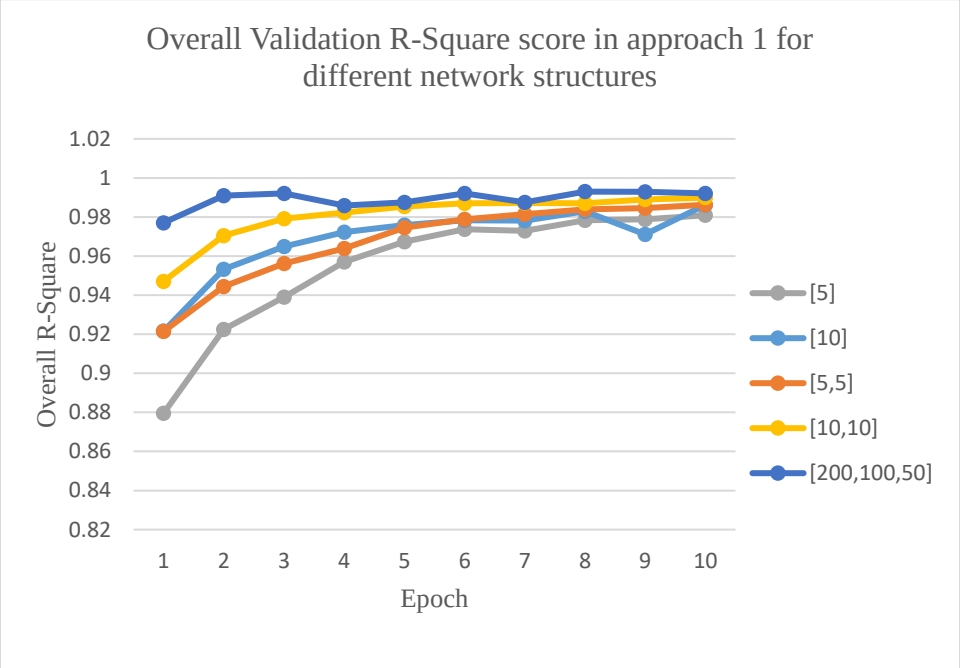

Figure 3.71 Comparison of validation accuracy for different network structures in approach 1

3.2.2 For approach 2 with training dataset 2

Eight MLP network structures are created from simple one hidden layer with 5 neurons to a large 3 layers network:

- Simple one hidden layer with 5 neurons [5]
- Simple one hidden layer with 10 neurons [10]
- Two hidden layers with 10 neurons each [10, 10]
- One hidden layer with 20 neurons [20]

- One hidden layer with 50 neurons [50]
- One hidden layer with 100 neurons [100]
- Two hidden layers with 100 neurons each [100, 100]
- Three hidden layers with 200, 100, 50 neurons [200, 100, 50]

From the comparison plot, apparently, the simple network with one hidden layer contains 5 neurons produced worst results (best R-Square is 0.8779) compared to other networks, due to the increased complexity of the 2nd approach, the simple network is not very good at capturing the non-linearity in the system. However, the wider network with 10 neurons in one within one hidden layer shows significant improvement with around 0.9611 final R-Square value. It is not hard to find out that when the network is becoming wider and deeper, the prediction accuracy is increasing correspondingly, and most of them are showing relatively stable and smooth training process. The best results are from the largest network structure [200, 100, 50] which produced the best 0.9914 final R-Square value. The following network model prediction results are depicting the impact from network structure to prediction quality, see Figure 3.72.

Network Structure	Input parameters	Best train $\boldsymbol{R^2}$	Best validation $\boldsymbol{R^2}$
[5]	All	0.8784	0.8779
[10]	All	0.9605	0.9611
[10, 10]	All	0.9752	0.9767
[20]	All	0.9655	0.9661
[50]	All	0.9815	0.9813
[100]	All	0.9837	0.9832
[100, 100]	All	0.9905	0.9900
[200, 100, 50]	All	0.9904	0.9914

Table 3.2 A summary of the best training and validation R-Square scores for different network structures in approach 2 with training dataset 2

