Multiple Tools for Automated Nanofiber Characterization by Image Processing

by

Erqian Gao

A thesis

presented to the University of Waterloo

in fulfillment of the

thesis requirement for the degree of

Master of Applied Science

in

Mechanical and Mechatronics Engineering

Waterloo, Ontario, Canada, 2022

Author's Declaration

Abstract

Nanofibers have been widely used in many engineering applications, including air filtration, energy storage, and biomedical engineering. Their performances largely depend on the morphology of nanofibers. The key morphological parameters include fiber diameter, pore size, porosity, and thickness homogeneity, which are often manually determined at this moment.

There is a need of automated tools for fast determination of nanofiber fiber diameters, pore size, porosity, and surface/thickness homogeneity. Researchers have developed automated tools to determine nanofiber diameters, primarily using MATLAB. However, none of the tools reported earlier can *automatically* process multiple images, which is essential to the accuracy of results. Regarding porosity, the most accurate approach to pore size determination is Brunauer-Emmett-Teller (BET) surface area analysis. This experimental approach is precise but time-consuming, costly, and destructive. Alternatively, the image processing method may offer a quick estimation of the porosity of the nanofiber mat.

In addition, many researchers consider the surface homogeneity with even fiber diameters as good a homogeneity. However, the diameters shown in a SEM image only indicate the local homogeneity with a minimal dimension bounding the SEM image. Alternatively, the thickness of the entire nanofiber sample is a more reliable criterion. However, experimental determination of the thickness throughout the entire nanofiber mat is challenging because of its fragility and thinness. If the thickness of only a few places is measured, the local unevenness may be overlooked during the sampling.

The main objective of this research is therefore to develop a set of automated tools for the characterization of diameter, inter-fiber and intra-fiber pores, porosity, and thickness homogeneity of nanofiber mats. Among them, three different approaches are used to determine the nanofiber diameter.

Specifically, the following tools are developed to achieve the preceding goals. First, multi-image processing tools are developed to determine the diameters of nanofiber mats using

MATLAB and machine learning based on UNet and residual neural network. Second, serval image processing tools using different image segmentation methods are proposed to determine the area of inter-fiber pores, intra-fiber pores, and the porosity. The most accurate one is identified by comparing their performances with experimental data. Finally, a characterization tool is proposed to quantitatively compare the nanofiber homogeneity by analyzing the light transmittance.

Acknowledgements

I would like to thank my supervisor, Dr. Zhongchao Tan, for his patient guidance and encouragement. His positive outlook and confidence in me inspired me and changed the way I look at life. In addition, my thanks go to my thesis readers, Drs. Fue-Sang Lien and Zhao Pan, for their contributions to the quality of the final thesis.

I would like to thank Dr. Hesheng Yu for his guidance and support on the thesis project, and Dr. Yifu Li and Mr. Yi Zhang, who generous share their experimental data with me.

My warm gratitude also goes to all the current members of Green Energy and Pollution Control Research Lab during my graduate studies.

I wish to thank my parents. Thanks for your constant unconditional support, and encouragements. Words cannot express my gratitude for everything all you have done for me.

Table of Contents

List of Figures

List of Tables

CHAPTER 1. INTRODUCTION

1.1 Background

Nanofibers have been developed for many engineering applications, for example, air filtration (Givehchi et al. 2016, 2018; Ahne et al. 2018; Homaeigohar & Elbahri, 2014), energy storage (Chen et al., 2014; Li et al. 2022; Li et al. 2020, 2019), drug delivery (Cui et al., 2018), and tissue engineering (Agarwal et al., 2008; Rijal et al., 2018). Nanofiber morphology can largely influence the properties and performances of the devices using nanofibers (Barhoum et al., 2019). A slight change in nanofiber morphology can significantly impact the macroscopic material properties. The critical morphological parameters include fiber diameter, pore size, porosity, and homogeneity.

Many potential applications in different areas require a high porosity material, for example, cancer cell enrichment (Zhang et al. 2021), sensing materials (Zhang et al., 2010), and catalyst (Patel et al., 2007). In tissue engineering, nanofibers are used as artificial scaffolds, which act as structural support and the environment for cell proliferating and tissue forming. The pore size on nanofiber surface greatly affects cell anchoring, proliferation, and migration. As another example, a smaller pore size on nanofiber can offer better cellular adhesion (Kulpreechanan et al., 2013). However, when the pore size is smaller than the cell, the over-dense nanofiber mat prevents the cells from migrating into the under layer of the scaffold, hindering the forming of proper tissue. Therefore, it is essential to characterize both porosity and pore size of nanofibers.

Finally, fiber surface homogeneity also plays a vital role in the performance of nanofiber-based devices. Two examples are battery separator (Li et al. 2019) and air filtration. When a nanofiber mat is used for air filtration, homogeneous surfaces can prevent local piercing of the mat and the resultant particle penetration (Zhang et al., 2010; Kim et al., 2020). As another example, enhanced homogeneity of a nanofiber mat can improve the uniformity of electrical conductivity, which is an essential feature for electrical usage (Kabir & Demirocak, 2017). Therefore, the characterization of nanofiber homogeneity is important.

1.1.1 Nanofiber diameter determination

Among all the characteristics of nanofibers, diameter is the most important because it dramatically affects related device performance. For example, when nanofibers are used for air filtration (Givehchi et al. 2021), fiber diameters close to the mean free path of air molecules may result in slip flow (Givehchi et al. 2014). The slip flow effect can greatly reduce the pressure drop trough the nanofibrous filter, which means energy consumption (Zhao et al., 2016a). In addition, smaller nanofiber diameters result in thinner mats with a larger volume to surface ratio. These factors, in turn, affect the mechanical properties, electrical conductivity, bioactivity of the nanofibers (Barhoum et al., 2019; Neal et al., 2009). In short, the diameter is the most crucial parameter of a nanofiber.

Despite the great importance of nanofiber diameter determination, the tools to measure the diameters of nanofibers are limited. To the best of my knowledge, image processing is the only feasible way to determine nanofiber diameters. The images for nanofiber characterization are often taken using a scanning electron microscope (SEM). Then, the nanofiber diameters can be determined by manually measuring the number of pixels between the fiber boundaries and converting the number into length based on the scale (Givehchi et al., 2016). This manual approach is time-consuming and often subjective. Thus, an *automated* approach for nanofiber diameter determination is needed.

Other researchers have developed some automated tools to determine nanofiber diameters, primarily using the MATLAB software. The SEM images are usually first binarized and then skeletonized to find its fiber centerline in those approaches. Finally, the nanofiber diameter is determined by double the distance between fiber centerline and fiber boundary. In this approach, it is noticed that most methods reported earlier, such as DiameterJ and SIMpoly, are subjective (Hotaling et al., 2015a; Murphy et al., 2020). In addition, the results depend on the image magnification as reported by Givehchi et al. (2016). Furthermore, none of the tools reported earlier can *automatically* process multi-images. Therefore, there is a lack of tools that can combine the results from multi-images with improved accuracy of diameter determination.

MATLAB scripts can help determine the nanofiber diameters by processing of multiple SEM images. However, MATLAB is a commercial software package that is costly and bulky, taking up many computer resources. Many built-in functions packaged in MATLAB are unneeded. Additionally, its capability in image processing is limited and highly dependent on functions provided in its toolbox.

Compared to MATLAB, an open-source script is more budget-friendly, occupies fewer computing resources, and offers a greater flexibility to the end users. The source codes can be further customized and changed for different applications. Thus, this thesis tackles the preceding challenges by developing a Python-based open-source image processing script (*see* Sections 3.3 and 3.4). The Python script can be run on a free platform, such as Jupyter Notebook, using few computer resources.

In addition, existing image segmentation methods cannot accurately differentiate nanofiber from the background in the presence of rough surfaces. Comparison between the results obtained manually and automatedly shows that there is still room for improvement in accuracy (*see* Section 3.2). Thus, machine learning is used for image segmentation to improve the accuracy of the results and to extend the scope of application. Two different machine learning approaches are used separately to develop automated tools for nanofiber diameter determination (*see* Sections 3.3 and 3.4).

1.1.2 Determination of nanofiber porosity and pore size

Porosity and pore size are two essential characteristics of nanofibers and nanofibrous mats. The porosity of a nanofiber mat indicates its surface-to-volume ratio. Total porosity ca be measured by mercury intrusion, liquid adsorption, or gas adsorption. These experimental approaches are deemed accurate but time-consuming, costly, and destructive. Alternatively, the image processing method may offer a fast estimation of porosity. Using the image processing method, a preliminary result can be obtained in a few seconds.

The most accurate experimental approach to pore size determination is Brunauer-Emmett-Teller (BET) surface area analysis (Naderi, 2015). The basic idea is to measure the gas absorption and adsorption amount at different relative pressure. Standard BET is carried out using nitrogen under 1 atm pressure at 77 K, which is the boiling point of nitrogen. The total surface area and pore volume can be calculated from the nitrogen intake. With an assumption that all pores are uniform cylindrical pores in solid material, a theoretical value of pore size can be calculated. With the model of cylindrical pores, the total pore volume is

$$V_{(bet)} = \pi d^2 l \tag{1-1}$$

where *d* is the mean pore size, and *l* is the total length of the pores. The BET surface area is

$$S_{(bet)} = \pi d l \tag{1-2}$$

The length, *l*, can be eliminated from the two equations, and the average diameter, *d*, can thus be calculated. However, the pores in nanofiber mats are caused from the fiber cross structures, and it is erratic to regard them as solid and uniform cylinders. Thus, the BET analysis method can only offer a theoretical equivalent pore size. Furthermore, it provides neither real pore sizes nor pore size distributions. For nanofibers, this method is mainly used to determine the specific surface areas or porosities instead of pore sizes (Evora et al., 2010; Ji & Zhang, 2009; Nazari et al., 2018).

Alternatively, image processing may be more feasible and accurate for pore size determination. The pores sizes can be manually measured from the image, but again the manual method is time-consuming and subjective, and the number of sampling points is limited. Thus, a reliable automated tool for the determination of nanofiber diameter and pore size is needed. Accordingly, Section 3.2 of this thesis reports a MATLAB tool that can automatically calculate the sizes of inter-fiber and intra-fiber pores. The procedure involves image segmentation, edge detection, and pixels calculation.

1.1.3 Determination of nanofiber surface homogeneity

The surface homogeneity can be considered as the homogeneity across the thickness, or thickness uniformity, throughout a nanofiber mat. Many earlier studies have reported the effects of surface homogeneity on the performances of nanofiber mats. For example, Arshad et al. (2011) indicate that a great homogeneity can help eliminate structural weakness and increase the tensile strength of the nanofibers. Kim et al. (2020) also point out the negative effects of inhomogeneity on air filtration. The overly dense area causes a great air resistance, which increases the pressure drop cross the filter. At the thin part, however, local piercing occurs and decreases the filtration efficiency (Kim et al., 2020). In short, fiber inhomogeneity plays a negative role in both pressure drop and filtration efficiency.

Though the negative impact that inhomogeneity can cause has been realized, earlier studies have overlooked the impact of this morphological parameter on nanofibers. Many researchers equate the surface homogeneity with diameter distribution and regard a concentrated diameter distribution as acceptable homogeneity(Zheng et al., 2020). However, the reported homogeneity depends on the observation window, *i.e.*, size scale. The nanofiber diameters observed in a SEM image can only be used to indicate the homogeneity at the minimal dimension of the SEM image. A nanofiber appearing uniform in the SEM image may be otherwise. Alternatively, the thickness of a nanofiber mat is deemed a reliable criterion for thickness homogeneity. However, measuring the thickness throughout the entire fiber mat is impractical. On the other hand, if the thickness of only a few places is measured, the local unevenness may be overlooked because of limited sampling.

The preceding points may be the reasons that most previous research only qualitatively analyze surface homogeneity. To tackle this challenge, this thesis work reports a simple method to determine surface heterogeneity using images taken with an optical microscope. The hypothesis is as follows. For the same sample under the same optical microscope, the light transmittance is low through the spot where the nanofiber is thick, resulting in low brightness, and *vice versa*. Thus, a quantitative evaluation of homogeneity can be determined for comparison by analyzing the difference in brightness variation of the same microscopic image. This principle has not been tested for its feasibility in the field of nanofibers, although it has been used in the paper industry (Chhabra, 2003) to determine the evenness of paper thickness.

1.2 Research Objectives

The overall objective of this research is to develop a set of automated tools for the characterization of the diameter, size of inter-fiber and intra-fiber pores, porosity, and surface homogeneity of nanofibers and nanofiber mats. Among them, three different approaches are used to determine the nanofiber diameters. Specifically, the following tasks were completed to achieve the goal:

1. Multi-image processing using MATLAB to determine the diameters of nanofibers
2. Machine learning, specifically the U-shape neural network, for image segmentation to determine nanofiber diameters.
3. Diameter determination using the residual neural network.
4. Serval image processing tools using different image segmentation methods to determine the area of inter-fiber pores, intra-fiber pores and the porosity.
5. Quantitative comparison of nanofiber surface homogeneity using optical microscopic images.

1.3 Thesis Organization

This thesis is structured as follows. Chapter 1 briefly introduces the background and objectives of the study. Chapter 2 presents a state-of-the-art literature review on related works and background knowledge required for this thesis research project. Chapter 3 introduces different approaches for nanofiber diameter determination. Chapter 4 introduces the methods for automated measurement of inter-fiber pores, intra-fiber pores, porosity, and surface homogeneity. Finally, Chapter 5 summarizes the conclusions drawn from this work and recommendations to the future research.

1.4 Highlight of Contributions

To the best of my knowledge, this thesis is the first to report an image processing tool that can process multi-images for nanofiber diameter determination. In addition, it is the first attempt to apply deep learning neural network to nanofiber diameter determination. It is also the first to apply UNet, a biomedical image segmentation method, to the nanofiber diameter determination.

Regarding the pore size and porosity, this thesis project implements some representative approaches in the previous research and analyzed their results. The thesis also compares the advantages and limits of the earlier approaches.

Last, this thesis is the first to innovatively use a simple image processing method to analyze the thickness homogeneity of nanofiber mats. This method has not been reported for nanofibers, regardless of its applications in the paper industry.

CHAPTER 2. LITERATURE REVIEW

2.1 Electrospinning and Nanofiber

Fibers with diameters in the nanometer range are often called *nanofibers*. Nanofibers can be produced from various materials with different morphologies and hence have different properties and applications. The methods for fabricating nanofiber materials include electrospinning, air-jet spinning, centrifugal spinning, template synthesis, phase inversion, and spinneret-based tunable engineered parameters (STEP) techniques (Stojanovska et al., 2016). Among all these methods, electrospinning is regarded as the most promising fabrication method for sub-100 nm fibers due to its simplicity and effectiveness. The main advantages of electrospinning include its ability to fabricate fibers from various polymers, its ability to produce fibers with diameters below 100 nm, ease of setup for the device, and ease of functionalization for various purposes (Givehchi et al. 2016).

Figure 2-1 shows a schematic of a typical lab-scale electrospinning device, which includes a high voltage power supply, a spinneret (e.g., a syringe), and a grounded collector. It produces nanofibers by applying a high-voltage supply to polymer solutions. The feedstock, normally a polymer solution, is stored behind the spinneret. When the polymer solution is pushed out of the spinneret at a constant rate, forming a droplet at the spinneret tip, a high voltage is applied to that droplet. The high voltage applies an electrostatic repulsion force on the droplet, stretching it and deforming its shape. When the high voltage reaches a critical value, the electrostatic repulsion force becomes strong enough to overcome the surface tension of the droplet. As a result, the polymer solution jets out, forming a straight generatrix of a cone; this cone is often referred to as *Taylor cone* (Taylor, 1969). After the solution jet out, it travels through the air, with randomly whipping, and lands on the grounded metal collector. The solvent evaporates at the jetting and whipping stages, and the polymer precipitates on the collector, forming solid nanofibers (Givehchi et al.; 2016; Li et al. 2021).

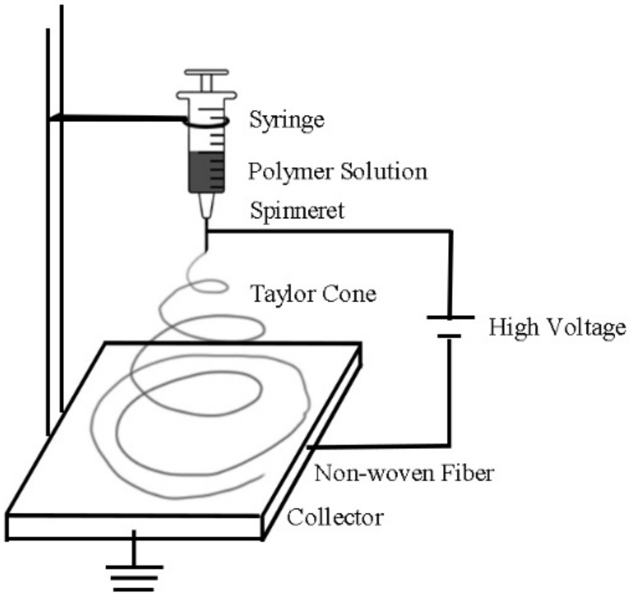


Figure 2-1. Schematic of a typical electrospinning setup in a laboratory setting

Humans began to use electrospinning to make nanofibers as early as 1995 (Doshi & Reneker, 1995). In the following two decades, the efforts to study the principle of electrospinning and improve the electrospinning process have never stopped. However, fabricating nanofiber with well-defined morphology at a high yield is still a significant challenge to this day.

2.1.1 Nanofiber and its emerging applications

Nanofibers have been extensively studied in the past two decades. This section introduces three application areas where nanofibers are most frequently used: energy, environment, and health care.

Nanofibers are widely used electrode and separator materials in the energy fields, such as solar cells (Pierini et al., 2017), fuel cells (Skupov et al., 2017), electrochemical hydrogen production (Sebastián et al., 2010), and batteries (Chen et al., 2013; Miao et al., 2013). The high porosity and flexibility of nanofiber structures enable them to outperform other materials in energy applications (Yu et al., 2016; Iqbal et al., 2017).

Nanofibers are also used for air filtration in the environmental field. Research has proven that filters made by nanofibers have lower pressure drops and higher filtration efficiencies compared to conventional air filters. The sizeable surface-area-to-volume ratio and large porosity of nanofibers enable them to absorb more contaminants (Yoon et al., 2008; Givehchi et al., 2016).

Nanofibers can also be used as drug delivery media and wound dressings in the biomedical field. When nanofibers are used as drug delivery media, their tortuous and inner-connected structures offer a great potential for controlled drug delivery (Sebe et al., 2015). The high surface area to volume ratio of nanofibers also help increase the contact area of drugs with the vivo environment, and therefore expedite the dissolution of water-soluble drugs (Nagy et al., 2012). Therapeutic agents can be either loaded in the posttreatment process of nanofibers (Kataria et al., 2014) or added in a polymer solution before electrospinning (Xu et al., 2009).

Electrospun nanofibers overlapping scaffolds are similar to extracellular matrix in terms of structure, giving them an advantage as wound dressings. The structure of nanofiber naturally mimics the extracellular matrix of the human body. Compared to traditional wound dressings, nanofibers are more favorable to body tissue and have better cell attachment and proliferation, thus lowering the possibility of scar tissue formation (Choi et al., 2008).

Nanofibers can also be used for circulating tumour cell analysis for the early diagnosis of cancer, evaluating cancer progression and assessing treatment efficacy (Zhang et al., 2021). For example, electrospun nanofibers may be a simple and efficient way to create a nanoroughened surface. The extracellular matrix can be well mimicked by nanofibers for cell adhesion and culture.

2.1.2 Morphology characterization of nanofibers

The morphology of nanofibers plays a vital role in the physical and chemical properties of the nanofibers (Moon et al., 2011; Givehchi et al., 2016; Barhoum et al., 2019). Accurate control of the nanofiber morphology has often been the objectives of nanofiber studies. For example, Huang et al. (2011) controlled the diameters and porosities of nanofibers by adjusting environmental humidity during electrospinning. There are also many other studies examining the impact of other factors on nanofiber morphology. The factors include polymer molecular weight (Eda & Shivkumar, 2007; Koski et al., 2004; Mckee et al., 2006; Zhao et al., 2005), solution concentration (Amiraliyan et al., 2009; Eda & Shivkumar, 2007), solvent's volatility (Veleirinho et al., 2008), electrospinning voltage(Dhanalakshmi et al., 2015), solvent flow rate (Yuan et al., 2004) and needle-to-collector distance (Yuan et al., 2004).

In this context, accurate and effective characterization is particularly important. The characterization of morphology not only can offer information about the fiber's properties, but also can offer guidance for fabrication protocol optimization and future fabrication (Khajavi & Abbasipour, 2017). However, the current morphology characterization techniques are still in their infancy, and many improvements can be made to them.

2.2 Diameter Determination

2.2.1 Existing image processing algorithms

Currently, the nanofiber diameters can only be accurately determined by image processing. Scanning electron microscope (SEM) is frequently used for the characterization of nanofiber morphology. The number of pixels between the fiber boundaries in a SEM image is measured and converted to length according to the scale of the SEM image. However, this manual measurement is time-consuming and the number of samples is limited.

Alternatively, Tomba et al. (2010) developed a multivariate image analysis (MIA) system for the determination of nanofiber diameters. The MIA can extract essential information from a SEM image and reduce the date size and data dimensionality. Multiway principal component analysis (MPCA) is also performed on the SEM image. The MPCA linearly combines the original variables with principal components to help identify the greatest variability in data. However, it is noticed that the MPCA method has many shortcomings. For example, it cannot effectively separate fibers from each other in the overlapped areas, resulting in deviation. For the best denoising effect, this method also assumes that all fibers are straight, which can be a significant disadvantage for bent nanofibers. Finally, the thin fibers in low-resolution images are likely to be overlooked because of repeated denoising operations.

To tackle the preceding challenge of overlapping fibers, Shin et al. (2008) reproted another method for automatically measuring nanofiber diameters. Their method consists of the following three main steps:

1. fiber boundary detection
2. fiber individualization
3. distance transformation

The fiber individualization is at the core of this approach. In this step, fiber boundaries are detected using the Canny edge detection, in which an edge point is defined as the point where strength has a local maximum in the direction of the gradient. The most significant advantage of the Canny edge detection is its accuracy and effectiveness in separating overlapped fibers in SEM images. On the other hand, the most significant disadvantage of Canny edge detection is that it cannot be used on highly oriented nanofibers. When dealing with high-oriented fibers, this method can easily outline the fiber boundaries, but has difficulty distinguishing the fiber sides and the background.

Thus, Murphy et al. (2020) proposed an alternative method for automatically measuring nanofiber diameter. This method consists of the following three main steps:

1. morphologically reconstructing
2. image skeletonization
3. fiber diameter calculation

The morphologically reconstructing step first binarizes the image using the Otsu method (Otsu, 1979). Then, the centerline of the fiber is determined and reduced into 1 pixel. Finally, the diameter is calculated by doubling the distance between the fiber boundary and fiber centerline.

Babashakoori et al. (2019) improved the algorithm for nanofiber diameter determination. They used the Hough transform (Gonzalez et al., 2002) to extract the features of the image, generating skeletonization of the SEM image. The most significant advantage of the Hough transform is its tolerance for discontinued features and image noises. After the skeletonization, the Gabor filtering bank (Field 1987) is used to further improve the accuracy of measurement.

Alternatively, Dehghan et al. (2016) used the Fuzzy Clustering Method (FCM) to determine the nanofiber diameters. The SEM image is segmented into a binary image by FCM method.

The optimal cluster value is determined by trying out different clustering numbers ranging from one to six. The optimal clustering value is the value that results in the most similarity between the binary image and the original image. Finally, the image similarity is calculated using the structural similarity index method.

Among all the studies mentioned above, image binarization is a major point of difference. The algorithm for the image binarization can be divided into two categories: global threshold and local threshold. The global threshold chooses a single threshold value, and all values smaller than the threshold value are regarded as 0, corresponding to pure black. All values greater than the threshold value are regard as 255, corresponding to pure white. Covnersely, local threshold is the method in which multi-threshold value is used at a different spot on the image. It is a necessary means for images with uneven luminance.

The global and local thresholds are not totally opposite. Many global threshold-based methods can also be applied locally. To locally apply a global threshold algorithm, the image needs to be first divided into many sub-images. Then the global threshold algorithm is applied to each sub-image, and the result is obtained by stitching together all sub-images. That is why some algorithms can be applied either locally or globally.

With the recent advances in algorithms, the third categories appear. They include the clustering algorithm, such as K-means and FCM (Dehghan et al., 2016). Dehghan et al. (2016) explained the approach of using the fuzzy clustering method (FCM) to measure the diameter of polyblend fibers.

2.2.2 Current image processing platforms for nanofiber characterization

Existing image processing platforms for nanofiber characterization can be divided into commercial software and open-source platforms. MATLAB is the most widely used commercial software, which has been used by many researchers such as Shin's group(Ho Shin et al., 2008), Tan's group (Ahne et al. 2018; Givehchi, 2016, 2018, 2021; Li et al. 2020; 2021), MIA(Tomba et al., 2010), and SIMpoly (Murphy et al., 2020). The most used open-source software is ImageJ (Ferreira and Rasband, 2012) and its plug-in DiameterJ (Hotaling et al., 2015). Both commercial software and open-source software have their own advantages and disadvantages. Table 2-1 compares their advantages and disadvantages.

Table 2-1. Comparison of different image processing platforms

Platform	Pro	Con
MATLAB	• Well-maintained environment • Good training materials and tech help	• Expensive • Commercial software has its own restriction • Take up a lot of computer resources
Open-source software	• Free • Software takes up little space in computer	• Steep learning curve • DiameterJ's author stopped maintenance and update

2.2.3 Machine learning approach for image segmentation

The determination of nanofiber diameters consists of the following three steps:

1. image segmentation
2. skeletonization
3. distance measurement

The image segmentation step can be done by machine learning. There are three main categories for image segmentation in the machine learning field:

1) semantic segmentation
2) instance segmentation
3) panoptic segmentation

The semantic segmentation is the most basic type of segmentation that identifies different objects in an image. The objects can be a person, book, flower, car, *etc*. Typical algorithms for the semantic segmentation include UNet (Ronneberger et al., 2015), DeepLab (Chen et al., 2014, 2018), and FCN (Long et al., 2014).

An instance segmentation combines target detection and semantic segmentation. The instance segmentation first identifies the bounding box of the instance by target detection. Then it performs the semantic segmentation in the detection box, and each segmentation result is output as a different instance. The instances of each semantic category are often be regarded as an extension of the semantic segmentation. For example, if the semantic segmentation method identifies a flock of birds in an image, then the instance segmentation further eases the object detection by identifying individual birds in the flock. Frequently used algorithms for instance segmentation include Mask R-CNN (He et al., 2017), YOLCAT (Bolya et al., 2019), and Gated-SCNN(Takikawa et al., 2019).

The panoptic segmentation is an emerging method compared to the other two methods (Kirillov et al., 2018). First proposed by Kirillov et al. (2018), this algorithm combines the semantic and instance segmentations and generates a coherent scene segmentation.

In this thesis project, semantic segmentation is sufficient in the context of nanofiber morphology analysis because there is only one object category (*i.e.*, fiber) throughout this thesis project. Locating the classified items is unneeded. In addition, all fibers are equally analyzed for characterization and distinguishing the fibers is unnecessary. For these two reasons, the semantic segmentation is enough for this project.

The algorithms of semantic segmentation reported earlier include UNet (Ronneberger et al., 2015), Deeplab (Chen et al., 2018), and FCN (Long et al., 2014). Among all these algorithms, UNet is one of the earliest segmentation algorithms that has been repeatedly studied and implemented. Thus, the principles and implementation paths of UNet are relatively clear, and useful information are easily available in the literature. In addition, UNet is well known for its low requirement of training data and short training time, which can significantly reduce the time for preparing the database and reduce the requirements for computer hardware. For example, Ronneberger et al. (2015) obtained a high accuracy of 0.9203 using the PhC-U373 dataset, which contains only 35 partially annotated images for training. Furthermore, in the ISBI 2015 Challenge[1], the UNet model outperformed all other models with an accuracy of 0.7756 on DIC-HeLa dataset, while the second-best algorithm can only achieve an accuracy of 46% (Ronneberger et al., 2015).

Therefore, the UNet model is used in this project, and it is further explained as follows. The UNet model was first proposed by Ronneberger et al (2015) for biomedical image

[1] https://biomedicalimaging.org/2015/program/isbi-challenges/

segmentation. Its name comes from a U-shaped neural network structure of this model. This model contains a contracting path to capture image features and a symmetric expanding path for precise localization. There are four conventional steps in the contracting path, and each step contains two 3x3 convolutions layers and a max-pooling. The expanding path has a relatively symmetric structure. It also contains four steps, in each of which an up-convolution is performed, followed by two convolution layers. This model has 23 layers in total.

2.2.4 Determination of nanofiber diameters by image recognition

Some researchers have been trying to combine machine learning with nanofiber characterization. Most of them are for defect detection and distinguishing the homogenous and nonhomogeneous nanomaterials. For example, Matson et al. (2019) proposed a machine learning approach to distinguish between the transmission electron microscopy images of carbon nanotubes and carbon nanofibers. A dataset of 5000 images was built for model training. The neural network structures they used are ResNet50 and VGG16. They came up with a new structure by combining the VGG16 with hyper-column representation and testing the accuracy of this structure. Regarding the classification accuracy, ResNet50 has the highest accuracy on identifying carbon nanofibers and clusters; VGG16 with hyper-columns has the highest accuracy on carbon nanofiber matrix, and VGG16 has the highest accuracy on non-carbon nanotubes. Overall, VGG16 with hyper-columns has the highest total accuracy.

Leracitano et al. (2021) proposed an automated classification system based on hybrid unsupervised and supervised machine learning for electrospun nanofibers. His study aims to distinguish between the homogenous and non-homogenous nanofibers, which are interpreted as two different categories. This study builds and implements a network that combines the autoencoder and multilayer perceptron. They used 160 SEM images for the dataset buildup. This model achieved an accuracy of over 90% and outperformed other similar algorithms such as support vector machine and linear discriminant analysis.

Napoletano et al. (2018) proposed an approach for nanofiber defect detection using CNN. This study uses ResNet18 for feature extraction and defect detection. The accuracy is over 90%, outperforming a comparative algorithm by about 5%.

Others attempted to use machine learning figure out the relationship between the fabrication parameters and nanofiber characteristics. For example, Oflza et al. (2021) used the extreme gradient boosting model to predict the diesel oil sorption capacities of nanofibers and determine the optimal conditions.

To my best knowledge, however, no research has been conducted to determine the nanofiber diameter using machine learning. The machine learning approach may allow complete abandoning of complex image processing steps in former non-machine learning approaches. It determines the diameters of nanofibers in SEM images in one single step. This thesis project uses the ResNet for machine learning, which is elaborated as follows.

ResNet stands for deep residual neural network. This network is well known for its unique skip connection, which is illustrated using Figure 2-2. The skipping connection behaves like an identity mapping in the structure and ensures that the result does not worsen as the network layer increases. These skip connection structures can effectively prevent the gradient vanishing problem and enable extremely deep neural network (He et al., 2015). From the perspective of gradient calculation, the skip connection brings additions to the gradient calculation. Thus, the gradient is composed of multiplications, preventing the vanish of the gradient.

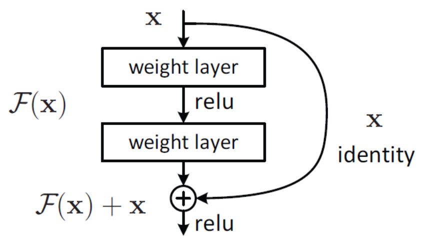

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2-2. The structure of ResNet (He et al., 2015)

Table 2-2 explains the neural network structure of ResNet50. This structure can be divided into five stages as listed in the first column, and each stage may contain multiple layers of the network. A point worth emphasizing is the bottleneck structure used in the ResNet50. A 1×1 convolutional layer is applied before and after each 3×3 convolutional layer in the block. These 1×1 convolutional layers are used to reduce the number of channels and therefore reduce the parameter that needs to be trained in the 3×3 convolution. Another 1×1 convention layer is placed after the 3×3 convolution to restore the channel number, and the consequent output have the same dimension as the input does. By reducing the number of training parameters, this structure can significantly reduce the computation time and hardware requirements for training. When visualizing the network structure, the 1×1 convolutional layer is smaller than other features, extracting convolutional layers placed near it and giving the neural network a bottleneck shape. That is why this is called the bottleneck structure.

Table 2-2. The Structure of ResNet50

Layer name	**ResNet50**
conv1	7×7, 64, stride 2
conv2_x	3×3 max pool, stride 2
	$\begin{bmatrix} 1\times 1, & 64 \\ 3\times 3, & 64 \\ 1\times 1, & 256 \end{bmatrix} \times 3$
conv3_x	$\begin{bmatrix} 1\times 1, & 128 \\ 3\times 3, & 128 \\ 1\times 1, & 512 \end{bmatrix} \times 3$
conv4_x	$\begin{bmatrix} 1\times 1, & 256 \\ 3\times 3, & 256 \\ 1\times 1, & 1024 \end{bmatrix} \times 3$
conv5_x	$\begin{bmatrix} 1\times 1, & 512 \\ 3\times 3, & 512 \\ 1\times 1, & 2048 \end{bmatrix} \times 3$
Output layer	Average pool, 1000-d fc, Softmax

Transfer learning, as the name indicates, is the process of transferring knowledge from one task that has already been learned to another (Torrey and Shavlik, 2010). In the transfer learning, a base network is first trained on a base dataset and then repurposing, or transferring, the learned features to a second task. The features that are general enough to be suitable for both tasks rather than task-specific will tend to work in this method. When training the model for the second task, the pre-trained model can vastly reduce the calculation amount and time.

Transfer learning becomes quite necessary for a structure like ResNet50, which has 3.8×10^9 parameters.

In this thesis work, the dataset used for model pre-training is the ImageNet dataset (Russakovsky et al., 2015). The ImageNet dataset is a massive dataset for image classification. This dataset contains millions of images belonging to thousands of different categories. It was originally used for an image classification competition called ImageNet Challenge. After that, as the scale of the competition continued to expand, the sample of the database became more and more complete. The ImageNet has gradually become a common benchmark in the field of image classification.

2.3 Nanofiber Porosity and Determination

2.3.1 Existing approach for porosity determination

The porosity of a nanofiber mat can be determined experimentally or by image processing. The technologies used in experimental studies includes mercury intrusion porosimetry, liquid extrusion porosimetry, and capillary flow porometry. Alternatively, the porosity can be determined by image processing. In most earlier studies, the characterization by micrographs is mainly used as a preliminary characterization technique, but there are still a few attempts to determine the porosity by image processing.

The mercury intrusion porosimetry is a conventional technique for porosity determination. In this method, pressure is applied to a mercury column, and the mercury is pressurized into a sample. With the increasing pressure, mercury intrudes the pores in the sample. The total pore volume of the sample can be obtained from the volume of mercury that intruded into the sample. Note that both inter-fiber pores and intra-fiber pores are characterized during the measurement. An equivalent pore size can also be calculated by the Washburn equation, which relates the applied pressure to pore size using the physical properties of the non-wetting liquid. However, this technique is destructive because up to 60,000 psi can be applied to the sample. The high pressure applied also brings the possibility of the sample structure being distorted (Širc et al., 2012).

The liquid extrusion porosimetry is developed from the mercury intrusion porosimetry. The main difference between liquid extrusion porosimetry and mercury intrusion porosimetry is the type of liquid used for testing -- The non-wetting liquid mercury is changed into wetting liquid, and a much lower pressure is required for liquid extrusion porosimetry. Regardelss of the simial the measurement principles, there is a particular gap between the results obtained from the two techniques. The liquid extrusion porosimetry is likely to underestimate the sample's pore volume and pore size because of a low pressure used, and it normally gives a result smaller than that by mercury intrusion porosimetry.

The capillary flow porometry is a technique evolved from the liquid extrusion porosimetry, and their principles are similar. In this technique, the sample is first saturated with a liquid. Then, inert gas is used to displace liquid from the saturated sample. The air pressure at which liquid is pushed out and the gas can flow through is called the *bubble point*. With the obtained air pressure, the porosity and pore size of the nanofiber can be calculated using the Young-Laplace equation (Barhoum et al., 2019).

Regarding image processing, the nanofiber image used for analysis is a SEM image. A SEM image is a grayscale image, which only has different shades of black and white. In the SEM image, there are 256 gray scales, numerically represented by integrals from 0 to 255, with 0 and 255 representing pure black and pure white, respectively. A fiber in the SEM image normally has a bright color, and the background has a much dark color.

Many earlier studies on determining nanofiber porosity by image processing follow these steps: First, the SEM image is segmented into a binary image, which is purely black and white. Then, all black and white pixels are regarded as background and fiber, respectively. Finally, the number ratio of the black pixels to total pixels is calculated and considered as porosity. The difference between different studies is mainly in the first step, image segmentation, using various algorithms. The most reasonable method for image segmentation still to be determined.

Ghasemi-Mobarakeh et al. (2007) calculated the porosity of a nanofiber mat from the corresponding SEM image. In their method, the global threshold is used for image segmentation. All pixels with grayscale smaller and larger than the threshold are considered background and fiber pixels, respectively. Three different values are used in their research as global thresholds: the average grayscale, average grayscale plus standard deviation, and average grayscale minus standard deviation. They believe that segmenting the image using average grayscale plus standard deviation as a global threshold gives the porosity at the upper layer of the nanofiber. Using average as global threshold can obtain the middle layer's porosity of nanofiber. Using average minus standard deviation as a global threshold can obtain the porosity of nanofiber in the lower layers.

Wang et al. (2020) also used the global threshold method for image segmentation for porosity determination. The global threshold they chose for image segmentation is 85% average grayscale. By comparing the porosity result calculated from the image processing and mercury intrusion methods, they found that the values of these two results satisfy a particular relationship. The ratio of the two is the second Feigenbaum constant.

Other researchers, such as Sun et al. (2007) and Wang et al. (2018), determine the threshold for image segmentation by human observation and subjective judgment. Sun et al. (2007) state that the brightness and condition of different SEM images vary, and no algorithm is robust enough to segment all images automatically. In addition, their porosity result obtained from image processing is larger than measurement.

The deviation between experimental and image processing results prevails in the literature, regardless of the approach. For example, Tomba et al. (2010) used multivariate image analysis (MIA) to analyze the SEM images of nanofibers. They examined pore relative open area (ROA), defined as the ratio between the pore open area and the total area. Dierickx (1999) also reported that there should be a linear relationship between the ROA and the porosity. However, they did not mention whether this claim matched their results.

Others used a similar procedure for image processing, and the results were regarded as fiber compactness (Ganjkhanlou et al., 2014). This calculation is reasonable to some extent. However, packing density is more commonly used to measure the compactness of nanofibers. There are generally two methods for packing density determination. Liang et al. (2019) used the polymer density and nanofiber mass to calculate the packing density, using Eq. (2-1. Another is a method called the impregnation method (IM) (Charvet et al., 2018).

$$\alpha = \frac{W}{\rho_f Z} \tag{2-1}$$

where α is the nanofiber packing density, W is the basis weight of the nanofiber mat, ρ_f is the density of nanomaterial, and Z is the nanofiber film thickness.

The thickness of a nanofiber mat can be measured using the laser trigonometry method (LTM) (Charvet et al., 2018). The LTM determines the fiber thickness by measuring the displacement of the same laser between two different images (Ribeyre et al., 2017).

Impregnation is another method to determine packing density. In this method, the nanofiber is first embedded in an epoxy matrix. Then, the SEM image of the nanofiber cross-section is taken. Finally, the SEM image is binarized, and the ratio of black and white pixels is plotted along with the fiber thickness. This profile extracts the thickness and the mean packing density (Bourrous et al., 2014). This thesis project adopts the same basic principles of the impregnation method (IM) for image processing. However, the IM combines experimentation with image processing. Compared to the total image processing method, the IM can obtain more rigorous results and the steps are more complicated.

2.3.2 Limitations of the existing porosity determination by image processing

Although using image processing for nanofiber porosity sounds plausible, this approach has some fundamental challenges. No matter how thin the nanofiber is, it is a three-dimensional (3D) structure. All attempts to analyze three-dimensional space with two-dimensional (2D) images have limitations and are based on the ideal assumptions that 2D porosity is equal to 3D porosity.

Admittedly, an experimental approach is still the most accurate method for porosity determination. The experimentally measured porosity combines the void in both inter-fiber space and intra-fiber space, while the image processing approach overlooked the influence of intra-fiber pores. On the other hand, the image processing approach offers a fast and low-cost estimation, which can be used on solid and smooth nanofibers.

2.4 Nanofiber Surface Pore Size Determination

2.4.1 Surface intra-fiber pores

The pore sizes *in* nanofibers are crucial to some applications. For example, when nanofibers are used in the lithium-ion battery as the separator, numerous small pores in the fibers can enhance channels for lithium-ion, accelerating the mobility of lithium-ions. Therefore, nanofibers with small pores enhances the performance of a battery (Zhang et al., 2014). Another example is using the nanofiber for air filtration. The sizes of the nanofibers in an air filter affects the filtration effectiveness because the pore sizes influence the way particles travel through the fibers. Furthermore, nanosized pores can promote the slip effect when air flows through and reduce the air resistance. By optimizing the pore sizes, the pressure drop through an air filter can be vastly reduced (Zhao et al., 2016).

The methods for characterization of pores in nanofiber can be generally divided into two categories, experimental and image processing methods. The experimental method for measuring nanofiber pore sizes is basically the same as measuring porosity. Techniques such as BET, mercury intrusion porosimetry, and liquid extrusion porosimetry can also be used for pore size determination. However, they can only provide theoretical pore sizes with simplifications besides time-consuming and tedious operations. Alternatively, image processing may offer a fast, and possibly accurate, estimation.