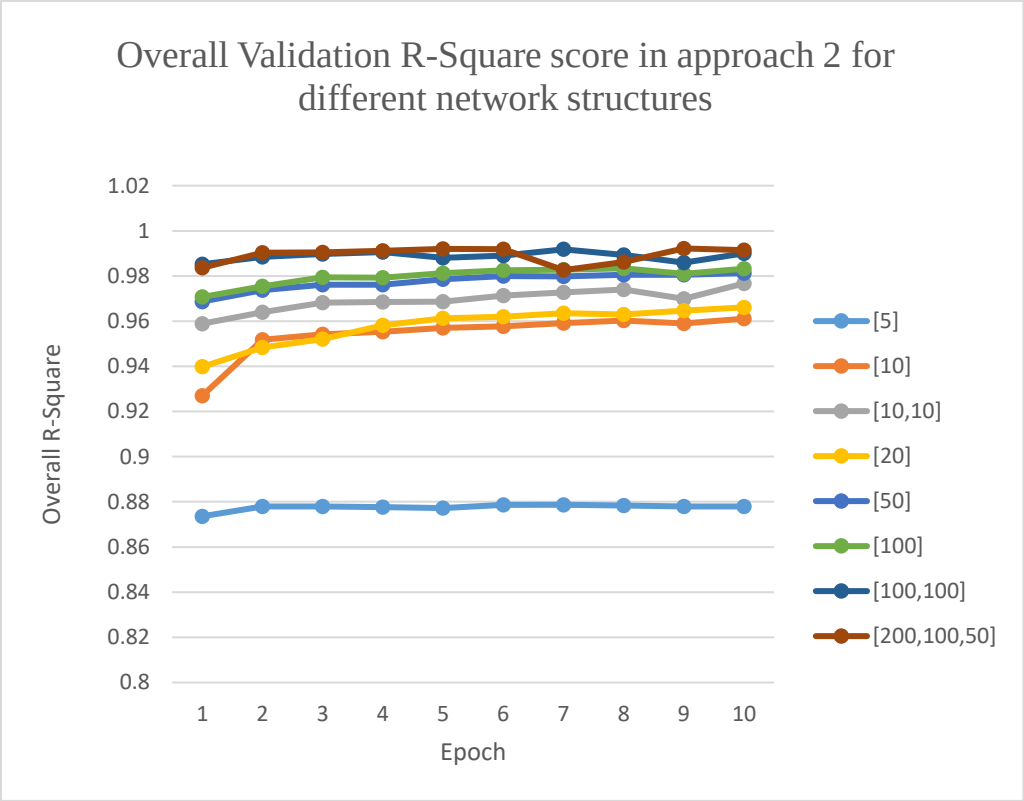

Figure 3.72 Comparison of validation accuracy for different network structures in approach 2

3.3 Results of sensitivity analysis for input features

3.3.1 For approach 1 with training dataset 2

One hidden layer with 5 neurons MLP network is used for this test, and various input parameter combinations are tested in the Neural Network training and prediction process; Table 3.3 shows the results of the training and validation accuracy scores for the combinations we tested in approach 1, and sorted by best validation R^2 scores.

Network Structure	Input parameters	Best train R^2	Best validation R^2
[5]	Remove A	0.9829	0.9821
[5]	Remove R	0.9817	0.981
[5]	g_el,A	0.9812	0.9806
[5]	Remove L	0.9804	0.9804
[5]	A,E,R	0.9809	0.9804
[5]	A,L,R	0.9799	0.979
[5]	All	0.9795	0.9782
[5]	Remove g_el	0.9786	0.9777
[5]	A,E,L,R	0.9773	0.9773
[5]	E,L,R	0.9771	0.9762
[5]	g_cc,g_el,L	0.9759	0.9753
[5]	Remove E	0.9766	0.973
[5]	g_cc,R	0.9722	0.9725
[5]	g_cc,g_el,R	0.9707	0.9716
[5]	g_cc,A	0.9714	0.9712
[5]	Remove g_cc	0.9682	0.9693
[5]	g_cc,L	0.9692	0.9669
[5]	g_el,L	0.9638	0.963
[5]	A,E,L	0.9572	0.958
[5]	g_el,R	0.9557	0.9558
[5]	g_cc,g_el,A	0.9497	0.9487
[5]	g_cc,E	0.8665	0.8664
[5]	g_cc,g_el	0.8667	0.8654
[5]	g_cc,g_el,E	0.8634	0.8628
[5]	g_cc	0.8629	0.8613
[5]	g_el	0.8576	0.8584
[5]	g_el,E	0.8416	0.8386

Table 3.3 A summary of the best training and validation R-Square scores for different input parameters combination in approach 1 with training dataset 2 (A is area, R is inscribed radius, E

is number of edges, L is perimeter, g_cc is shape factor, g_el is elongation factor, and the results are sorted by Best validation R^2).

Results in Table 3.3 illustrated the impact to the training and validation accuracy from various input parameters in approach 1. Select all inputs is not producing the best results which mean there is might have redundancy within the dataset. The model with only shape factor (g_cc) and elongation factor (g_el) shows relative poor performance (0.8654 R-Square scores for validation). After additional parameters are added to the model, the performance increased as expected: adding perimeter shows the best result (0.9753), and inscribed radius (0.9716), Area (0.9487), but when adding 'number of edges' the result was even worse. When we only consider the inputs without shape factor and elongation factor such as 'A, E, R' combination, 'A, L, R' combination or 'A, E, L, R' combination, the results are relative good: 0.9804, 0.9790, 0.9773 correspondingly. Another observation is that the model with elongation factor (g_el) and area (A) is giving us 0.9806 final validation R-Square score by only two inputs which are more desired model regarding computation efficiency and performance.

Approach 1, inputs without shape factor and elongation factor are good for most cases because of the correlations between the factors and geometrical properties. Model with elongation factor (g_el), Area (A) input combination produce very good results for only two input parameters. The number of edges is not giving positive contribution, even negative impact on results in some cases.

3.3.2 For approach 2 with training dataset 2

One hidden layer with 10 neurons MLP network is used for input sensitivity analysis in approach 2, the Table 3.4 shows the results of the training and validation accuracy scores which are sorted by best validation R^2 scores.