Despite many limitations in experimental measurement of nanofiber diameter, they remain the most frequently used for their wide acceptance and validity. Nonetheless, the image processing method is more frequently used to determine the pore size in some fields such as mud cake deposition (Rabbani & Salehi, 2017). A mud cake is a layer of solid particles left behind after mud passes through porous rocks during drilling. A schematic for mud cake is shown in Figure 2-3. Similarly, it is called dust cake in air filtration using baghouses in power plants (Tan, 2014).

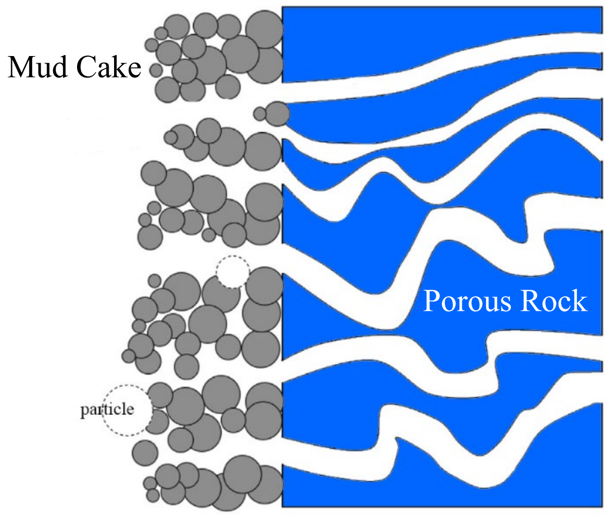


Figure 2-3. Schematic for mud cake

Rabbani et al. (2017) determined the porosity of a mud cake by image processing. They first used the Otsu multi-level threshold on the image, followed by a series of noise reducing and contrast adjustments. Then the watershed algorithm is used to divide the overlapped pore boundaries on the image. After that, each pore is separated and labelled, the pore size is calculated as the radius of the equivalent circles with the same area, and the equivalent area of the pore is determined. Then, the porosity is calculated by dividing the total area of all the pores over the total area of the mud cake. The algorithm in this paper indicates that a 3D porosity can be estimated using the 2D planar porosity of the mud cake. Admittedly, this may introduce some errors, but it is the considered the best technology possible. Similar image processing approach was also used for analyzing the pore opening size on nonwoven geotextiles (Aydilek et al., 2002).

2.4.2 Existing image processing approach

Several image processing approaches have been reported in the literature for nanofiber pore size determination. Most of these approaches start with binarization to convert the grayscale images into black and white only. Then the pore sizes are determined by the pixels, which are converted into nanometers, and the resulting sizes are presented in terms of area-based equivalent diameters. Area-based equivalent diameter is also a common practice in aerosol technologies (Tan, 2014).

The final equivalent diameters of the pores depend on the method for binarization, which differentiates the existing image-processing approaches. The most sketchy and straightforward binarization method binarizes the image using a global threshold. Regarding software to process the images, *ImageJ* is widely used. For example, Yanilmaz et al. (2014) reported the pore sizes using ImageJ, which allows manual adjustment of the threshold for binarization. Therefore, the threshold is a subjective variable in this type of image-processing. Consequently, the results are subjective and deviations may be introduced in this step.

Ziabari et al. (2008) proposed another image processing approach to pore sizes of electrospun nanofibers. They used the Otsu local threshold for image segmentation. The Otsu approach, with reasonable theoretical justification, reduces the interference of uneven brightness of the image.

Tomba et al. (2010) proposed a more complicated approach for pore size determination. They added a series of de-nosing steps to pre-process the images and used a multivariate image analysis (MIA) system for image binarization.

2.4.3 Limitation of the pore size determination

Despite the values of image processing for the determination of nanofiber pore sizes, it also has some fundamental challenges. First, it is still a 2D approach. Regardless of thickness of a nanofiber mat, it is essentially a 3D structure. Thus, using 2D images to represent 3D structures inevitably introduces deviations and reduces the accuracy of the results. The reasons behind the inaccuracy are different when image-processing is used for inter-fiber and intra-fiber pores. For inter-fiber pores, the limitation mainly comes from the stacking of nanofibers. For intra-fiber pores, the limitation mainly comes from the curve surface of nanofiber and the angle of the camera.

2.5 Surface Homogeneity

2.5.1 Existing methods of determining surface homogeneity

Surface homogeneity can be considered as thickness uniformity, which is an essential characteristics of nanofiber mats. The thickness uniformity can greatly influence the properties and performance of a nanofiber mat, depending on the application.

The surface homogeneity can be measured by a micrometer screw gauge, visualized by cross-section imaging, and quantified with light transmittance. Many engineering applications prefer nanofibers with greater strength, which often means a greater thickness. If the nanofiber thickness reaches the order of millimeters, its thickness can be measured by a micrometer screw gauge. Measurements taken at different points on a nanofiber mat can be used to estimate the average thickness of the nanofiber mat as well as the surface homogeneity using. Using the micrometer screw gauge is a simple method, but it can only provide a rough estimation. In addition, the pressure applied on nanofiber during measurement may cause structural distortion and damage to nanofiber mat at the thinnest (or weakest) spots.

Alternatively, the cross-section imaging can be used for nanofiber surface homogeneity determination. Both SEM and transmission electron microscope (TEM) can generate cross-sectional images of nanofiber mats. The cross-sectional images can not only show the inner structures of nanofiber mats, but also enables the determination of the thickness along the cross-section. This approach offers much more comprehensive information than single-point measurement because the images visualize the variation of thickness along the cross-section. However, there is a possibility that the selected cross-section does not accurately reflect the nanofiber homogeneity, and biases due to sampling still exists.

The light transmittance provides more information on the nanofiber surface homogeneity because it is applicable to the entire nanofiber mat. Examining the surface homogeneity through light transmittance is a common method in the paper industry. Chhabra (2003) proposed a similar method to measure the uniformity of nonwoven fibers. This method focuses on the spatial variation of nonwoven fiber uniformity by comparing the grayscale differences in the image. The parameter calculated for uniformity description is the variance-to-mean ratio of the grayscale. Chhabra (2003) referred to this parameter as a standardized index of dispersion. In addition, the variance-to-mean ratio on the downward direction and sideward direction are calculated separately and compared to offer more information about the nonwoven fiber.

However, very few attempts have been made to implement this technology for nanofiber mats. Nonetheless, Ryu et al. (2020) developed a real-time nanofiber thickness measuring system using light transmittance for uniform-thickness electrospun nanofiber mats. They added bar LEDs, a light diffuser panel, and a CCD camera to a conventional electrospinning device for real time thickness measurement. The light emitted from the LED lamp successively passes through the light diffuser panel and electrospun nanofiber. The CCD camera placed behind the electrospun nanofiber captures the light transmitted. The light intensity of pixels in the CCD camera image is measured and converted to thickness following the Beer-Lambert law, Eq. (2-2.

$$T = e^{-at} \tag{2-2}$$

where *T* is the light transmittance, *a* is the extinction coefficient, and *t* is the thickness of the nanofiber mat. Practically speaking, the calculated nanofiber thickness homogeneity can be used to guide the collector position to ensure uniform nanofiber mats are produced.

The advantage of this design is that its real-time measurement of fiber thickness during electrospinning and that the measurement is non-invasive. However, this approach requires expensive equipment, precise assembly, and professional operation. It also assumes that each measurement of the nanofiber surface homogeneous is accurate, which may not be the case in reality (Ryu et al., 2020).

Alternatively, some researchers quantify the uniformity as web mass variation (Bresee & Daniluk, 1997) and the web mass variation is regarded to correlate with grayscale variation. The coefficient of variation is used as a measure of web uniformity.

CHAPTER 3. DETERMINATION OF NANOFIBER SIZES USING MACHINE LEARNING AND MATLAB

3.1 Multi-Image diameter determination tool

This thesis reports an automated multi-image processing tool developed for the characterization of nanofibers by processing scanning electron microscopy (SEM) images. The basic procedure is inspired by the works of Shin et al. (2008), Givehchi (2016), Givehchi et al., 2016), *etc.* This thesis work contributes to the field by two improvements, one is the ability to simultaneously process multiple images, and another is the optimized denoising procedure. This tool was developed using MATLAB, and the code is in Appendix A.

The first improvement is to increase the number of images that can be processed at the same time. This feature allows end users to simultaneously process multiple SEM images with different magnifications and to combine the results for accurate presentation. Our research group has realized that the accuracy of calculated nanofiber diameters depends on the magnification of the SEM image used for image processing. Normally, the SEM image with a higher magnification gives a smaller mean diameter with a smaller standard deviation, and *vice versa* (Raheleh, 2016). Thus, more statistically reliable data can be obtained by combining the results from the images with different magnifications. After all, combining the results of different images still offers a larger sample area and more sampling points, leading to a more accurate result, even if the SEM images of the sample are obtained under the same magnification.

The second improvement is to optimize the denoising step. The earlier works only use linear filtering, that is, the median filter on the original SEM image for denoising. Compared to linear filtering, adaptive filtering is more selective and can preserve edges and high-frequency parts better (Chen et al. 2006). A Wiener filter is applied adaptively to the image to achieve a better result.

An example of SEM original image is shown in Figure 3-1. In my approach, the image processing follows this procedure shown in Figure 3-2. It consists of the following ten steps, which are explained one by one after Figure 3-2.

1. Read scale and crop
2. Linear filtering and adaptive filtering for denoising
3. Histogram equalization
4. Opening and closing
5. Binarization
6. Main de-nosing
7. Edge detection
8. Dilation and thinning
9. Skeletonization
10. Distance transformation

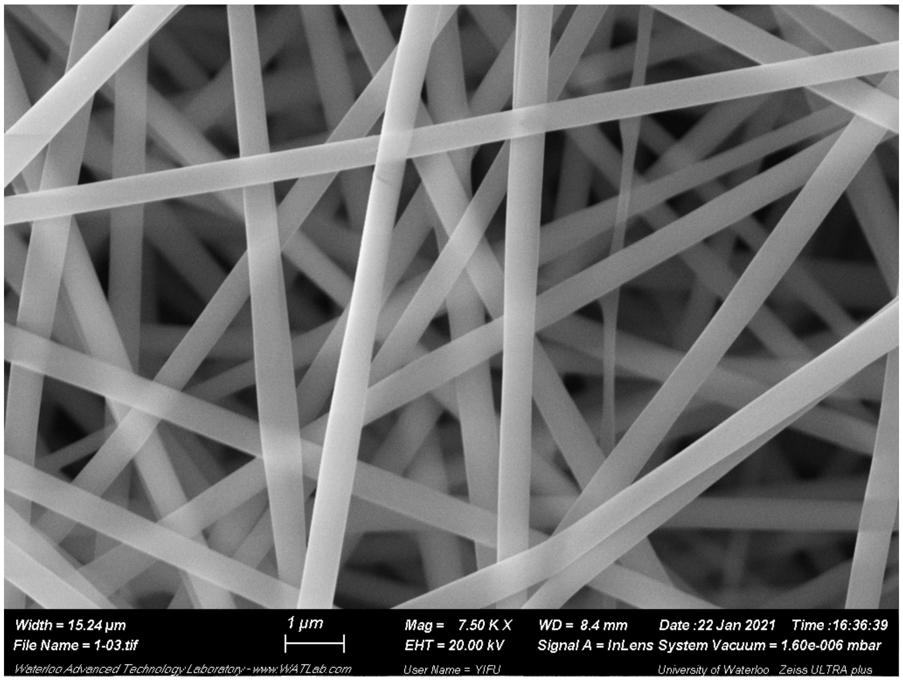
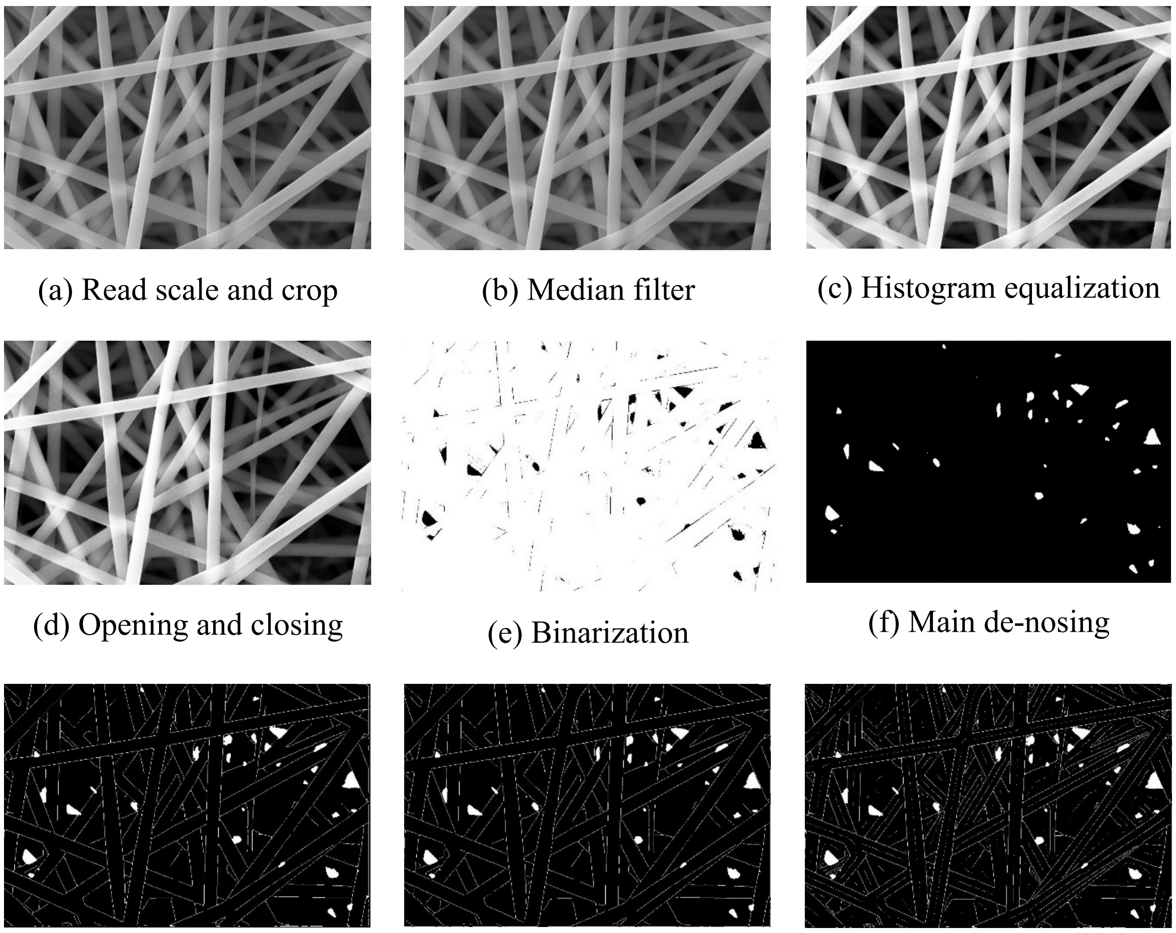


Figure 3-1. An original SEM image



(a) Read scale and crop (b) Median filter (c) Histogram equalization

(d) Opening and closing (e) Binarization (f) Main de-nosing

(g) Edge detection (h) Dilation and thinning (i) Skeletonization

Figure 3-2. The image processing procedure for diameter determination

1) Read scale and crop

The bottom par to the original SEM image (*e.g.* Figure 3-1) contains the date, user, institution, setting, scale, and so on. After importing the original image, MATLAB automatically creates a coordinate system with the upper left corner of the image as the origin. The scale is located near coordinate in the image. First, the scale part of the image is cropped into a small sub-image with 90 pixels wide and 30 pixels high. After that, a label matrix that contains labels for the 4-connected objects in the sub-image is returned. Since the sub-image is a binary image with only one object, the scale can be distinguished from the background. Then, the coordinates of the pixels that make up the scale are obtained. The difference between max X coordinate and min X coordinate is calculated, and the result is the number of pixels the scale occupies in the image. By doing so, the pixel grid length of the scale is read. The length represented by the scale bar in reality will be manually entered in nanometers. Finally, the image is cropped into an image with dimension 1024×680 pixels to remove the title bar from the original image.

Figure 3-2a is acquired after all these steps.

2) Linear filtering and adaptive filtering for denoising

Linear filtering and adaptive filtering are performed for initial noise reduction. The linear filtering performed is a median filter, and the adaptive filtering performed is a Wiener filter. The main idea of a median filter is to replace a pixel with the median value of its adjacent pixels. The Wiener filter minimizes the mean square error among the pixel intensities.

Figure 3-2b is obtained after this step.

3) Histogram equalization

Histogram equalization is performed to increase the contrast in the images. A histogram is the graphical representation of the grayscale intensity distribution of pixels in grayscale images. By performing histogram equalization, the intensity peak is spread out, and objects in the image become more prominent.

Figure 3-2c is acquired after this step.

4) Opening and closing

Opening and closing are performed for further noise reduction. Opening and closing are basic operations used for denoising and smoothing contours. An opening contains an erosion operation followed by a dilation operation, while the closing is the process in which dilation operation happens first and erosion happens afterwards.

Figure 3-2d is obtained after the opening and closing.

5) Binarization

Binarization is performed to segment the fiber from the background. Image binarization is the process of converting a grayscale image into a binary image. This process greatly reduces the information contained in the image from 256 shades of gray to 2 shades, black and white. The result depends on the binarization algorithm.

In this thesis work, preliminary trials were conducted to test some common binarization algorithms for binarization. The results show that the diameter results actually is insensitive to binarization algorithm. Thus, commonly used binarization methods such as the Otsu method can be used in this step. In this thesis study, the Sauvola local threshold method is used for consistency because our research group has been using this approach (*e.g.,* Givehchi, 2016; Givehchi et al., 2016). The Sauvola local threshold allows retrieval of information from a 4 by 4 pixel grid, which is used to determine a localized value for the binarization.

The sample binarization result is shown in Figure 3-2e.

6) Main de-nosing

With the binary image, a series of operations are performed for future denoising. In addition to the opening and closing that have already been performed, the *'majority'* and *'clean'* operations built in MATLAB are also performed. After the *'majority'* and *'clean'* operations, all graphics on the image with an area less than 3×3 pixels are eliminated.

Then the image color is inverted to get Figure 3-2f.

7) Edge detection

The Canny edge detection is used to detect the fiber boundaries. The Canny edge detector was developed by John F. Canny in 1986 and it detects edges in images using a multi-stage algorithm (Sahir, 2019). The Canny edge detection uses the intensity gradients of pixels to determine the boundaries of objects. In simple terms, it regards the vertical direction of the direction in which the pixel intensity gradient changed the most as the edge.

Figure 3-2g is obtained after this step.

8) Dilation and thinning

The Dilation operation is used to seal the small openings on the detected edges (MathWorks, 2022a). The Dilation operation give an output in which the output pixel is the maximum value of all pixels in the neighborhood. Then, the thinning operation is performed to reduce the thickness of the boundary. This step helps fix the edge profile and avoids inaccurate thickening.

Figure 3-2h is obtained after this step.

9) Skeletonization

In the image processing field, the skeleton of an object can be considered as a one-pixel thin version of the original shape, which is equidistant to the original boundaries (Aslan et al., 2008). After the skeletonization, the resulting skeletons of nanofibers are located right in the middle of the fiber with the equal distance to both boundaries. Therefore, the skeleton of the fiber can be considered as the centerline of the fiber. With the identified centerlines, the *pruning* operation is then performed to delete sporadic branches at the ends of the skeletons.

Figure 3-2i is obtained after this step.

10) Distance transformation

Three commonly used distance measurement algorithms for image processing are the city block distance transformation, the chessboard distance transformation, and the Euclidean distance transformation (MathWorks, 2022b). They are briefly introduced as follows.

The city block distance transformation calculates the length of the path between the pixels according to the four-connected neighborhood. The path only moves horizontally and vertically, and one pixel is the smallest unit of distance. Pixels whose edges touch are one unit apart; pixels diagonally touching are two units apart. Equation (3-1) can be used to calculate the city block distance (MathWorks, 2022c).

$$D_{CB} = (|x_1 - x_2| + |y_1 - y_2|) \tag{3-1}$$

where D_{CB} is the distance between two pixels calculated by the city block algorithm; (x_1, y_1) defines the location of pixel 1 and (x_2, y_2) defines the location of pixel 2.

The chessboard distance transformation calculates the distance between the pixels based on the eight-connected neighborhood, which means that the path can move horizontally, vertically, and diagonally. One pixel is the smallest unit of distance. The pixels that edge or corner touch are one unit apart. Equation (3-2) can be used calculate the chessboard distance (MathWorks, 2022c).

$$D_{Ch} = Max(|x_1 - x_2|, |y_1 - y_2|) \tag{3-2}$$

where D_{Ch} is the distance between two pixels calculated by the chessboard algorithm.

The Euclidean distance transformation calculates the length of the straight line between two pixels, which means the path can go across the pixels, and the smallest length unit can be smaller than one pixel. Equation (3-3) is for calculating the Euclidean distance.

$$D_{Eu} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{3-3}$$

where D_{Eu} is the distance between two pixels calculated by the Euclidean algorithm.

Among all the preceding methods, the Euclidean distance transformation is the most accurate one (Wang, 2002), and therefore it is chosen for this thesis research project.

3.2 Diameter determination using machine learning for image segmentation

3.2.1 UNet model

The UNet model is used for the segmentation of the SEM images, followed by central line determination and distance transformation for diameter determination. The UNet model has received great attention since it was first reported by Ronneberger et al. (2015). The related literature has been cited tens of thousands of times to date. This model was initially designed for image segmentation in the biomedical field, but follow-up studies have extended its applicability.

The network architecture of UNet is illustrated in Figure 3-3 (source: Ronneberger et al., 2015) The most notable feature of this network is its U-shape structure. The left side of the U shape is a contracting path consisting of a series of convolutional layers. The right side is an expansive path consisting of a series of up-convolutional layers. There are also four concatenations between the contacting path and the expansive path.

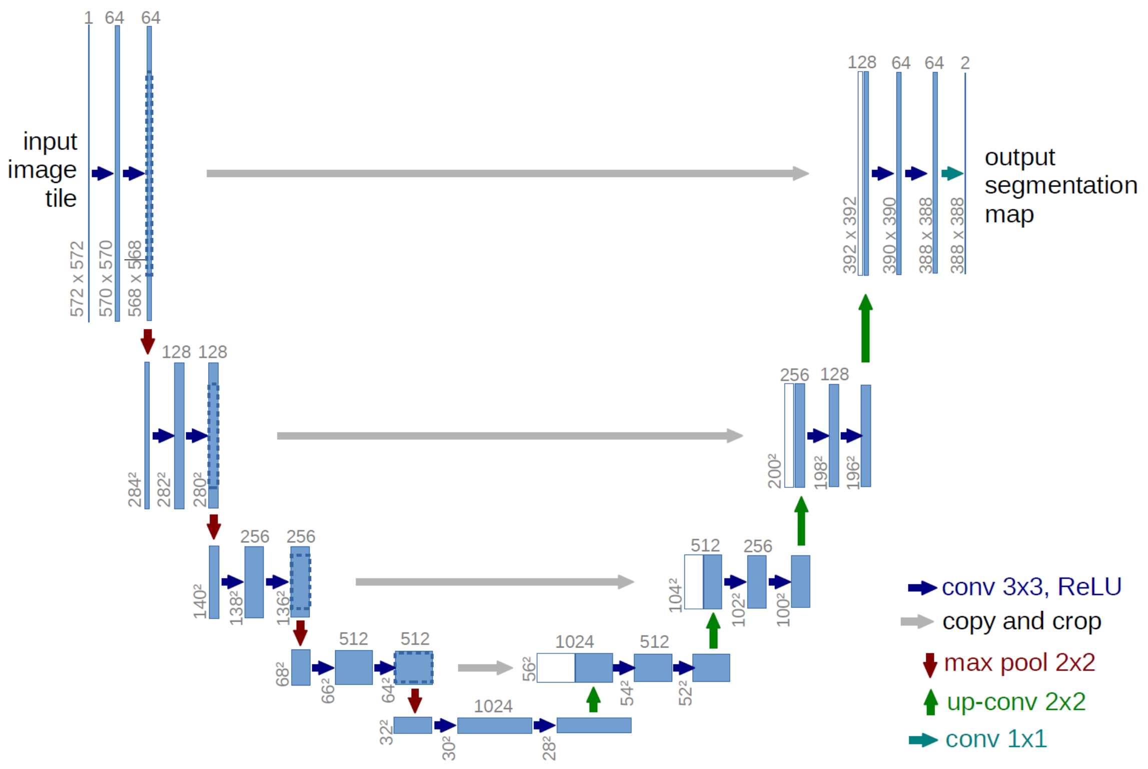


Figure 3-3. The structure of U-net.

The contracting path is composed of a series of typical convolutional networks. This path contains four convolutional steps. Each step contains two convolution layers and a max pooling layer. In each step, a 3x3 convolutions kernel is first applied twice, each followed by a rectified linear activation function (ReLU). This convolutional operation is unpadded and results in the loss of boundary pixels. After two convolutional layers, a 2x2 max pooling operation with stride two is applied to the feature map. The number of feature channels is doubled after each down sampling step.

As for the expansive path, each up-sampling step of the feature map is followed by a 2x2 up-convolution layer that halves the number of feature channels. ReLU is used as activation function. A SoftMax layer is placed after all neural network layers.

In the original UNet model reported by Ronneberger et al. (2015), border pixels may be lost during each convolution. Therefore, it is necessary to crop the image into the desired size before the concatenation operation. There are 23 layers in this network. The loss function used is the cross-entropy function. Apart from the network structure, overlap-tile strategy (Tsang, 2018) is also used in this model for a better performance. When predict a pixel in the image, the surrounding pixels are also considered to provide environmental information for enhanced accuracy. For this reason, padding is done by preprocessing to expand the original image size.

In this thesis project, the implementation of the UNet model is not exactly the same as originally reported by Ronneberger et al. (2015). Unlike medical image segmentation, the images segmented in this project are post-processed to determine the diameter of the nanofiber. The result of diameter is highly related to the number of pixels in the image. A smaller overall size of the image results in a smaller diameter too, deviating from the fact.

To minimize the deviation mentioned in the preceding paragraph, padding is added to each convolutional step in this thesis work. Therefore, the pixels lost during the convolution will be complemented by zeros, and the size of the image remain unchanged during the convolution operation. The size of the output image then equals that the input image. Batch normalization (Ioffe & Szegedy, 2015) is added between the convolution operation and the ReLU activation function (Agarap, 2019).

I implement the UNet code on the SEM images of nanofibers collected in our research group to segment the nanofibers from the background. I changed the padding method of the original model, but I did not make any major change to the original neural network structure. *See also* Section 5.2.1 for recommended future works.

3.2.2 Procedure for implementation of UNet model

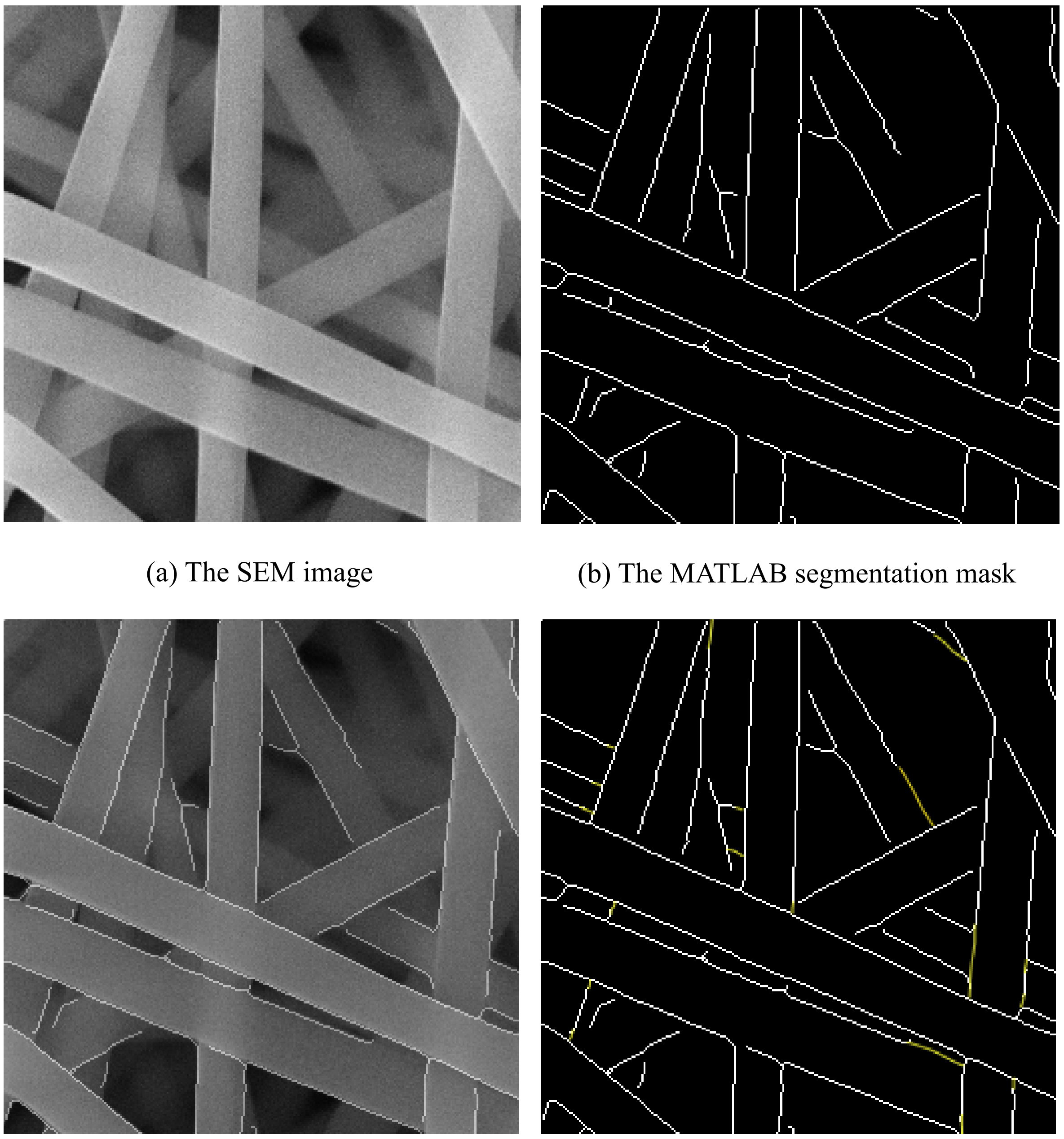
The implementation of machine learning comprises of three steps: build the database, run the model, and tune the hyperparameter. They are introduced one by one as follows.

1) Dataset build-up

The nanofiber samples were prepared in our research lab by other graduate students studying electrospun nanofibers for energy and clean air applications. Data presented in this thesis project include those from the same research group (*e.g.*, Ahne et al., 2018; Li et al., 2019, 2020, 2021; Givehchi et al., 2016, 2018, 2021). All electrospun nanofiber samples are polymer-based, including PAN and PVP.

The SEM images are prepared as follows. After removing the title bar (*i.e.*, the word description under the main image), the size of SEM images becomes 600×1000 or so. The original SEM images come in different sizes because they were taken at different times using different equipment, but all images are cropped into 512×512 pixels. During cropping, the best effort has been made to ensure as many non-repeating positions as the image size allows. I wrote a Python script to quickly batch process images (*see* Appendix B for the Python code).

Segmentation masks are generated from MATLAB using the same binarization and edge detection algorithm in the diameter determination part (*see* Section 3.1), specifically the Sauvola local threshold and the Canny edge detection method. In addition, the resultant segmentation mask is manually corrected using Photoshop to ensure that the results obtained by machine learning exceed those by MATLAB. The procedure is depicted in Figure 3-4. Figure 3-4 (a) is an example of SEM image after cropping, and Figure 3-4 (b) is the segmentation mask generated using a MATLAB script that I wrote. Figure 3-4 (c) is a composite image made in Photoshop. The opacity of the MATLAB segmentation mask is adjusted to 50% and overlapped with the original SEM image to show the deviation between them and the imperfection of this segmentation result. Figure 3-4 (d) is the manually corrected segmentation mask with the changes highlighted.



(a) The SEM image

(b) The MATLAB segmentation mask

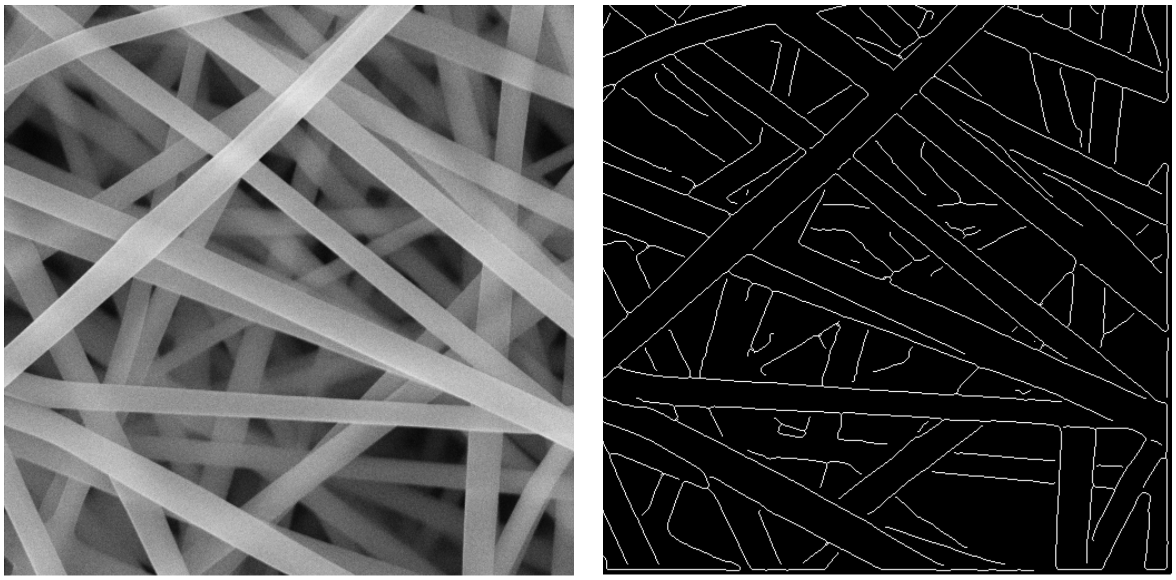
(c) Comparison of the SEM image and the segmentation mask

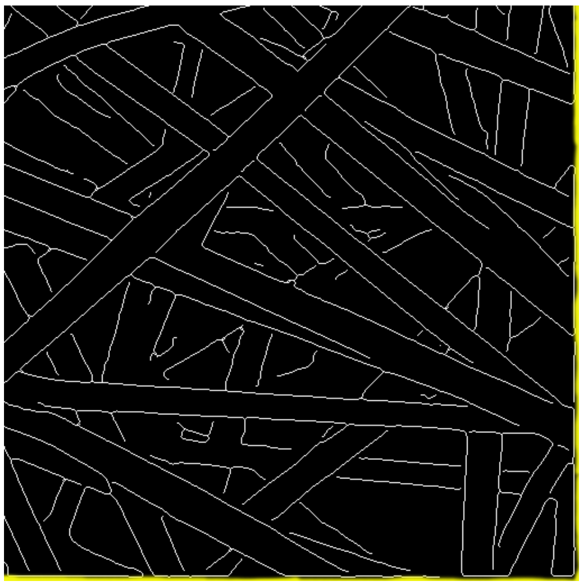
(d) The manually corrected segmentation mask with correction highlighted

Figure 3-4. Procedure for mask correction using Photoshop

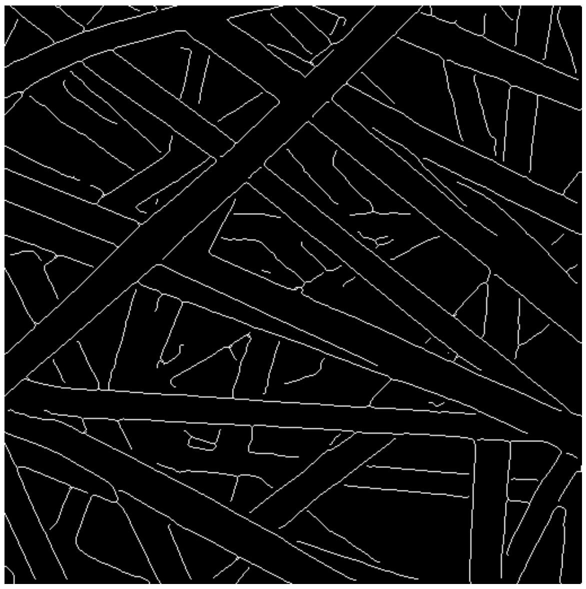
During the segmentation mask generation, mask generated by MATLAB may have a minor displacement compared to the original image. Again, the mask was generated by combining the Sauvola local threshold method and The Canny edge detection. During the Sauvola local thresholding, the image information was retrieved from the upper left corner of the image by 4×4 pixel grids. The pixels at the very upper left edge do not have enough neighbors for the algorithm to calculate the threshold value, and they are ignored during the thresholding process, resulting in a shift of 4 pixels in both left and upward direction within the segmentation mask. There is an additional step in the image processing to line up the image and the segmentation mask for further training.

As illustrated in Figure 3-5, the segmentation mask was 4 pixels downward and 4 pixels right compared to the original image. Thus, a Python script (*see* Appendix C) is used to quickly batch process the images by cropping the displacement boundary quickly.


(a) The SEM image (b) The MATLAB segmentation result



(c) Highlight of the segmentation mask displacement



(d) The segmentation mask with displacement corrected

Figure 3-5. Displacement of MATLAB generated mask

2) Implementation of UNet model

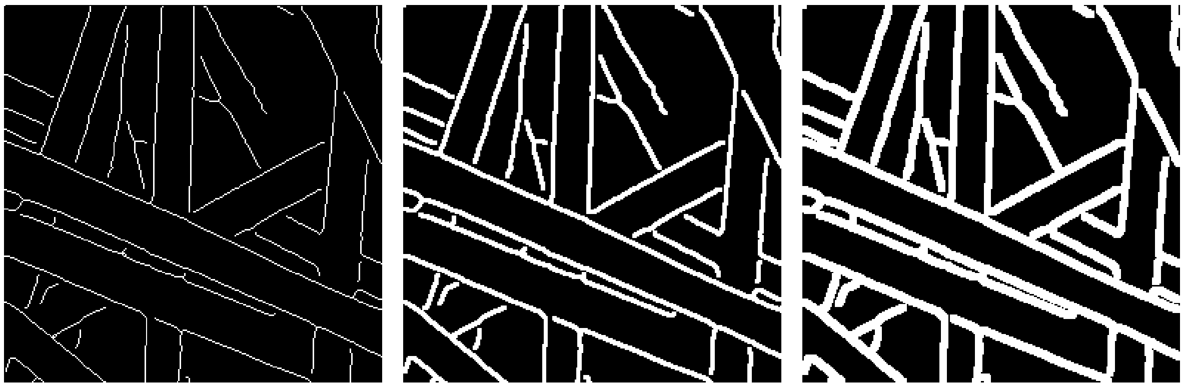
Appendix C shows the code for the UNet model. The code for the implementation of UNet is from https://github.com/zhixuhao/unet. In the first few attempts to implement the UNet model, I noticed that the model failed to generate a meaningful result. Regardless of the hyperparameter, the model did not output reasonable masks but fully black images.

To solve the problem, I made a series of efforts for trouble-shooting. The first step of troubleshooting was to identify problem -- the neural network code, the hyperparameter setting, or the dataset. To do so, I ran the UNet model using the code for a biomedical image dataset provided by Ronneberger et al. (2015) for training. When using that biomedical dataset, nontrivial segmentation results could be generated. However, with the same code and same setting of hyperparameter, the UNet model still output fully black images. Therefore, the problem was identified as in my dataset.

Then, I attempted to increase the size of my dataset. My original dataset contained only 50 images, and I increased the number to 100. However, this increment did not improve the result. Considering the dataset used by Ronneberger et al. (2015) has only 41 images, I do not believe that the amount of data is the problem.

Opting out the possibility of failure caused by the amount of data, I think the failure may be caused by the low variance of my images. The segmentation masks used for model training are basically black images with a very thin white line. Even if the picture gives a pure black image as the result, the accuracy can be as high as 90%. Therefore, I wrote a Python script (*see* Appendix F) to thicken the fiber in the image. Segmentation mask images with fibers from one pixel to five pixels wide were generated and used for the model training.

The images with thick fiber boundaries are shown in Figure 3-6. Figure 3-6 (a) (b) and (c) are images of fiber with 1-pixel wide, 3-pixel wide, and 5-pixel wide, respectively. However I still could not get a non-trivial result, even if the edge of the fiber increased to five pixels thick.



(a) Image with 1-pixel fiber (b) Image with 3-pixel fiber (c) Image with 5-pixel fiber

Figure 3-6. The segmentation mask image with different fiber thickness

Another attempt I made was to invert the image color (*see* Figure 3-7), changing the image from pure black to pure white. However, the output were still pure black images. With this, it is safe to conclude that the layout of the image is not the problem.