Network Structure	Input parameters	Best train $\boldsymbol{R^2}$	Best validation $\boldsymbol{R^2}$
[10]	All	0.9803	0.9805
[10]	Remove A	0.9799	0.98
[10]	g_cc,g_el,A, R, p_test	0.9691	0.9696
[10]	Remove E	0.9699	0.9696
[10]	Remove R	0.9700	0.9691
[10]	Remove L	0.9638	0.9639
[10]	g_cc,g_el,A, E, p_test	0.9589	0.9592
[10]	g_cc,g_el,A, p_test	0.954	0.9544
[10]	g_el,A,p_test	0.9417	0.9417
[10]	g_cc,g_el,L, p_test	0.9361	0.9372
[10]	g_el,A,,R, p_test	0.919	0.9195
[10]	g_cc,A,p_test	0.9145	0.9172
[10]	g_cc,g_el,E,R,p_test	0.8668	0.8667
[10]	g_cc,g_el,R,p_test	0.8661	0.8664
[10]	g_el,R,p_test	0.8428	0.8432
[10]	g_cc, g_el,E, L, p_test	0.7779	0.778
[10]	g_cc, R, p_test	0.7399	0.7399
[10]	g_cc, g_el, p_test	0.7262	0.7257
[10]	g_cc, g_el,E, p_test	0.7196	0.7195

Table 3.4 A summary of the best training and validation R-Square scores for different input parameters combination in approach 2 with training dataset 2 (A is area, R is inscribed radius, E is number of edges, L is perimeter, g_cc is shape factor, g_el is elongation factor, p_test is the testing pressure for Kro, Krw calculation and the results are sorted by best validation R^2)

Results in Table 3.4 shows the impact on the training and validation accuracy from various input parameters in approach 2. We found the best result was given by the model with all

input parameters (0.9805 final validation R-Square score). The models with the removal of one or two parameters from all inputs have the least impact on the overall validation accuracy scores. A similar trend with the test in approach 1 can be found that the model inputs with only shape factor and elongation factor do not perform well unless at least one parameter is added to describe somewhat the size-related information, such as when inscribed radius (R) is added, the score increased to 0.8664 from 0.7257. Moreover, almost nothing improved when the number of edges (E) is added to this model, this is similar to the results in approach 1, the number of edges is not essential in our training process. Another notable model is the 'g_el, A, p_test' combination which contains only three input parameters but performed well with validation R-Square of 0.9417; the similar trend can be found in the test for approach 1.

3.4 Comparison of process speed between Neural Network and direct calculation approach

The computation time is growing fast when the pore network is becoming more complicated, but it is still acceptable for properties calculation in the small-scale model. Regarding Neural Network approaches, more time will be used in preprocessing, training/validation, and so forth. When the model is big enough, the Neural Network overhead can be neglected. Table 3.5 shows the direct calculation and Neural Network speed test results for different model scale. Figure 3.73 shows the relative speed-up by using Neural Network predictions compared to direct calculations.

Number of tubes	Tubes*Test Pressures	Calculation time used for $\boldsymbol{P_{cow}, K_{ro}, K_{rw}}$	NN Predict time used for $\boldsymbol{P_{cow}, K_{ro}, K_{rw}}$
100	20,000	1	1
500	100,000	8	3
1,000	200,000	17	5
2,000	400,000	41	9
3,000	600,000	73	13
5,000	1,000,000	99	21
7,000	1,400,000	162	30
10,000	2,000,000	239	42
20,000	4,000,000	562	86
50,000	10,000,000	1831	212
100,000	20,000,000	4699	424

Table 3.5 A summary of the comparison for conventional calculation speed and Neural Network prediction speed based on 200 testing pressures and various number of capillary tubes

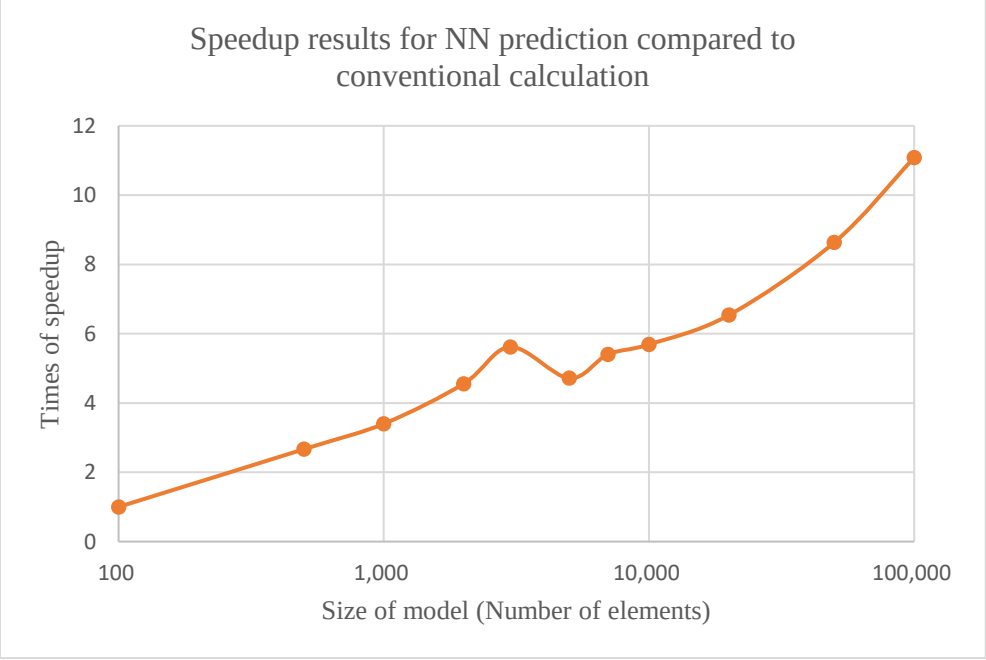

Figure 3.73 Speedup results for NN predictions compared to conventional calculations

From the above results, when the model is large, the conventional direct calculation is very time-consuming even for such a simple model with a bundle of the parallel capillary, not to mention the complex 3D pore-throats model with more connectivity.The computation time will grow exponentially with the increasing complexities. However, from our results, the Neural Network prediction time will increase just linearly with the increasing size of the model, and predict much faster even without help from multi-cores CPU parallel computing and GPU.

4. Discussion and summary

4.1 Summary

- Training dataset No.2 is more representative in most cases (either in approach 1 or approach 2) due to the better coverage in terms of the polygon geometries. As a result, the training data preparation is critical for generating a generalize NN model.
- From the results comparison plots, the model with S_w_area as one of output is perform better than the model with S_w_frac as one of the output in approach 2. This observation shows the Neural Network with proper data preprocessing can deal with very small input and output values.
- In approach 1, the problem is relatively simple compared to approach 2 because of the inputs corresponding to one output (P_{cow}) explicitly, and they are independent from each other. As the result, one hidden layer MLP network is good enough for predicting capillary pressures with properly select input parameters, e.g. elongation factor, and area of the cross-section combination. But the predicting results oscillation can be observed compared to calculated pressures even the overall accuracy is acceptable.
- In approach 2, the problem is relatively complicated due to the relative permeability curves are generated by predicting intermediate parameters such as S_w_area (or S_w_frac), q_o, q_w then convert them to final results, the Neural Network is trying

to ‘learn’ the behavior of multi-phase flow within capillary tube with specific geomeatry and at specific invading pressure, which contains more nonlinearity than the problem in approach 1. The overall results are acceptable by apply several strategies for training dataset creation, pre-processing and post-processing, and utilizing multiple hidden layers MLP networks even single hidden layer network with all input parameters. The models in approach 2 can predict capillary pressures and intermediate parameters for relative permeabilities at same time, also the results for P_{cow} is more stable and smooth than the results from approach 1.

- Data pre-processing is critical for Neural Network modeling and training. Appropriate pre-processing is helping Neural Network to avoid some converging issues, for example, the strategy for creating training dataset in approach 2 improved the robustness of the network prediction, also the pre-scaling of the training and prediction dataset will help stabling training process when dealing with very small values.
- The sophisticated problems may need multiple hidden layers network or even some new network structures. However, regarding the complexity in this study, one or two hidden layers MLP networks with specific data pre-processing, post-processing and hyperparameter tuning are satisfactory, more layers and more neurons may increase the expressive power of Neural Network, but may introduce unnecessary over-fitting issues and consume more computation resources.
- Overall R-Square scores or some other accuracy indicator such as Mean Squared Error (MSE) are essential indicators for Neural Network prediction quality control, but the final predictions versus actual data comparison and comparison plot would be a better standard for the final examination.

4.2 Future works

- Apply the workflow to imbibition process in pore-scale modeling and simulation.
- Extend work to 3 phase flows
- Extend work to concave polygons
- Neural Network prediction for more complex ‘pore-throat-pore’ scenarios in 3D.
- The wettability impact and wetting layer formation and collapse during displacements will be added to our study.
- Data preprocessing analysis (clustering, dimensionality reduction, and so forth.) to reduce the redundant information within training dataset, will lead to more efficient training process without some unnecessary noise.
- Improve the random polygon generator:
    - 1. Improve efficiency by introducing parallel computing scheme.
    - 2. Improve the flexibility by adding more customized constraints that can mimic any geometrical properties distribution from real rock samples.
    - 3. May extend to 3D to generate more complex geometries in terms of pore network modeling.
    - 4. Add the function to create random concave polygons and calculate related characteristics.
- Build a flexible Neural Network research framework customized for pore network modeling and simulation of multi-phase fluid flow through porous media.

5. References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., … Zheng, X. (2016). TensorFlow : A System for Large-Scale Machine Learning This paper is included in the Proceedings of the TensorFlow : A system for large-scale machine learning. *Proc 12th USENIX Conference on Operating Systems Design and Implementation*, 272–283.