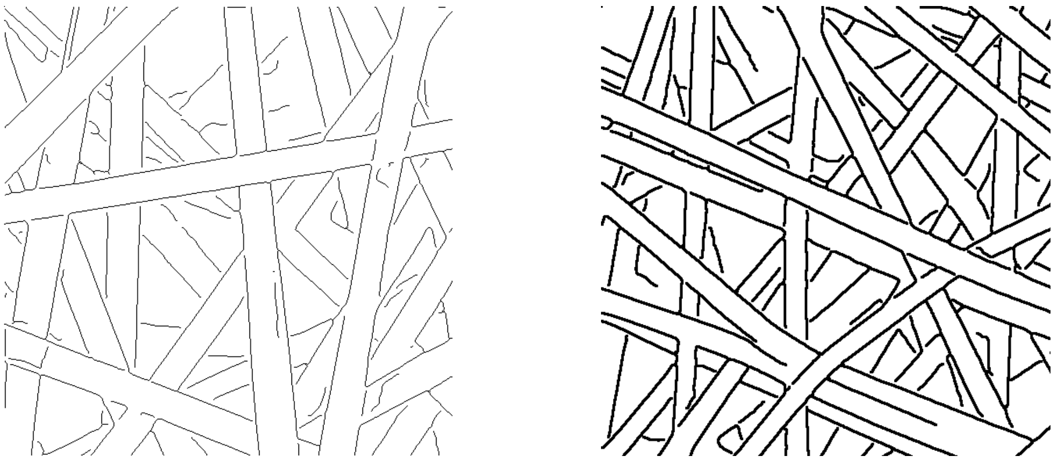

(a) Inverted color image with 1-pixel fiber (b) Inverted color image with 3-pixel fiber

Figure 3-7. The segmentation mask image with inverted color

After excluding all the preceding possible sources of problem, I believe that the problem lies in the format and properties of the images used as training dataset. The original images obtained from the experimental equipment are in tag image file format (.tif format). Then, converted them into PNG format with 8-bits bit depth using Photoshop. A bit depth of eight bits allows 256 different intensities (shades of gray) to be recorded. After this attempt, I finally got non-trivial results from the model.

3) Optimization of the hyperparameter

The learning rate, batch size, and epoch were tuned and optimized by trial and error. The loss function used for training is binary cross-entropy. The range of learning rate is from 1E-3 to 1E-5 as a best practice in the field. The range of batch size varies from 2 to 10. The epoch varies from 5 to 50 epochs. The effects of the hyperparameter are shown in Figure 3-8.

Figure 3-8 shows the effects of epoch on the accuracy and loss of the testing set. The greater the epoch number, the less the loss of the model, which indicates a smaller variance between the calculated and generated mask. However, the accuracy did not increase further, rather it decreased when the epoch number reached 25. This result shows that 25 epochs are already sufficient for the model training. Otherwise, problems such as overfitting may occur when the epoch number increases further.

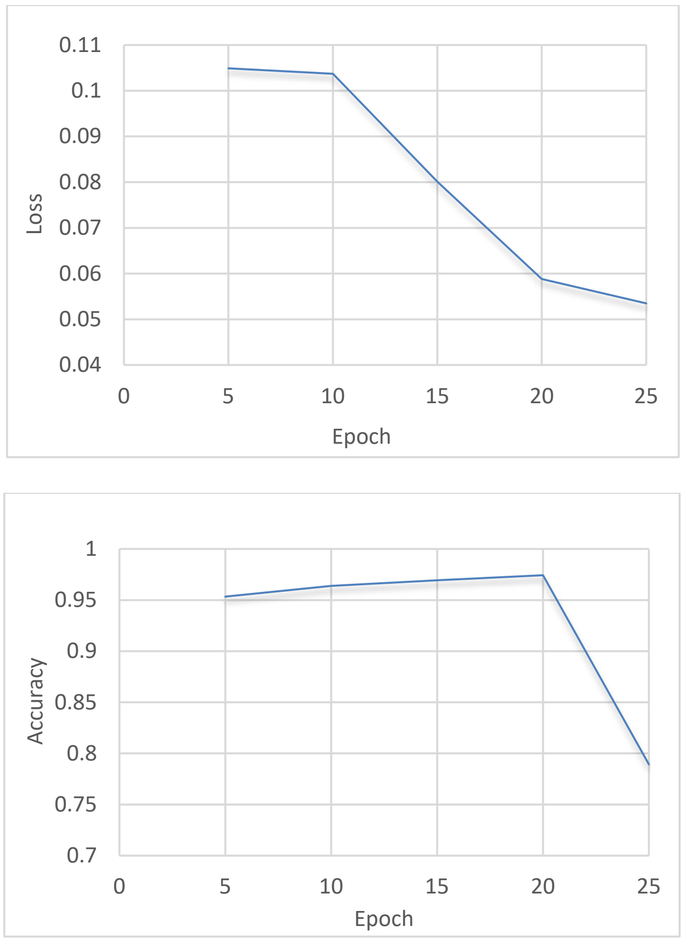
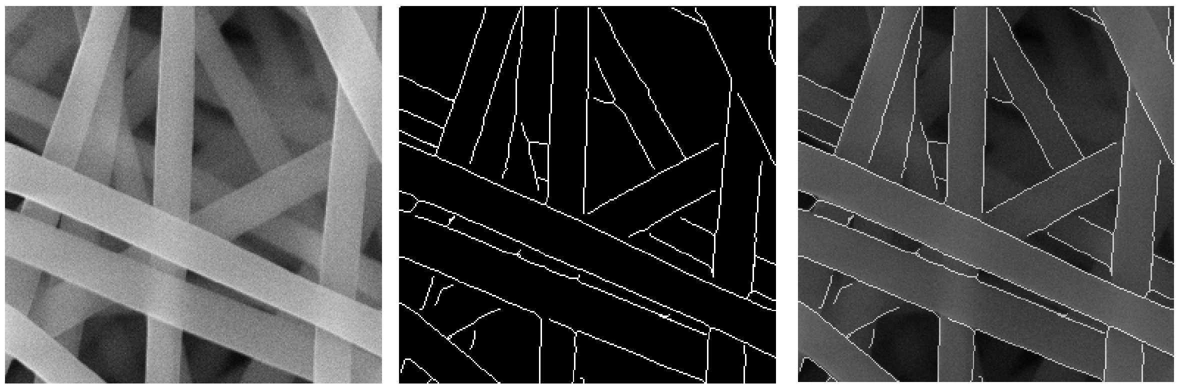


Figure 3-8. The accuracy and loss under different epochs

4) Diameter determination

The algorithm for diameter determination is the same as that explained in Section 3.1, using the skeletonization of image. The Python library used are *skimage* and *quanfima* for the image processing. Details are available in Appendix D.

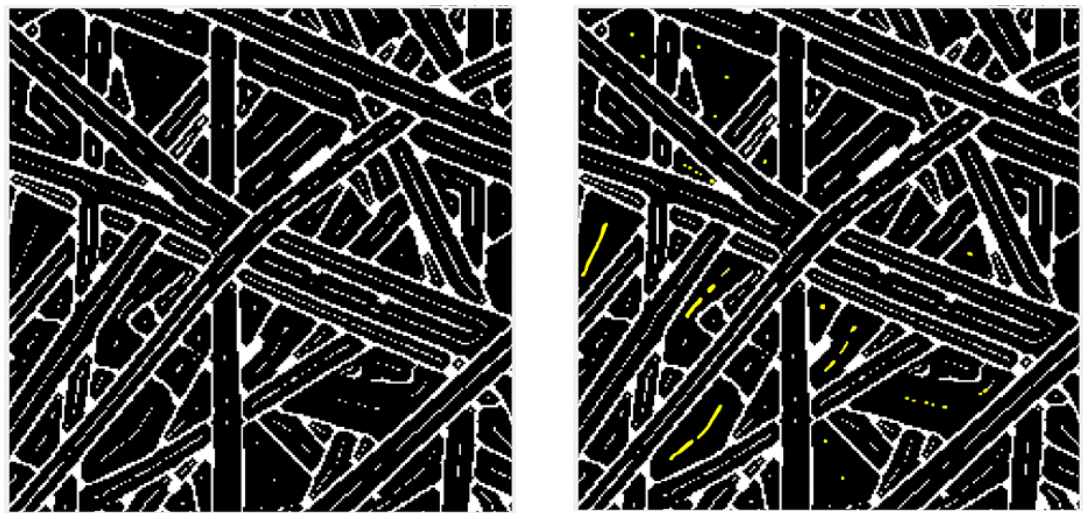
Figure 3-9 shows the generated boundary result of the nanofiber SEM images. The overlapping picture (Figure 3-9c) shows a promising result for the fiber segmentation. When used for diameter determination, precision and integrity are two most essential criteria for image segmentation. Precision means the generated image boundary should be located precisely at the place where the fiber boundary appears. The dislocation of fiber boundary will lead to deviation in diameter. Integrity means that both sides of the fiber need to appear in the image segmentation mask. The fibers in the background, which do not appear on the segmentation mask, do not introduce deviation to the diameter result. Therefore, the segmentation result does not need to outline every single fiber on the image. However, the algorithm for diameter calculation may misrecognize fiber with only one side boundary and bring deviation into the result. Therefore, the segmentation is regarded as satisfactory only when both criteria are satisfied.



a) b) c)

Figure 3-9. a) original SEM image; b) The generated boundary of nanofiber; c) a and b overlap for comparison

One misjudgment problem is noticed during the diameter determination as illustrated in Figure 3-10. As seen from the skeleton of the image in Figure 3-10a, the centerlines are drawn between not only the fibers, but also some gaps, and the misjudged fibers are highlighted in Figure 3-10b. This means that some gaps between fibers are also recognized as the nanofiber boundary, and the distance between gaps is calculated as the fiber diameter. The algorithm itself causes this error and it is considered challenging for the program to distinguish between the gap and the fiber. However, this is not a big concern because the gap usually leads to a ridiculous large number, which is hundreds of times larger than the average fiber diameter. This obvious outlier can be easily identified and excluded.



(a) Skeletonization result of the image

(b) Skeletonization with misjudgment highlighted in yellow

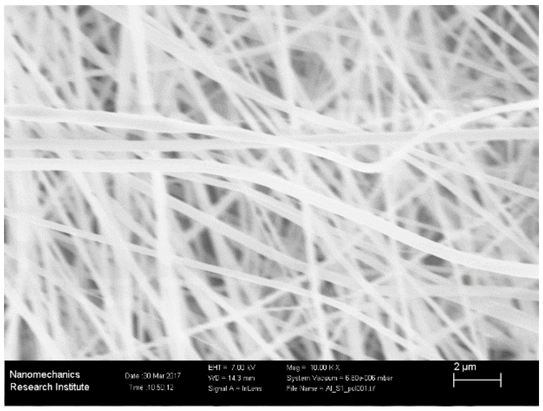
Figure 3-10. Skeletonization result of the image

3.3 Diameter Determination Using Residual Neural Network

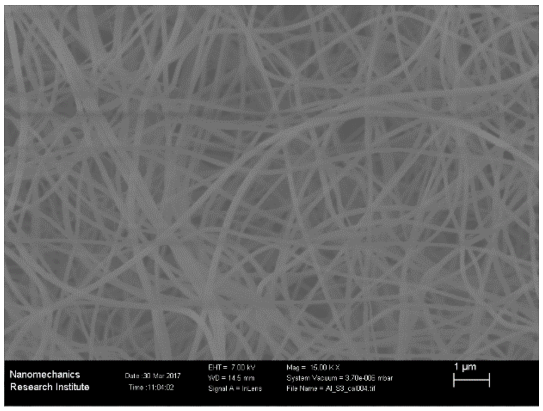
3.3.1 Dataset of real SEM data

Using machine learning to solve practical problems normally requires sufficiently large volume of historical data, which is also a key challenge. This challenge was also encountered when I was constructing the SEM image dataset for this thesis work. SEM is often a shared costly equipment in a university environment, and it an appointment is needed to use the instrument. In addition, it is time-consuming and requires professional skills to operate.

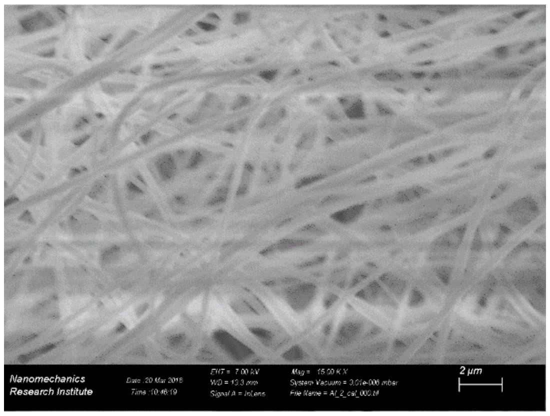
The proportion of usable images is also very small among the collected data. Example of unqualified images are shown in Figure 3-11. There are many reasons why pictures are unusable. For example, the fiber mat is either too thick or too thin, the camera is not always in focus, and the image may be too bright or too dark. In addition, not all projects aim to produce smooth fibers with good surface homogeneity. Some researchers (*e.g.*, Zhang et al. 2021) need porous fibers or fibers with rough surfaces for specific purposes. However, only pictures of smooth, solid, and surface homogeneity nanofibers are selected for the dataset to be built.



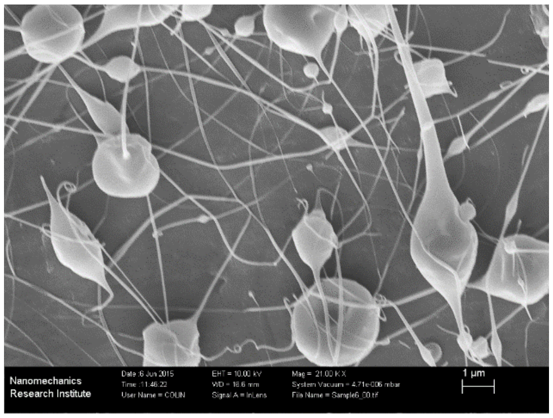
(a) Too bright image



(b) Too dark image



(c) Out of focus image

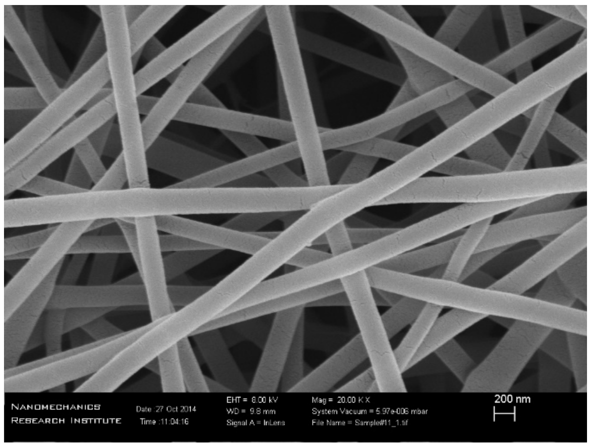


(d) Image with uneven fibers

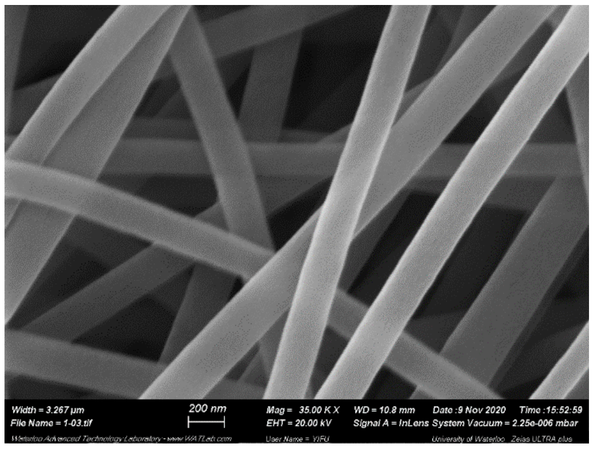
Figure 3-11. Examples of unqualified SEM images

The preceding factors limit the size of our SEM image dataset, although many have been collected over the last ten years. After careful selection, 172 images useable images are chosen to build the SEM image dataset for this thesis work.

All selected images are cut into squares with a size of 341x341 pixels. These images were created by different group members over many years, and the settings of the equipment are also different. Figure 3-12 shows two SEM images taken in 2014 and 2020. Therefore, the size of the SEM original data is also slightly different.



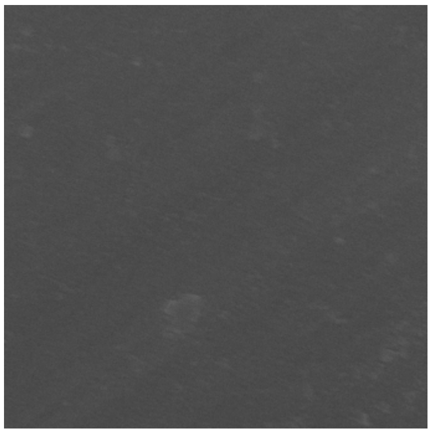
(a) The SEM image taken in 2014



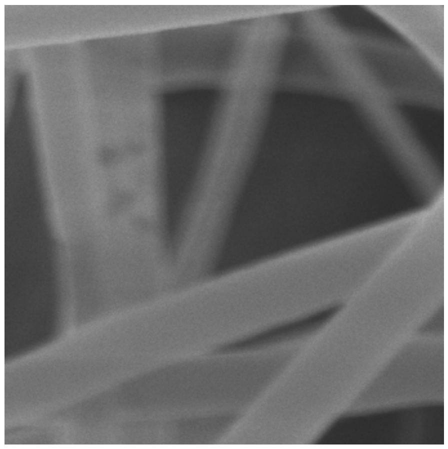
(b) The SEM image taken in 2020

Figure 3-12. Two SEM images take at different year with different size

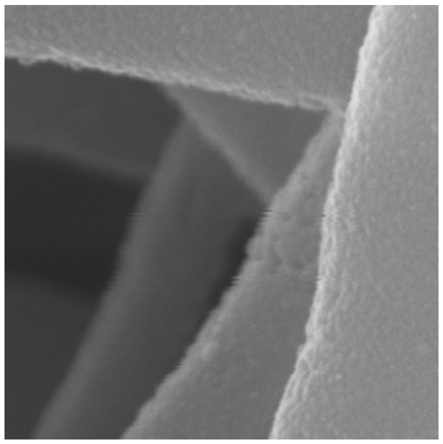
Although the sizes of these images vary, six square images with a size of 341x341 pixels can be cropped out of each of the original SEM image, which is referred to as sub-images. As a result, 1032 sub-images were cropped out of the 172 original SEM images. However, not all the sub-images can be used for dataset buildup. Some sub-images, for example, contain very few fibers or no fibers at all. Examples of unqualified sub-images are shown in Figure 3-13. These types of sub-images do not have enough information for diameter determination, and they are excluded from the dataset.



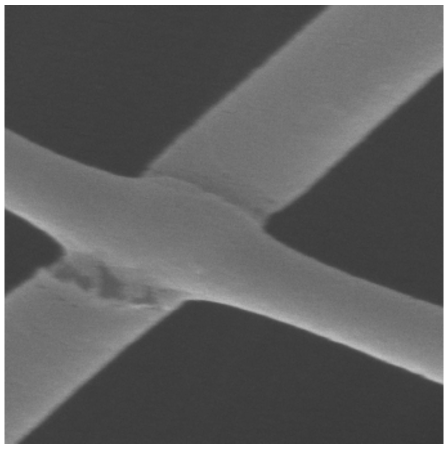
(a) Sub-image with no fibers



(b) Blurry sub-image



(c) Sub-image partial fibers



(d) Sub-image with two fibers only

Figure 3-13. Examples of unqualified sub-images

After the screening the 1032 sub-images, 912 of them are considered valid or the dataset. Figure 3-14 illustrates an example a qualified sub-image. This sub-image has enough fibers in the image, and the image is clear; most of the fibers in the image have both sides of the border in the field of view simultaneously. These features ensure sufficient information for diameter determination.

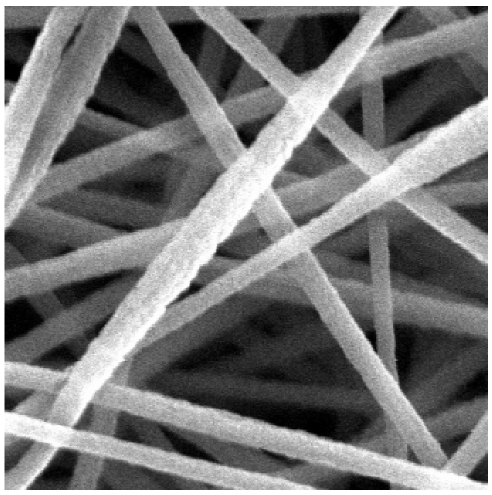


Figure 3-14. An example of qualified sub-image for ResNet training dataset

When using machine learning for classification tasks like this, each image is assigned to one class, and the class number is called a label in machine learning. The label is regarded as ground truth and used to judge whether the result is good enough. In this thesis project, the labels of images are the mode of nanofiber diameter in terms of pixels. The diameter result label is created using an autonomous MATLAB script for fiber diameter determination using the same algorithm as the one described in Section 3.1; the MATLAB code is in Appendix A. The resultant diameter distribution from a SEM sub-image normally follows the normal distribution. For this dataset buildup, the mode number of the diameter distribution is chosen to be the label used in the result. This procedure is to simplify the label from a distribution function to an integer, reducing the difficulty of implement the Resnet model.

After getting the label of the image, which is an integer, the diameter results are also validated by manually measuring the fiber diameter in ImageJ. Figure 3-15 shows the sample image used for the validation. The result generated by the MATLAB script is 17 pixels. After manually measuring the same sub-image in ImageJ, it is found out that this picture does have multiple fibers with a diameter equivalent to 17 pixels. Therefore, the method is deemed plausible. The locations where the fiber has 17-pixel diameters are marked using the think white lines in Figure 3-15.

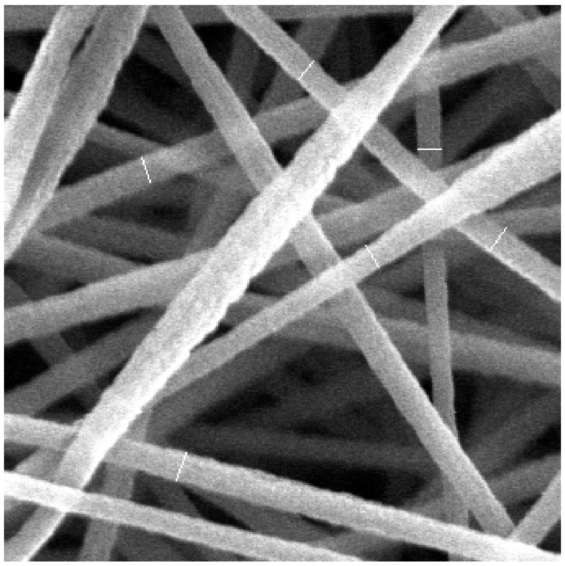


Figure 3-15. The input image with manual measured location highlighted in the picture

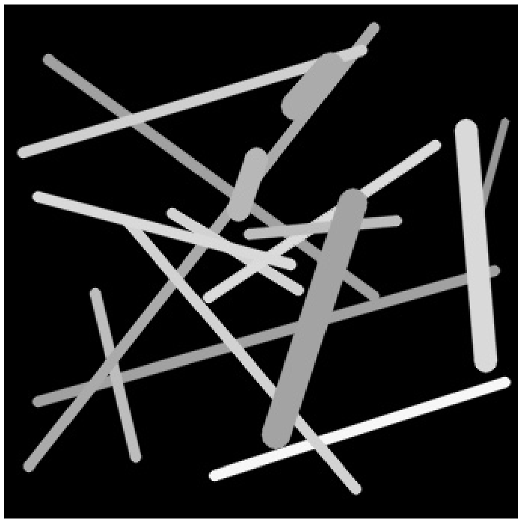
Nanofiber diameters in the dataset range from 1 to 76 pixels. Correspondingly, all images are separated into a total of 76 classes. As the best practice, 80% of the images are divided into the training set, 10% of the images belong to the testing set, and the rest (10%) of the images belong to the validation set. The image has been scrambled multiple times during the dataset buildup and has never been deliberately arranged or distinct by categories. The division of the training set and testing set is done by a Python script (*see* Appendix E) automatically. Therefore, the distribution of image categories in the training set and testing set is deemed random.

3.3.2 Build dataset of synthetic image data

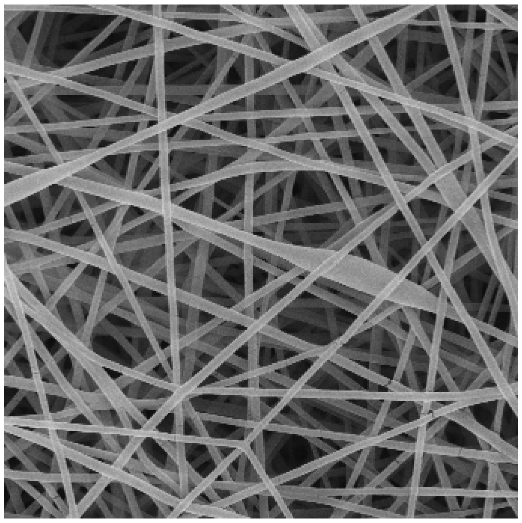
Synthetic images are the images generated by a computer using certain algorithm. As explained in Section 3.3.1, the number of valid SEM images are limited, which his much less than the amount required for training ResNet -- According to previous studies (Shahinfar et al., 2020) report that the data for ResNet training is usually on the order of thousands to tens of thousands. When real SEM data are limited, a synthetic dataset can be built up for the model training. Therefore, synthetic data were generated for this thesis project using a Python script (*see* Appendix E) running on the Jupyter notebook platform because my dataset of real SEM images is less than 1000.

After examining the morphology of the real SEM images, I divided my synthetic images into two categories, random and aligned fibers, to best mimic the layout of real SEM images. In the random-fiber images, all lines are in random orientation. The sizes of all generated images are 341x341 pixels square. The compositions of the images are basically gray lines on a black background. The grayscale of each line is an independent random number between 150 and 250. The location and orientation of each line are random. The width of the line is also random, between 1 and 20 pixels.

Figure 3-16a shows an example of random fiber image. This synthetic image is designed to mimic the real SEM data like the one shown in Figure 3-16b. Despite inevitable differences between these two images, my best effort is made to make them have features in common such as dark background and bright fiber lines.



(a)



(b)

Figure 3-16. (a) An example of random synthetic image; (b) An example of real SEM image with randomly oriented nanofiber

For the synthetic dataset, the label was generated following the same criteria described in Section 3.4.3.1 below, using the mode of nanofiber diameter as the image label. The mode of nanofiber diameter is expressed in terms of pixels, which is an integer. Unlike real SEM data, the synthetic image data are designed and generated by my will and the label is known before the images are generated. Since the label is the mode of fiber diameter distribution, the synthetic images were designed to have one most frequently occurring line width for each image. Considering the size of the picture, 341×341 pixels, the number of lines is set to 16 for a reasonable layout. This number is obtained after many trials and error. With the number of 16, the number of lines is just sufficient to cause the lines heavily overlapping each other but provide enough information.

In this dataset, every image contains 16 lines in total, 11 of them have the same width, and the rest 5 have different random widths. The number of pixels of the mode of the line width is the label of the image. An example of generated image is shown in Figure 3-17, where all 16 lines are marked. This image belongs to the class whose label is 4, whereas the mode of line width in this image is 4 pixels. In the image, lines 1 to 11 have the same width, 4 pixels, and lines 12 to 16 have other different widths.

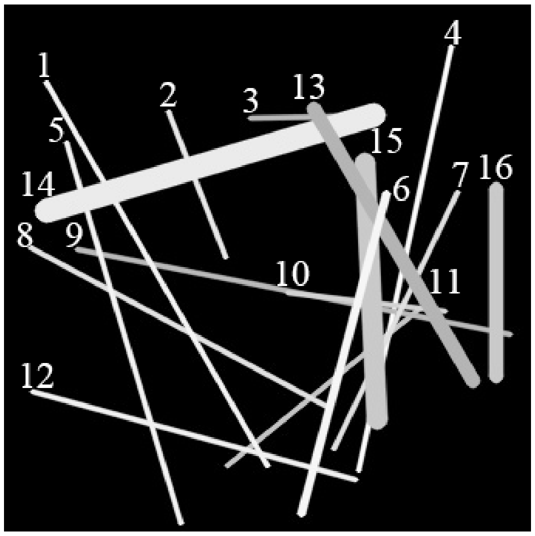


Figure 3-17. An example of Class 4 synthetic image data

In a synthetic aligned-fiber image, all lines are aligned in two orientations at an angle of 90 degrees. These images are designed to mimic aligned nanofibers. Figure 3-18 compares the synthetic aligned-fiber and real SEM images. The synthetic images are also square images, each with 341x341 pixels. The compositions of the images are also gray lines on a black background. The grayscale of each line is an independent random number between 150 and 250. (Note: 0 for black and 255 for white) The orientation and distance between each line are randomized, which means each fiber forms an angle with the edge of the image. In addition, the distances between fibers are randomized too. However, the thickness of all lines in the same synthetic image is the same. The thickness of fibers varies from 1 to 20 pixels for all this type of images used in this thesis work.

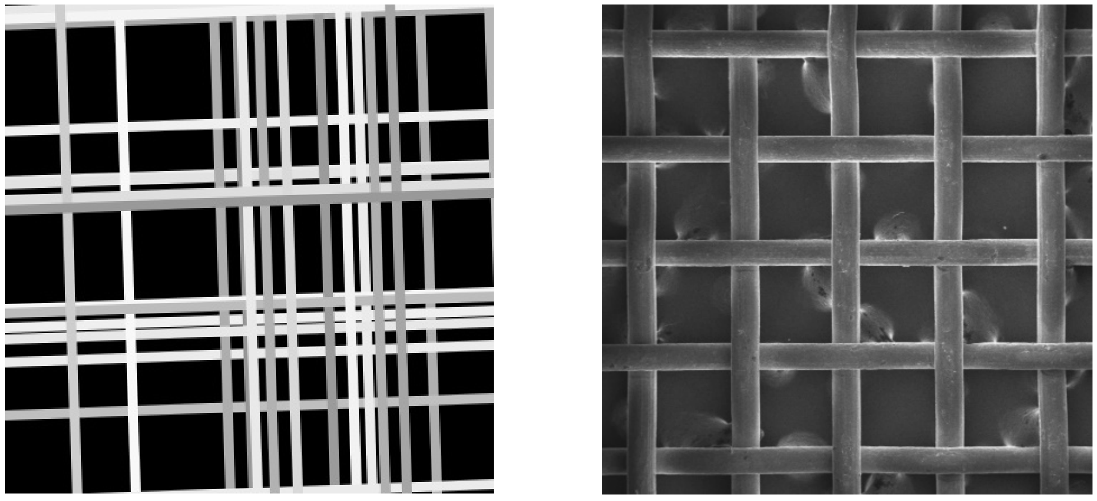


Figure 3-18. (a) an example of category two synthetic image data; (b) an example of real SEM image data of aligned nanofiber

These synthetic images with aligned lines are generated following these steps:

1. Various lines perpendicular and parallel to the bottom are generated on an image with a slightly larger size of 380x380 pixels.
2. This image with is rotated at a random angle between -5 to 5 degrees to ensure that the fiber do not always align to the edge of image.

3. It is cropped into a size of 341x341 pixels to match other images.

The preceding two categories of images, the images with randomly oriented and the aligned fibers, have the same image sizes, colors of fiber lines, and the ranges of fiber widths. There are 20 classes in each category of images, corresponding to the line diameters of 1 to 20 pixels, which are written as 0 to 19 in the code. There are 50 images in each class, resulting in 10,000 images for both categories. A total of 20,000 images were generated for this data set, where 16,000 images belong to the training set, 2,000 images belong to the testing set, and 2,000 images belong to the validation set. In the training, testing, and validation sets, each class in each category has the same number of images.

3.3.3 ResNet model and structure highlight

The ResNet structure is constructed under the TensorFlow[2] framework using the Keras library[3]. TensorFlow is an end-to-end open-source platform for machine learning. Transfer learning is implemented in this thesis project. The structure of ResNet and the implementation of the transfer learning for the two datasets mentioned in Sections 3.3.1 and 3.3.2, which are basically the same.

The ResNet model is pre-trained using the ImageNet dataset. The convolutional weights of the ResNet50 pre-trained model act as a feature extractor in the model. These convolution weights of the pre-trained model are frozen, which means they are not updated during the training. The last fully connected layer of the pre-trained ResNet50 model is replaced by customized predicting layers.

After pre-training the ResNet50 structure, the ResNet50 was implemented. The implementation and the top structure of this model is inspired by the example in Pattanayak's book 'Intelligent Projects Using Python, Chapter 2 Transfer Learning' (Pattanayak, 2019). I added a global average pooling layer and a drop-out layer to the structure, followed by a fully connected layer. The average pooling layer and drop-out layer are used to reduce the amount of calculation and improve computing performance. For the drop-out layer, the drop out ratio

[2] https://www.tensorflow.org/, accessed March 25, 2022
[3] https://keras.io/, accessed March 25, 2022

is set at 0.5, which is a commonly used default value. The fully connected layer is used to modify the dimension of the output, converting the calculation result from a matrix to an integer.

3.3.4 Hyperparameter and computer setup

The computer is setup as follows. The computer is a personal laptop with a CPU of AMD Ryzen 9 5980HS with Radeon Graphics and a GPU of NIVIDIA GeForce RTX 3080. The calculation times are automatically recorded by the computer.

Table 3-1 summarizes the hyperparameter setup for the model trained with synthetic dataset. The max number of training epoch is 50. The training is set with an early stopping guide: patience equals 3, which means that the calculation will stop if the accuracy does not improve in 3 epochs, regardless of epochs.

The loss function used is the categorical cross-entropy. Cross entropy means the difference between actual and predicted probability distributions for each class. This loss function is specifically designed for multi-class classification tasks, is regarded as the default loss function for this type of problems, and assumes that only one class is correct among all classes. The loss function is calculated using Eq. (3-4 (Zhang and Sabuncu, 2018). In practice, the categorical cross-entropy, as a commonly used function, is stored in the Keras library and can be called directly.

$$\boldsymbol{L} = -\sum_{\boldsymbol{i}=\boldsymbol{1}}^{\boldsymbol{n}} (\boldsymbol{y}_{\boldsymbol{i}} \mathbf{log} \widetilde{\boldsymbol{y}_{\boldsymbol{i}}}) \tag{3-4}$$

where L is the loss function, y_i is the *i*-th scalar value of the model output, $\widetilde{y_i}$ is the target value and n is the number of scalar values in the model output.

Table 3-1. The hyperparameter setup for the model trained with synthetic dataset

Hyperparameter	Specification	Remarks
Learning Rate	0.001	
Epochs	Max epoch = 64	Early stopping: Patience = 3
Batch Size	128	

Loss Function	Categorical cross entropy	
Metrics	'accuracy'	
Output Layer Activation Function	SoftMax function	Number of classes = 20
Normalization	Input and labels normalized between 0 to 1	From range of (0,255) to (0,1)
Trian-Test-Split	Train=80%, Train=10%, Val= 10%	Train = 16000, Test = 2000, Val = 2000

Table 3-2 summarizes the hyperparameter setup for the model trained with real SEM image dataset trained. The main setup is almost the same as the model trained with hyperparameter of synthetic dataset, expect for the number of classes, number of images and batch size. These three differences are all due to the features of real SEM images. For real SEM images, the larger number of classes is due to the wider range of nanofiber diameters in images. The number of real images is smaller than synthetic images, and therefore the batch size is also reduced to make sure the parameters in model get enough updates.

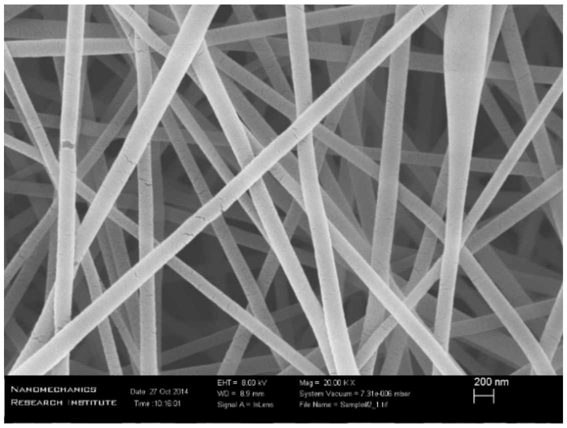
Table 3-2. The hyperparameter setup for the model trained with real SEM image dataset

Hyperparameter	**Specification**	**Remarks**
Learning rate	0.001	
Epoch	Max epoch = 64	Early stopping: Patience = 3
Batch size	16	
Loss function	Categorical cross entropy	
Metrics	'accuracy'	
Output layer activation function	SoftMax function	Number of classes = 76
Normalization	Input and labels normalized between 0 to 1	From range of (0,255) to (0,1)
Trian-Test-Split	Train=80%, Train=10%, Val= 10%	Train = 730, Test = 91, Val = 91

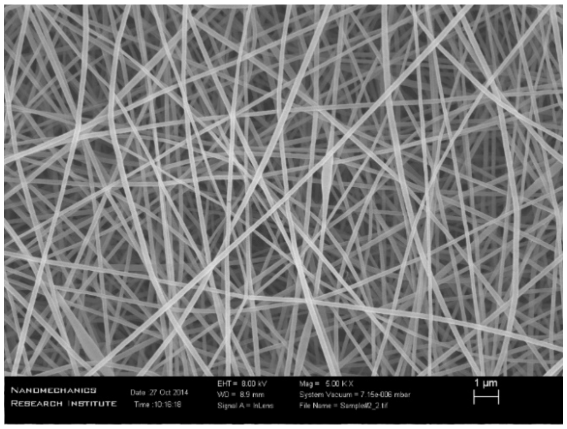
3.4 Results and Discussion for Diameter Determination

3.4.1 MATLAB approach results

Figure 3-19 shows the SEM images of same nanofiber sample at different magnifications, and Figure 3-20 shows the corresponding diameter distributions obtained using the multi-image automated diameter determination tool introduced in Section 3.1. Givehchi et al. (2016) also reported that the diameter determination depends on SEM magnification.



(a)The SEM image of nanofiber at 20K magnification



(b) The SEM image of nanofiber at 5K magnification

Figure 3-19. SEM images of one nanofiber sample at two different magnifications

Considering the dependence of diameter distribution on the SEM magnification, results from images at different magnifications should be combined for more statistically reliable data. For the combined result, the average diameter and standard deviation are closer to the low-magnification image result. This is likely because the low-magnification images contain a larger area of sample, and therefore its results are closer to the fact. The mean diameter obtained from the image with a lower magnification is smaller. It is likely because the high magnification means a zoom-in on the sample, and thus characterizes the finer fibers that are ignored in the low-magnification scenario. Therefore, combination of the results from different magnifications is deemed more comprehensive and accurate in diameter determination.

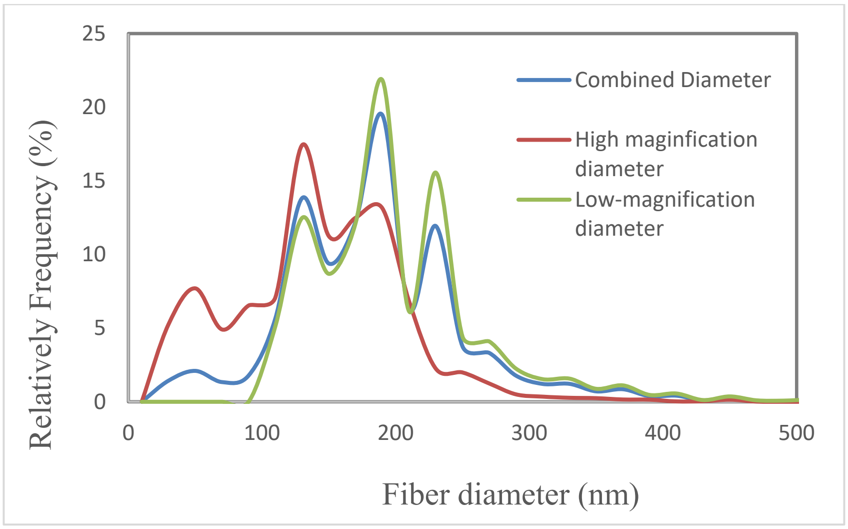
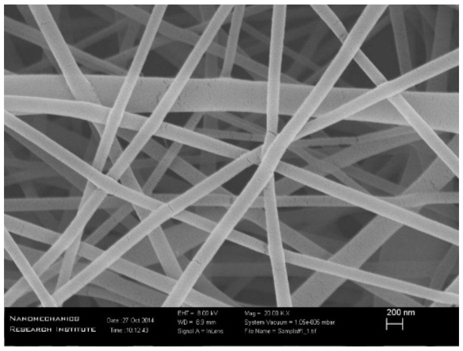


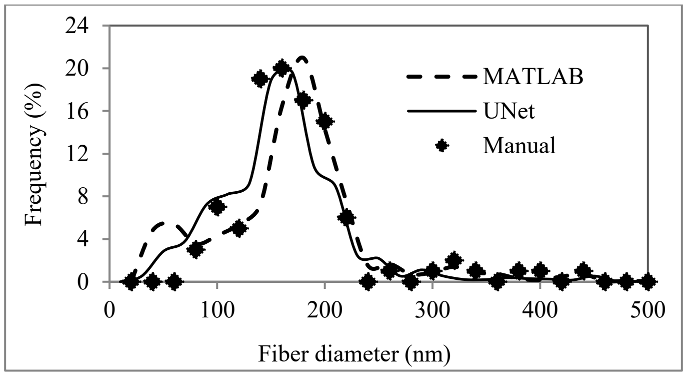
Figure 3-20. Diameter distributions from two images at different magnifications

3.4.2 UNet approach result

Figure 3-21 compares the diameter distributions obtained using MATLAB, manual measurement, UNet approach and the corresponding SEM image used. The manual measurement is regarded as the benchmark for accuracy evaluation. The manual measurement is from the doctoral thesis of Givehchi (2016), who also uses the manual measurement as a benchmark. The same benchmark is used herein to compare the UNet method with the MATLAB method. Figure 3-21b shows that that the UNet method is closer to the manual measurement than MATLAB does. However, their difference is not distinctive. Further in-depth analyses are as follows.



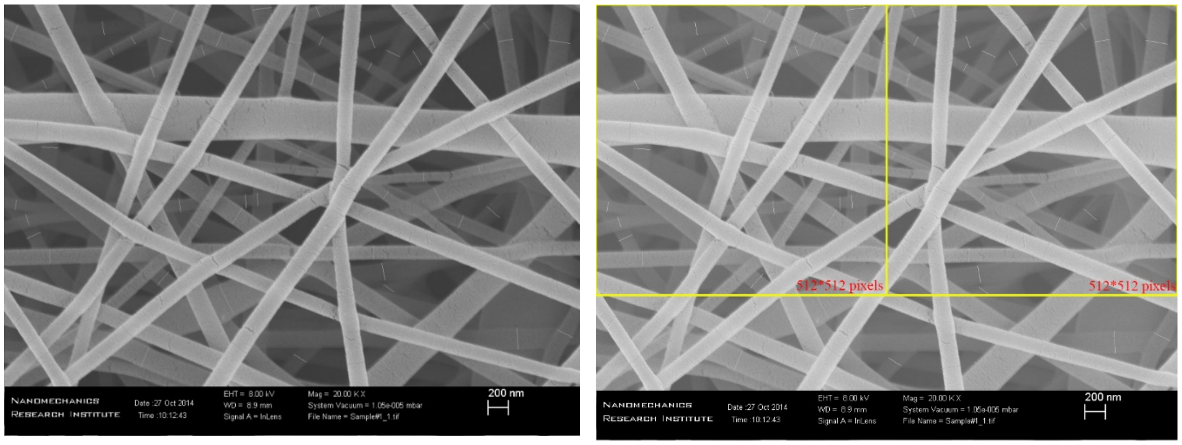
(a)



(b)

Figure 3-21. (a) The subject SEM image; (b) The results measured using MATLAB, UNet, and manual counting

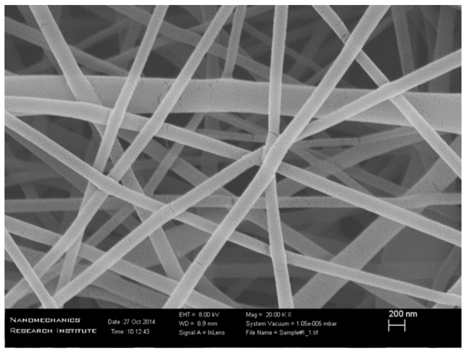
First, the difference between manual measurement and UNet result may be induced by the difference in area of examination. For example, Figure 3-22 shows the original SEM image (Figure 3-22a) for manual measurement and the image for the UNet method highlighted in yellow (Figure 3-22b). Figure 3-22 (a) shows the image with manually measured points marked - the short white lines crossing the fibers in this figure mark the random locations for manual measurement. Note that all fibers in the original SEM image are counted when using the MATLAB method. The size of the complete SEM image is 1024×681 pixels, which can be measured twice in a lateral direction using UNet because when using UNet for image segmentation, the input image size is fixed at 512×512 pixels. This means that the UNet method cannot measure the entire area of the SEM image.



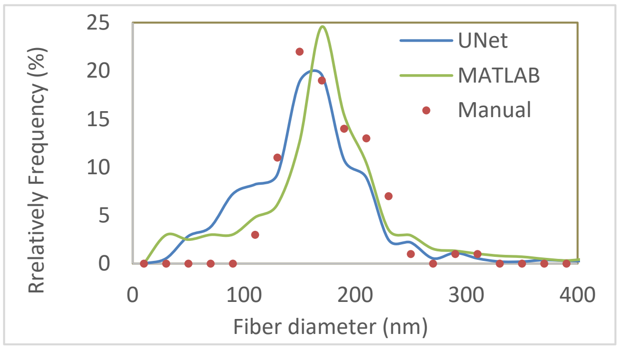
a) b)

Figure 3-22. a) The SEM image with the manually measure location marked; b) Image with the UNet method measuring area highlighted in yellow

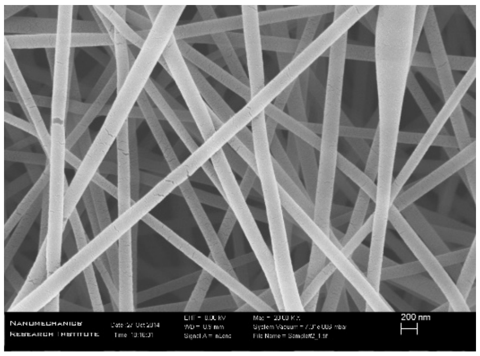
To exclude the interference of different examination area, I compared the results using the same area, which is the one for UNet. Figure 3-23 compares the results obtained using the same size of SEM image, and the corresponding mean diameters and standard deviations from all methods are shown in Table 3-3. According to the table, compared to the MATLAB method, mean diameter from UNet method has a smaller deviation with the manual result. It means the UNet method has a better accuracy compared to the MATLAB method. Overall, the UNet model is deemed reliable. The distributions of UNet data points are very close to the manual measured data points and the overall trend is also the same.



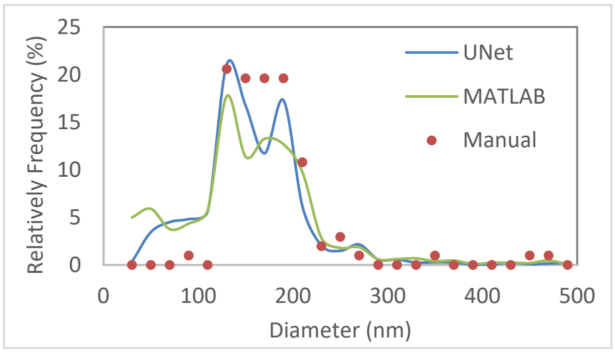
(a) Original SEM sample 1



(b) The diameter distribution of Sample 1



(c) Origianl SEM sample 2



(d) The diameter distribution of Sample 2

Figure 3-23 The comparison of diameter distribution curve from different methods

Table 3-3 The mean and standard deviation of diameter results

Sample	**Manual**		**MATLAB**			**UNet**		
	Mean (nm)	Stdev* (nm)	Mean (nm)	Stdev (nm)	Mean's deviation with manual (%)	Mean (nm)	Stdev (nm)	Mean's deviation with manual (%)
#1	178.61	43.98	166.75	80.19	6.64	177.80	71.96	0.50
#2	177.07	54.50	156.02	72.72	11.89	156.77	58.02	11.46

*Stdev stands for *standard deviation*.

3.4.3 Result from ResNet approach

3.4.3.1 Result from real SEM image dataset

Figure 3-24 compares the accuracy and loss of the ResNet50 model results from the SEM dataset, of nearly 1000 images. The final training accuracy is 0.315, and the final training loss is 2.470, which are not ideal. Normally, an accuracy of 0.7 or more can be considered as satisfactory for multi-classification task. The current accuracy 0.315 in Figure 3-24 is much less than this threshold. The inaccuracy may result from the insufficient sample size and uneven distribution of data samples in the SEM database. They are elaborated as follows.

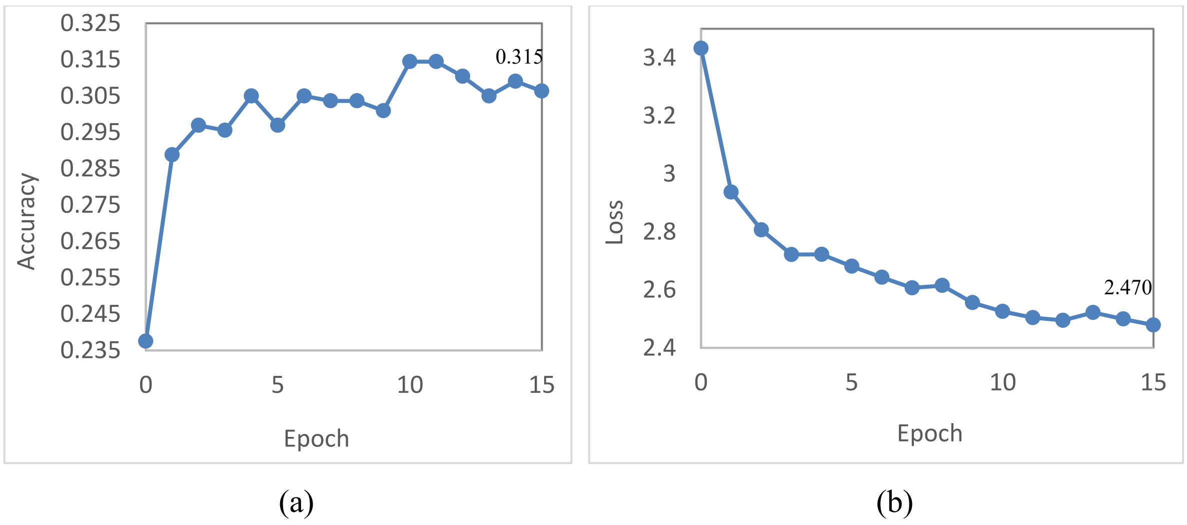


Figure 3-24. (a) The epoch accuracy plot for model trained on real SEM images; (b)The epoch loss plot

First, the number of samples in the database is less than that normally required for deep learning training. It is a best practice in machine learning that the more data the better. For example, Shahinfar et al. (2020) analyzed the effects of image number on machine learning result. They reported that the more images, the better the training result, ideally in the range of ten to one thousand images per category. Even when transfer learning is used to reduce the number of images needed, the number of images in each category should still be on the order of 100. However, there are only about ten images for each category in this thesis project because there are only 912 images in this dataset for 76 classes in this thesis project.

Another challenge is the uneven distribution of data. Ideally, there should be the same amount of data in each category when using machine learning for multi-classification tasks. Figure 3-25 shows the uneven distribution of data in the categories. The category with most images, diameter at six pixels, has 135 pictures, which is ten times the average value. On the other hand, there are 35 categories with less than 10 images. Moreover, the data points of diameter value are not continuously. For example, the frequency of diameter at 19, 58, 59 pixels are zero. The uneven data distribution adds more difficulty to this task and reduce the accuracy of results. This should be part of the future work.

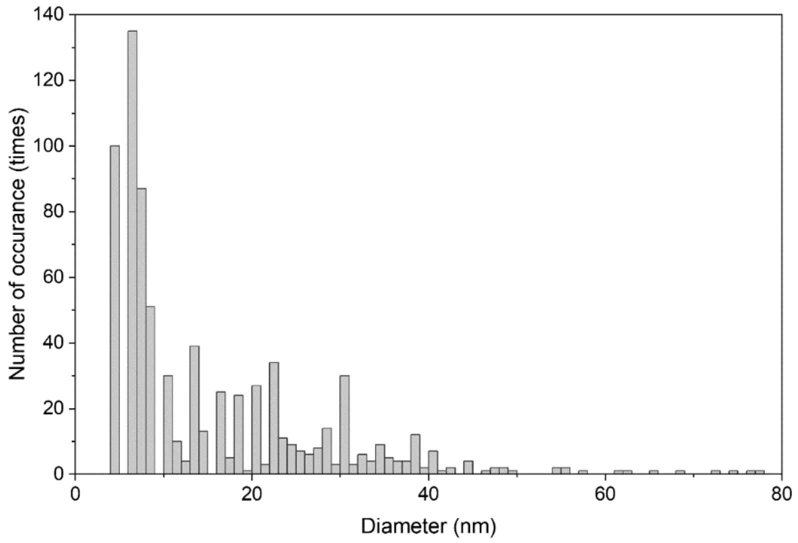


Figure 3-25. The data distribution of the real SEM dataset

Data augmentation is the most common solution to insufficient in machine learning. Data augmentation refers to adding slightly modified copies of existing data to increase the total amount of data. Common data augmentation methods include geometric transformation, flipping, cropping, rotation, and random erasing (Shorten & Khoshgoftaar, 2019). However, methods like geometric transformation are not applicable to fiber diameter determination because it changes the shape of the items in the images. Nonetheless, data augmentation cannot solve the problem of uneven and discontinuous data distribution.

Alternatively, a synthetic dataset is generated for the modeling training to solve both problems. Details of results can be found in the next section. The training performed on synthetic data can also be regarded as a trail to verify the feasibility of this method.

3.4.3.2 Results using synthetic image dataset

Figure 3-26 shows the accuracy and loss *vs*. epoch of synthetic dataset. The continuous black line represents training set accuracy and loss, and the scattered dots present the validation set accuracy and loss. The result shows that the final training accuracy is 0.7747. The training loss is 0.3241. The validation accuracy is 0. 7832. The validation loss is 0.3241. Empirically speaking, an accuracy over 0.7 can be regarded as good enough for a multi-classification task. The validation accuracy is not lower than the training accuracy which means that the hyperparameters in the model are optimized well and the model converges properly.

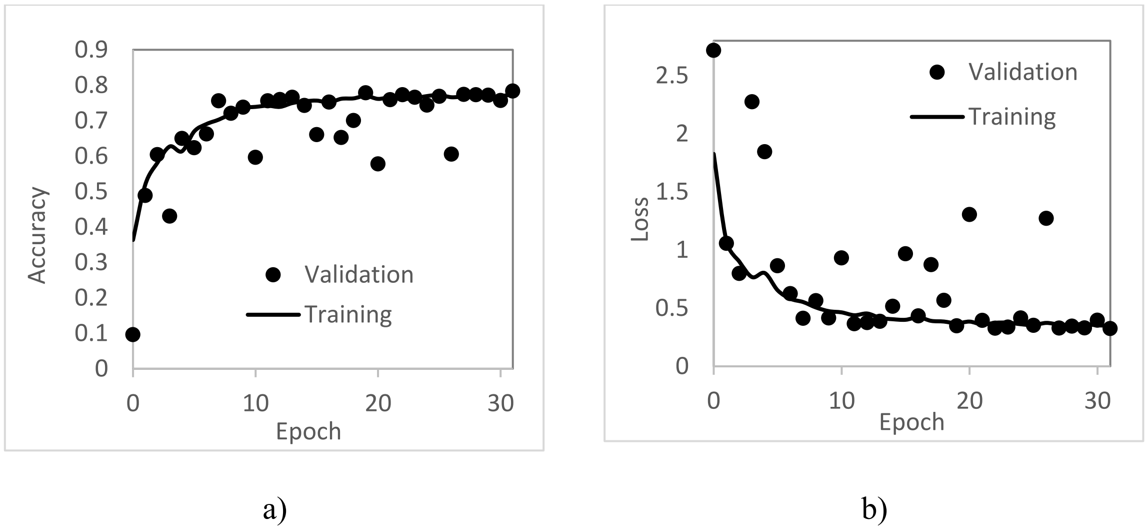
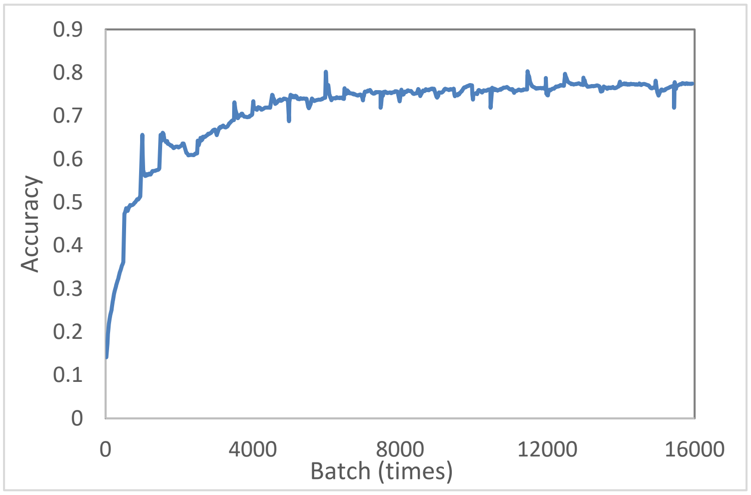
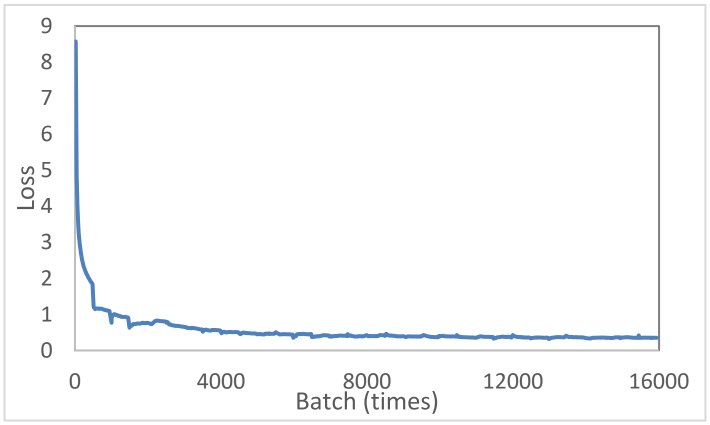


Figure 3-26. (a) The epoch accuracy plot; (b) The epoch loss plot

Figure 3-27 shows the batch accuracy and loss. The batch accuracy is the accuracy change after each batch and it. The batch accuracy shows the same accuracy and loss result. The shape of the accuracy and loss plot confirm that the model does convergence at the end of training.



a)



b)

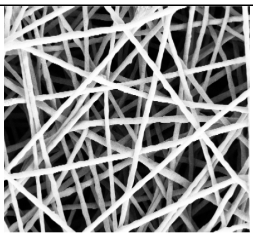
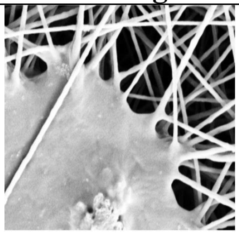
Figure 3-27. (a) The batch accuracy (b) The batch loss plot

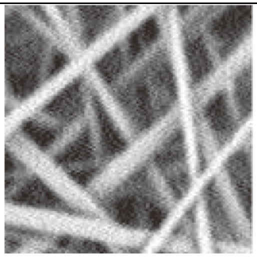
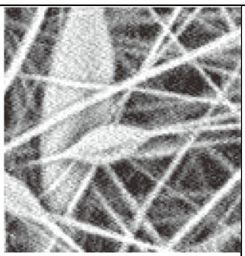
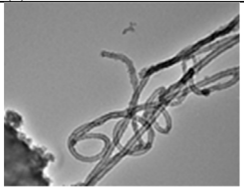
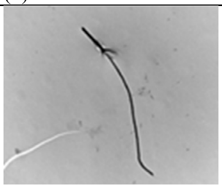
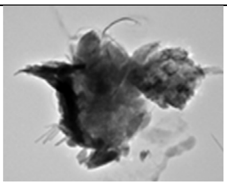
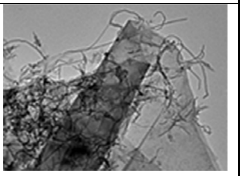
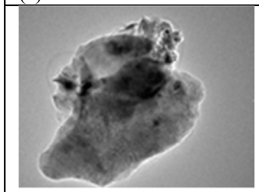
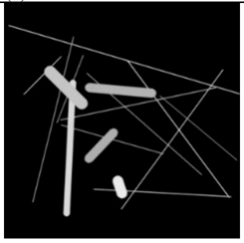
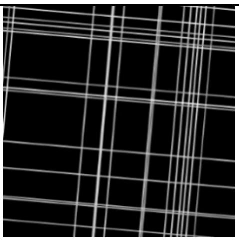
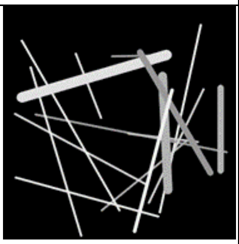
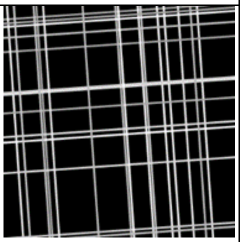
When apply machine learning to the nanofiber field, the previous studies used it on distinguish different kind of fiber (Ieracitano et al., 2021; Matson et al., 2019; Napoletano et al., 2018). Napoletani et al (2018) used CNN to distinguish anomaly in nanofiber, and their binary classification accuracy reached 97%. Ieracitano et al (2021), proposed a classification system that combined unsupervised and supervised machine learning to distinguish different fibers. They classified nanofibers into two categories, homogenous and non-homogenous nanofiber. Their classification accuracy was as high as 92.5%. Matson et al (2019) used three different neural networks, i.e., pre-trained ResNet50, VGG16, and a modified VGG16, to distinguish five categories of nanofibers, including nanocarbon tube (CNT), nanocarbon fiber, and nanocarbon cluster. The three neutral networks gave similar accuracies between 60% and 85% for each category.

In this thesis study, the accuracy retrieved from the validation result is 78%, which falls in the range of 60-85% reported earlier, despite the larger size of categories. Conversely, Napoletani et al. (2018) and Ieracitano et al. (2021) only required the model to distinguish between two categories; Matson et al. (2019) required their model to distinguish between 5 categories. In this thesis project, however, the model aims to distinguish between 20 categories, which greatly increase the difficulty and reduce the accuracy.

In addition, the images in this thesis work have more subtle difference between different categories than those in earlier studies (Napoletani et al., 2018; Ieracitano et al. 2021; Matson et al. 2019). To illustrate, Table 3-4 compares the images in different categories of different studies. Unlike the images in other studies, which show different shapes in different categories, all images in this thesis work contain lines with similar colors and shapes and the only difference is their widths. The similar shape gives them similar morphological features and making it more challenging to distinguish between them.

Table 3-4 Comparison of images in different categories of different studies

Research	**Categories of images**			
Napoletani et al. (2018)	 (a) without anomalies	 (b) with anomalies		

Ieracitano et al. (2021)	 (a) Non-defective fiber	 (b)Defective fiber		
Matson et al. (2019)	 (a) CNT Cluster	 (b) CNT Fiber	 (c) CNT Matrix	 (d) CNT Matrix-Surface
	 (e) Non-CNT			
	 (a) Category 1, 1-pixel width line	 (b) Category 2, 1-pixel width line	 (c) Category 1, 2-pixel width line	 (d) Category 2, 2-pixel width line

3.5 Summary

In summary, this thesis study points out an application of machine learning for nanofiber characterization; its feasibility is verified. In the future, work can be done to further optimize the neural network structure and further increase the accuracy. It would be also useful to reconstruct a dataset using real SEM images and try to achieve a high accuracy using this model on real SEM images.

CHAPTER 4. CHARACTERIZATION OF PORES AND SURFACE HOMOGENEITY BY IMAGE PROCESSING

4.1 Surface inter-fiber pores determination

4.1.1 The Methodology

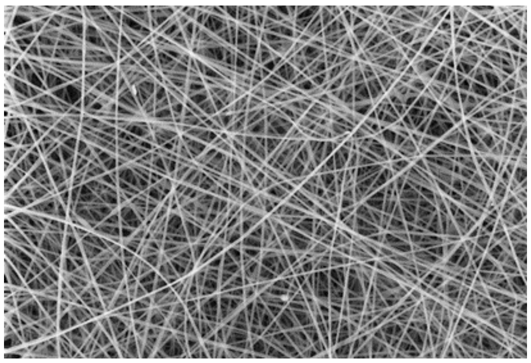
An automated image processing tool is developed using MATLAB to characterize the surface inter-fiber pores of nanofibers, and the code is in Appendix J. Figure 4-1 shows the image processing procedure. It consists of the following four steps, which are going to be explained one by one.

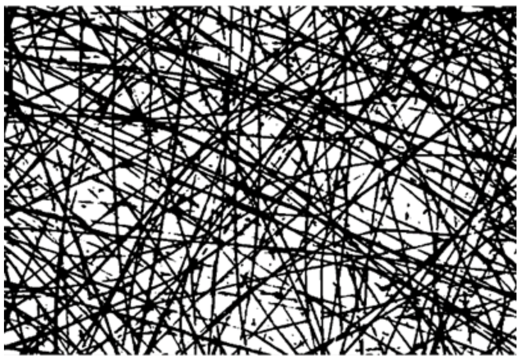
a) read scale and crop

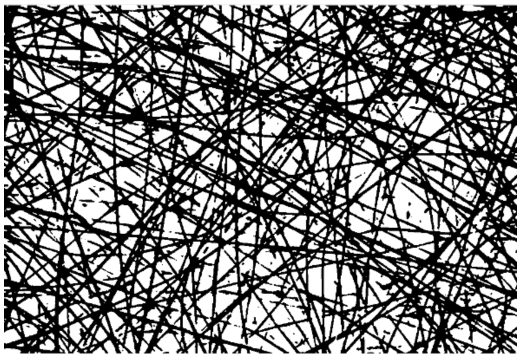
b) binarization

c) denoising

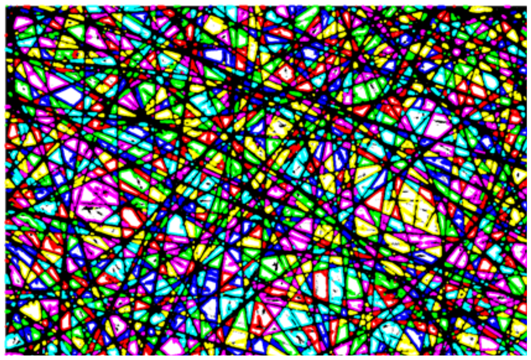
d) size measurement and visualization


(a) Read scale and crop


(b) Binarization



(c) Denoising



(d) Size measurement and visualization

Figure 4-1. The image processing procedure for inter-fiber pore size determination

a) Read scale and crop

This step removes the information bar at the bottom of the image. The scale of the image is also cropped out, and its length is measured.

b) Binarization

Convert the image into grayscale one, if necessary. Then, binarization by thresholding to convert the image into a black-and-white binary image. Three thresholding methods are tested herein for image binarization: global Otsu, locally Otsu and manually-input threshold.

The first edition of the script is written using the global Otsu threshold method, the most frequently used method in relevant research. After the script is done, it is noticed that the picture's brightness varies, and the morphology of the nanofibers is also different. With only one criterion, the binary image does not always look reasonable.

Therefore, another version of the script with a manually input threshold is attempted. The manual option gives the initiative to the operator, and the appropriate values are determined by subjective judgment. The thresholding value determines how deep we want to consider the fiber as one layer. A large thresholding value means only fiber on the very surface is considered, resulting in large pore sizes. A smaller thresholding value means fiber at a certain depth on the surface is considered as one layer, resulting in relatively smaller pore sizes.

After the second edition, the binarization operation always results in some fibers with a fading edge. The fading edge comes from uneven brightness in the image. Even for one fiber, there are light and dark sides of the fiber, and the fading edge will result in a significant deviation in pore size determination. Therefore, attempts are made to use Otsu threshold methods locally.

Local Otsu method is used in the third edition. The binarization result looks more reasonable and accurate after this improvement.

c) **Denoising**

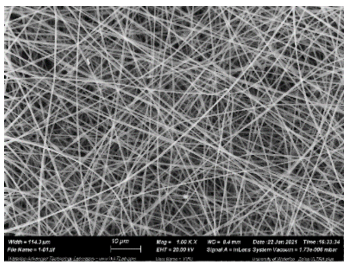
After the binarization, a series of denoising operations is performed. "Majority" and "clean" operations are performed to eliminate the dots on the image that are less than 3×3 pixels. A median filter and histogram equalization are performed for denoising and increasing image contrast. "Opening" and "closing" operations are also performed for further reduce the noise.

d) **Pore size measurement**

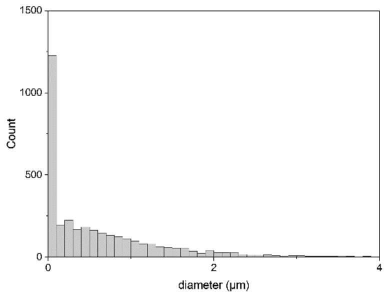
After the denoising, the boundaries of pore are identified and labelled. The area size inside each separated boundary is measured using the "regionprops" function in MATLAB. The number of pixels in the region is measured and returned. The pore size is the equivalent diameter of a circle of equal area.

4.1.2 Results of surface inter-fiber pores determination

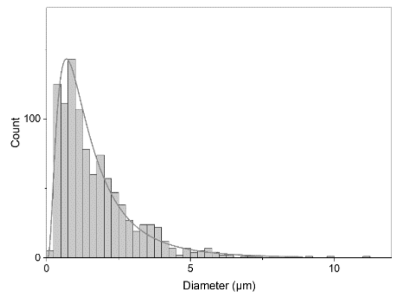
Figure 4-2 shows the SEM image of a nanofiber mat and the determined pore size distributions with and without denoising operation. The result in Figure 4-2 (b) is obtained a preliminary edition script without denoising. The script for this example is the same as the one in Appendix J, except for the denoising step introduced in c) of Section 4.1.1 is excluded. The pore size distribution is extremely *right-skewed*, with a peak close to 0 on the left end. Such a high peak is unlikely to reflect the true characteristics, and likely it results from noises on the image.



(a) Sample



(b) Pore size distribution without denoising



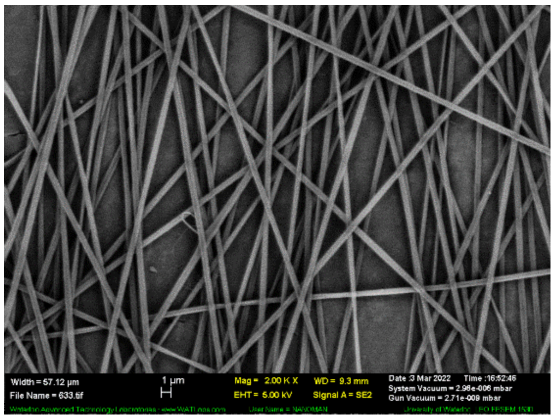
(c) Pore size distribution with denoising

Figure 4-2. The pore size distribution with and without denoising operation

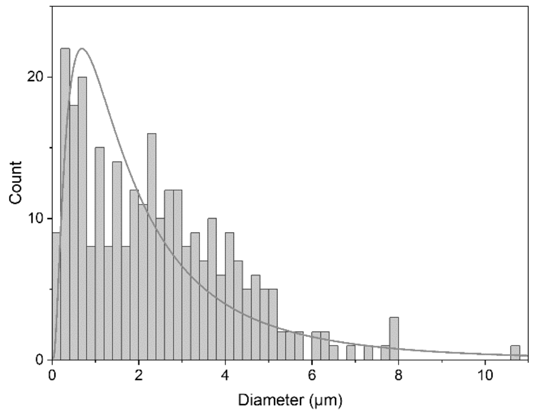
To improve the accuracy of measurement, serval denoising operations were added in the script (*see* Section 4.1.1 and Appendix J). Figure 4-2c shows the pore size distribution with denoising applied. It appears to be a lognormal distribution with a coefficient of determination, R^2, of 0.9653. Usually, correlation with an R^2 whose value higher than 0.8 can be regarded as statistically significant.

Log-normal distribution is typical in many engineering fields, including aerosol particle size distribution. However, we cannot verify the accuracy of the preceding results, regardless of denoising or not. The reason is that a 3D nanofiber mat with a high porosity may appear otherwise in 2D because of the misleading perception, especially for thick nanofiber mats. Nonetheless, this automated tool for pore size distribution may find usefulness for thin nanofiber mat. The thinner the better, ideally the most accurate for a single-layer nanofiber mat because all fibers in the sample can be captured by the SEM images.

To verify the preceding hypothesis, a single-layer nanofiber mat is tested, and related results are shown in Figure 4-3. The pore size determination result is shown in **Error! Reference source not found.**. The pore size distribution also appears to be lognormal with a R^2 value of 0.8689, which means it is statistically significant.



(a) Single-layer nanofiber mat



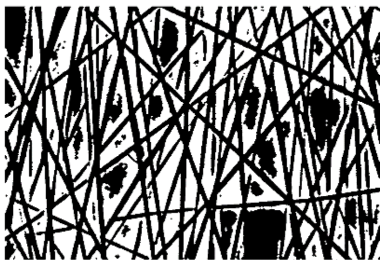
(b) Pore size distribution with denoising

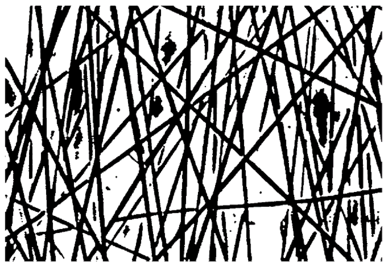
Figure 4-3. A single-layer nanofiber mat and its pore size distribution

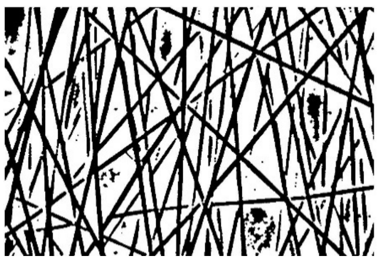
4.1.3 Effects of binarization on pore size distribution measurement

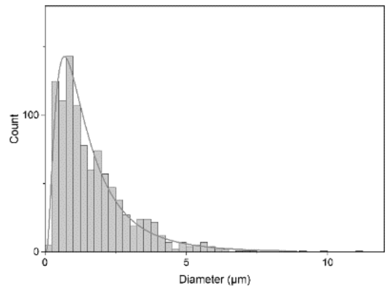
Another factor of concern is the brightness of the background. A dark background of image generated may influence the effectiveness of binarization, which depends on the method of binarization. This thesis work compares 1) Otsu global binarization, 2) Otsu local binarization, and 3) the adaptive binarization function built in MATLAB. The adaptive binarization is performed by using the MATLAB built function "adaptthresh". This function calculates the locally adaptive threshold value based on the Gaussian weighted mean intensity (first-order statistics) in the neighborhood of each pixel.

Figure 4-4 compares the image segmentation results and pore sizes counted using these three methods for the original SEM image shown in Figure 4-3. Comparing the first row of images after binarization (Figure 4-4a-c) with the original SEM image in Figure 4-3 shows that global Otsu binarization introduces the largest errors. However, the resultant segmentation images local Otsu method and adaptive method are comparable with minor artifacts appear at different spots. While it is safe to conclude that the measurement using global Otsu binarization is inaccurate, we cannot tell which one of local Otsu and adaptive methods is more reliable. Nonetheless, a comparison between the x-axes in Figure 4-4e and f shows that the equivalent pore diameters obtained using local Otsu are slightly larger than those using adaptive binarization.

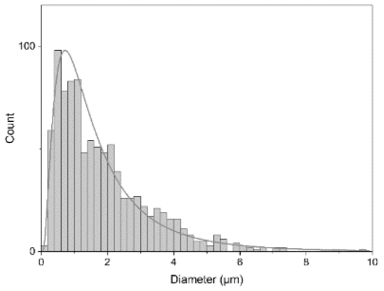

(a) global Otsu binarization


(b) local Otsu binarization

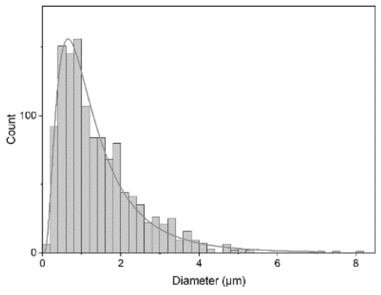

c) adpactive binarizaiton



d) using global Otsu binarization



e) using local Otsu binarization



f) using adaptive binarization Method

Figure 4-4. The image segmentation result from different binarization methods and resultant pore size distributions

Despite the differences in results, one cannot draw an affirmative conclusion that adaptive binarization is better than local Otsu binarization, or vice versa. The reason is that the ground truth, manual measurement, is impractical. Note that the pore sizes presented of this section are centered on the area-equivalent pore sizes, which can only be determined by image processing. The current experimental method such as BET can only generate a hypothesis pore size with many simplifications. Finally, we can further compare the mean value and standard deviations of the three size distributions. Table 4-1 shows the mean pore sizes and standard deviations of the lognormal fitting curves.

Table 4-1 The expected value and standard deviation of pore size result from different binarization method

Binarization Method	Mean	Standard Deviation
Global Otsu (Figure 4-4d)	1.383	0.87
Locally Otsu (Figure 4-4e)	1.653	0.962
Adaptive (Figure 4-4f)	1.199	0.794

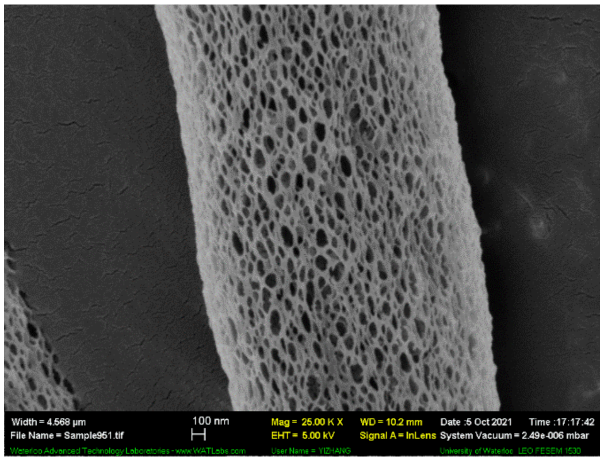
Despite lack of ground of validation, the preceding knowledge is still useful for quality inspection or comparison of the quality of the nanofiber mats. For example, it can be used to check the maximum hole size of the fiber. However, in most engineering application, a pore size distribution is less important than the information of the largest pores, where unwanted leakage and penetration of working fluids start. For example, when nanofiber mats are used as lithium-ion battery separators, the battery separators are designed to separate battery's anode and cathode, preventing short circuit and allow ion transport at the same time. To achieve this goal, the maximum pore size of Li-ion battery separators usually is carefully controlled to below 1 μm (Wang et al., 2016). In general, the separators must have pores smaller than the particle size of the electrode components. The average pore size of a battery separator is normally in the range of 0.03 μm to 0.1 μm. As another example, when nanofiber mats are used for air filtration, the maximum pore size may also determine the smallest particle size that can be captured. Therefore, in the field of air filtration, maximum pore size is an important indicator.

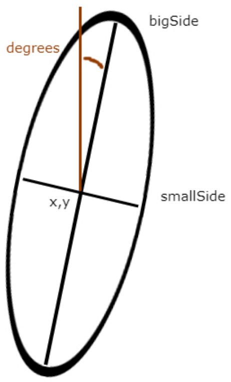
4.2 Surface intra-fiber pores determination

4.2.1 Methodology for Surface intra-fiber pores determination

Figure 4-5 shows a sample nanofiber with intra-fiber pores. This sample nanofiber was obtained by electrospinning of cellulose acetate solution using acetone and DCM as a solvent. Nanofiber with intra-pores can also be characterized with the method introduced in Section

4.1, which gives the equivalent circle diameter of the same area. In many cases, however, the pores on nanofiber are of regular, often oval, shape. Thus, the small side of the oval is taken as the size of size of the pore (*see* Figure 4-5b) for the ease of manual measurement. This is considered as the best technology available for irregular pores between nanofibers.

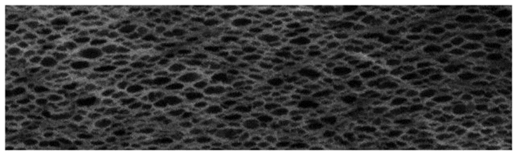

Width = 4.568 µm
File Name = Sample951.tif
100 nm
Mag = 25.00 K X
EHT = 5.00 kV
WD = 10.2 mm
Signal A = InLens
Date :5 Oct 2021
Time :17:17:42
System Vacuum = 2.49e-006 mbar
Waterloo Advanced Technology Laboratories - www.WATLabs.com
User Name = YIZHANG
University of Waterloo LEO FESEM 1530



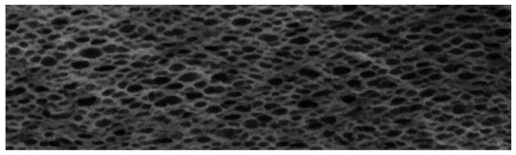
a b

Figure 4-5. a) SEM image of nanofibers with intra-fiber pores; b) a model pore (the small side stands for the size of the pore)

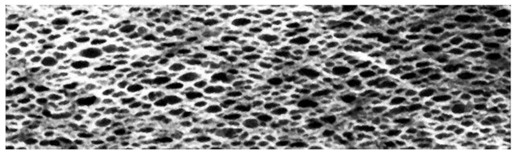
Figure 4-6 illustrates the procedure for the determination of surface intra-fiber pores in this thesis work. This procedure is inspired by the diameter determination procedure in Section 3.1, using the same Canny edge detection for image segmentation. Conversely, there are two main differences. First, the pore size determination procedure in this current section uses only Canny edge detection for boundary determination. In contrary, the diameter determination procedure in Section 3.1 uses both the Canny edge detection and image thresholding method for image segmentation. Another difference is the requirement of image size. The pore size determination procedure requires manual pre-crop of the area full of pores. The cropped images may be different in shape and size. Therefore, the script does not require on the image size to maintain its adaptability. However, the script for diameter measurement imports the image size to obtain certain information from the image. Thus, only image with fix size can be used as input; otherwise, the parameters in the script must be revised accordingly.