Aghaei, A., & Piri, M. (2015). Direct pore-to-core up-scaling of displacement processes: Dynamic pore network modeling and experimentation. *Journal of Hydrology*, *522*, 488–509. https://doi.org/10.1016/j.jhydrol.2015.01.004

Ahmadi, M. A., Zendehboudi, S., Dusseault, M. B., & Chatzis, I. (2016). Evolving simple-to-use method to determine water–oil relative permeability in petroleum reservoirs. *Petroleum*, *2*(1), 67–78. https://doi.org/10.1016/j.petlm.2015.07.008

Al-Alawi, S., Kalam, M. Z., & Al-Mukheini, M. (1996). Application of artificial neural networks to predict wettability and relative permeability of sandstone rocks. *Engineering Journal of the University of Qatar*, *9*, 29–43.

Al-Bulushi, N. I., King, P. R., Blunt, M. J., & Kraaijveld, M. (2012). Artificial neural networks workflow and its application in the petroleum industry. *Neural Computing and Applications*, *21*(3), 409–421. https://doi.org/10.1007/s00521-010-0501-6

Alizadeh, A. H., & Piri, M. (2014). Three-phase flow in porous media: A review of experimental studies on relative permeability. *Reviews of Geophysics*. https://doi.org/10.1002/2013RG000433

Arshadi, M., Zolfaghari, A., Piri, M., Al-Muntasheri, G. A., & Sayed, M. (2017). The effect of deformation on two-phase flow through proppant-packed fractured shale samples: A micro-scale experimental investigation. *Advances in Water Resources*, *105*, 108–131. https://doi.org/10.1016/j.advwatres.2017.04.022

Bengio, Y. (2009). *Learning Deep Architectures for AI*. *Foundations and Trends® in Machine Learning* (Vol. 2). https://doi.org/10.1561/2200000006

Blunt, M. J. (1998). Physically-based network modeling of multiphase flow in intermediate-wet porous media. *Journal of Petroleum Science and Engineering*, *20*(3–4), 117–125. https://doi.org/10.1016/S0920-4105(98)00010-2

Blunt, M. J. (2017). *Multiphase Flow in Permeable Media A Pore-Scale Perspective*. Cambridge University Press.

Bruyelle, J., & Guérillot, D. (2014). Neural networks and their derivatives for history matching and reservoir optimization problems. *Computational Geosciences*, *18*(3–4), 549–561. https://doi.org/10.1007/s10596-013-9390-y

Chaki, S., Verma, A. K., Routray, A., Mohanty, W. K., & Jenamani, M. (2014). Well tops guided prediction of reservoir properties using modular neural network concept: A case study from western onshore, India. *Journal of Petroleum Science and Engineering*, *123*, 155–163. https://doi.org/10.1016/j.petrol.2014.06.019

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. https://doi.org/10.3115/v1/D14-1179

Chollet, F., & others. (2015). Keras. GitHub.

Dong, H. (2007). Micro-Ct Imaging and Pore Network Extraction a Dissertation Submitted To the Department of Earth Science and Engineering of Imperial College Lodon in Partial Fulfilment of the Requirements for the Degree of, 1–213.

Drla, D. E., Pope, G. A., Sepehrnoorl, K., & Texas, U. (1993). Three - Phase Gas / Oil / Brine Relative Permeabilities Measured Under C02 Flooding Conditions, (May), 143–150.

Edris Joonaki, S. G. (2013). Prediction of Relative Permeability for Multiphase Flow in Fractured Oil Reservoirs by using a Soft Computing Approach. *International Journal of Computer Applications*, *73(16):45*-(16), 45–55. https://doi.org/10.5120/12829-0286

Fenwick, D., & Blunt, M. (1998). Network Modeling of Three-Phase Flow in Porous Media. *SPE Journal*, *3*(1), 5–8. https://doi.org/10.2118/38881-PA

Fenwick, D. H., & Blunt, M. J. (1998). Three-dimensional modeling of three phase imbibition and drainage. *Advances in Water Resources*, *21*(2), 121–143. https://doi.org/10.1016/S0309-1708(96)00037-1

Frasconi, P., Gori, M., & Sperduti, A. (1998). A general framework for adaptive processing of data structures. *IEEE Transactions on Neural Networks*, *9*(5), 768–786. https://doi.org/10.1109/72.712151

G.E. Hinton and R. Salakhutdinov. (2006a). Reducing the Dimensionality of Data with Neural Networks, *313*(July), 504–507. https://doi.org/10.1126/science.1127647

G.E. Hinton and R. Salakhutdinov. (2006b). Reducing the Dimensionality of Data with Neural Networks, *313*(July), 504–507. https://doi.org/10.1126/science.1127647

Gesho, M., Zolfaghari, A., Piri, M., Pereira, F. (n.d.). Pore Network Extraction and Upscaling: A Big Data Approach. In *CMWR Conference*. Toronto ON: University of Toronto.

Gholanlo, H. H., Amirpour, M., & Ahmadi, S. (2016). Estimation of water saturation by using radial based function artificial neural network in carbonate reservoir: A case study in Sarvak formation. *Petroleum*, *2*(2), 166–170. https://doi.org/10.1016/j.petlm.2016.04.002

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. *AISTATS '11: Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, *15*, 315–323. https://doi.org/10.1.1.208.6449

Goodfellow, Ian, Bengio, Yoshua, Courville, A. (2016). Deep Learning. *MIT Press*. Retrieved from http://www.deeplearningbook.org/

Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., & Bengio, Y. (2013). Maxout Networks. https://doi.org/10.1093/bib/bbw065.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., … Bengio, Y. (2014). Generative Adversarial Nets. *Advances in Neural Information Processing Systems 27*, 2672–2680. https://doi.org/10.1017/CBO9781139058452

Guler, B., Ertekin, T., & Grader, A. S. (1999). AN Artificial Neural Network baded relative permeability predictor, (99).

Hamada, G. M., & Elshafei, M. A. (2010). Neural network prediction of porosity and permeability of heterogeneous gas sand reservoirs using NMR and conventional logs. *Nafta*, *61*(10), 451–465.

Han, J., & Moraga, C. (1995). The influence of the sigmoid function parameters on the speed of backpropagation learning. In J. Mira & F. Sandoval (Eds.), *From Natural to Artificial Neural Computation: International Workshop on Artificial Neural Networks Malaga-Torremolinos, Spain, June 7--9, 1995 Proceedings* (pp. 195–201). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-59497-3_175

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778. https://doi.org/10.1109/CVPR.2016.90

Hochreiter, J. (1991). Untersuchungen zu dynamischen neuronalen Netzen. *Master's Thesis, Institut Fur Informatik, Technische Universitat, Munchen*, 1–71. Retrieved from http://www.bioinf.jku.at/publications/older/3804_2.pdf%0Ahttp://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Untersuchungen+zu+dynamischen+neuronalen+Netzen#0

Hochreiter, S., & Frasconi, P. (2009). Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies. *A Field Guide to Dynamical Recurrent Networks*. https://doi.org/10.1109/9780470544037.ch14

Hochreiter, S., & Urgen Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Honarpour, M., Koederitz, L., & Harvey, a H. (1986). Relative Permeability of Reservoirs Petroleum, 1–1.

Hui, M.-H., & Blunt, M. J. (2000a). Effects of wettability on three-phase flow in porous media. *The Journal of Physical Chemistry B*, *104*, 3833–3845. https://doi.org/10.1021/jp9933222

Hui, M.-H., & Blunt, M. J. (2000b). Pore-Scale Modeling of Three-Phase Flow and the Effects of Wettability. *SPE*. https://doi.org/10.2523/59309-MS

Hui, M. H., & Blunt, M. J. (2000). Effects of wettability on three-phase flow in porous media. *Journal of Physical Chemistry B*, *104*(16), 3833–3845. https://doi.org/10.1021/jp9933222

Jreou, G. N. S. (2012). Application of neural network to optimize oil field production. *Asian Transactions on Engineering*, *2*(3), 10–23.

Kingma, D. P., & Ba, J. L. (2015). Adam: a Method for Stochastic Optimization. *International Conference on Learning Representations 2015*, 1–15. https://doi.org/http://doi.acm.org.ezproxy.lib.ucf.edu/10.1145/1830483.1830503

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems* (pp. 1097–1105).

Krose, B., & Smagt, P. Van Der. (1996). *An Introduction to Neural Networks*. *Networks*.

Kuwauchi, Y., Abbaszadeh, M., Shirakawa, S., & Yamazaki, N. (1996). Development and Applications of a Three Dimensional Voronoi-Based Flexible Grid Black Oil Reservoir Simulator. *SPE Asia Pacific Oil and Gas Conference*. https://doi.org/10.2118/37028-MS

Lago, M., & Araujo, M. (2001). Threshold Pressure in Capillaries with Polygonal Cross Section. *Journal of Colloid and Interface Science*, *243*(1), 219–226. https://doi.org/10.1006/jcis.2001.7872

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2323. https://doi.org/10.1109/5.726791

Li, F.-F. (n.d.). CS231n Convolutional Neural Networks for Visual Recognition. Retrieved from http://cs231n.github.io/neural-networks-1

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier Nonlinearities Improve Neural Network Acoustic Models. *Proceedings of the 30 Th International Conference on Machine Learning*, *28*, 6. Retrieved from https://web.stanford.edu/~awni/papers/relu_hybrid_icml2013_final.pdf

Mason, G., & Morrow, N. R. (1984). Coexistence of menisci and the influence of neighboring pores on capillary displacement curvatures in sphere packings. *Journal of Colloid And Interface Science*, *100*(2), 519–535. https://doi.org/10.1016/0021-9797(84)90455-7

May, R., Dandy, G., & Maier, H. (2011). Review of Input Variable Selection Methods for Artificial Neural Networks. *Artificial Neural Networks - Methodological Advances and Biomedical Applications*, (August 2016), 362. https://doi.org/10.5772/644

Mayer, R. P., & Stowe, R. A. (1965). Mercury porosimetry—breakthrough pressure for penetration between packed spheres. *Journal of Colloid Science*, *20*(8), 893–911. https://doi.org/10.1016/0095-8522(65)90061-9

Miao, X., Gerke, K. M., & Sizonenko, T. O. (2017a). A new way to parameterize hydraulic conductances of pore elements: A step towards creating pore-networks without pore shape simplifications. *Advances in Water Resources*, *105*, 162–172. https://doi.org/10.1016/j.advwatres.2017.04.021