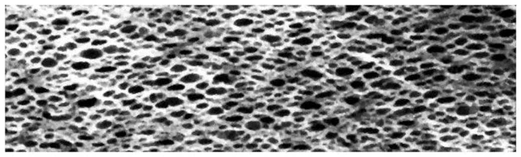
(a) Cropping and resizing



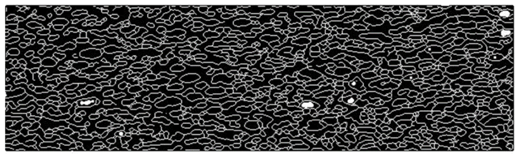
(b) Median filtering



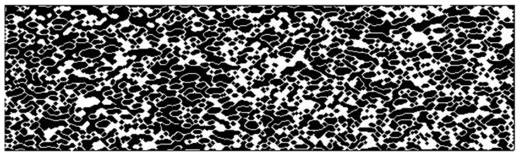
(c) Histogram equalization



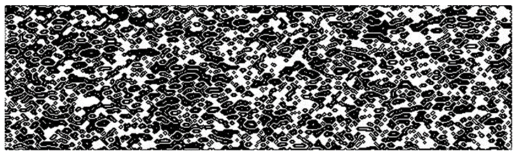
(d) Opening and closing



(e) Canny edge detection



(f) Dilation and thinning



(g) Skeletonization and distance transformation

Figure 4-6. The image processing procedure for intra-fiber pores size determination

The pore size determination procedure in Figure 4-6 consists of the following seven steps, which are explained one by one.

a) Cropping and resizing

An original SEM image usually contains multiple fibers with intra-fiber pores. The algorithm cannot distinguish the boundaries between the fibers and the boundaries between intra-pores and outer fibers. Therefore, it is necessary to identify and manually crop out the image part that contains only one or part of a porous fiber like the one shown in Figure 4-6a.

b) Median filtering

Median filtering changes each pixel value with the median of its neighbors by scanning the image pixel by pixel. It denoises the image and the resultant image is shown in Figure 4-6b. However, Figure 4-6a and b appear to be almost the same. The reason is that the image contains no fiber-less background like a SEM image for fiber mat does. Therefore, the benefit of this step is marginal. As a result, the end user can choose to omit this step.

c) **Histogram equalization**

Histogram equalization increases the image contrast to distinguish the boundary between pores in the fiber. The resultant image is shown in Figure 4-6c.

d) **Opening and closing**

An "opening and closing" operation is performed for image denoising again. The resultant image is shown in Figure 4-6d. Similarly, Figure 4-6c and d appear to be identical. Again, the reason is that the image is filled with a fiber without fibreless background. Therefore, the benefit of this step is marginal too. As a result, the end user can choose to omit this step.

e) **Canny edge detection**

The boundaries of the pores are identified using the canny edge detection method, and the resultant image becomes Figure 4-6e.

f) **Dilation and thinning**

This the same as Step 8 on page 29 introduced in Section 3.1. The resultant image becomes Figure 4-6f.

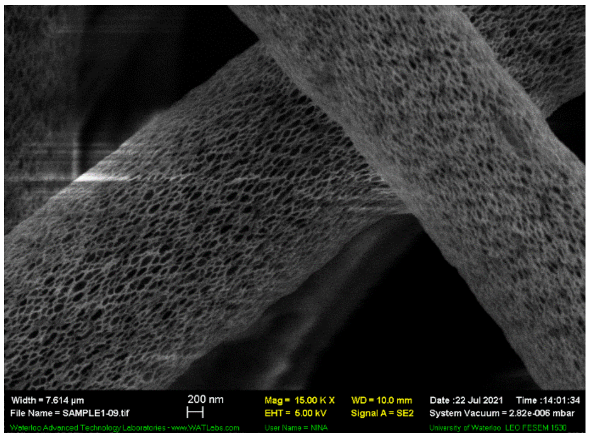
g) **Skeletonization and distance transformation**

They are the same as Steps 9 and 10 introduced in Section 3.1. The image is skeletonized to find the centerline of the pores, and the distance between the centerline to the boundary is measured using Euclidean transformation. The resultant image becomes Figure 4-6g.

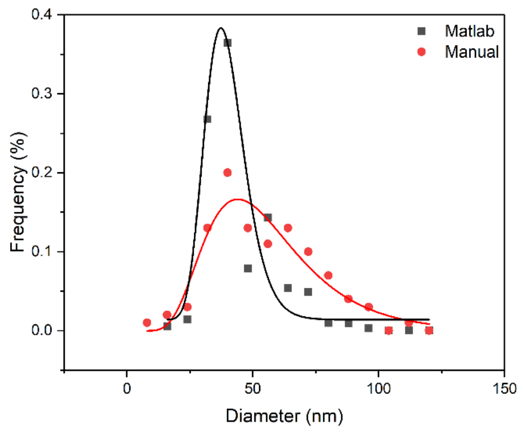
4.2.2 Results of surface intra-fiber pores determination

1) Preliminary trial with 100-sample manual measurement

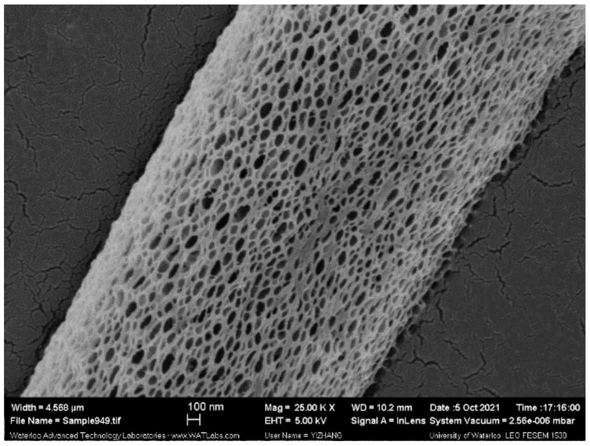
Figure 4-7 compares the pore size distributions obtained by automated image processing and those and manually. The black lines are for the results obtained by automated image processing using MATLAB, and the red lines are for the results obtained by manual measurement using ImageJ. Regarding the manual measurements, more than 100 sample pores are randomly taken from each sample. To ensure randomness of sampling, the subject image is first divided into 100 equal areas with equal lengths and widths. Then, a random location is sampled in each area. In this project, the manually measured result are regarded as ground truth. Finally, the *x* axis is size of pore (in nm) about 10 and the *y* axis is the corresponding diameter's frequency.



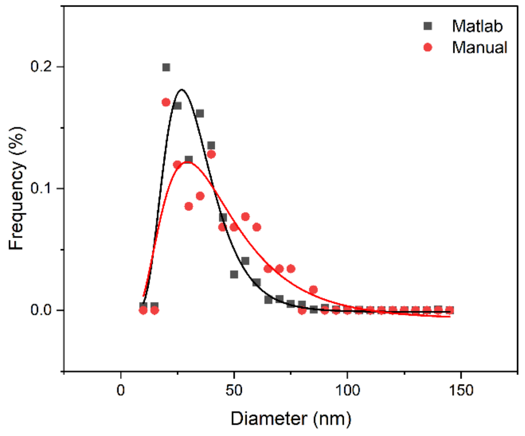
(a) Sample 1



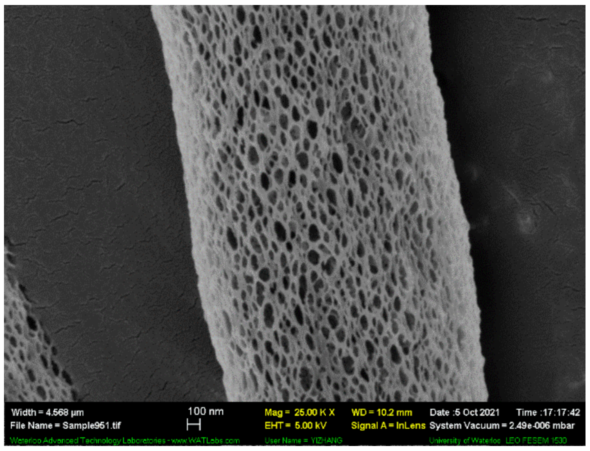
(b)



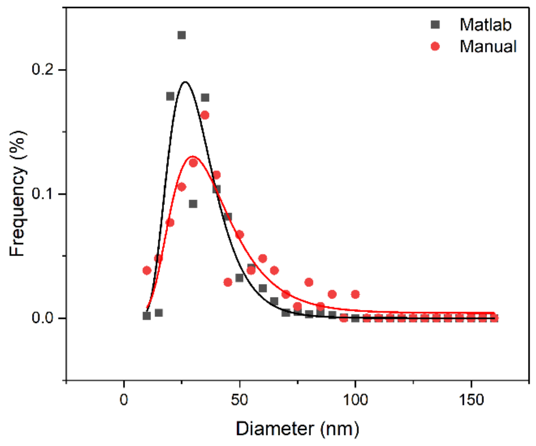
(c) Sample 2



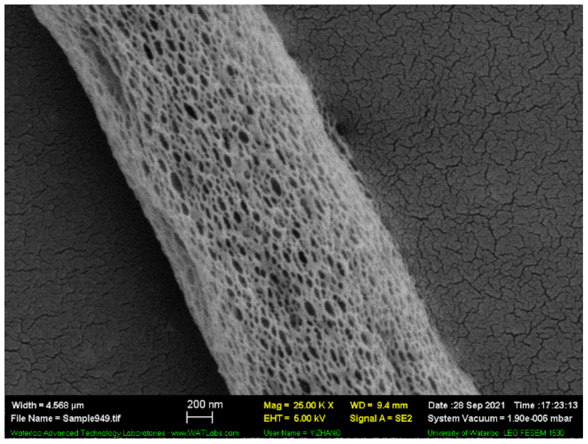
(d)



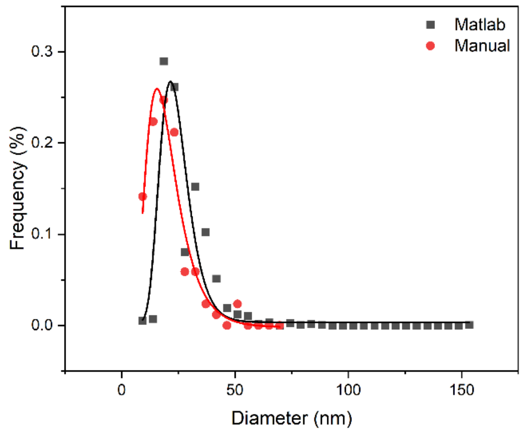
(e) Sample 3



(f)



(g) Sample 4



(h)

Figure 4-7. The original SEM images of four porous nanofiber samples and their corresponding pore size distributions obtained by image processing

The curves in Figure 4-7 show that the MATLAB and manual measurements provide similar results. However, there are also differences. The manual measurements generally have a greater standard deviation than the MATLAB based image processing result. Meanwhile, the pore size distributions generated by image processing usually have higher peaks than the manual measurements do. In addition, the distribution curves of manually measurements appear to be smoother than those obtained using image processing.

To quantify the preceding differences, the mean and standard deviation of diameter are calculated and summarized in Table 4-2. The deviation between the MATLAB and manual measurements of are calculated using Eq. (4-1.

$$d = \frac{|\mu_{Manual} - \mu_{MATLAB}|}{\mu_{Manual}} \tag{4-1}$$

where d is the deviation of results, μ_{Manual} is the mean value of the distribution obtained manually, and μ_{MATLAB} is the mean value of the distribution obtained by image processing using MATLAB.

Table 4-2. Comparison of pore size distributions result from image processing and manual measurements

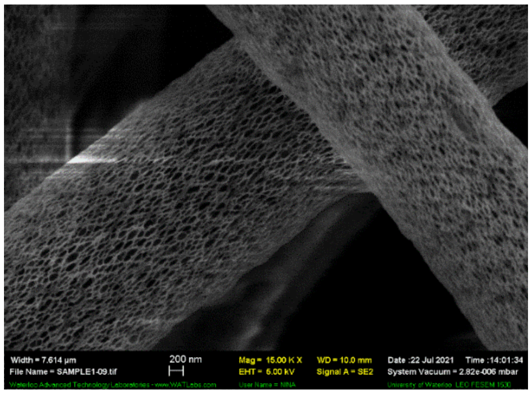
Sample#	MATLAB Method		Manual Method 100 sample		Deviation (Eq. (4-1)	
	Mean (nm)	Stdev (nm)	Mean (nm)	Stdev (nm)	Of mean	Of Stdev
1	43.41	4.42	59.18	20.26	0.27	0.78
2	36.17	10.74	42.54	24.31	0.15	0.56
3	34.83	8.54	39.82	11.33	0.13	0.25
4	26.66	4.42	22.54	9.60	0.18	0.54

Table 4-2 shows that the deviation between the mean values of manual and automated measurements varies from 0.13-0.27. The great deviation result from the small pore sizes: a difference of few nanometers may lead to a significant deviation. The deviations of the results from the two different methods vary widely mainly because the magnification and size of nanofibers in images. The smaller size makes it much harder to distinguish the edge of the pore and brings more difficulties for the manual measurement.

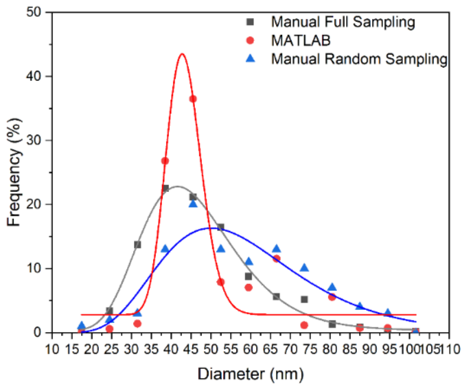
This preliminary test reveals a problem: the accuracy and reliability of the image processing methods is not validated by the manual measurement because of the limited (*i.e.* 100) manual sampling points. Considering the sample size, the manual measurement cannot be used as the ground truth, unless all pores are manually measured. This leads to the next section.

2) Comparison of full manual measurement and image processing results.

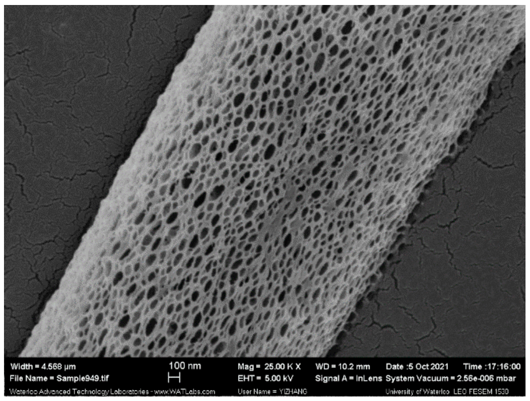
To improve the reliability of manual measurement, the pore sizes were manually measured again by counting all pores in the images, which is referred to as Manual Full Sampling. The sizes of all pores in the nanofiber samples are measured, and the full sampling is expected to eliminate the basis of insufficient number of sampling points. Figure 4-8 shows the results of full sampling manual measurements.



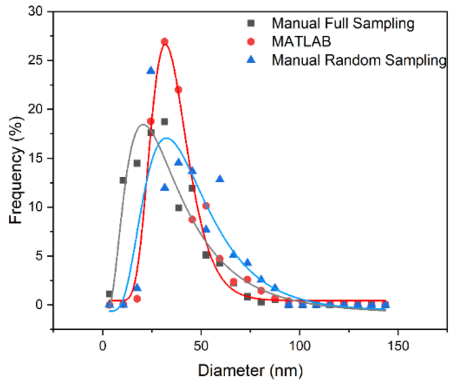
(a) Sample 1



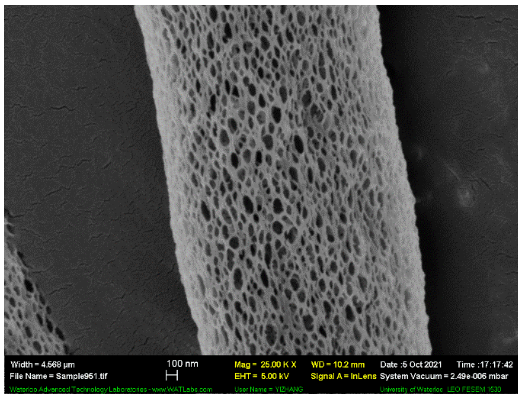
(b) Sample 1 pore size distributions



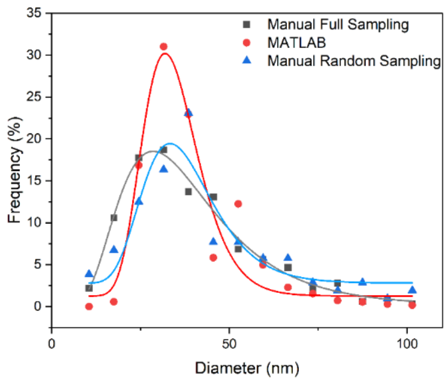
(c) Sample 2



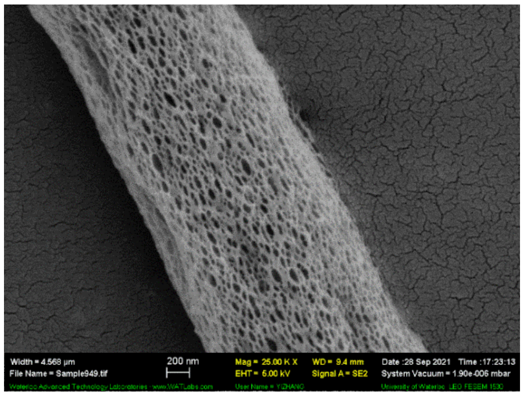
(d) Sample 2 pore size distributions



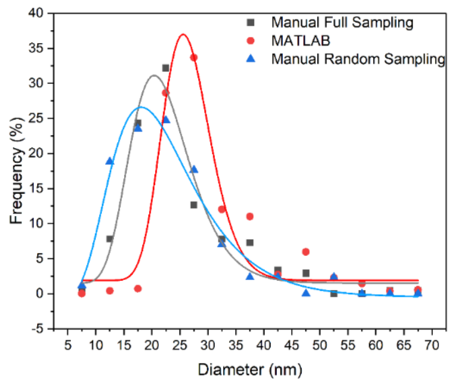
(e) Sample 3



(f) Sample 3 pore size distributions



(g) Sample 4



(h) Sample 4 pore size distributions

Figure 4-8. The comparison of full sampling manual measurements, 100-sample manual measurement, and MATLAB results

Figure 4-8 also shows lognormal distributions. To quantitative describe the pore diameter distribution, the coefficient of determination, $\boldsymbol{R^2}$, is calculated and shown in Table 4-3. The $\boldsymbol{R^2}$ for the data obtained using MATLAB (*i.e.*, image processing) varies from 0.9134 to 0.9626. The $\boldsymbol{R^2}$ for manual measurements varies from 0.8513 to 0.9906. Correlation with an $\boldsymbol{R^2}$ greater than 0.8 can be regarded as statistically significant. Therefore, the $\boldsymbol{R^2}$ values verified that the distribution of nanofiber pore size are lognormally distributed. In addition, the measurement with full manual sampling has a higher $\boldsymbol{R^2}$ that those using 100 samples. It is likely because the full sampling manual results take more data points and form a smoother distribution curve, giving a more comprehensive presentation of the data.

Table 4-3 The coefficient of determination for diameter frequency's fitting curve

Sample #	***R^2***		
	MATLAB	**Manual 100 sample**	**Manual full sample**
1	0.9134	0.8628	0.9906
2	0.9626	0.7951	0.9371
3	0.9202	0.8513	0.9826
4	0.9397	0.9517	0.9686

To quantify the accuracy of the measurements, the mean diameter and standard deviation from each method is calculated and summarized in Table 4-4. The deviation between the mean values of full manual and automated measurements varies from 0.07 to 0.16, which are much lower than the values of 0.13-0.27 in Table 4-2. The reduced deviations indicate an improved accuracy and reliability in terms of mean pore sizes of the automated measurements.

Figure 4-9 shows a parity plot comparing the mean fiber diameters given by MATLAB and two manual methods for the four samples. The results demonstrate that the manual and automated methods agree well in terms of mean pore size. Comparing the two manual methods shows that the full sample result has a linear coefficient of 1.0592, which is closer to 1 than that the 100-sample manual result, 1.19. It also confirms that the MATLAB result is closer to the full sampling result.

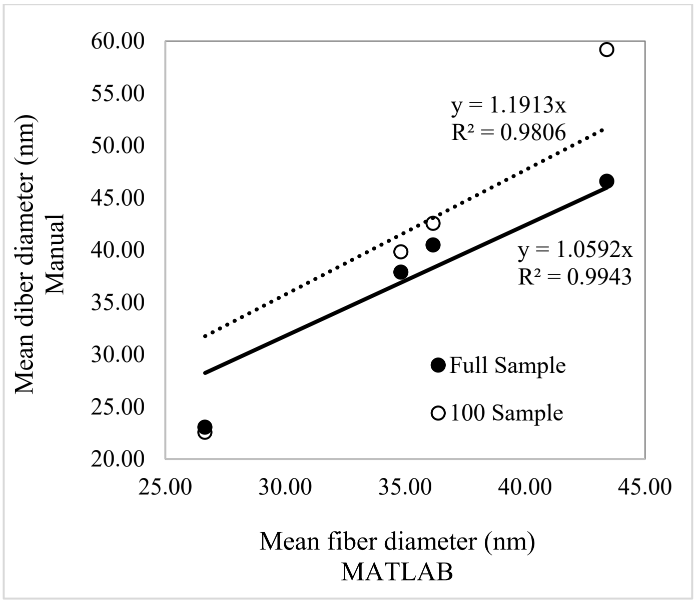


Figure 4-9. Correlation between the mean fiber diameters for automated and manual methods

On the other hand, the standard deviations of two methods still vary significantly, indicting the shape of the curves do not agree with each other. This disagreement in the curve shapes is obvious in Figure 4-8. The distribution curve of manual method result has a relatively smooth curve, and the distribution curve of the MATLAB method has a much steeper curve. This problem deserves attention it the future works.

Table 4-4 The comparison of mean and standard deviations from different methods

Sample#	MATLAB Method		Manual Method Full sample		Deviation (Eq. (4-1)	
	Mean (nm)	Std (nm)	Mean (nm)	Std (nm)	Of Mean	Of Std
1	43.41	4.42	46.57	13.04	0.07	0.66
2	36.17	10.74	40.48	30.52	0.11	0.65
3	34.83	8.54	37.88	19.83	0.08	0.57
4	26.66	4.42	23.04	5.87	0.16	0.25

Admittedly, this automated method for intra-fiber pore size characterization does have limitations. In addition the inaccuracy mentioned in the preceding paragraphs, it does not calculate the area of the intra-fiber pores and only gives a diameter number. The result would be accurate or at least have a reference value when the shape of the pores is circle, oval or in a slender narrow shape. For circular and oval pores, the result is its diameter or the short diameter. For slender narrow pores, the number is the distance between narrow sides. However, for very irregular hole shapes, the result will have a large deviation. This is mainly due to the algorithm itself.

4.3 Porosity Determination

4.3.1 Methodology for porosity determination

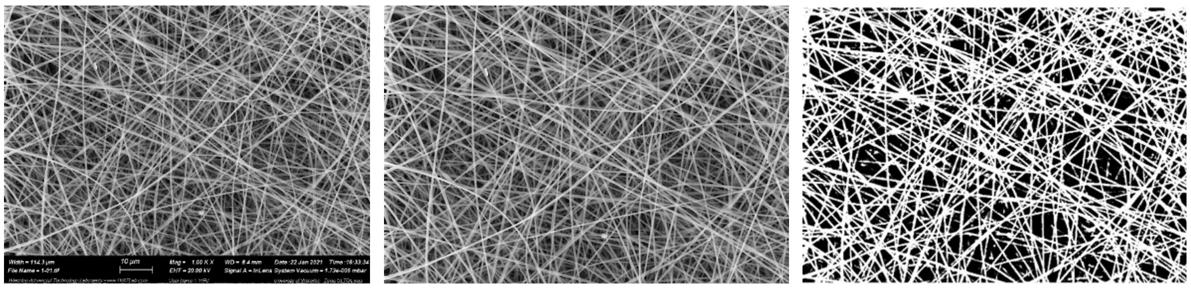
The procedure of nanofiber porosity determination consists of three steps:

1) image cropping
2) image binarization
3) ratio calculation.

As shown in Figure 4-10, the original SEM image in Figure 4-10a is first cropped to remove the title bar, resulting in Figure 4-10b. Binarization converts the original grayscale SEM image into a binary image that only has two colors, black and white. In grayscale images, all colors can be regarded as different shades of gray and are represented by a number in the range 0 to 255 (under 8-bit format) in computer. During the binarization process, a threshold value is chosen. All values smaller than the threshold value are regarded as 0, pure black, and all values larger than the threshold value are regarded as 255, pure white. Then, a grayscale image is converted into a binary image. During the binarization, a key step is to determine an appropriate threshold value to separate the fiber from the background. The final step is to calculate the porosity using the equation below. With a properly defined binarization threshold, the estimated porosity should be close to the experimental result.

$$P = \frac{N_{wh}}{N_{tot}} = 1 - \frac{N_{blk}}{N_{tot}} \tag{4-2}$$

where P is the porosity, N_{wh} is the number of white pixels in the image after binarization, N_{blk} is the number of black pixels in the image after binarization, and N_{tot} is the number of total pixels in the image.



(a) Original image (b) Cropped image (c) Binarized image

Figure 4-10. The image processing procedure for porosity determination

Section 2.3.1 lists three case studies on this topic that follow similar steps. Those three case studies use three different criteria for the threshold value chosen. In this study, all thresholding chosen criteria mentioned in the three previous studies are performed. Results obtained with different thresholding values are compared to the experimental result, which served as a ground truth here. The three studies in Section 2.3.1 and the thresholding chosen criteria they use are explained as follows.

1) **85% average grayscale**

   Wang et al. (2020) used 85% average grayscale as the image thresholding value. The thresholding value can be expressed in Eq. (**4-3**:

$$T_1 = 85\%\bar{g} \tag{4-3}$$

   where T_1 means threshold value, $\bar{g}$ is the average grayscale value.

2) **Average grayscale and average grayscale ± standard deviation**

   Ghasemi-Mobarakeh et al. (2007) used three thresholding values for porosity determination, the average grayscale, average grayscale plus standard deviation, and average grayscale minus standard deviation.

$$T_1 = 85\%\bar{g} \tag{4-4}$$

$$T_2 = 85\%\bar{g} + g_{std} \tag{4-5}$$

$$T_3 = 85\%\bar{g} - g_{std} \tag{4-6}$$

where g_{std} is the standard deviation of grayscale.

3) Globally Otsu method

Sun et al. (2007) used the Otsu method for the threshold value determination. Otsu method is a common method used in image processing and computer vision that uses the intra-class intensity variance to separate pixels. In the process of binarization, the image pixels are separated into two classes, the foreground and background. The Otsu method would find out the threshold value at which the intra-class intensity variance is minimized, or equivalently, inter-class variance is maximized. The sum of variances of the two classes can be calculated using the equation below:

$$\sigma_\omega^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t) \tag{4-7}$$

where $\omega_0(t)$ and $\omega_1(t)$ are the probability a pixel is classified in class 0 and class 1 when the threshold value is t, $\sigma_0^2(t)$ and $\sigma_1^2(t)$ are the inter-class variance of two classes. The Otsu method will return the t value at which $\sigma_\omega^2(t)$ is minimized.

The threshold value needs to be determined first before the porosity can be calculated. Twelve SEM images are selected in different batches and different magnification levels for a more comprehensive examination. The description of their naming scheme is available in Table 4-4. The first digit in the naming number represents which sample the image comes from. Images with the same first number but a different last number in the name mean these pictures are from the same sample but at different magnification. Same batch of sample means these samples are fabricated at the same time under the same experimental conditions and used as parallel samples.

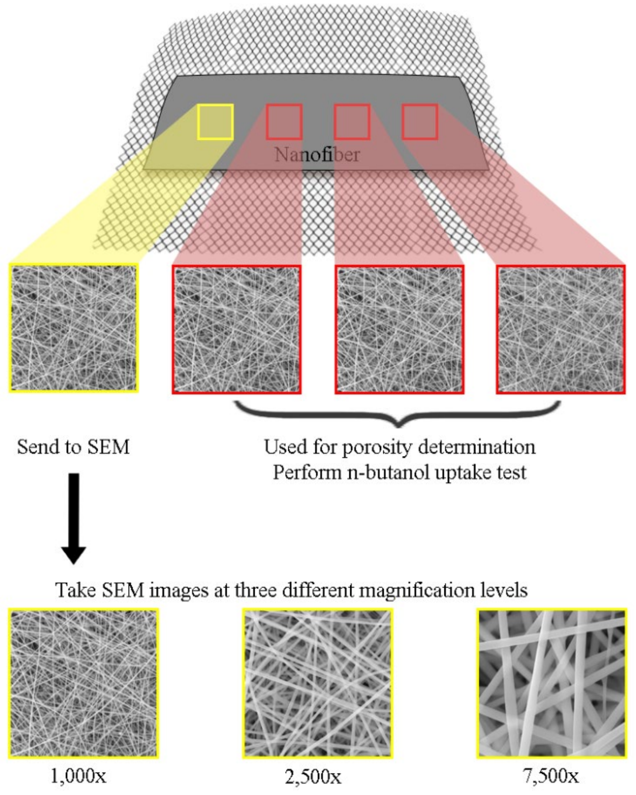
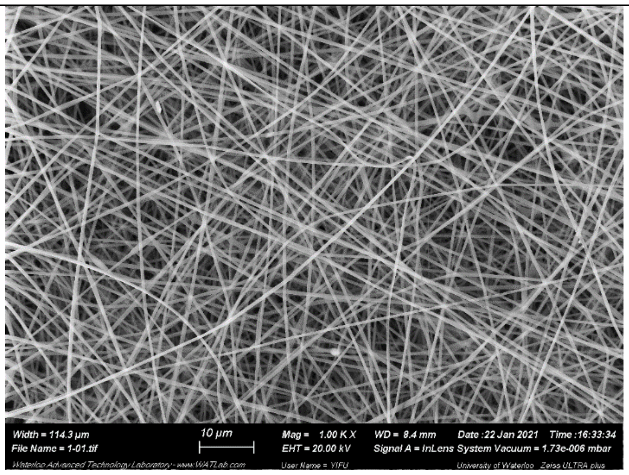
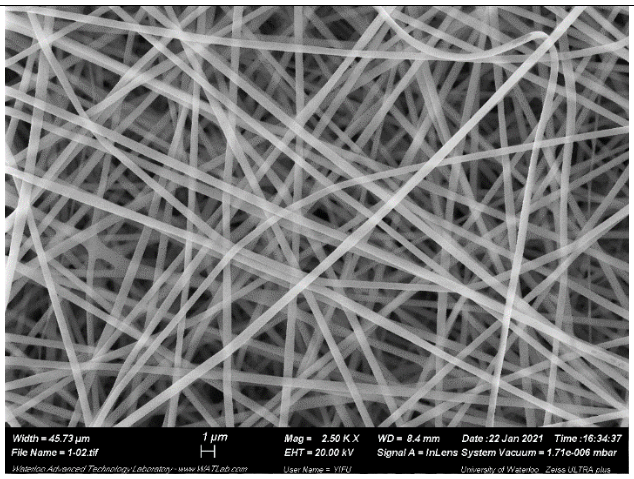
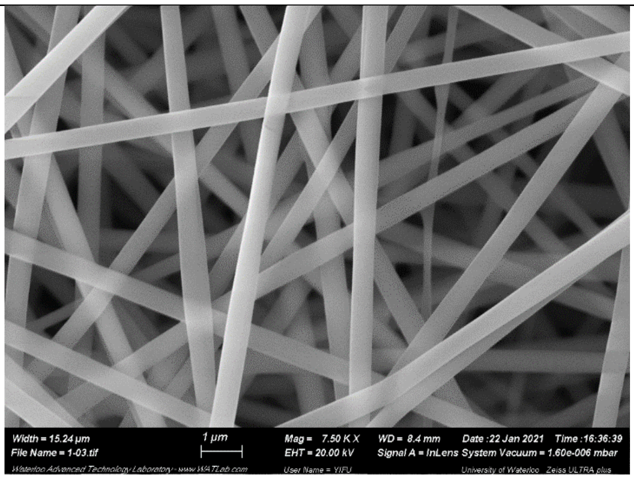
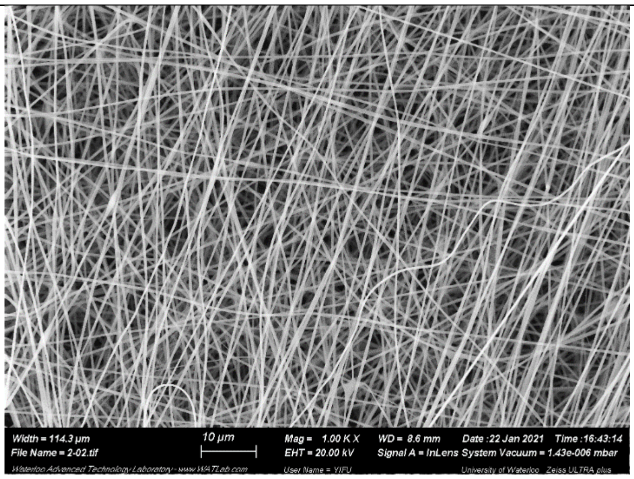
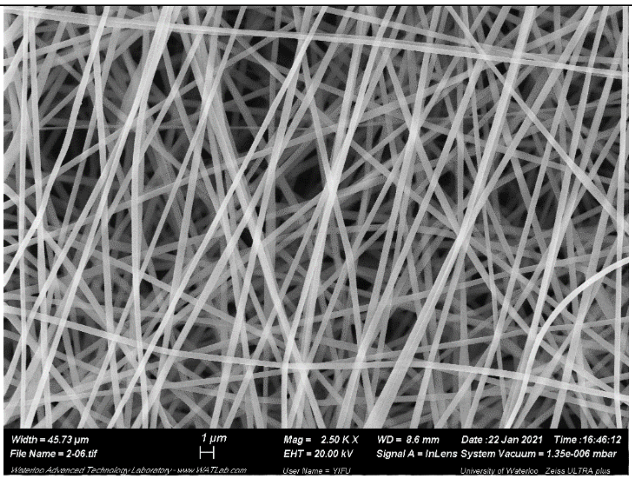
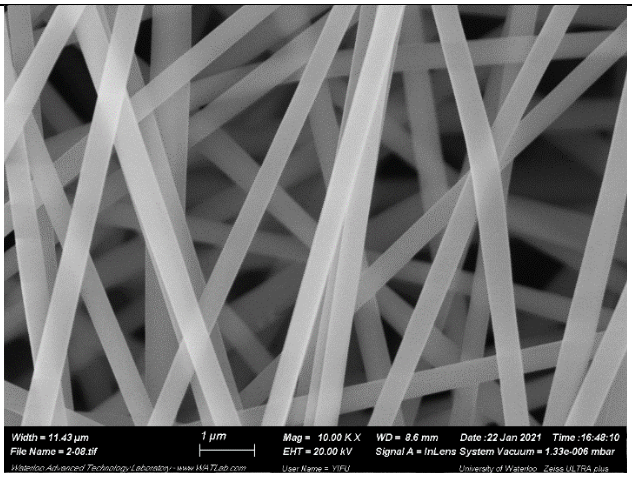
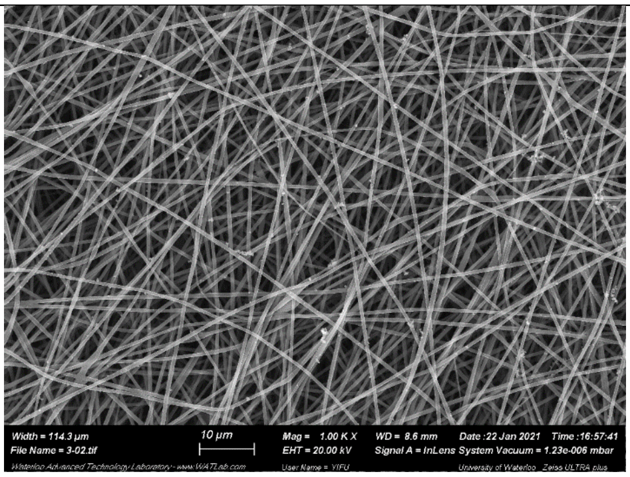

Nanofiber
Send to SEM
Used for porosity determination
Perform n-butanol uptake test
Take SEM images at three different magnification levels
1,000x
2,500x
7,500x

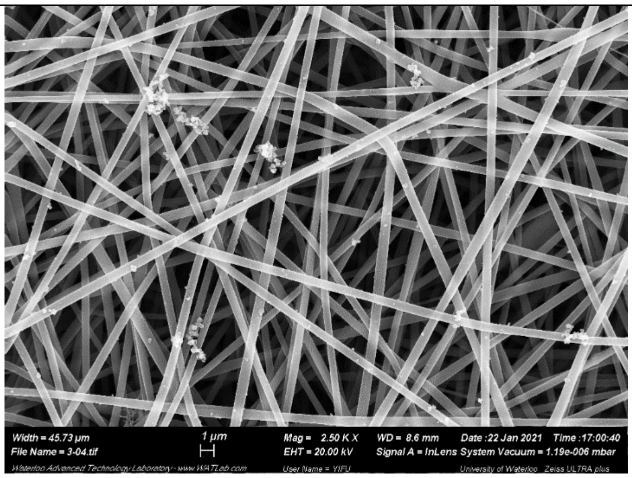
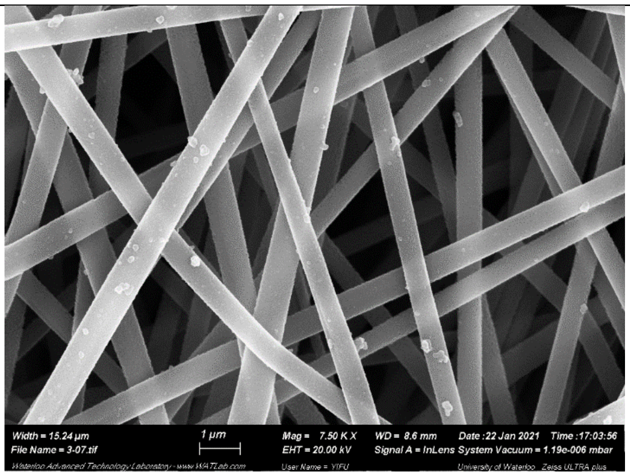
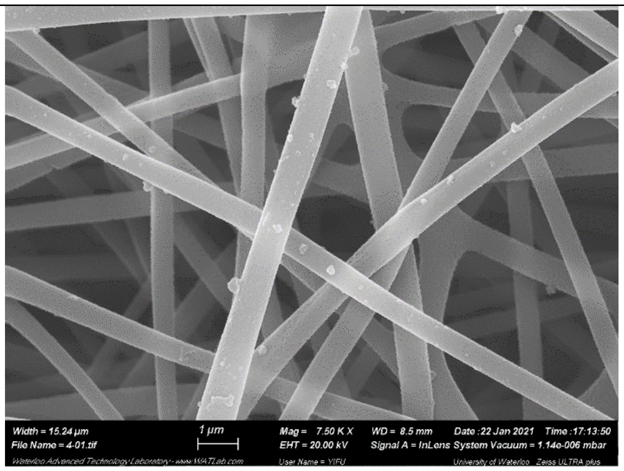
Figure 4-11 procedure for porosity determination

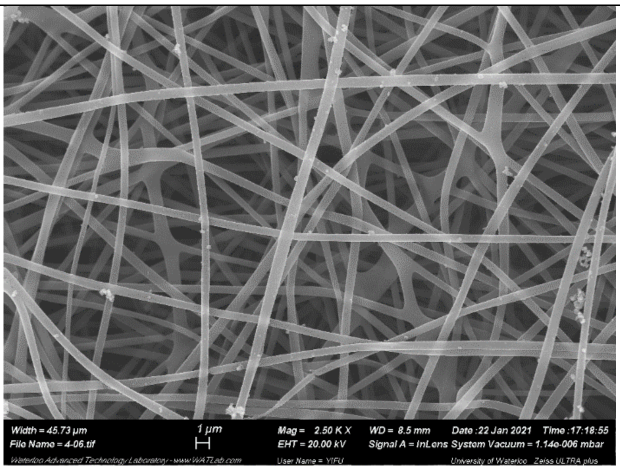
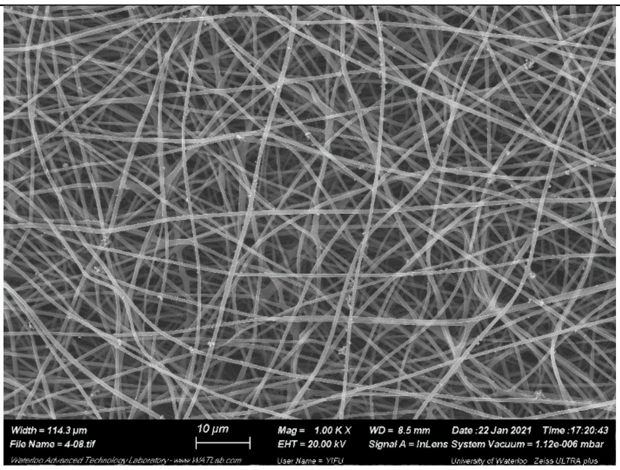
Table 4-5 Descriptions for selected SEM images

Sample #	Description	SEM Image
SEM 101	Nanofiber Mat A, Sample 1, at magnification 1.00 kx	

SEM 102	Nanofiber Mat A, Sample 1, at magnification 2.50 kx	 Width = 45.73 µm File Name = 1-02.tif 1 µm Mag = 2.50 K X EHT = 20.00 kV WD = 8.4 mm Signal A = InLens Date :22 Jan 2021 Time :16:34:37 System Vacuum = 1.71e-006 mbar Waterloo Advanced Technology Laboratory - www.WATLab.com User Name = YIFU University of Waterloo Zeiss ULTRA plus
SEM 103	Nanofiber Mat A, Sample 1, at magnification 7.50 kx	 Width = 15.24 µm File Name = 1-03.tif 1 µm Mag = 7.50 K X EHT = 20.00 kV WD = 8.4 mm Signal A = InLens Date :22 Jan 2021 Time :16:36:39 System Vacuum = 1.60e-006 mbar Waterloo Advanced Technology Laboratory - www.WATLab.com User Name = YIFU University of Waterloo Zeiss ULTRA plus
SEM 202	Nanofiber Mat A, Sample 1, at magnification 1.00 kx	 Width = 114.3 µm File Name = 2-02.tif 10 µm Mag = 1.00 K X EHT = 20.00 kV WD = 8.6 mm Signal A = InLens Date :22 Jan 2021 Time :16:43:14 System Vacuum = 1.43e-006 mbar Waterloo Advanced Technology Laboratory - www.WATLab.com User Name = YIFU University of Waterloo Zeiss ULTRA plus

SEM 206	Nanofiber Mat A, Sample 1, at magnification 2.50 kx	 Width = 45.73 µm File Name = 2-06.tif 1 µm Mag = 2.50 K X EHT = 20.00 kV WD = 8.6 mm Signal A = InLens Date :22 Jan 2021 Time :16:46:12 System Vacuum = 1.35e-006 mbar Waterloo Advanced Technology Laboratory - www.WATLab.com User Name = YIFU University of Waterloo Zeiss ULTRA plus
SEM 208	Nanofiber Mat A, Sample 1, at magnification 10.00 kx	 Width = 11.43 µm File Name = 2-08.tif 1 µm Mag = 10.00 K X EHT = 20.00 kV WD = 8.6 mm Signal A = InLens Date :22 Jan 2021 Time :16:48:10 System Vacuum = 1.33e-006 mbar Waterloo Advanced Technology Laboratory - www.WATLab.com User Name = YIFU University of Waterloo Zeiss ULTRA plus
SEM 302	Nanofiber Mat B, Sample 1, at magnification 1.00 kx	 Width = 114.3 µm File Name = 3-02.tif 10 µm Mag = 1.00 K X EHT = 20.00 kV WD = 8.6 mm Signal A = InLens Date :22 Jan 2021 Time :16:57:41 System Vacuum = 1.23e-006 mbar Waterloo Advanced Technology Laboratory - www.WATLab.com User Name = YIFU University of Waterloo Zeiss ULTRA plus

SEM 304	Nanofiber Mat B, Sample 1, at magnification 2.50 kx	 Width = 45.73 µm 1 µm Mag = 2.50 K X WD = 8.6 mm Date :22 Jan 2021 Time :17:00:40 File Name = 3-04.tif EHT = 20.00 kV Signal A = InLens System Vacuum = 1.19e-006 mbar Waterloo Advanced Technology Laboratory - www.WATLab.com User Name = YIFU University of Waterloo Zeiss ULTRA plus
SEM 307	Nanofiber Mat B, Sample 1, at magnification 7.50 kx	 Width = 15.24 µm 1 µm Mag = 7.50 K X WD = 8.6 mm Date :22 Jan 2021 Time :17:03:56 File Name = 3-07.tif EHT = 20.00 kV Signal A = InLens System Vacuum = 1.19e-006 mbar Waterloo Advanced Technology Laboratory - www.WATLab.com User Name = YIFU University of Waterloo Zeiss ULTRA plus
SEM 401	Nanofiber Mat B, Sample 1, at magnification 7.50 kx	 Width = 15.24 µm 1 µm Mag = 7.50 K X WD = 8.5 mm Date :22 Jan 2021 Time :17:13:50 File Name = 4-01.tif EHT = 20.00 kV Signal A = InLens System Vacuum = 1.14e-006 mbar Waterloo Advanced Technology Laboratory - www.WATLab.com User Name = YIFU University of Waterloo Zeiss ULTRA plus

SEM 406	Nanofiber Mat B, Sample 1, at magnification 2.50 kx	 Width = 45.73 µm File Name = 4-06.tif Waterloo Advanced Technology Laboratory - www.WATLab.com 1 µm Mag = 2.50 K X EHT = 20.00 kV User Name = YIFU WD = 8.5 mm Signal A = InLens Date :22 Jan 2021 System Vacuum = 1.14e-006 mbar University of Waterloo Zeiss ULTRA plus Time :17:18:55
SEM 408	Nanofiber Mat B, Sample 1, at magnification 1.00 kx	 Width = 114.3 µm File Name = 4-08.tif Waterloo Advanced Technology Laboratory - www.WATLab.com 10 µm Mag = 1.00 K X EHT = 20.00 kV User Name = YIFU WD = 8.5 mm Signal A = InLens Date :22 Jan 2021 System Vacuum = 1.12e-006 mbar University of Waterloo Zeiss ULTRA plus Time :17:20:43

To start with, SEM 101 is used as an example to show the difference between the threshold value from different algorithms. The determined threshold values are presented below in Table 4-5. The thresholding value obtained varies from 72.097 to 121.436. The difference in numbers is huge, and it can be easily predicted that choosing different threshold values will result in different porosity results. The result is presented and discussed in the next section.