Miao, X., Gerke, K. M., & Sizonenko, T. O. (2017b). A new way to parameterize hydraulic conductances of pore elements: A step towards creating pore-networks without pore shape simplifications. *Advances in Water Resources*, *105*, 162–172. https://doi.org/10.1016/j.advwatres.2017.04.021

Miles, J. (2014). R Squared, Adjusted R Squared. In *Wiley StatsRef: Statistics Reference Online*. https://doi.org/10.1002/9781118445112.stat06627

Mohaghegh, S., & Ameri, S. (1995). Artificial Neural Network As A Valuable Tool For Petroleum Engineers. *Network*. https://doi.org/10.1.1.112.4627

Mosser, L., Dubrule, O., & Blunt, M. J. (2017). Reconstruction of three-dimensional porous media using generative adversarial neural networks. Retrieved from http://arxiv.org/abs/1704.03225

Moujahid, A. (2016). A Practical Introduction to Deep Learning with Caffe and Python. Retrieved from http://adilmoujahid.com/posts/2016/06/introduction-deep-learning-python-caffe/

Name, L., Name, F., Training, O., Training, P., Darin, C., Training, R. O., … Co-investigator, N. (1998). Convolution Networks for Images, Speech, and Time-Series. *Igarss 2014*, (1), 1–5. https://doi.org/10.1007/s13398-014-0173-7.2

Nielsen, M. (2015). Neural Networks and Deep Learning. *Determination Press*. Retrieved from http://neuralnetworksanddeeplearning.com/index.html

Oak, M. J., Baker, L. E., & Thomas, D. C. (1990). Three-Phase Relative Permeability of Berea Sandstone. *Journal of Petroleum Technology*, *42*(8), 1054–1061. https://doi.org/10.2118/17370-pa

Oren, P.-E., Bakke, S., & Arntzen, O. J. (1998). Extending Predictive Capabilities to Network Models. *SPE Journal*, *3*(4), 324–336. https://doi.org/10.2118/52052-PA

Patzek, T. W., & Silin, D. B. (2001). Shape Factor and Hydraulic Conductance in Noncircular Capillaries. *Journal of Colloid and Interface Science*, *236*(2), 295–304. https://doi.org/10.1006/jcis.2000.7413

Pedregosa, F., & Varoquaux, G. (2011). *Scikit-learn: Machine learning in Python*. *... of Machine Learning ...* (Vol. 12). https://doi.org/10.1007/s13398-014-0173-7.2

Piri, M. (2003). Pore-Scale Modelling of Three-Phase Flow.

Piri, M., & Blunt, M. J. (2005a). Three-dimensional mixed-wet random pore-scale network modeling of two- And three-phase flow in porous media. I. Model description. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, *71*(2). https://doi.org/10.1103/PhysRevE.71.026301

Piri, M., & Blunt, M. J. (2005b). Three-dimensional mixed-wet random pore-scale network modeling of two- and three-phase flow in porous media. II. Results. *Physical Review E*, *71*(2), 26302. https://doi.org/10.1103/PhysRevE.71.026302

Princen, H. . (1969a). Capillary phenomena in assemblies of parallel cylinders: II. Capillary rise between two cylinders. *Journal of Colloid and Interface Science*, *30*(3), 359–371. https://doi.org/10.1016/0021-9797(69)90379-8

Princen, H. M. (1969b). Capillary phenomena in assemblies of parallel cylinders. I. Capillary Rise between Two Cylinders. *Journal of Colloid And Interface Science*, *30*(1), 171–184. https://doi.org/10.1016/0021-9797(70)90167-0

Princen, H. M. (1970). Capillary phenomena in assemblies of parallel cylinders. III. Liquid Columns between Horizontal Parallel Cylinders. *Journal of Colloid And Interface Science*, *34*(2), 171–184. https://doi.org/10.1016/0021-9797(70)90167-0

Rabbani, A., Assadi, A., Kharrat, R., Dashti, N., & Ayatollahi, S. (2017). Estimation of carbonates permeability using pore network parameters extracted from thin section images and comparison with experimental data. *Journal of Natural Gas Science and Engineering*, *42*, 85–98. https://doi.org/10.1016/j.jngse.2017.02.045

Reichhardt, D., & Isaiah, J. (2013). Performing Reservoir Simulation with Neural Network Enhanced Data. *2013 SPE Digital Energy ...*, (March), 5–7. Retrieved from http://www.onepetro.org/mslib/servlet/onepetropreview?id=SPE-163691-MS

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in …. *Psychological Review*, *65*(6), 386–408. https://doi.org/10.1037/h0042519

Rowell, R. L. (1998). Physical Chemistry of Surfaces, 6th ed. *Journal of Colloid and Interface Science*. https://doi.org/10.1006/jcis.1998.5823

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *ArXiv E-Prints*.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. https://doi.org/10.1038/323533a0

Ryazanov, A. V., van Dijke, M. I. J., & Sorbie, K. S. (2009). Two-phase pore-network modelling: Existence of oil layers during water invasion. *Transport in Porous Media*, *80*(1), 79–99. https://doi.org/10.1007/s11242-009-9345-x

Sharak, A. Z., Samimi, A., Mousavi, S. A., & Bozarjamhari, R. B. (2014). Investigation of membrane preparation condition effect on the PSD and porosity of the membranes using a novel image processing technique. *Journal of Applied Polymer Science*, *131*(4). https://doi.org/10.1002/app.39899

Shokooh Saljooghi, B., & Hezarkhani, A. (2015). A new approach to improve permeability prediction of petroleum reservoirs using neural network adaptive wavelet (wavenet). *Journal of Petroleum Science and Engineering*, *133*, 851–861. https://doi.org/10.1016/j.petrol.2015.04.002

Shu, L., McIsaac, K., Osinski, G. R., & Francis, R. (2017). Unsupervised feature learning for autonomous rock image classification. *Computers and Geosciences*, *106*(March 2016), 10–17. https://doi.org/10.1016/j.cageo.2017.05.010

Sperduti, A., & Starita, A. (1997). Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, *8*(3), 714–735. https://doi.org/10.1109/72.572108

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, *15*, 1929–1958. https://doi.org/10.1214/12-AOS1000

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems (NIPS)*, 3104–3112. https://doi.org/10.1007/s10107-014-0839-0

Van Dijke, M. I. J., Lago, M., Sorbie, K. S., & Araujo, M. (2004). Free energy balance for three fluid phases in a capillary of arbitrarily shaped cross-section: Capillary entry pressures and layers of the intermediate-wetting phase. *Journal of Colloid and Interface Science*, *277*(1), 184–201. https://doi.org/10.1016/j.jcis.2004.05.021

Van Dijke, M. I. J., Piri, M., Helland, J. O., Sorbie, K. S., Blunt, M. J., & Skj??veland, S. M. (2007). Criteria for three-fluid configurations including layers in a pore with nonuniform wettability. *Water Resources Research*, *43*(12), 1–11. https://doi.org/10.1029/2006WR005761

Van Dijke, M. I. J., & Sorbie, K. S. (2003). Three-phase capillary entry conditions in pores of noncircular cross-section. *Journal of Colloid and Interface Science*, *260*(2), 385–397. https://doi.org/10.1016/S0021-9797(02)00228-X

Van Dijke, M. I. J., & Sorbie, K. S. (2006). Existence of fluid layers in the corners of a capillary with non-uniform wettability. *Journal of Colloid and Interface Science*, *293*(2), 455–463. https://doi.org/10.1016/j.jcis.2005.06.059

Weisstein, E. W. (n.d.). “Hyperbolic Tangent.”From MathWorld--A Wolfram Web Resource. Retrieved from http://mathworld.wolfram.com/HyperbolicTangent.html

Werbos, P. J. (1990). Backpropagation Through Time: What It Does and How to Do It. IEEE.

Windows, M., Os, M., When, C. P., Wei, Y., Yildirim, P., den Bulte, C., … Szegedy, C. (2014). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *Uma Ética Para Quantos?*, *XXXIII*(2), 81–87. https://doi.org/10.1007/s13398-014-0173-7.2

Yun, W. (n.d.). Deep Learning : Automated Surface Characterization of Porous Media to Understand Geological Fluid Flow.

Zhong, Z. (2017). *STORAGE CAPACITY ESTIMATION OF COMMERCIAL SCALE INJECTION AND STORAGE OF CO2 IN THE JACKSONBURG-STRINGTOWN OIL FIELD*. West Virginia University. Retrieved from https://search.proquest.com/docview/1900944977?accountid=14556

Zhong, Z., & Carr, T. R. (2016). Application of mixed kernels function (MKF) based support vector regression model (SVR) for CO2– Reservoir oil minimum miscibility pressure prediction. *Fuel*, *184*, 590–603. https://doi.org/10.1016/j.fuel.2016.07.030

Zolfaghari, A., & Piri, M. (n.d.-a). Effects of Thermodynamically Consistent Threshold Capillary Pressures and Composite Menisci on Residual Oil Saturation and Relative Permeability. In *Gordon Research Conference - Flow & Transport in Permeable Media*. Lewiston, Main: Bates College.

Zolfaghari, A., & Piri, M. (n.d.-b). Pore-scale Network Modeling of Two- and Three-phase Flow based on Thermodynamically Consistent Threshold Capillary Pressures. In *Gordon Research Conference - Flow & Transport in Permeable Media*. Lewiston, Main: Bates College.

Zolfaghari, A., & Piri, M. (2017a). Pore-Scale Network Modeling of Three-Phase Flow Based on Thermodynamically Consistent Threshold Capillary Pressures. I. Cusp Formation and Collapse. *Transport in Porous Media*, *116*(3), 1093–1137. https://doi.org/10.1007/s11242-016-0814-8

Zolfaghari, A., & Piri, M. (2017b). Pore-scale Network Modeling of Three-Phase Flow Based on Thermodynamically Consistent Threshold Capillary Pressures. II. Results. *Transport in Porous Media*, *116*(3), 1093–1137. https://doi.org/10.1007/s11242-016-0814-8

Zolfaghari Shahrak, A. (2014). Pore-Scale Network Modeling of of Two- and Three-Phase Flow Based on Thermodynamically Consistent Threshold Capillary Pressures.