Table 4-6. The threshold value SEM 101

Sample 101	**Threshold Value**
Average grayscale	121.436
85% grayscale	103.221
Otsu	117.450

Standard deviation of grayscale	49.340
Average plus deviation	72.097
Average minus deviation	170.776

4.3.2 Result of porosity determination

The porosity determination results of different samples using different methods are shown in Table 4-7. The experimental porosities in Table 4-7 are obtained using n-butanol uptake test. The main idea of the uptake test is to measure the mass of the n-butanol that the nanofiber can absorb and convert the absorbed n-butanol mass into volume. The n-butanol volume is also the pore volume in the nanofiber. Then the porosity is obtained by dividing the pore volume by the nanofiber volume. In the uptake test, the porosity can be calculated using the following equation:

$$P(\%) = \frac{W_w - W_d}{p_b V} \tag{4-8}$$

where P is the fiber porosity, W_w is the weight of wet fibers, W_d is the weight of dry fibers, p_b is the density of n-butanol, and V is the geometric volume of the fiber.

The experimental porosity is regarded as the ground truth in this study. By examining Table 4-7, the following conclusions can be drawn. Firstly, results in Table 4-7 show that the image magnification has little influence on the porosity result. For example, SEM 101, 102, 103 are from on single sample and the difference between their porosity result is no more than 2%, as long as the same thresholding criteria is used. This phenomenon was also observed in other samples. When the same thresholding criteria is used, no matter which threshold criteria is using, images from the same fiber sample always generate relatively consistent results despite the magnification difference.

For different threshold criteria, as expected, a larger threshold value would result in a higher porosity. In Table 4-6, the threshold value of descending in order of average grayscale, Otsu method and 85% grayscale. Correspondingly, the porosity obtained using these three methods also decreasing in this order. Among all threshold criteria mentioned above, mean grayscale plus deviation has the largest value and result in the highest porosity. For example, for sample2, from image SEM 202, 206 and 208, the mean grayscale plus deviation method has an average of 81.81, as of the rest methods are providing results under 50.

Under the same method, the output porosity for one sample is very consistent, but it can be far from accurate compared to the experimental result. The image processing results retrieved from 85% mean grayscale method, Otsu method and mean minus deviation method are all smaller than the experimental results. For SEM 302, 304 and 307, the 85% mean grayscale reported an average of 43.44, while the Otsu method reported an average of 51.38. On the other hand, the mean minus deviation reported an average of 20.84 and the mean plus deviation reported an average of 80.46. Among all four porosity values, the mean plus deviation has an answer that falls inside the uncertainty range of 79.3±7.1 which is the experimental result. Thus, a conclusion can be made, using mean plus deviation as threshold can obtain the result that has the closest values to the experimental results.

Table 4-7. The result of porosity using different binarization method and experimental value

Mat #	**Sample #**	**Sample #**	**Experimental Porosity**	**85% Mean greyscale porosity (%)**	**Ostu porosity (%)**	**Mean minus deviation porosity (%)**	**Mean plus deviation porosity (%)**
A	A1	SEM x1000		36.61	45.43	19.36	82.12
		SEM x2500		35.38	44.77	19.47	82.49
		SEM x7500		37.81	45.12	20.63	81.85
		SEM x1000		36.27	44.12	19.54	81.59
		SEM x2500		36.11	42.36	20.87	81.66
		SEM x7500		36.26	45.19	20.34	82.19
	A2	Porosity	84.95				

	A3	Porosity	79.15				
	A4	Porosity	93.05				
B	B1	SEM x1000		42.37	52.07	20.46	81.36
		SEM x2500		43.46	50.51	21.33	79.24
		SEM x7500		43.49	51.55	20.72	80.79
		SEM x1000		38.74	57.5	21.84	80.93
		SEM x2500		39.11	58.06	20.36	80.37
		SEM x7500		38.89	56.2	20.52	81.48
	B2	Porosity	86.23				
	B3	Porosity	79.62				
	B4	Porosity	71.96				

Table 4-8 shows each sample's average image processing porosity when mean plus deviation is used as threshold and the deviation between the experimental and image processing results. It is noticed that the deviation varies from 1.47 % to 4.54 %. Considering that the image processing method is only a simple and preliminary method, this deviation can be considered acceptable.

Table 4-8. The deviation between image processing result and experimental result

Sample	Average grayscale plus deviation Porosity (%)	Experimental (%)	Deviation (%)
101, 102, 103	82.15	85.7±7	4.14
201, 202, 203	81.81		4.54
301, 302, 303	80.46	79.3±7.1	1.47

401, 402, 403	80.93		2.05

As it mentioned above, serval studies have tried this image processing method for porosity determination. Most of the studies only discuss the trends in results and only one of them, the research made by Wang et al., made a quantitative statement on the result.

Wang et al. (2020) state that the ratio between experimental and image processing results is the second Feigenbaum constant when using 85% grayscale as the threshold criteria. The second Feigenbaum constant is a term in in mathematics. This constant expresses the ratios in a bifurcation diagram for a non-linear map (Briggs 1991). The second Feigenbaum constant α has a value of around 2.5029. Though I doubt how does a constant in bifurcation theory related to the image analysis of nanofibers, I still calculated the ratio between image processing porosity and experimental porosity in Wang et al.'s way using data from our research group. I did so since it is the only quantitative statement I found in similar research.

The experimental porosity, image processing porosity and the ratio of the porosity results obtained by the two methods is shown in Table 4-9. From Table 4-9, it can be seen that the ratio between two results varies from 1.82 to 2.37. Deviation between the ratio and second Feigenbaum constant α varies from 3.22% to 27.15%. With a deviation as high as 27.15%, I highly doubt whether the relationship between the second Feigenbaum constant α and nanofiber porosity result exists.

Table 4-9. The result of porosity and the ratio between image processing porosity and experimental porosity

Sample	85% Mean Grayscale (%)	Experimental Porosity (%)	Ratio	Deviation (%)	Second Feigenbaum constant α
101	36.61	85.7±7	2.34	6.47	2.5029
102	35.38		2.42	3.22	
103	37.81		2.27	9.44	
202	36.27		2.36	5.60	

206	36.11		2.37	5.18	
208	36.26		2.36	5.57	
302	42.37	79.3±7.1	1.87	25.22	
304	43.46		1.82	27.10	
307	43.49		1.82	27.15	
401	38.74		2.05	18.22	
406	39.11		2.03	18.99	
408	38.89		2.04	18.53	

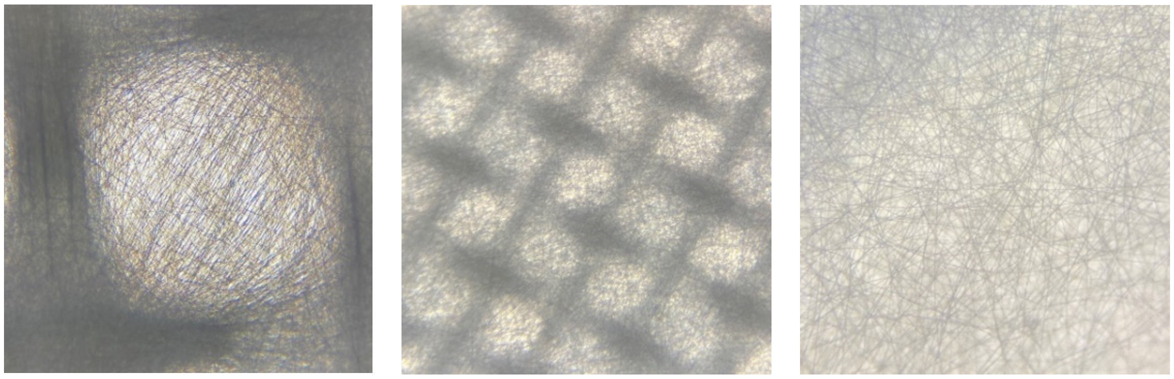
Although Wang et al.'s statement does not apply on my data, their study does bring new insight on data analyze. It is noticed that for different images taken on the same sample, the ratios between the image processing result and experimental result are close in value. For example, the standard deviation is 0.01 for samples 101, 102 and 103. Even under the worst case from sample 302, 304 and 307, they have a standard deviation of 0.028, which is still considered relatively small.

However, this constant ratio has little practical value for nanofiber characterization because the ratios for all fiber samples are different. We cannot know the ratio unless we measure porosity experimentally in advance. Therefore, taking the 85% mean of grayscale as the threshold value and trying to find a fixed ratio between the experimental result and image processing result will not take us anywhere.

4.4 Surface homogeneity determination

This study uses the optical microscope (SW380B, swift, Canada) images of nanofibers for surface homogeneity determination. An example of nanofibers with different homogeneity is shown in Figure 4-12. All three images are polypyrrole nanofiber made under the same experimental conditions but on different sample collectors. In this case, the texture of the different collectors may have an effect on the surface homogeneity of nanofibers. All images were shot at 400x magnification. Since the model of the optical microscope used does not have the function of connecting to the monitor, images were taken by pressing the camera on the microscope's ocular lens. The equipment used for taking pictures is iPhone12.

In the optical microscope images, nanofibers with more surface homogeneity will produce images with a more uniform color. On the contrary, nanofibers with less surface homogeneity will produce images with more significant color variance. Therefore, the variance of color is used to describe the surface homogeneity of nanofiber.



(a) (b) (c)

Figure 4-12 The optical microscope pictures of nanofiber with different homogeneity

The parameter used to describe image color variance is the coefficient of variation (CV). CV is also known as relative standard deviation. The CV is a standardized measure of dispersion of a probability distribution, and it is defined as the ratio of the standard deviation to the mean. *See* Eq. (4-9.

$$C_v = \frac{\sigma}{\mu} \tag{4-9}$$

where C_v is the CV, σ is the standard deviation and μ is the mean value of grayscale. In this thesis project, the CV is calculated using the equation above. The corresponding MATLAB code is attached in Appendix M.

For the three samples in Figure 4-12, their standard deviation of grayscale, average and variation of coefficient of grayscale calculated using MATLAB are listed in Table 4-10. The average grayscale represents the overall color of images and the standard deviation represent the difference in image color. The increasing standard deviation indicates the decreasing surface homogeneity.

In this study, the varication of coefficient of grayscale is used as indicator instead of standard deviation to reduce the influence of image contrast. During the process of taking optical microscope images, the usage of optical microscope and camera can introduce inconsistency in different images' contrast. When optical microscope focusing on sample, the process of aligning the focal length will change the amount of light coming into the lens and result in changes in image brightness and contrast. While taking pictures with the camera, the camera automatically focuses and adjusts the light coming in the lens. The brightness of the camera's focus point will affect the contrast and brightness of the picture.

If the influence of image contrast is not considered, it would be less reasonable to compare the results between different images. Therefore, the variation of coefficient of grayscale is used as the indicator for nanofiber surface homogeneity. Again, the CV of the three samples in Figure 4-12 is shown in the third column of Table 4-10. Among the three images in Figure 4-12, Figure 4-12 (a) is the one with the least homogeneity which has the smallest value of CV, 1.08, and Figure 4-12 (c) is the one with the most homogeneity which has the largest value of CV, 10.09.

Table 4-10 The calculated homogeneity results of different nanofiber sample

Sample #	Standard Deviation of Grayscale	Average Grayscale	Coefficient of variation of Grayscale
a	1462.73	144.96	10.09
b	597.17	173.86	3.43
c	222.04	205.39	1.08

This surface homogeneity determination approach could be very useful when used for parallel comparisons for the same batch of samples or comparison of different positions of the same sample. The difference of homogeneity between different samples may be very slight and not as apparent as the ones in Figure 4-12. Under that scenario, this tool would be very useful for easily distinguishing and comparing samples' homogeneity.

However, I currently cannot give a specific number as a criterion for this indicator, the CV. It is really hard to justify under which CV value the fiber can be considered as homogeneous and above which CV value the fiber can be considered as inhomogeneous. The homogeneity of sample cannot be quantified by a single CV value. Even for the concept of homogeneous itself, it is a relative concept and conclusions need to be drawn in the comparison.

At this stage, this tool can only offer information about nanofiber's surface homogeneity. Future work can be done to combine this tool with other nanofiber images and offer more information. If this tool can be used with a microscope that can accurately control the incoming light; it might be possible to calculate the thickness of nanofiber throughout the whole fiber mat.

CHAPTER 5. CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

The following conclusions can be drawn from this thesis work. They are mentioned at the end of sections in the thesis and compiled here for the ease of reading.

5.1.1 Diameter determination

This thesis presents three approaches to fiber diameter measurement, the MATLAB approach, the UNet approach, and the ResNet approach. The results obtained using the MATLAB approach was verified by manually measurement. In this part, the influence of image magnification was verified and considered during the diameter determination procedure. Two images at different magnification can be processed at the same time and their results are combined as the result, which is deemed more accurate.

For the UNet approach, a machine learning model UNet is used for diameter determination, and the results by the UNet approach was more accurate than those obtained by the MATALB approach. The UNet approach was also implemented using Python, which can be run on a free platform instead of commercial software like MATLAB. This change reduces the of operation.

For the ResNet approach, the feasibility of determining nanofiber diameter using this model was verified. The acceptable result was obtained from synthetic data.

5.1.2 Characterization of pores and surface homogeneity

This thesis presents two tools for the characterization of pores in a nanofiber mat, one for inter-fiber pores and the other for the intra-fiber pores. The sizes of inter-fiber pores are expressed as the equivalent area diameter. The equivalent area diameters largely depended on the binarization method. This approach can be useful for examining the largest pores on a fiber mat.

The intra-fiber pores are characterized by the short sides of oval pores. The result was verified by to manually measurement. The mean diameters obtained from the automated tool were close to the manual measurement diameter, but the standard deviations had large deviations between them.

The tool for porosity measurement was validated using experiments. The measured porosity by image processing depended on the thresholding value. Using average greyscale plus standard deviation as thresholding value resulted in the most accurate porosity result.

Last, this thesis presents a simple method for surface homogeneity determination is present by image processing. This method is deemed simple and quick for comparing different homogeneities of the nanofiber mats.

5.2 Recommendations to Future works

Despite the contributions of this thesis work to the field of nanofiber, limitations exist as presented in the thesis. The following research is recommended to further improve the accuracy and reliability of the tools.

5.2.1 UNet model

As mentioned at the end of Section 3.2.1, I implement the UNet code on the SEM images of nanofibers collected in our research group to segment the nanofibers from the background. I changed the padding method of the original model, but I did not make any major change to the original neural network structure.

In the future, some changes can be made to the original model structure to improve the accuracy of the results. For example, the structure can be changed by adding attention mechanisms such as channel attention(Q. Wang et al., 2019), pixel attention(H. Zhao et al., 2020), convolutional block attention module(Woo et al., 2018), and bottleneck attention module(Park et al., 2018). Using different optimization methods and loss functions such as weighted cross-entropy(Phan et al., 2020), dice loss(Milletari et al., 2016), and focal loss(Lin et al., 2017) may also help improve the results. Furthermore, using different feature extraction methods such as Resnet backbone (He et al., 2015) , VGG16 backbone (Simonyan &

Zisserman, 2014), and DenseNet (Huang et al., 2016) can also bring improvement to the result (Zhang et al., 2021).

5.2.2 ResNet50 model

As mentioned at the end of Chapter 3, this thesis study points out an application of machine learning for nanofiber characterization; its feasibility is verified. In the future, work can be done to further optimize the neural network structure and further increase the accuracy. It would be also useful to reconstruct a dataset using real SEM images and try to achieve a high accuracy using this model on real SEM images.

As explained in Section 3.4.3, the current ResNet50 model has a low accuracy due to insufficient sample size and uneven distribution of data samples in the SEM database. This part of work deserves further investigation. For the problems existing in the real SEM database mentioned in Section 3.4.3.1, I come up with the following three-step solution:

The first step and most important step is to collect more data. Considering the problem of uneven data distribution, extra attention is required to collecting more data that is different from the existing data which means different nanofiber diameters in this case.

The second step is to narrow down the diameter range of the training data. For classification tasks, it is generally accepted that the more categories, the greater the difficulty. Reducing the diameter range can reduce the number of categories that need to be classified and thus reduce the difficulty of classification. In the case of decreasing difficulty, a higher accuracy can be more easily achieved under the same model. When the category number is reduced, the amount of data required is also reduced which also reduced the workload.

The third step is to perform data augmentation. By performing data augmentation, the amount of data can be largely increased. Only when the sample size is large enough that the accuracy of the machine learning can be ensured.

5.2.3 Automated intra-fiber pore size distribution

As indicated at the end of Section 4.2.2, the automatic intra-fiber pore size distribution is different from the full manual measurement, which is the ground truth. Further improvement is needed in this part of work.

REFERENCES

Agarwal, S., Wendorff, J. H., & Greiner, A. (2008). Use of electrospinning technique for biomedical applications. In *Polymer* (Vol. 49, Issue 26, pp. 5603–5621). Elsevier BV. https://doi.org/10.1016/j.polymer.2008.09.014

Agarap, A. F. (2019). Deep Learning using Rectified Linear Units (ReLU). https://arxiv.org/abs/1803.08375

Ahne, J., Li, Q., Croiset, E., & Tan, Z. (2018). Electrospun cellulose acetate nanofibers for airborne nanoparticle filtration. *Textile Research Journal*, 89 (15), 3137-3149. https://doi.org/10.1177/0040517518807440

Amiraliyan, N., Nouri, M., & Kish, M. H. (2009). Effects of some electrospinning parameters on morphology of Natural silk-based nanofibers. *Journal of Applied Polymer Science*, *113*(1), 226–234. https://doi.org/10.1002/app.29808

Arshad, S. N., Naraghi, M., & Chasiotis, I. (2011). Strong carbon nanofibers from electrospun polyacrylonitrile. *Carbon*, *49*(5), 1710–1719. https://doi.org/10.1016/j.carbon.2010.12.056

Aslan, C., Erdem, A., Erdem, E., & Tari, S. (2008). Disconnected skeleton: Shape at its absolute scale. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *30*(12), 2188–2203. https://doi.org/10.1109/TPAMI.2007.70842

Aydilek, A. H., Asce, A. M., Seyfullah., Oguz, H., Edil, T. B., & Asce, M. (2002). *Digital Image Analysis to Determine Pore Opening Size Distribution of Nonwoven Geotextiles*. https://ascelibrary.org/doi/epdf/10.1061/%28ASCE%290887-3801%282002%2916%3A4%28280%29

Barhoum, A., Bechelany, M., Salam, A., & Makhlouf, H. (2019). *Handbook of Nanofibers*.

Bolya, D., Zhou, C., Xiao, F., & Lee, Y. J. (2019). *YOLACT: Real-time Instance Segmentation*. http://arxiv.org/abs/1904.02689

Bourrous, S., Bouilloux, L., Ouf, F. X., Appert-Collin, J. C., Thomas, D., Tampère, L., & Morele, Y. (2014). Measurement of the nanoparticles distribution in flat and pleated filters during clogging. *Aerosol Science and Technology*, *48*(4), 392–400. https://doi.org/10.1080/02786826.2013.878453

Bresee, R., & Daniluk, T. (1997). *Characterizing nonwoven web structure using image analysis techniques*.

Briggs, K. (1991). A precise calculation of the Feigenbaum constants. Mathematics of computation, 57(195), 435-439. https://home.csulb.edu//~scrass/teaching/math456/articles/calcFeigenbaum.pdf

Charvet, A., Pacault, S., Bourrous, S., & Thomas, D. (2018). *Association of fibrous filters for aerosol filtration in predominant Brownian diffusion conditions*. https://doi.org/10.1016/j.seppur.2018.06.045ï

Chen, J., Benesty, J., Huang, Y., & Doclo, S. (2006). New insights into the noise reduction Wiener filter. IEEE Transactions on audio, speech, and language processing, 14(4), 1218-1234.

Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2018). DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *40*(4), 834–848. https://doi.org/10.1109/TPAMI.2017.2699184

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2014). *Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs*. http://arxiv.org/abs/1412.7062

Chen, W., Liu, Y., Ma, Y., Liu, J., & Liu, X. (2014). Improved performance of PVdF-HFP/PI nanofiber mat for lithium ion battery separator prepared by a bicomponent cross-electrospinning method. *Materials Letters*, *133*, 67–70. https://doi.org/10.1016/j.matlet.2014.06.163

Chen, Y., Li, X., Park, K., Song, J., Hong, J., Zhou, L., Mai, Y. W., Huang, H., & Goodenough, J. B. (2013). Hollow carbon-nanotube/carbon-nanofiber hybrid anodes for Li-ion batteries. *Journal of the American Chemical Society*, *135*(44), 16280–16283. https://doi.org/10.1021/ja408421n

Chhabra, R. (2003). *Nonwoven Uniformity — Measurements Using Image Analysis*.

Choi, J. S., Leong, K. W., & Yoo, H. S. (2008). In vivo wound healing of diabetic ulcers using electrospun nanofibers immobilized with human epidermal growth factor (EGF). *Biomaterials*, *29*(5), 587–596. https://doi.org/10.1016/j.biomaterials.2007.10.012

Cui, Z., Zheng, Z., Lin, L., Si, J., Wang, Q., Peng, X., & Chen, W. (2018). Electrospinning and crosslinking of polyvinyl alcohol/chitosan composite nanofiber for transdermal drug delivery. In *Advances in Polymer Technology* (Vol. 37, Issue 6, pp. 1917–1928). John Wiley and Sons Inc. https://doi.org/10.1002/adv.21850

Dehghan, N., Payvandy, P., & Tavanaie, M. A. (2016). *Measuring the Diameter of Nanofibers Extracted from Polyblend Fibers Using FCM Clustering Method*.

Dhanalakshmi, M., Lele, A. K., & Jog, J. P. (2015). Electrospinning of Nylon11: Effect of processing parameters on morphology and microstructure. *Materials Today Communications*, *3*, 141–148. https://doi.org/10.1016/j.mtcomm.2015.01.002

Doshi, J., & Reneker, D. H. (1995). Electrospinning process and applications of electrospun fibers. *Journal of Electrostatics*, *35*(2), 151–160. https://doi.org/https://doi.org/10.1016/0304-3886(95)00041-8

Eda, G., & Shivkumar, S. (2007). Bead-to-fiber transition in electrospun polystyrene. *Journal of Applied Polymer Science*, *106*(1), 475–487. https://doi.org/10.1002/app.25907

Evora, M. C., Klosterman, D., Lafdi, K., Li, L., & Abot, J. L. (2010). Functionalization of carbon nanofibers through electron beam irradiation. *Carbon*, *48*(7), 2037–2046. https://doi.org/10.1016/j.carbon.2010.02.012

Ferreira, T. & Rasband, W.S., 2012. ImageJ User Guide, ImageJ/Fiji1.46. https://imagej.nih.gov/ij/docs/guide/user-guide.pdf (accessed on March 25, 2022)

Field, D.J. (1987). Relations between the statistics of natural images and the response properties of cortical cells. *Journal of the Optical Society of America, A4*(12), 2379–2394. https://doi.org/10.1364/JOSAA.4.002379

Ganjkhanlou, Y., Bayandori Moghaddam, A., Hosseini, S., Nazari, T., Gazmeh, A., & Badraghi, J. (2014). Application of Image Analysis in the Characterization of Electrospun Nanofibers. In *J. Chem. Chem. Eng* (Vol. 33, Issue 2).

Ghasemi-Mobarakeh, L., Semnani, D., & Morshed, M. (2007). A novel method for porosity measurement of various surface layers of nanofibers mat using image analysis for tissue engineering applications. *Journal of Applied Polymer Science*, *106*(4), 2536–2542. https://doi.org/10.1002/app.26949

Givehchi, R. & Tan, Z. (2014). An overview of airborne nanoparticle filtration and thermal rebound theory. *Aerosol and Air Quality Research, 14* (1), 46-63. https://doi.org/10.4209/aaqr.2013.07.0239

Givehchi, R. (2016). *Filtration of NaCl and WOx Nanoparticles using Wire Screens and Nanofibrous Filters*. University of Waterloo PhD Thesis. https://uwspace.uwaterloo.ca/handle/10012/10289

Givehchi, R., Li, Q. & Tan, Z. (2016). Quality factors of PVA nanofibrous filters for airborne particles in the size range of 10–125 nm. *Fuel, 181*, 1273-1280. https://doi.org/10.1016/j.fuel.2015.12.010

Givehchi, R., Li, Q. & Tan, Z. (2018). Filtration of sub-3.3 nm tungsten oxide particles using nanofibrous filters. *Materials, 11*(8), 1277. https://doi.org/10.3390/ma11081277

Givehchi, R., Du, B., Razavi, M., Tan, Z. & Siegel, J. (2021). Performance of nanofibrous media in portable air cleaners. *Aerosol Science and Technology 55*(7), 805-816. https://doi.org/10.1080/02786826.2021.1901846

Gonzalez, R.C. & Woods, R.E. (2002) Digital Image Processing. Prentice-Hall, Inc. Upper Saddle River, New Jersey, USA.

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (n.d.). *Mask R-CNN*. Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2961-2969. https://openaccess.thecvf.com/content_iccv_2017/html/He_Mask_R-CNN_ICCV_2017_paper.html

He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Deep Residual Learning for Image Recognition*. http://arxiv.org/abs/1512.03385

Ho Shin, E., Soo Cho, K., Hwo Seo, M., & Kim, H. (2008). Determination of Electrospun Fiber Diameter Distributions Using Image Analysis Processing. In *Macromolecular Research* (Vol. 16, Issue 4).

Homaeigohar, S., & Elbahri, M. (2014). Nanocomposite electrospun nanofiber mats for environmental remediation. In *Materials* (Vol. 7, Issue 2, pp. 1017–1045). https://doi.org/10.3390/ma7021017

Hotaling, N. A., Bharti, K., Kriel, H., & Simon, C. G. (2015a). DiameterJ: A validated open source nanofiber diameter measurement tool. *Biomaterials*, *61*, 327–338. https://doi.org/10.1016/j.biomaterials.2015.05.015

Hotaling, N. A., Bharti, K., Kriel, H., & Simon, C. G. (2015b). DiameterJ: A validated open source nanofiber diameter measurement tool. *Biomaterials*, *61*, 327–338. https://doi.org/10.1016/j.biomaterials.2015.05.015

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).

Ieracitano, C., Paviglianiti, A., Campolo, M., Hussain, A., Pasero, E., & Morabito, F. C. (2021). A novel automated classification system based on hybrid unsupervised and supervised machine learning for electrospun nanofibers. *IEEE/CAA Journal of Automatica Sinica*, *8*(1), 64–76. https://doi.org/10.1109/JAS.2020.1003387

Iqbal, N., Wang, X., Babar, A. A., Zainab, G., Yu, J., & Ding, B. (2017). Flexible Fe3O4@carbon nanofibers hierarchically assembled with MnO2 particles for high-performance supercapacitor electrodes. *Scientific Reports*, *7*(1). https://doi.org/10.1038/s41598-017-15535-x

Ji, L., & Zhang, X. (2009). Fabrication of porous carbon/Si composite nanofibers as high-capacity battery electrodes. *Electrochemistry Communications*, *11*(6), 1146–1149. https://doi.org/10.1016/j.elecom.2009.03.042

Kabir, M. M., & Demirocak, D. E. (2017). Degradation mechanisms in Li-ion batteries: a state-of-the-art review. In *International Journal of Energy Research* (Vol. 41, Issue 14, pp. 1963–1986). John Wiley and Sons Ltd. https://doi.org/10.1002/er.3762

Kataria, K., Sharma, A., Garg, T., Goyal, A. K., & Rath, G. (2014). *Novel Technology to Improve Drug Loading in Polymeric Nanofibers*.

Khajavi, R., & Abbasipour, M. (2017). Controlling nanofiber morphology by the electrospinning process. In *Electrospun Nanofibers* (pp. 109–123). Elsevier Inc. https://doi.org/10.1016/B978-0-08-100907-9.00005-2

Kim, S. C., Kang, S., Lee, H., Kwak, D. bin, Ou, Q., Pei, C., & Pui, D. Y. H. (2020). Nanofiber filter performance improvement: Nanofiber layer uniformity and branched nanofiber. *Aerosol and Air Quality Research*, *20*(1), 80–88. https://doi.org/10.4209/aaqr.2019.07.0343

Kirillov, A., He, K., Girshick, R., Rother, C., & Dollár, P. (2018). *Panoptic Segmentation*. http://arxiv.org/abs/1801.00868

Koski, A., Yim, K., & Shivkumar, S. (2004). Effect of molecular weight on fibrous PVA produced by electrospinning. *Materials Letters*, *58*(3–4), 493–497. https://doi.org/10.1016/S0167-577X(03)00532-9

Kulpreechanan, N., Bunaprasert, T., Damrongsakkul, S., Kanokpanont, S., & Rangkupan, R. (2013). Effect of polycaprolactone electrospun fiber size on L929 cell behavior. *Advanced Materials Research*, *701*, 420–424. https://doi.org/10.4028/www.scientific.net/AMR.701.420

Leung, W. W. F., Hung, C. H., & Yuen, P. T. (2010). Effect of face velocity, nanofiber packing density and thickness on filtration performance of filters with nanofibers coated on a substrate. *Separation and Purification Technology*, *71*(1), 30–37. https://doi.org/10.1016/j.seppur.2009.10.017

Li, Y., Li, Q. & Tan, Z. (2019). A review of electrospun nanofiber-based separators for rechargeable lithium-ion batteries. *Journal of Power Sources, 443*, 227262. https://doi.org/10.1016/j.jpowsour.2019.227262

Li, Y. & Tan, Z. (2020). Effects of a separator on the electrochemical and thermal performances of Lithium-ion batteries: a numerical study. *Energy & Fuels, 34*(11), 14915-14923. https://doi.org/10.1021/acs.energyfuels.0c02609

Li, Y., Yu, H., Zhang, Y., Zhou, N. & Tan, Z. (2022). Kinetics and characterization of preparing conductive nanofibrous mat by In-situ polymerization of Polypyrrole on electrospun nanofibers. *Chemical Engineering Journal. 433*, 133531. https://doi.org/10.1016/j.cej.2021.133531

Liang, W., Xu, Y., Li, X., Wang, X. X., Zhang, H. di, Yu, M., Ramakrishna, S., & Long, Y. Z. (2019). Transparent Polyurethane Nanofiber Air Filter for High-Efficiency PM2.5 Capture. *Nanoscale Research Letters*, *14*(1). https://doi.org/10.1186/s11671-019-3199-0

Lin, M., Chen, Q., & Yan, S. (2013). *Network In Network*. http://arxiv.org/abs/1312.4400

Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. In Proceedings of the IEEE international conference on computer vision (pp. 2980-2988).

Ioffe, S. & Szegedy, C. (2015). Batch normalization: accelerating deep network training by reducing internal covariate shift, *Proceedings of Machine Learning Research, 37*:448-456, 2015. http://proceedings.mlr.press/v37/ioffe15.html

Long, J., Shelhamer, E., & Darrell, T. (2014). *Fully Convolutional Networks for Semantic Segmentation*. http://arxiv.org/abs/1411.4038

MathWorks, 2022a. Types of Morphological Operations https://www.mathworks.com/help/images/morphological-dilation-and-erosion.html (accessed March 25, 2022.)

MathWorks, 2022b. Distance Transform of a Binary Image, https://www.mathworks.com/help/images/distance-transform-of-a-binary-image.html (accessed March 25, 2022.)

MathWorks, 2022c, Distance transform of binary image, https://www.mathworks.com/help/images/ref/bwdist.html (accessed March 25, 2022.)

Matson, T., Farfel, M., Levin, N., Holm, E., & Wang, C. (2019). Machine Learning and Computer Vision for the Classification of Carbon Nanotube and Nanofiber Structures from Transmission Electron Microscopy Data. *Microscopy and Microanalysis*, *25*(S2), 198–199. https://doi.org/10.1017/s1431927619001727

Mckee, M. G., Layman, J. M., Cashion, M. P., & Long, T. E. (2006). Phospholipid Nonwoven Electrospun Mats. In *Source: Science, New Series* (Vol. 311, Issue 5759).

Miao, Y. E., Zhu, G. N., Hou, H., Xia, Y. Y., & Liu, T. (2013). Electrospun polyimide nanofiber-based nonwoven separators for lithium-ion batteries. *Journal of Power Sources*, *226*, 82–86. https://doi.org/10.1016/j.jpowsour.2012.10.027

Milletari, F., Navab, N., & Ahmadi, S. A. (2016, October). V-net: Fully convolutional neural networks for volumetric medical image segmentation. In 2016 fourth international conference on 3D vision (3DV) (pp. 565-571). IEEE.

Moon, R. J., Martini, A., Nairn, J., Simonsen, J., & Youngblood, J. (2011). Cellulose nanomaterials review: Structure, properties and nanocomposites. *Chemical Society Reviews*, *40*(7), 3941–3994. https://doi.org/10.1039/c0cs00108b

Murphy, R., Turcott, A., Banuelos, L., Dowey, E., Goodwin, B., & Cardinal, K. O. H. (2020). SIMPoly: A MATLAB-Based Image Analysis Tool to Measure Electrospun Polymer Scaffold Fiber Diameter. *Tissue Engineering - Part C: Methods*, *26*(12), 628–636. https://doi.org/10.1089/ten.tec.2020.0304

Naderi, M. (2015). Surface Area: Brunauer-Emmett-Teller (BET). In *Progress in Filtration and Separation* (pp. 585–608). Elsevier Ltd. https://doi.org/10.1016/B978-0-12-384746-1.00014-8

Nagy, Z. K., Balogh, A., Vajna, B., Farkas, A., Patyi, G., Kramarics, Á., & Marosi, G. (2012). Comparison of electrospun and extruded soluplus®-based solid dosage forms of improved dissolution. *Journal of Pharmaceutical Sciences*, *101*(1), 322–332. https://doi.org/10.1002/jps.22731

Napoletano, P., Piccoli, F., & Schettini, R. (2018). Anomaly detection in nanofibrous materials by CNN-based self-similarity. *Sensors (Switzerland)*, *18*(1). https://doi.org/10.3390/s18010209

Nazari, M., Kashanian, S., Moradipour, P., & Maleki, N. (2018). A novel fabrication of sensor using ZnO-Al2O3 ceramic nanofibers to simultaneously detect catechol and hydroquinone.

Journal of Electroanalytical Chemistry, *812*, 122–131. https://doi.org/10.1016/j.jelechem.2018.01.058

Neal, R. A., Mcclugage, S. G., Link, M. C., Sefcik, L. S., Ogle, R. C., & Botchwey, E. A. (2009). *Laminin Nanofiber Meshes That Mimic Morphological Properties and Bioactivity of Basement Mats*. www.liebertpub.com

Oflaz, K., Oflaz, Z., Ozaytekin, I., Dincer, K., & Barstugan, R. (2021). Time and volume-ratio effect on reusable polybenzoxazole nanofiber oil sorption capacity investigated via machine learning. *Journal of Applied Polymer Science*, *138*(30). https://doi.org/10.1002/app.50732

Otsu, N. (1979). A threshold selection method from gray-level histograms, in IEEE Transactions on Systems, Man, and Cybernetics, vol. 9, no. 1, pp. 62-66, Jan. 1979, doi: 10.1109/TSMC.1979.4310076.

Park, J., Woo, S., Lee, J. Y., & Kweon, I. S. (2018). Bam: Bottleneck attention module. arXiv preprint arXiv:1807.06514.

Patel, A. C., Li, S., Wang, C., Zhang, W., & Wei, Y. (2007). Electrospinning of porous silica nanofibers containing silver nanoparticles for catalytic applications. *Chemistry of Materials*, *19*(6), 1231–1238. https://doi.org/10.1021/cm061331z

Pattanayak, S. (2019). Chapter 2. In Intelligent projects using python: 9 real-world AI projects Leveraging Machine Learning and deep learning with TensorFlow and keras. essay, Packt Publishing.

Phan, T. H., & Yamamoto, K. (2020). Resolving class imbalance in object detection with weighted cross entropy losses. arXiv preprint arXiv:2006.01413.

Pierini, F., Lanzi, M., Nakielski, P., Pawlowska, S., Urbanek, O., Zembrzycki, K., & Kowalewski, T. A. (2017). Single-Material Organic Solar Cells Based on Electrospun Fullerene-Grafted Polythiophene Nanofibers. *Macromolecules*, *50*(13), 4972–4981. https://doi.org/10.1021/acs.macromol.7b00857

Rabbani, A., & Salehi, S. (2017). Dynamic modeling of the formation damage and mud cake deposition using filtration theories coupled with SEM image processing. *Journal of Natural Gas Science and Engineering*, *42*, 157–168. https://doi.org/10.1016/j.jngse.2017.02.047

Ribeyre, Q., Charvet, A., Vallières, C., & Thomas, D. (2017). Impact of relative humidity on a nanostructured filter cake – Experimental and modelling approaches. *Chemical Engineering Science*, *161*, 109–116. https://doi.org/10.1016/j.ces.2016.12.013

Rijal, N. P., Adhikari, U., Khanal, S., Pai, D., Sankar, J., & Bhattarai, N. (2018). Magnesium oxide-poly(ε-caprolactone)-chitosan-based composite nanofiber for tissue engineering applications. *Materials Science and Engineering B: Solid-State Materials for Advanced Technology*, *228*, 18–27. https://doi.org/10.1016/j.mseb.2017.11.006

Ronneberger, O., Fischer, P., & Brox, T. (2015). *U-Net: Convolutional Networks for Biomedical Image Segmentation*. http://arxiv.org/abs/1505.04597

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., & Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, *115*(3), 211–252. https://doi.org/10.1007/s11263-015-0816-y

Ryu, H. il, Koo, M. S., Kim, S., Kim, S., Park, Y. A., & Park, S. M. (2020). Uniform-thickness electrospun nanofiber mat production system based on real-time thickness measurement. *Scientific Reports*, *10*(1). https://doi.org/10.1038/s41598-020-77985-0

Sahir, S., Jan 25, 2019, Canny Edge Detection Step by Step in Python — Computer Vision; https://towardsdatascience.com/canny-edge-detection-step-by-step-in-Python-computer-vision-b49c3a2d8123

Sebastián, D., Calderón, J. C., González-Expósito, J. A., Pastor, E., Martínez-Huerta, M. v., Suelves, I., Moliner, R., & Lázaro, M. J. (2010). Influence of carbon nanofiber properties as electrocatalyst support on the electrochemical performance for PEM fuel cells. *International*

Journal of Hydrogen Energy, *35*(18), 9934–9942. https://doi.org/10.1016/j.ijhydene.2009.12.004

Sebe, I., Szabó, P., Kállai-Szabó, B., & Zelkó, R. (2015). Incorporating small molecules or biologics into nanofibers for optimized drug release: A review. In *International Journal of Pharmaceutics* (Vol. 494, Issue 1, pp. 516–530). Elsevier. https://doi.org/10.1016/j.ijpharm.2015.08.054

Shahinfar, S., Meek, P., & Falzon, G. (2020). "How many images do I need?" Understanding how sample size per class affects deep learning model performance metrics for balanced designs in autonomous wildlife monitoring. *Ecological Informatics*, *57*. https://doi.org/10.1016/j.ecoinf.2020.101085

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, *6*(1). https://doi.org/10.1186/s40537-019-0197-0

Širc, J., Hobzová, R., Kostina, N., Munzarová, M., Juklíčková, M., Lhotka, M., Kubinová, Š., Zajícová, A., & Michálek, J. (2012). Morphological characterization of nanofibers: Methods and application in practice. *Journal of Nanomaterials*, *2012*. https://doi.org/10.1155/2012/327369

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Skupov, K. M., Ponomarev, I. I., Razorenov, D. Y., Zhigalina, V. G., Zhigalina, O. M., Ponomarev, I. I., Volkova, Y. A., Volfkovich, Y. M., & Sosenkin, V. E. (2017). Carbon Nanofiber Paper Electrodes Based on Heterocyclic Polymers for High Temperature Polymer Electrolyte Mat Fuel Cell. *Macromolecular Symposia*, *375*(1). https://doi.org/10.1002/masy.201600188

Stojanovska, E., Canbay, E., Pampal, E. S., Calisir, M. D., Agma, O., Polat, Y., Simsek, R., Gundogdu, N. A. S., Akgul, Y., & Kilic, A. (2016). A review on non-electro nanofibre spinning techniques. In *RSC Advances* (Vol. 6, Issue 87, pp. 83783–83801). Royal Society of Chemistry. https://doi.org/10.1039/c6ra16986d

Sun, W., Chen, T., Chen, C., & Li, J. (2007). A study on mat morphology by digital image processing. *Journal of Mat Science*, *305*(1–2), 93–102. https://doi.org/10.1016/j.memsci.2007.07.040

Takikawa, T., Acuna, D., Jampani, V., & Fidler, S. (2019). *Gated-SCNN: Gated Shape CNNs for Semantic Segmentation*. http://arxiv.org/abs/1907.05740

Tan, Z. (2014). Post-combustion Air Emission Control, in Air Pollution and Greenhouse Gases: from Basic Concepts to Engineering Applications for Air Emission Control, Springer, Singapore, https://doi.org/10.1007/978-981-287-212-8_10

Taylor, G. I. S. (1969). Electrically driven jets. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, *313*, 453–475.

Torrey, L., & Shavlik, J. (2010). Transfer learning. In Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques (pp. 242-264). IGI Publishing, Hershey, PA, USA. ISBN:978-1-60566-766-9

Tomba, E., Facco, P., Roso, M., Modesti, M., Bezzo, F., & Barolo, M. (2010). Artificial vision system for the automated measurement of interfiber pore characteristics and fiber diameter distribution in nanofiber assemblies. *Industrial and Engineering Chemistry Research*, *49*(6), 2957–2968. https://doi.org/10.1021/ie901179m

Tsang, S.-H. (2018). Review: U-Net (Biomedical Image Segmentation). Towards Data Science. URL: https://towardsdatascience.com/review-u-net-biomedical-image-segmentation-d02bf06ca760 (accessed on March 25, 2022).

Veleirinho, B., Rei, M. F., & Lopes-Da-Silva, J. A. (2008). Solvent and concentration effects on the properties of electrospun polyethylene terephthalate nanofiber mats. *Journal of Polymer Science, Part B: Polymer Physics*, *46*(5), 460–471. https://doi.org/10.1002/polb.21380

Wang, H., Zhang, Y., Gao, H., Jin, X., & Xie, X. (2016). Composite melt-blown nonwoven fabrics with large pore size as Li-ion battery separator. International Journal of Hydrogen Energy, 41(1), 324-330.

Wang, J., Chen, Y., Chakraborty, R., & Yu, S. X. (2020). Orthogonal convolutional neural networks. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 11505-11515).

Wang, T., Chen, Y., Dong, W., Liu, Y., Shi, L., Chen, R., & Pan, T. (2020). Fractal Characteristics of Porosity of Electrospun Nanofiber Mats. *Mathematical Problems in Engineering*, *2020*. https://doi.org/10.1155/2020/2503154

Wang, T., Dong, W., Chen, Y., Pan, T., & Chen, R. (2018). Study on porosity of electrospun nanofiber mat by neural network. *Applied Mathematical Sciences*, *12*(22), 1059–1074. https://doi.org/10.12988/ams.2018.8582

Wang, W., Chapter 31 - Modeling and Processing with Quadric Surfaces, Editor(s): Gerald Farin, Josef Hoschek, Myung-Soo Kim, Handbook of Computer Aided Geometric Design, North-Holland, 2002, Pages 777-795, https://doi.org/10.1016/B978-044451104-1/50032-0.Wightman, R., Touvron, H., & Jégou, H. (2021). *ResNet strikes back: An improved training procedure in timm*. http://arxiv.org/abs/2110.00476

Woo, S., Park, J., Lee, J. Y., & Kweon, I. S. (2018). Cbam: Convolutional block attention module. In Proceedings of the European conference on computer vision (ECCV) (pp. 3-19).

Xu, W., Yang, W., & Yang, Y. (2009). Electrospun starch acetate nanofibers: Development, properties, and potential application in drug delivery. *Biotechnology Progress*, *25*(6), 1788–1795. https://doi.org/10.1002/btpr.242

Yoon, K., Hsiao, B. S., & Chu, B. (2008). Functional nanofibers for environmental applications. *Journal of Materials Chemistry*, *18*(44), 5326–5334. https://doi.org/10.1039/b804128h

Yu, T., Lin, B., Li, Q., Wang, X., Qu, W., Zhang, S., & Deng, C. (2016). First exploration of freestanding and flexible Na2+2: XFe2- x(SO4)3@porous carbon nanofiber hybrid films with superior sodium intercalation for sodium ion batteries. *Physical Chemistry Chemical Physics*, *18*(38), 26933–26941. https://doi.org/10.1039/c6cp04958c

Yuan, X. Y., Zhang, Y. Y., Dong, C., & Sheng, J. (2004). Morphology of ultrafine polysulfone fibers prepared by electrospinning. *Polymer International*, *53*(11), 1704–1710. https://doi.org/10.1002/pi.1538

Zhang, C., Benz, P., Argaw, D.M., Lee, S., Kim, J., Rameau, F., Bazin, J.-C. & Kweon, I.S. (2021). ResNet or DenseNet? Introducing Dense Shortcuts to ResNet. Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2021, pp. 3550-3559. https://openaccess.thecvf.com/content/WACV2021/html/Zhang_ResNet_or_DenseNet_Introducing_Dense_Shortcuts_to_ResNet_WACV_2021_paper.html

Zhang, F., Ma, X., Cao, C., Li, J., & Zhu, Y. (2014). Poly(vinylidene fluoride)/SiO2 composite mats prepared by electrospinning and their excellent properties for nonwoven separators for lithium-ion batteries. *Journal of Power Sources*, *251*, 423–431. https://doi.org/10.1016/j.jpowsour.2013.11.079

Zhang, Q., Welch, J., Park, H., Wu, C. Y., Sigmund, W., & Marijnissen, J. C. M. (2010). Improvement in nanofiber filtration by multiple thin layers of nanofiber mats. *Journal of Aerosol Science*, *41*(2), 230–236. https://doi.org/10.1016/j.jaerosci.2009.10.001

Zhang, Y., Li, Y. & Tan, Z. (2021). A review of enrichment methods for circulating tumor cells: from single modality to hybrid modality, *Analyst, 146* (23)*,* 7048-7069. https://doi.org/10.1039/D1AN01422F

Zhang, Y., Li, J., An, G., & He, X. (2010). Highly porous SnO2 fibers by electrospinning and oxygen plasma etching and its ethanol-sensing properties. *Sensors and Actuators, B: Chemical*, *144*(1), 43–48. https://doi.org/10.1016/j.snb.2009.10.012

Zhang, Z., & Sabuncu, M. (2018). Generalized cross entropy loss for training deep neural networks with noisy labels. Advances in neural information processing systems, 31.

Zhao, H., Kong, X., He, J., Qiao, Y., & Dong, C. (2020, August). Efficient image super-resolution using pixel attention. In European Conference on Computer Vision (pp. 56-72). Springer, Cham.

Zhao, X., Wang, S., Yin, X., Yu, J., & Ding, B. (2016a). Slip-Effect Functional Air Filter for Efficient Purification of PM 2.5. *Scientific Reports*, *6*. https://doi.org/10.1038/srep35472

Zhao, X., Wang, S., Yin, X., Yu, J., & Ding, B. (2016b). Slip-Effect Functional Air Filter for Efficient Purification of PM 2.5. *Scientific Reports*, *6*. https://doi.org/10.1038/srep35472

Zhao, Y. Y., Yang, Q. B., Lu, X. F., Wang, C., & Wei, Y. (2005). Study on correlation of morphology of electrospun products of polyacrylamide with ultrahigh molecular weight. *Journal of Polymer Science, Part B: Polymer Physics*, *43*(16), 2190–2195. https://doi.org/10.1002/polb.20506

Zheng, G., Jiang, J., Wang, X., Li, W., Liu, J., Fu, G., & Lin, L. (2020). Nanofiber mats by multi-jet electrospinning arranged as arc-array with sheath gas for electrodialysis applications. *Materials and Design*, *189*. https://doi.org/10.1016/j.matdes.2020.108504

Ziabari, M., Mottaghitalab, V., & Khodaparast Haghi, A. (2008). Evaluation of electrospun nanofiber pore structure parameters. *Korean Journal of Chemical Engineering,* 25 (4), 923-932. https://doi.org/10.1007/s11814-008-0151-x

APPENDICES

Appendix A MATLAB code for fiber diameter determination

```matlab

close all
clear all

file1 = '1-04.tif';
file2 = '1-05.tif';
I1 = imread(file1);
I2 = imread(file2);

original1 = I1;
original2 = I2;

% Analyse the part of the SEM image with the scale to obtain the
    pixel length of the ruler automatically
scale1 = imcrop(I1, [305 722 90 30]);
[scale1, noOfRegions1] = bwlabel(scale1, 4);
scale2 = imcrop(I2, [305 722 90 30]);
[scale2, noOfRegions2] = bwlabel(scale2, 4);

% We need to find the ruler; it should be the object with the
    largest area
% Get the information about the image
measurements1 = regionprops(scale1, 'Area', 'PixelList');
area1 = cat(1, measurements1.Area);
[m1, index1] = max(area1);
measurements2 = regionprops(scale2, 'Area', 'PixelList');
area2 = cat(1, measurements2.Area);
[m2, index2] = max(area2);

% process file1 first
ruler1 = ismember(scale1, index1);
% Find the difference between the min and max X value to get the
    length
coordinates1 = measurements1.PixelList;
x1 = coordinates1(:,1);

% The measurement should actually start and end in the same
    position on the two opposite vertical lines (ie if start from
    center of left bar, must end at centre of right bar). The
    correction is a measurement of the thickness of the bar and
    subtracting it so that the pixel length start from the left bar
    's first pixel, to the right bar's first pixel.
pixelDistance1 = range(x1) - 2;
fprintf('The pixel length of the scale ruler is: %.2f pixels for
    the first file.\n', pixelDistance1)

scalevalue1 = input('Enter the scale (nm) for the first file: ');
cannySensitivity1 = input(['\nEnter the Canny Border Detection
    sensitivity (0-0.999). \nThe lower the number, the easier
    matlab detects a region as an edge\n'...
    +'-If set less than 0.2 then a blurring filter will be done on
    the image (for high res)\n'...
    +'-If set above, no blurring is required (for low res)\nChoose
    0.2 for high res images, and 0.21-0.3 for low res images: ']);

isBlur1 = false;
if cannySensitivity1 <= 0.2
```

```
    isBlur1 = true;
end

% distance = input('Enter the pixel length of the scale: ');
%close(scaleinput)

scale1 = pixelDistance1/scalevalue1;

% processing file 2
ruler2 = ismember(scale2, index2);
% Find the difference between the min and max X value to get the
    length
coordinates2 = measurements2.PixelList;
x2 = coordinates2(:,1);

% The measurement should actually start and end in the same
    position on the
% two opposite vertical lines (ie if start from center of left bar,
     must
% end at centre of right bar). The correction is a measurement of
    the
% thickness of the bar and subtracting it so that the pixel length
    start
% fro mthe left bar's first pixel, to the right bar's first pixel.
pixelDistance2 = range(x2) - 2;
fprintf('The pixel length of the scale ruler is: %.2f pixels for
    the second file.\n', pixelDistance2)

% scaleinput = imtool(file);
% disp('*Use the ''Measure Distance'' tool (ruler icon) to retrieve
     scale*')
scalevalue2 = input('Enter the scale (nm) for the second file: ');
cannySensitivity2 = input(['\nEnter the Canny Border Detection
    sensitivity (0-0.999). \nThe lower the number, the easier
    matlab detects a region as an edge\n'...
    +'-If set less than 0.2 then a blurring filter will be done on
    the image (for high res)\n'...
    +'-If set above, no blurring is required (for low res)\nChoose
    0.2 for high res images, and 0.21-0.3 for low res images: ']);

isBlur2 = false;
if cannySensitivity2 <= 0.2
    isBlur2 = true;
end

% distance = input('Enter the pixel length of the scale: ');
%close(scaleinput)

scale2 = pixelDistance2/scalevalue2;

%% Start the preprocessing image manipulation
% processing first file
I = I1

I = imcrop(I, [0 0 1024 680]);
ASLI=I;
% 2. Initial noise reduction for the grayscale image
```

```matlab
I = medfilt2(I);
I = wiener2(I,[5 5]);

% 3. Even out the contrast profile
I = imadjust(I);
I = histeq(I);

% 4. Noise Reduction
SE = strel('square', 2);
I = imclose(I, SE);
I = imopen(I, SE);

% We need this to use edge detection later (edge detection must be
    done on
% the original image)
i = I;

% 5. Local thresholding method
% This method goes to each pixel and processes the best theshold
    value for
% each pixel.

preprocessed = I;

BW = localthresh(I);
%BW = im2bw(I, 0.16);
I = BW;

% 6. MAIN Noise Reduction
SE = strel('disk', 2);
I = imclose(I, SE);
I = imopen(I, SE);
I = bwmorph(I, 'majority');
I = bwmorph(I, 'clean');

BWreduced = I;

% 7. Reverse colors of the binary image
I = imcomplement(I);

% 8. Store the edge profile of the image and combine with the
    binary image into N
if isBlur1

    i = gaussianFilter(i, 8);

end

i = edge(i, 'canny', cannySensitivity1);

% 9.Perfect the boundaries by first dilating (to seal openings) and
     thinning
% to reduce the edges to their more accurate thickness
SE = strel('disk', 1);
i = imdilate(i, SE);
i = bwmorph(i, 'thin', Inf);

```

```
N = I | i;

% 10. Noise reduction. 'clean' gets rid of single pixel outliers
SE = strel('disk', 2);
N = imclose(N, SE);
N = bwmorph(N, 'clean');
N = imcomplement(N);
N = bwmorph(N, 'clean');
N = imcomplement(N);

withEdge = N;

% 11. Diameter Measurement
dist = bwdist(N, 'euclidean');
dist = dist*2/scale1;
% 7. Create a skeleton image and use pruning to delete sporadic
    branches
skeleton = imregionalmax(dist, 4);
D1 = dist(find( skeleton == 1 ));

% Display image with blue skeleton
% convert to rgb first
rgb = repmat(N, [1 1 3]);
rgb = cat(3, N, N, N);
rgb = repmat(double(N)./255,[1 1 3]);

finalImage = N | skeleton;

m1 = max(D1);
% 12. Create the histogram from the measured distance data (radius
    x 2).
% Find the ranges necessary for plotting
range = 0:20:m1-mod(m1, 20)+20;
noOfRanges1 = length(range) -1;
frequency1 = zeros(1,noOfRanges1);

% Store the diameters in the ranges
for i = 1:noOfRanges1

    frequency1(i)=length(find( D1 > range(i) & D1 < range(i+1) ));

end

[magnitude, direction] = imgradient(dist);
angle = skeleton .* direction;
negIndices = find(angle < 0);
if ~isempty(negIndices)
    angle(negIndices) = angle(negIndices) + 180;
end

angle = angle + 90;
fixIndices = find(angle > 180);
if ~isempty(fixIndices)
    angle(fixIndices) = angle(fixIndices) + 180;
end

fprintf('\n\nThe average fiber orientation is %.2f degrees for the
    first file.\n\n', mean(mean(angle)))

nelements1 = sum(frequency1);

% processing second file
I = I2
I = imcrop(I, [0 0 1024 680]);
ASLI=I;

% 2. Initial noise reduction for the grayscale image
I = medfilt2(I);
I = wiener2(I,[5 5]);

% 3. Even out the contrast profile
I = imadjust(I);
I = histeq(I);

% 4. Noise Reduction
SE = strel('square', 2);
I = imclose(I, SE);
I = imopen(I, SE);

% We need this to use edge detection later (edge detection must be
    done on
% the original image)
i = I;

% SE = strel('disk', 50);
% IO = imopen(I, SE);
% I = I - IO;

% 5. Local thresholding method
% This method goes to each pixel and processes the best theshold
    value for
% each pixel.

preprocessed = I;

BW = localthresh(I);
%BW = im2bw(I, 0.16);
I = BW;

% 6. MAIN Noise Reduction
SE = strel('disk', 2);
I = imclose(I, SE);
I = imopen(I, SE);
I = bwmorph(I, 'majority');
I = bwmorph(I, 'clean');

BWreduced = I;

%BW2 = bwmorph(I, 'skel', Inf);
%skeleton = bwmorph(BW2, 'spur');

% Reverse colors of the binary image
I = imcomplement(I);

```

```
% 8a. Store the edge profile of the image and combine with the
    binary image into N
if isBlur2

    i = gaussianFilter(i, 8);

end

i = edge(i, 'canny', cannySensitivity2);

% 8b.Perfect the boundaries by first dilating (to seal openings)
    and thinning
% to reduce the edges to their more accurate thickness
SE = strel('disk', 1);
i = imdilate(i, SE);
i = bwmorph(i, 'thin', Inf);

N = I | i;

% 9. Noise reduction. 'clean' gets rid of single pixel outliers
SE = strel('disk', 2);
N = imclose(N, SE);
N = bwmorph(N, 'clean');
N = imcomplement(N);
N = bwmorph(N, 'clean');
N = imcomplement(N);

withEdge = N;

% 10. Diameter Measurement
dist = bwdist(N, 'euclidean');
dist = dist*2/scale2;
% 7. Create a skeleton image and use pruning to delete sporadic
    branches
skeleton = imregionalmax(dist, 4);
D2 = dist(find( skeleton == 1 ));

% Display image with blue skeleton
% convert to rgb first
rgb = repmat(N, [1 1 3]);
rgb = cat(3, N, N, N);
rgb = repmat(double(N)./255,[1 1 3]);

finalImage = N | skeleton;

m2 = max(D2);
% 11a. Create the histogram from the measured distance data (radius
     x 2).
% Find the ranges necessary for plotting
range = 0:20:m2-mod(m2, 20)+20;
noOfRanges2 = length(range) -1;
frequency2 = zeros(1,noOfRanges2);

% Store the diameters in the ranges
for i = 1:noOfRanges2

    frequency2(i)=length(find( D2 > range(i) & D2 < range(i+1) ));
```

```

end

[magnitude, direction] = imgradient(dist);
angle = skeleton .* direction;
negIndices = find(angle < 0);
if ~isempty(negIndices)
    angle(negIndices) = angle(negIndices) + 180;
end
angle = angle + 90;
fixIndices = find(angle > 180);
if ~isempty(fixIndices)
    angle(fixIndices) = angle(fixIndices) + 180;
end

fprintf('\n\nThe average fiber orientation is %.2f degrees for the
    second file.\n\n', mean(mean(angle)))

nelements2 = sum(frequency2);

% 11a. Create the histogram for the combined average

DCombined = [D1;D2];
% Find the ranges necessary for plotting
mCombined = max([m1 m2]);
noOfRangesCombined = max([noOfRanges1 noOfRanges2]);
range = 0:20:mCombined-mod(mCombined, 20)+20;
noOfRangesCombined = length(range) -1;
frequencyCombined = zeros(1,noOfRangesCombined);

% Store the diameters in the ranges
for i = 1:noOfRangesCombined

    frequencyCombined(i)=length(find( DCombined > range(i) &
    DCombined < range(i+1) ));

end

negIndices = find(angle < 0);
if ~isempty(negIndices)
    angle(negIndices) = angle(negIndices) + 180;
end
angle = angle + 90;
fixIndices = find(angle > 180);
if ~isempty(fixIndices)
    angle(fixIndices) = angle(fixIndices) + 180;
end

fprintf('\n\nThe Total combining average fiber orientation is %.2f
    degrees for the second file.\n\n', mean(mean(angle)))

nelementsCombined = sum(frequencyCombined);
% Convert the frequency into a percentage
frequencyCombined = frequencyCombined ./ nelementsCombined * 100;
%frequency range

if noOfRanges1 < noOfRanges2
```

```
    frequency1 = cat(2,frequency1,zeros(1,noOfRanges2 - noOfRanges1));
end
if noOfRanges2 < noOfRanges1
    frequency2 = cat(2,frequency2,zeros(1,noOfRanges1 - noOfRanges2));
end

% 12. Plot the data
figure(1);
hold on;
combinedGraph = [frequency1(:), frequency2(:), frequencyCombined(:)];
hb = bar(range(2:end)-10, combinedGraph, 'grouped')
endlimit = floor(range(end)/20)*20;
if mod(range(end), 20) > 0
    endlimit = endlimit + 20;
end

set(gca, 'XTick', [0:60:endlimit])

xlabel('Diameter [nm]')
ylabel('Frequency [%]')
```

Appendix B Python Code for image cropping

```python
import glob
import cv2
import os
from PIL import Image

save_file_to = "/home/e4gao/image_crop/input/croppedSquare/"
file_count = 0

for file in glob.glob("/home/e4gao/image_crop/input/*.tif"):
    image = Image.open(file)

    # top left 1
    cropped_image1 = image.crop((0, 0, 256, 256))
    cropped_image1.save(save_file_to + file[29:-4] + str(file_count
    ) + '.tif')
    file_count = file_count + 1

    # top left 2
    cropped_image1 = image.crop((255, 0, 511, 256))
    cropped_image1.save(save_file_to + file[29:-4] + str(file_count
    ) + '.tif')
    file_count = file_count + 1

    # top right 1
    cropped_image1 = image.crop((510, 0, 766, 256))
    cropped_image1.save(save_file_to + file[29:-4] + str(file_count
    ) + '.tif')
    file_count = file_count + 1

    # top right 2
    cropped_image1 = image.crop((764, 0, 1020, 256))
    cropped_image1.save(save_file_to + file[29:-4] + str(file_count
    ) + '.tif')
    file_count = file_count + 1

    # bottom left 1
    cropped_image1 = image.crop((0, 256, 256, 512))
    cropped_image1.save(save_file_to + file[29:-4] + str(file_count
    ) + '.tif')
    file_count = file_count + 1

    # bottom left 2
    cropped_image1 = image.crop((255, 256, 511, 512))
    cropped_image1.save(save_file_to + file[29:-4] + str(file_count
    ) + '.tif')
    file_count = file_count + 1

    # bottom right 1
    cropped_image1 = image.crop((510, 256, 766, 512))
    cropped_image1.save(save_file_to + file[29:-4] + str(file_count
    ) + '.tif')
    file_count = file_count + 1

    # bottom right 2
    cropped_image1 = image.crop((764, 256, 1020, 512))
    cropped_image1.save(save_file_to + file[29:-4] + str(file_count
    ) + '.tif')
```

```
file_count = 0
```

Appendix C Python Code for correct the displacement of segmented image

```python
import glob
import cv2
import os
from PIL import Image

save_file_to = "/home/e4gao/input/"
file_count = 0
for file in glob.glob("/home/e4gao/input/*.tif"):
    image = Image.open(file)
    cropped_image1 = image.crop((0, 0, 512, 512))
    cropped_image1.save(save_file_to + file[29:-4] + str(file_count
    ) + '.tif')
file_count = file_count + 1
```

Appendix D UNet Code

The code of UNet model and code for UNet training is from https://github.com/zhixuhao/unet.

D1. Code of UNet model

```
import numpy as np
import os
import skimage.io as io
import skimage.transform as trans
import numpy as np
from keras.models import *
from keras.layers import *
from keras.optimizers import *
from keras.callbacks import ModelCheckpoint, LearningRateScheduler
from keras import backend as keras
from tensorflow.keras.optimizers import Adam

def unet(pretrained_weights = None,input_size = (256,256,1)):
    inputs = Input(input_size)
    conv1 = Conv2D(64, 3, activation = 'relu', padding = 'same',
    kernel_initializer = 'he_normal')(inputs)
    conv1 = Conv2D(64, 3, activation = 'relu', padding = 'same',
    kernel_initializer = 'he_normal')(conv1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)
    conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same',
    kernel_initializer = 'he_normal')(pool1)
    conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same',
    kernel_initializer = 'he_normal')(conv2)
    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
    conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same',
    kernel_initializer = 'he_normal')(pool2)
    conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same',
    kernel_initializer = 'he_normal')(conv3)
    pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)
    conv4 = Conv2D(512, 3, activation = 'relu', padding = 'same',
    kernel_initializer = 'he_normal')(pool3)
    conv4 = Conv2D(512, 3, activation = 'relu', padding = 'same',
    kernel_initializer = 'he_normal')(conv4)
    drop4 = Dropout(0.5)(conv4)
    pool4 = MaxPooling2D(pool_size=(2, 2))(drop4)

    conv5 = Conv2D(1024, 3, activation = 'relu', padding = 'same',
    kernel_initializer = 'he_normal')(pool4)
    conv5 = Conv2D(1024, 3, activation = 'relu', padding = 'same',
    kernel_initializer = 'he_normal')(conv5)
    drop5 = Dropout(0.5)(conv5)

    up6 = Conv2D(512, 2, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(drop5))
    merge6= concatenate([drop4,up6],axis=3)
    conv6 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(merge6)
    conv6 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv6)

    up7 = Conv2D(256, 2, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(conv6))
    merge7= concatenate([conv3,up7],axis=3)
    conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(merge7)
```

```
    conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv7)

    up8 = Conv2D(128, 2, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(conv7))
    merge8= concatenate([conv2,up8],axis=3)
    conv8 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(merge8)
    conv8 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv8)

    up9 = Conv2D(64, 2, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(conv8))
    merge9= concatenate([conv1,up9],axis=3)
    conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(merge9)
    conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv9)
    conv9 = Conv2D(2, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv9)
    conv10 = Conv2D(1, 1, activation = 'sigmoid')(conv9)

    model = Model(inputs, conv10)

    model.compile(optimizer = Adam(lr = 1e-5), loss = 'binary_crossentropy', metrics = ['accuracy'])

    #model.summary()

    if(pretrained_weights):
      model.load_weights(pretrained_weights)

    return model
```

D2. Code for UNet training

```python
from model import *
from data import *

#os.environ["CUDA_VISIBLE_DEVICES"] = "0"

data_gen_args = dict(rotation_range=0.1,
                    width_shift_range=0.1,
                    height_shift_range=0.1,
                    shear_range=0.1,
                    zoom_range=0.1,
                    horizontal_flip=True,
                    fill_mode='nearest')
myGene = trainGenerator(2,'data/membrane/train','image','label',
    data_gen_args,save_to_dir = None)

model = unet()
model_checkpoint = ModelCheckpoint('unet_membrane.hdf5', monitor='
    loss',verbose=1, save_best_only=True)
model.fit_generator(myGene,steps_per_epoch=300,epochs=1,callbacks=[
    model_checkpoint])

testGene = testGenerator("data/membrane/test")
results = model.predict_generator(testGene,30,verbose=1)
saveResult("data/membrane/test",results)
```

Appendix E The Python code for SEM image dataset build up

This MATLAB code will automatically run the diameter determination code on all images in the folder it is located.

```
    myFolder = 'C:\Users\e4gao\OneDrive - University of Waterloo\
    desktop\img_resize';
    d_all = [];
    if ~isdir(myFolder)
      errorMessage = sprintf('Error: The following folder does not
    exist:\n%s', myFolder);
      uiwait(warndlg(errorMessage));
      return;
    end
    filePattern = fullfile(myFolder, '*.tif');
    dataFiles = dir(filePattern);
    for k = 1:length(dataFiles)
      baseFileName = dataFiles(k).name;
      fullFileName = fullfile(myFolder, baseFileName);
      fprintf(1, 'Now reading %s\n', fullFileName);
      I = imread(fullFileName);

% we are not using the scale this time
% the Canny Sensitivity
cannySensitivity = 0.25;

isBlur = false;
if cannySensitivity <= 0.2
    isBlur = true;
end

%% image preprocessing

% Initial noise reduction for the grayscale image
I = medfilt2(I);

% Even out the contrast profile
I = imadjust(I);
I = histeq(I);

% Noise Reduction
SE = strel('square', 2);
I = imclose(I, SE);
I = imopen(I, SE);

% We need this to use edge detection later (edge detection must be
    done on
% the original image)
i = I;
```

```
% MAIN Noise Reduction
SE = strel('disk', 2);
I = imclose(I, SE);
I = imopen(I, SE);
I = bwmorph(I, 'majority');
I = bwmorph(I, 'clean');

BWreduced = I;

% Reverse colors of the binary image
I = imcomplement(I);

% Store the edge profile of the image and combine with the binary
    image into N
if isBlur

    i = gaussianFilter(i, 8);

end

i = edge(i, 'canny', cannySensitivity);

% Perfect the boundaries by first dilating (to seal openings) and
    thinning
% to reduce the edges to their more accurate thickness
SE = strel('disk', 1);
i = imdilate(i, SE);
i = bwmorph(i, 'thin', Inf);

N = I | i;

% Noise reduction. 'clean' gets rid of single pixel outliers
SE = strel('disk', 2);
N = imclose(N, SE);
N = bwmorph(N, 'clean');
N = imcomplement(N);
N = bwmorph(N, 'clean');
N = imcomplement(N);

withEdge = N;

%Diameter Measurement
dist = bwdist(N, 'euclidean');
dist = dist*2;

% Create a skeleton image and use pruning to delete sporadic
    branches
skeleton = imregionalmax(dist, 4);
D = dist(find( skeleton == 1 ));
```

```matlab

% Display image with blue skeleton
% convert to rgb first
rgb = repmat(N, [1 1 3]);
rgb = cat(3, N, N, N);
rgb = repmat(double(N)./255,[1 1 3]);

finalImage = N | skeleton;

m = max(D);

% Create the histogram from the measured distance data (radius x 2)
    .
% Find the ranges necessary for plotting
range = 0:1:m-mod(m, 20)+20;
noOfRanges = length(range) -1;
frequency = zeros(1,noOfRanges);

% Store the diameters in the ranges
for i = 1:noOfRanges
```

```matlab

    frequency(i)=length(find( D > range(i) & D < range(i+1) ));

end

[magnitude, direction] = imgradient(dist);
angle = skeleton .* direction;
negIndices = find(angle < 0);
if ~isempty(negIndices)
    angle(negIndices) = angle(negIndices) + 180;
end
angle = angle + 90;
fixIndices = find(angle > 180);
if ~isempty(fixIndices)
    angle(fixIndices) = angle(fixIndices) + 180;
end

nelements = sum(frequency);

D_rounded = round(D);
D_out = mode(D_rounded);
d_all = [d_all; D_out];
    end
```

Appendix F ResNet Code

F1. Synthetic image generation

```python
import os
import time
import cv2 as cv
import numpy as np

IMAGE_HEIGHT = 341
IMAGE_WIDTH = 341
LARGE_IMAGE_HEIGHT = 380
LARGE_IMAGE_WEIGHT = 380
MIN_PIXEL = 1
MAX_PIXEL = 20
PIXEL_IMAGE_NUMBER = 500
LINE_PATH = './line'
RECTANGLE_PATH = './rectangle'
COMMON_LINE = 11
RANDOM_LINE = 16
COLOR_PIXEL_START = 150
COLOR_PIXEL_END = 255

def get_position(min_number=0, max_number=255):
    return np.random.randint(min_number, max_number + 1)

def top_to_bottom(image, pixes, index):
    color = get_position(COLOR_PIXEL_START, COLOR_PIXEL_END)
    if index >= COMMON_LINE:
        random_pixel = get_position(MIN_PIXEL, MAX_PIXEL)
        position = get_position(0, LARGE_IMAGE_WEIGHT -
    random_pixel)
    else:
        position = get_position(0, LARGE_IMAGE_WEIGHT - pixes)
    cv.rectangle(image, (position, 0), (position + pixes,
    LARGE_IMAGE_HEIGHT), (color, color), thickness=-1)

def left_to_right(image, pixels, index):
    color = get_position(COLOR_PIXEL_START, COLOR_PIXEL_END)
    if index >= COMMON_LINE:
        random_pixel = get_position(MIN_PIXEL, MAX_PIXEL)
        position = get_position(0, LARGE_IMAGE_HEIGHT -
    random_pixel)
    else:
        position = get_position(0, LARGE_IMAGE_HEIGHT - pixels)
    cv.rectangle(image, (0, position), (LARGE_IMAGE_WEIGHT,
    position + pixels), (color, color), thickness=-1)

def generate_rectangle_image(pixes):
    for pixel_number in range(PIXEL_IMAGE_NUMBER):
        image = np.zeros((LARGE_IMAGE_WEIGHT, LARGE_IMAGE_HEIGHT),
    'uint8')
        index = np.arange(1, 31)
        np.random.shuffle(index)
        odd_index = 0
        even_index = 0
        for i in index:
            if i % 2 == 0:
                even_index += 1
                top_to_bottom(image, pixes, even_index)
            else:
                odd_index += 1
                left_to_right(image, pixes, odd_index)
        angle = get_position(-5, 5)
        M = cv.getRotationMatrix2D((LARGE_IMAGE_WEIGHT // 2, LARGE_IMAGE_WEIGHT // 2), angle, 1)
        image = cv.warpAffine(image, M, (LARGE_IMAGE_WEIGHT, LARGE_IMAGE_HEIGHT))
        image = image[20:20 + IMAGE_WIDTH, 20:20 + IMAGE_HEIGHT]
        cv.imwrite(os.path.join(RECTANGLE_PATH, '%d_%d.jpg' % (pixes, pixel_number)), image)

def generate_line_image(pixels):
    for pixel_number in range(PIXEL_IMAGE_NUMBER):
        image = np.zeros((IMAGE_HEIGHT, IMAGE_WIDTH), 'uint8')
        for i in range(COMMON_LINE):
            x_start = get_position(pixels, IMAGE_WIDTH - pixels)
            x_end = get_position(pixels, IMAGE_WIDTH - pixels)
            y_start = get_position(pixels, IMAGE_WIDTH - pixels)
            y_end = get_position(pixels, IMAGE_WIDTH - pixels)
            color = get_position(COLOR_PIXEL_START, COLOR_PIXEL_END)
            cv.line(image, (x_start, y_start), (x_end, y_end), (color, color), thickness=pixels, lineType=cv.LINE_AA)
        for i in range(COMMON_LINE, RANDOM_LINE):
            random_pixels = get_position(MIN_PIXEL, MAX_PIXEL)
            x_start = get_position(random_pixels, IMAGE_WIDTH - random_pixels)
            x_end = get_position(random_pixels, IMAGE_WIDTH - random_pixels)
            y_start = get_position(random_pixels, IMAGE_WIDTH - random_pixels)
            y_end = get_position(random_pixels, IMAGE_WIDTH - random_pixels)
            color = get_position(COLOR_PIXEL_START, COLOR_PIXEL_END)
            cv.line(image, (x_start, y_start), (x_end, y_end), (color, color), thickness=random_pixels,
                    lineType=cv.LINE_AA)
        cv.imwrite(os.path.join(LINE_PATH, '%d_%d.jpg' % (pixels, pixel_number)), image)

def main():
    start = time.time()
    for i in range(MIN_PIXEL, MAX_PIXEL + 1):
        print('loading', i)
        generate_line_image(i)
        generate_rectangle_image(i)
    print('time', time.time()     star)

if __name__ == '__main__':
main()
```

F2 Training set generation

```python
import cv2 as cv
import numpy as np
from sklearn.model_selection import train_test_split
import os

def load_image(path):
    image_data = []
    image_label = []
    for image_name in os.listdir(path):
        image = cv.imread(os.path.join(path, image_name))
        image_data.append(image)
        image_label.append(int(image_name.split('_')[0]))
    return image_data, image_label

def train_test_generate():
    line_image, line_label = load_image('./line')
    rectangle_image, rectangle_label = load_image('./rectangle')
    image = line_image + rectangle_image
    label = line_label + rectangle_label
    image = np.array(image, 'uint8')
    label = np.array(label)
    train_x, test_x, train_y, test_y = train_test_split(image, label, test_size=.2, random_state=1)
    print(train_x.shape, test_x.shape)
    np.savez('train_data.npz', train_x=train_x, train_y=train_y, test_x=test_x, test_y=test_y)

if __name__ == '__main__':
    train_test_generate()
```

F3 Training the model

```python
import numpy as np
from tensorflow import keras

IMAGE_HEIGHT = 341
IMAGE_WIDTH = 341
CLASSES = 20

def load_data(path='./train_data.npz'):
    data = np.load(path)
    train_x, train_y, test_x, test_y = data['train_x'], data['train_y'], data['test_x'], data['test_y']
    train_x = train_x.astype('float32')
    test_x = test_x.astype('float32')
    train_x = train_x / 255.
    test_x = test_x / 255.
    return np.reshape(train_x, (-1, IMAGE_HEIGHT, IMAGE_WIDTH, 1)), \
           np.reshape(test_x, (-1, IMAGE_HEIGHT, IMAGE_WIDTH, 1)), train_y, test_y
```

```python

def build_model(input_shape=(IMAGE_HEIGHT, IMAGE_WIDTH, 1)):
    base_model = keras.applications.ResNet50(include_top=False,
    input_shape=input_shape, weights='imagenet')
    base_model.trainable = False
    x = keras.layers.GlobalAveragePooling2D()(base_model.output)
    x = keras.layers.Dense(128, activation='relu')(x)
    x = keras.layers.Dropout(0.5)(x)
    x = keras.layers.Dense(32, activation='relu')(x)
    output = keras.layers.Dense(CLASSES, activation='softmax')(x)
    model = keras.models.Model(input=base_model.input, output=
    output)
    return model

def train_model():
    model = build_model()
    train_x, test_x, train_y, test_y = load_data()
    callbacks = [
        keras.callbacks.TensorBoard(update_freq='batch'),
        keras.callbacks.ModelCheckpoint(filepath='model_checkpoint'
    , save_best_only=True, save_weights_only=True),
        keras.callbacks.EarlyStopping(min_delta=0.05, patience=3)
    ]
    model.compile(optimizer='adam', loss=keras.losses.
    categorical_crossentropy, metrics=['accuracy'])
    model.fit(train_x, train_y, epochs=16, batch_size=128,
    validation_data=(test_x, test_y), callbacks=callbacks)
    model.save('my_model')

if __name__ == '__main__':
    train_model()
```

F4 Verify the Result

```python
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
import tensorflow.keras as keras
import os

IMAGE_HEIGHT = 341
IMAGE_WEIGHT = 341

def predict(path=None):
    model = keras.models.load_model('./my_model')
    image = cv.imread(path)
    image = cv.cvtColor(image, cv.COLOR_BGR2RGB)
    image = cv.resize(image, (IMAGE_HEIGHT, IMAGE_WEIGHT))
    show_image = image.copy()
    image = image.astype('float32') / 255.
    image = np.expand_dims(image, 0)
    predict_labels = model.predict(image)[0]
    predict_class = np.argmax(predict_labels) + 1
    plt.figure()
    plt.imshow(show_image)
    plt.title('predict class:%d' % predict_class)
    plt.show()
if __name__ == '__main__':
    predict('./train/1/line_1.jpg')
```

Appendix G MATLAB Code for thicken the fiber in image

```
index = 1;
se90 = strel('line',5,90);
se0 = strel('line',5,0);
for index = 1:41
    BWs = imread(append('test_',string(index),'_result.tif'));
    BWsdil = imdilate(BWs,[se90 se0]);
    imwrite(BWsdil,append('test_',string(index),'
    _result_imdilated_5.tif'))
end
```

Appendix H Diameter determination Python code

```python
import napari
import numpy as np
from quanfima import morphology as mrph
from skimage import filters, io, morphology
from skimage.color import rgb2gray

I = io.imread('0_predicted.tif')
#enable the line below if the image is not a greyscale image
# I = rgb2gray(I)

# skeletalization the image
skeleton = morphology.skeletonize(I)

# dmap and dvals provide fiber diameter maps and the fiber diameter
     values
cskel, fskel, omap, dmap, ovals, dvals = mrph.
    estimate_fiber_properties(I, skeleton)

 # return a list of the diameter arrays
dvals = [dvals]
np.savetxt('diameter',(dvals))
return dvals
```

Appendix I Python Code for synthetic data generation

```
## Generate training set
import os
import time
import cv2 as cv
import numpy as np

# image height
IMAGE_HEIGHT = 341
# image width
IMAGE_WIDTH = 341
# background height
LARGE_IMAGE_HEIGHT = 380
# background width
LARGE_IMAGE_WIDTH = 380
# minimum line pixel number
MIN_PIXEL = 1
# maximum line pixel number
MAX_PIXEL = 20
# Image number for each line width
PIXEL_IMAGE_NUMBER = 500
# line path
LINE_PATH = './line'
# rectangle path
RECTANGLE_PATH = './rectangle'
# 12 common line (0-11)
COMMON_LINE = 11
# 5 random line (12-16)
RANDOM_LINE = 16
# pixel color minimum grayscale
COLOR_PIXEL_START = 150
# pixle color maximum grayscale
COLOR_PIXEL_END = 255

# return random location
def get_position(min_number=0, max_number=255):
    # obtain random number between [min_number,max_number)
    return np.random.randint(min_number, max_number + 1)
# top and bottom random rectangle
def top_to_bottom(image, pixes, index):
    # obtain random grayscale
    color = get_position(COLOR_PIXEL_START, COLOR_PIXEL_END)
    # obtain random length
    if index >= COMMON_LINE:
        # random rectangle width
        random_pixel = get_position(MIN_PIXEL, MAX_PIXEL)
        # obtain random location
        position = get_position(0, LARGE_IMAGE_WEIGHT - random_pixel)
    else:
        # default location
        position = get_position(0, LARGE_IMAGE_WEIGHT - pixes)
    # draw rectangle
    cv.rectangle(image, (position, 0), (position + pixes, LARGE_IMAGE_HEIGHT), (color,
color), thickness=-1)

# left and right random rectangle
def left_to_right(image, pixels, index):
    # obtain a random color
    color = get_position(COLOR_PIXEL_START, COLOR_PIXEL_END)
    # obtain random length
    if index >= COMMON_LINE:
        # random width
        random_pixel = get_position(MIN_PIXEL, MAX_PIXEL)
        # random location
        position = get_position(0, LARGE_IMAGE_HEIGHT - random_pixel)
    else:
        # default location
        position = get_position(0, LARGE_IMAGE_HEIGHT - pixels)
    # draw rectangle
    cv.rectangle(image, (0, position), (LARGE_IMAGE_WEIGHT, position + pixels), (color,
color), thickness=-1)

# generate rectangle images
def generate_rectangle_image(pixes):
    # loop for generation
    for pixel_number in range(PIXEL_IMAGE_NUMBER):
        # generate black background
        image = np.zeros((LARGE_IMAGE_WEIGHT, LARGE_IMAGE_HEIGHT), 'uint8')
        # generate sequence numbers
        index = np.arange(1, 31)
        # shuffle the index
        np.random.shuffle(index)
        odd_index = 0
        even_index = 0
        # traverse index
        for i in index:
            if i % 2 == 0:
                even_index += 1
                top_to_bottom(image, pixes, even_index)
            else:
```

```
            odd_index += 1
            left_to_right(image, pixes, odd_index)
        # get an angle for the lines
        angle = get_position(-5, 5)
        # get rotated matrix
        M = cv.getRotationMatrix2D((LARGE_IMAGE_WEIGHT // 2,
LARGE_IMAGE_WEIGHT // 2), angle, 1)
        # Affine transformation
        image = cv.warpAffine(image, M, (LARGE_IMAGE_WEIGHT,
LARGE_IMAGE_HEIGHT))
        #get the center part of the image
        image = image[20:20 + IMAGE_WIDTH, 20:20 + IMAGE_HEIGHT]
        # save the image
        cv.imwrite(os.path.join(RECTANGLE_PATH, '%d_%d.jpg' % (pixes, pixel_number)),
image)

# generate straight line image
def generate_line_image(pixels):
    for pixel_number in range(PIXEL_IMAGE_NUMBER):
        # generate full black background
        image = np.zeros((IMAGE_HEIGHT, IMAGE_WIDTH), 'uint8')
        # Generate common lines
        for i in range(COMMON_LINE):
            x_start = get_position(pixels, IMAGE_WIDTH - pixels)
            x_end = get_position(pixels, IMAGE_WIDTH - pixels)
            y_start = get_position(pixels, IMAGE_WIDTH - pixels)
            y_end = get_position(pixels, IMAGE_WIDTH - pixels)
            # get random color
            color = get_position(COLOR_PIXEL_START, COLOR_PIXEL_END)
            #draw straight line
            cv.line(image, (x_start, y_start), (x_end, y_end), (color, color), thickness=pixels,
lineType=cv.LINE_AA)
        # generate random width images
        for i in range(COMMON_LINE, RANDOM_LINE):
            # obtain random width for straight line
            random_pixels = get_position(MIN_PIXEL, MAX_PIXEL)
            # obtain random location of straight line
            x_start = get_position(random_pixels, IMAGE_WIDTH - random_pixels)
            x_end = get_position(random_pixels, IMAGE_WIDTH - random_pixels)
            y_start = get_position(random_pixels, IMAGE_WIDTH - random_pixels)
            y_end = get_position(random_pixels, IMAGE_WIDTH - random_pixels)
            #obtain random grayscale
            color = get_position(COLOR_PIXEL_START, COLOR_PIXEL_END)
            #draw straight line
            cv.line(image, (x_start, y_start), (x_end, y_end), (color, color),
thickness=random_pixels,
                    lineType=cv.LINE_AA)
        cv.imwrite(os.path.join(LINE_PATH, '%d_%d.jpg' % (pixels, pixel_number)), image)

# main function
def main():
    start = time.time()
    #generate image
    for i in range(MIN_PIXEL, MAX_PIXEL + 1):
        print('Currently generating:', i)
        # generate line
        generate_line_image(i)
        # generate rectangle image
        generate_rectangle_image(i)
    print('Image generation complete, calculation time:', time.time() - start)

if __name__ == '__main__':
    main()
```

Appendix J MATLAB code for the determination of surface inter-fiber pores

```
% Pore size analysis from given samples
% Erqian Gao

close all
clear all
% input your sample within the same working directory
Input_img = imread('sample 1-01.tif');
Sample = imcrop(Input_img, [0 0 1024 691]);

figure;
imshow(Sample)

Sample = medfilt2(Sample);
Sample = imadjust(Sample);
Sample = histeq(Sample);
SE = strel('square', 2);
Sample = imclose(Sample, SE);
Sample = imopen(Sample, SE);

figure;
imshow(Sample)

scale = imcrop(Input_img, [320 719 94 23]);
[scale, noOfRegions1] = bwlabel(scale, 4);
measurements = regionprops(scale, 'Area', 'PixelList');
area = cat(1, measurements.Area);
[m1, index] = max(area);
ruler = ismember(scale, index);
coordinates = measurements.PixelList;
x1 = coordinates(:,1);
pixelDistance = range(x1) – 2;
fprintf('The pixel length of the scale ruler is: %.2f pixels.\n', pixelDistance)
scalevalue = input('Enter the scale (μm): ');
scale = pixelDistance/scalevalue;
micrometreperpixel = scalevalue / pixelDistance;

h = histogram(Sample);
[inputsizex,inputsizey] = size(Sample);
size_full = inputsizex * inputsizey;
% check if you image is correct / the one you want
% comment out if thats the correct one
```

```
%figure; imshow(Sample)

% set up the threshold
% the image is read in uint8 which means it ranges from 0 to 256
% anything above the pick up threshold will be consider as a valid fiber

% using Otsu’s method
% you can determine the bin size
% bin_size = 16;
% [counts,x] = imhist(Sample, bin_size);
% T = otsuthresh(counts);
% stem(x,counts);
% fprintf(‘The threshold is %f \n’,T*256);
% Density_map = imbinarize(Sample,T);

T = adaptthresh(Sample,0.8,’ForegroundPolarity’,’bright’,’Statistic’,’gaussian’);
Density_map = imbinarize(Sample,T);

% Density_map = imbinarize(Sample, ‘adaptive’);
Density_map_revert = imcomplement(Density_map);
figure;
imshow(Density_map_revert)

SE = strel(‘disk’, 2);
Density_map_revert= imclose(Density_map_revert, SE);
figure;
imshow(Density_map_revert)

Density_map_revert = imopen(Density_map_revert, SE);
figure;
imshow(Density_map_revert)

Density_map_revert = bwmorph(Density_map_revert, ‘majority’);
figure;
imshow(Density_map_revert)

Density_map_revert = bwmorph(Density_map_revert, ‘clean’);
figure;
imshow(Density_map_revert)

[B,L,N,A] = bwboundaries(Density_map_revert,’noholes’);

imshow(Density_map_revert); hold on;
colors=[‘b’ ‘g’ ‘r’ ‘c’ ‘m’ ‘y’];
for k=1:length(B),
  boundary = B{k};
  cidx = mod(k,length(colors))+1;
  plot(boundary(:,2), boundary(:,1),…
      colors(cidx),’LineWidth’,2);

  %randomize text position for better visibility
  %rndRow = ceil(length(boundary)/(mod(rand*k,7)+1));
  %col = boundary(rndRow,2); row = boundary(rndRow,1);
  %h = text(col+1, row-1, num2str(L(row,col)));
  %set(h,’Color’,colors(cidx),’FontSize’,14,’FontWeight’,’bold’);
End

Total_hole_area = sum(sum(Density_map_revert));
Density_percentage = sum(Density_map, ‘all’) / size_full;
average_hole_area = Total_hole_area * micrometreperpixel * micrometreperpixel / N;
fprintf(‘The solidity is %f.\n’,Density_percentage);

Area_chart = zeros(length(B), 3);
current_cell = 0;
while current_cell < length(B)
    current_x = B{current_cell + 1,1}(:,1);
    current_y = B{current_cell + 1,1}(:,2);
    current_cell_area = polyarea(current_x,current_y)* micrometreperpixel *
micrometreperpixel;
    current_cell_area_pixel = polyarea(current_x,current_y);
    estimate_hole_diameter = 2 * sqrt(current_cell_area / pi);
    Area_chart(current_cell + 1, 1) = current_cell + 1;
    Area_chart(current_cell + 1, 2) = current_cell_area_pixel;
    Area_chart(current_cell + 1, 3) = current_cell_area;
    Area_chart(current_cell + 1, 4) = estimate_hole_diameter;
    current_cell = current_cell + 1;
end

average_cavity_diameter = mean(Area_chart(:,3));
fprintf(‘The average cavity diameter is %f (µm).\n’,average_cavity_diameter);
fprintf(‘The average cavity area is %f (µm^2).\n’,average_hole_area);

figure
h_cavity_diameter = histogram(Area_chart(:,4));
xlabel(‘Cavity Diameter in µm’)
ylabel(‘Count’)
title(‘Histogram for Cavity Diameters’);
h_cavity_diameter.NumBins = 100;
figure
h_cavity_area = histogram(Area_chart(:,3));
```

```
xlabel(‘Cavity Area in µm^2’)
ylabel(‘Count’)
title(‘Histogram for Cavity Areas’);
h_cavity_area.NumBins = 100;
```

Appendix K MATLAB code for the determination of intra-fiber pores

```
%% Pore size determination on any given input photos
%% The result would be in unit of pixels
close all
clear all

% input specification
file = ['image1pore.tif'];
I = imread(file);

% we are not using the scale this time
% the Canny Sensitivity
cannySensitivity = input(['\nEnter the Canny Border Detection sensitivity (0-0.999). \nThe
lower the number, the easier matlab detects a region as an edge\n'...
    +'-If set less than 0.2 then a blurring filter will be done on the image (for high res)\n'...
    +'-If set above, no blurring is required (for low res)\nChoose 0.2 for high res images, and
0.21-0.3 for low res images: ']);

isBlur = false;
if cannySensitivity <= 0.2
    isBlur = true;
end

%% image preprocessing

% step one: display the original image

figure;
imshow(I)

% step two: Initial noise reduction for the grayscale image
I = medfilt2(I);

figure;
imshow(I)

% step three: Even out the contrast profile
I = imadjust(I);
I = histeq(I);

figure;
imshow(I)

% step four: Noise Reduction
```

```
SE = strel('square', 2);
I = imclose(I, SE);
I = imopen(I, SE);

figure;
imshow(I)
i = I;

% step six: MAIN Noise Reduction
SE = strel('disk', 2);
I = imclose(I, SE);
I = imopen(I, SE);
I = bwmorph(I, 'majority');
I = bwmorph(I, 'clean');

BWreduced = I;


%BW2 = bwmorph(I, 'skel', Inf);
%skeleton = bwmorph(BW2, 'spur');

% Reverse colors of the binary image
I = imcomplement(I);

% step seven. Store the edge profile of the image and combine with the binary image into N
if isBlur

    i = gaussianFilter(i, 8);

end

i = edge(i, 'canny', cannySensitivity);

% step eight: Perfect the boundaries by first dilating (to seal openings) and thinning
% to reduce the edges to their more accurate thickness
SE = strel('disk', 1);
i = imdilate(i, SE);
i = bwmorph(i, 'thin', Inf);

N = I | i;

figure;
imshow(N)

% step nine: Noise reduction. 'clean' gets rid of single pixel outliers
```

```
SE = strel('disk', 2);
N = imclose(N, SE);
N = bwmorph(N, 'clean');
N = imcomplement(N);
N = bwmorph(N, 'clean');
N = imcomplement(N);

withEdge = N;

figure;
imshow(withEdge)

% 10. Diameter Measurement
dist = bwdist(N, 'euclidean');
dist = dist*2;
% 7. Create a skeleton image and use pruning to delete sporadic branches
skeleton = imregionalmax(dist, 4);
D = dist(find( skeleton == 1 ));


% Display image with blue skeleton
% convert to rgb first
rgb = repmat(N, [1 1 3]);
rgb = cat(3, N, N, N);
rgb = repmat(double(N)./255,[1 1 3]);

finalImage = N | skeleton;

figure;
imshow(finalImage)



m = max(D);
% 11a. Create the histogram from the measured distance data (radius x 2).
% Find the ranges necessary for plotting
range = 0:1:m-mod(m, 20)+20;
noOfRanges = length(range) -1;
frequency = zeros(1,noOfRanges);

% Store the diameters in the ranges
for i = 1:noOfRanges

   frequency(i)=length(find( D > range(i) & D < range(i+1) ));

end
```

```
%11.b Angle distribution
[magnitude, direction] = imgradient(dist);
angle = skeleton .* direction;
% Find the negative values and convert to the corresponding positive value
% (just add 180 degrees)
negIndices = find(angle < 0);
if ~isempty(negIndices)
   angle(negIndices) = angle(negIndices) + 180;
end

% imgradient gets the angle towards the edge, but we need the angle along
% the fiber, so just add 90 degrees
angle = angle + 90;
fixIndices = find(angle > 180);
if ~isempty(fixIndices)
   angle(fixIndices) = angle(fixIndices) + 180;
end

fprintf('\n\nThe average fiber orientation is %.2f degrees\n\n', mean(mean(angle)))

nelements = sum(frequency);
% Convert the frequency into a percentage
frequency = frequency ./ nelements * 100;
%frequency
%range

%%
% 12. Plot the data

figure('Position', [20 20 1200 760]), bar(range(2:end)-10 , frequency, 1)
endlimit = floor(range(end)/20)*20;
if mod(range(end), 20) > 0
   endlimit = endlimit + 20;
end

set(gca, 'XTick', [0:60:endlimit])

xlabel('Diameter [pixel]')
ylabel('Frequency [%]')
```

Appendix L MATLAB code for the determination of fiber porosity

```
% input your sample within the same working directory
Sample = imread('NF1.tif');

% for image that is not grayscale
Sample = Sample(:, :, 1);

h = histogram(Sample);
[inputsizex,inputsizey] = size(Sample);
size_full = inputsizex * inputsizey;
% check if you image is correct / the one you want
% comment out if thats the correct one

figure; imshow(Sample)

% set up the threshold
% the image is read in uint8 which means it ranges from 0 to 256
% anything above the pick up threshold will be consider as a valid fiber
% Serval thresholding method is provided below. Comment out the one you do not want to
use.

% 1. using manually input threshold value
   pick_up_threshold = 85;
   Density_map = imbinarize(Sample,pick_up_threshold/256);
% Density_map = Sample > pick_up_threshold;
   Density_map_revert = imcomplement(Density_map);

% 2. using 85% average greyscale
meanGL = mean(Sample(:));
pick_up_threshold = 0.85*meanGL;

% 3. using Otsu's method:
% the bin size can be changed
bin_size = 256;
[counts,x] = imhist(Sample, bin_size);
T = otsuthresh(counts);
stem(x,counts);
fprintf('The threshold is %f \n',T*256);

Density_map = imbinarize(Sample,T);
Density_map_revert = imcomplement(Density_map);

% Check the density map
figure; imagesc(Density_map(:,:,1))
```

```
Solidity = sum(Density_map, 'all') / size_full;
fprintf('The solidity is %f \n',Solidity);
fprintf('The porosity is %f', 1-Solidity);
```

Appendix M MATLAB code for the determination of Surface homogeneity

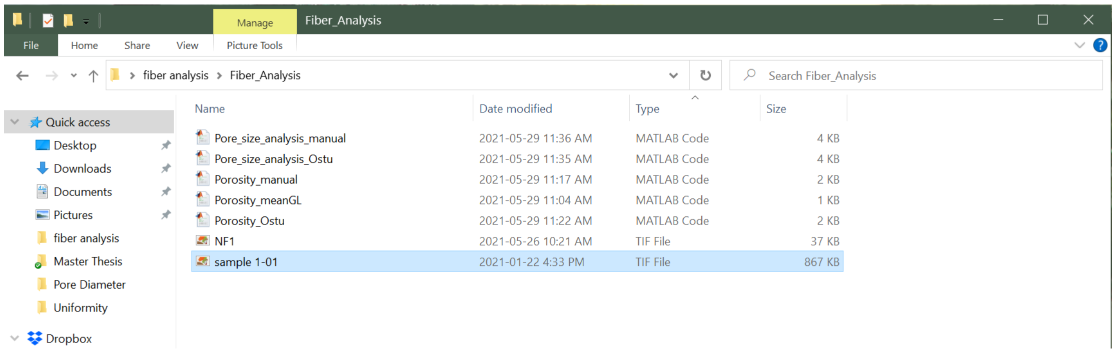
```
close all
clear all

i=imread('2_adjust.jpg');
i=rgb2gray(i);
i=double(i);
%sq1=var(i,0,1);
%sq2=var(i,0,2);
avg=mean2(i);
[m,n]=size(i);
s=0;
for x=1:m
   for y=1:n
   s=s+(i(x,y)-avg)^2;
   end
end
a1=var(i(:)); % Use the var function to calculate sd
a2=s/(m*n-1); % definition of sd
a3=(std2(i))^2; % use std2 function to get sd
cv = a1 / avg;

fprintf('The standard deviation is: %f.\n', a1);
fprintf('The average is: %f.\n', avg);
fprintf('The cv is: %f.\n', cv);
```

Appendix N Manual for Operation of the Automated Tools

1.Porosity Calculation

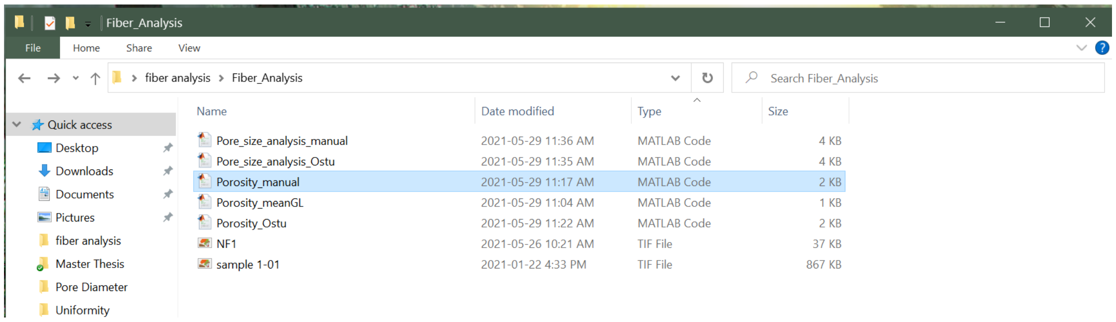
1.1 Manually Thresholding Method

Put the SEM image in the same folder with the script



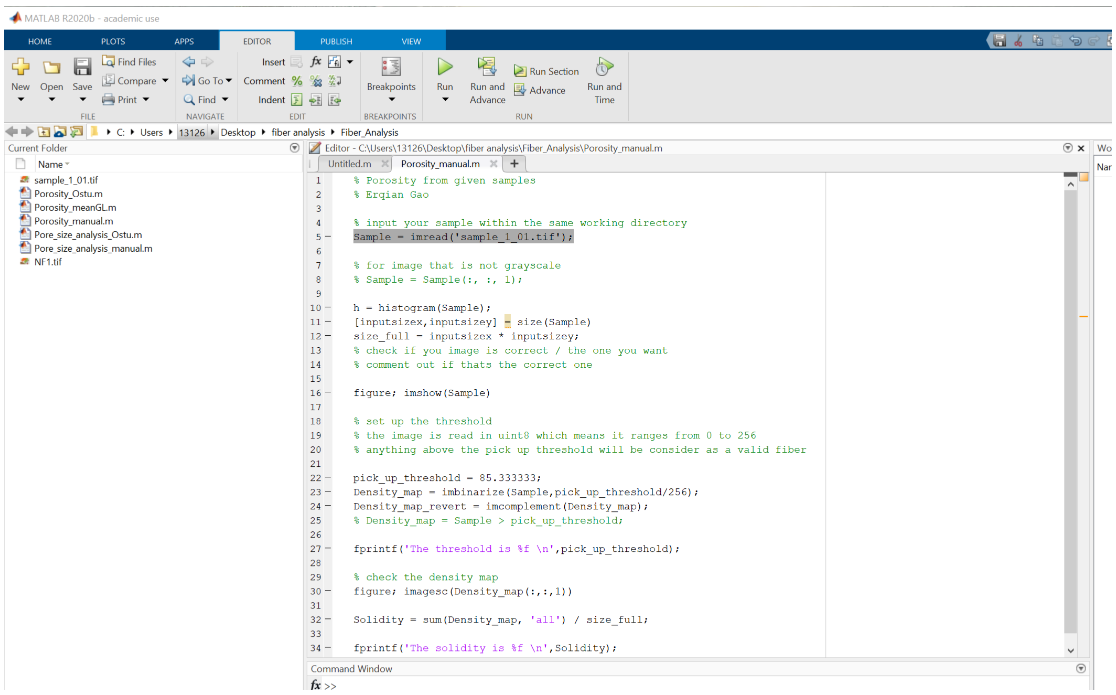
Name	Date modified	Type	Size
Pore_size_analysis_manual	2021-05-29 11:36 AM	MATLAB Code	4 KB
Pore_size_analysis_Ostu	2021-05-29 11:35 AM	MATLAB Code	4 KB
Porosity_manual	2021-05-29 11:17 AM	MATLAB Code	2 KB
Porosity_meanGL	2021-05-29 11:04 AM	MATLAB Code	1 KB
Porosity_Ostu	2021-05-29 11:22 AM	MATLAB Code	2 KB
NF1	2021-05-26 10:21 AM	TIF File	37 KB
sample 1-01	2021-01-22 4:33 PM	TIF File	867 KB

Open Porosity_manual file in the folder



Name	Date modified	Type	Size
Pore_size_analysis_manual	2021-05-29 11:36 AM	MATLAB Code	4 KB
Pore_size_analysis_Ostu	2021-05-29 11:35 AM	MATLAB Code	4 KB
Porosity_manual	2021-05-29 11:17 AM	MATLAB Code	2 KB
Porosity_meanGL	2021-05-29 11:04 AM	MATLAB Code	1 KB
Porosity_Ostu	2021-05-29 11:22 AM	MATLAB Code	2 KB
NF1	2021-05-26 10:21 AM	TIF File	37 KB
sample 1-01	2021-01-22 4:33 PM	TIF File	867 KB

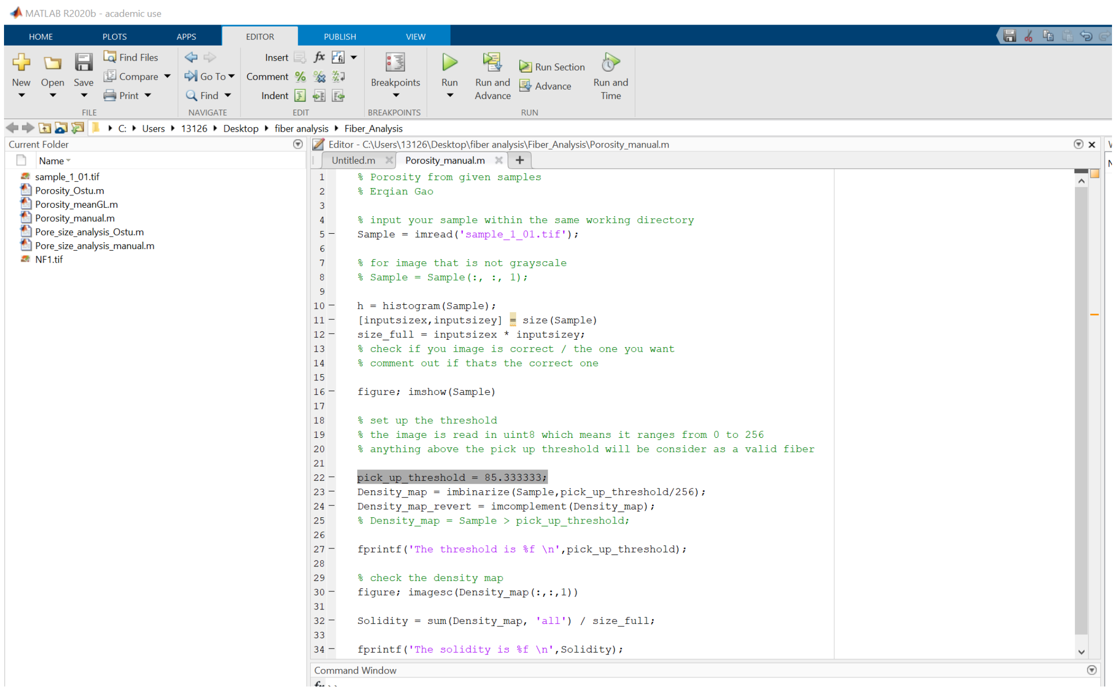
Change the highlighted line to the name of the SEM image file (In this example the file name is sample_1_01)


MATLAB R2020b - academic use
HOME PLOTS APPS EDITOR PUBLISH VIEW
New Open Save Find Files Compare Print FILE
Go To Find NAVIGATE
Insert Comment Indent EDIT
Breakpoints BREAKPOINTS
Run Run and Advance Run Section Advance Run and Time RUN
C: Users 13126 Desktop fiber analysis Fiber_Analysis
Current Folder
Name
sample_1_01.tif
Porosity_Ostu.m
Porosity_meanGL.m
Porosity_manual.m
Pore_size_analysis_Ostu.m
Pore_size_analysis_manual.m
NF1.tif
Editor - C:\Users\13126\Desktop\fiber analysis\Fiber_Analysis\Porosity_manual.m
Untitled.m Porosity_manual.m
Command Window

```
% Porosity from given samples
% Erqian Gao

% input your sample within the same working directory
Sample = imread('sample_1_01.tif');

% for image that is not grayscale
% Sample = Sample(:, :, 1);

h = histogram(Sample);
[inputsizex,inputsizey] = size(Sample)
size_full = inputsizex * inputsizey;
% check if you image is correct / the one you want
% comment out if thats the correct one

figure; imshow(Sample)

% set up the threshold
% the image is read in uint8 which means it ranges from 0 to 256
% anything above the pick up threshold will be consider as a valid fiber

pick_up_threshold = 85.333333;
Density_map = imbinarize(Sample,pick_up_threshold/256);
Density_map_revert = imcomplement(Density_map);
% Density_map = Sample > pick_up_threshold;

fprintf('The threshold is %f \n',pick_up_threshold);

% check the density map
figure; imagesc(Density_map(:,:,1))

Solidity = sum(Density_map, 'all') / size_full;

fprintf('The solidity is %f \n',Solidity);
```

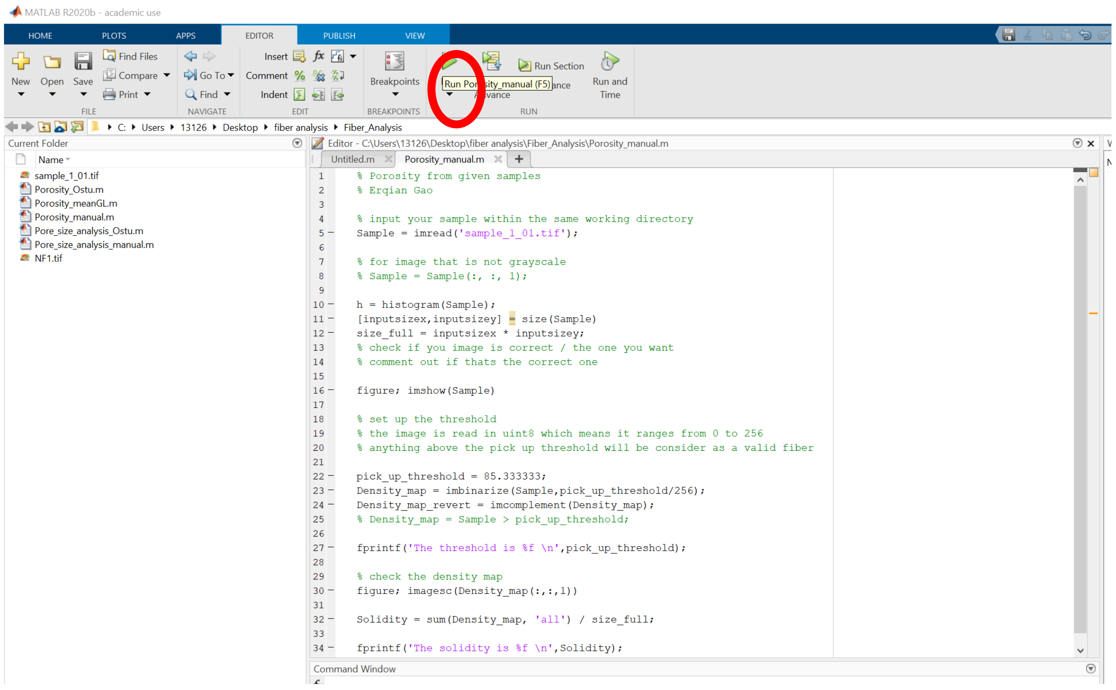
```
fx >>
```

Set up the threshold manually, the threshold value should be between 0~255


MATLAB R2020b - academic use
HOME PLOTS APPS EDITOR PUBLISH VIEW
New Open Save Find Files Compare Print FILE
Go To Find NAVIGATE
Insert Comment Indent EDIT
Breakpoints BREAKPOINTS
Run Run and Advance Run Section Advance Run and Time RUN
C: Users 13126 Desktop fiber analysis Fiber_Analysis
Current Folder
Name
sample_1_01.tif
Porosity_Ostu.m
Porosity_meanGL.m
Porosity_manual.m
Pore_size_analysis_Ostu.m
Pore_size_analysis_manual.m
NF1.tif
Editor - C:\Users\13126\Desktop\fiber analysis\Fiber_Analysis\Porosity_manual.m
Untitled.m Porosity_manual.m
Command Window

```
% Porosity from given samples
% Erqian Gao

% input your sample within the same working directory
Sample = imread('sample_1_01.tif');

% for image that is not grayscale
% Sample = Sample(:, :, 1);

h = histogram(Sample);
[inputsizex,inputsizey] = size(Sample)
size_full = inputsizex * inputsizey;
% check if you image is correct / the one you want
% comment out if thats the correct one

figure; imshow(Sample)

% set up the threshold
% the image is read in uint8 which means it ranges from 0 to 256
% anything above the pick up threshold will be consider as a valid fiber

pick_up_threshold = 85.333333;
Density_map = imbinarize(Sample,pick_up_threshold/256);
Density_map_revert = imcomplement(Density_map);
% Density_map = Sample > pick_up_threshold;

fprintf('The threshold is %f \n',pick_up_threshold);

% check the density map
figure; imagesc(Density_map(:,:,1))

Solidity = sum(Density_map, 'all') / size_full;

fprintf('The solidity is %f \n',Solidity);
```

Run the file (hit the run button or press F5)


MATLAB R2020b - academic use
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Run Porosity_manual (F5)
Run Section
Advance
Run and Time
Breakpoints
C: ▸ Users ▸ 13126 ▸ Desktop ▸ fiber analysis ▸ Fiber_Analysis
Current Folder
sample_1_01.tif
Porosity_Ostu.m
Porosity_meanGL.m
Porosity_manual.m
Pore_size_analysis_Ostu.m
Pore_size_analysis_manual.m
NF1.tif
Editor - C:\Users\13126\Desktop\fiber analysis\Fiber_Analysis\Porosity_manual.m
Untitled.m
Porosity_manual.m
Command Window

```
% Porosity from given samples
% Erqian Gao

% input your sample within the same working directory
Sample = imread('sample_1_01.tif');

% for image that is not grayscale
% Sample = Sample(:, :, 1);

h = histogram(Sample);
[inputsizex,inputsizey] = size(Sample)
size_full = inputsizex * inputsizey;
% check if you image is correct / the one you want
% comment out if thats the correct one

figure; imshow(Sample)

% set up the threshold
% the image is read in uint8 which means it ranges from 0 to 256
% anything above the pick up threshold will be consider as a valid fiber

pick_up_threshold = 85.333333;
Density_map = imbinarize(Sample,pick_up_threshold/256);
Density_map_revert = imcomplement(Density_map);
% Density_map = Sample > pick_up_threshold;

fprintf('The threshold is %f \n',pick_up_threshold);

% check the density map
figure; imagesc(Density_map(:,:,1))

Solidity = sum(Density_map, 'all') / size_full;

fprintf('The solidity is %f \n',Solidity);
```

Wait couple of minutes and get the results

Three figures will appear

Figure 1)

The histogram of greyscale value of every pixel in the image

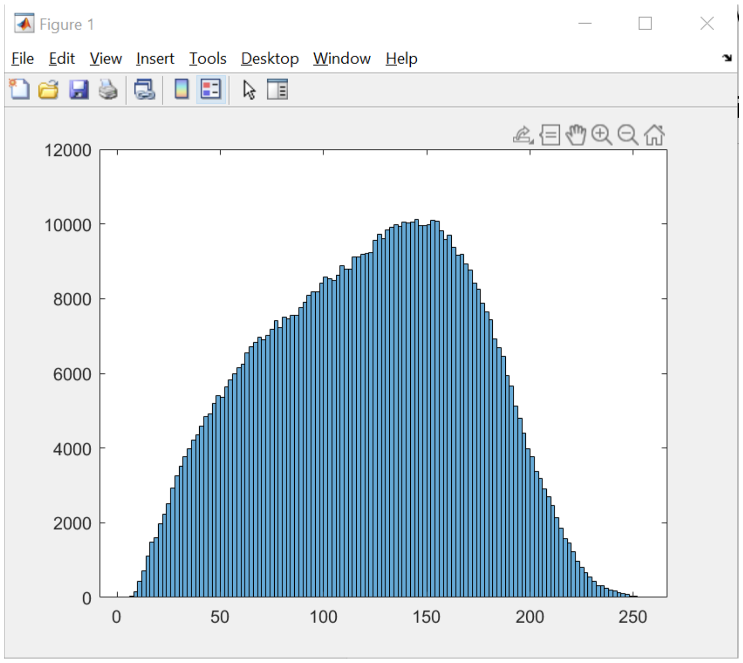


Figure 2)

The original picture

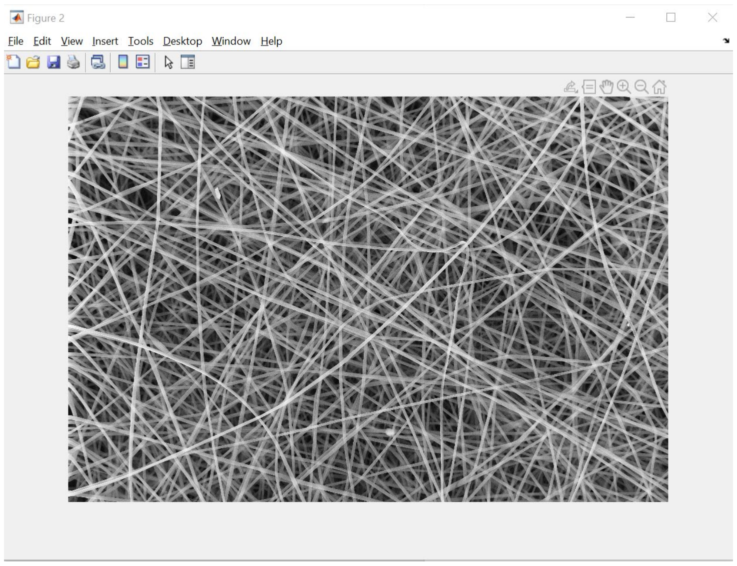
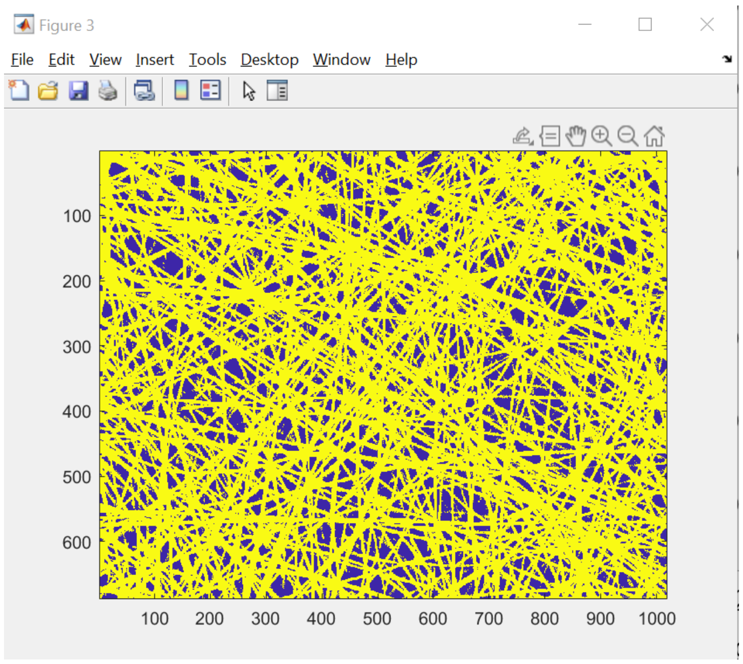
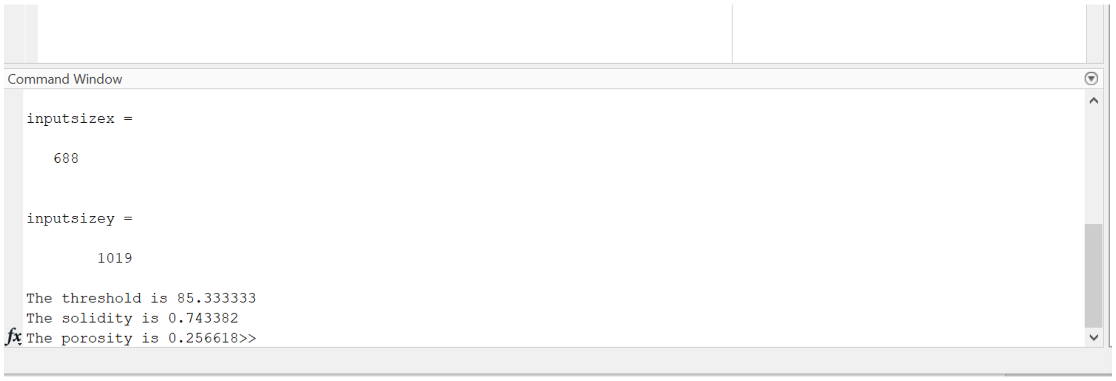


Figure 3)

The thresholding result of the picture



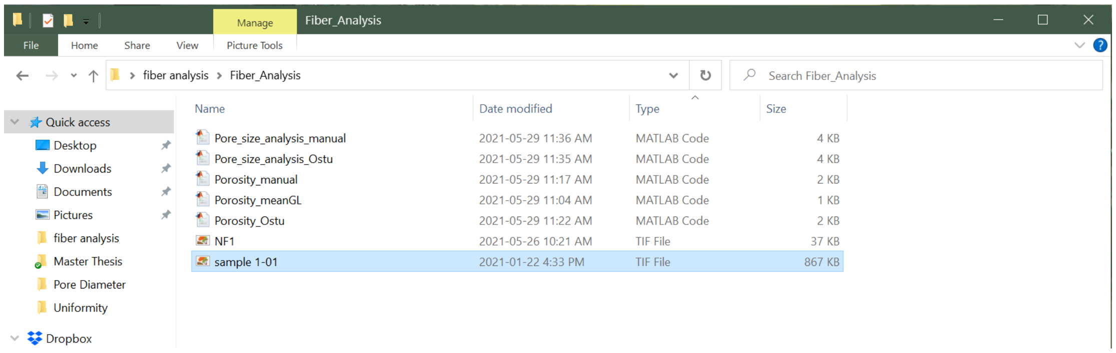
The porosity number will appear in the command window



```
Command Window

inputsizex =

   688


inputsizey =

        1019

The threshold is 85.333333
The solidity is 0.743382
The porosity is 0.256618>>
```

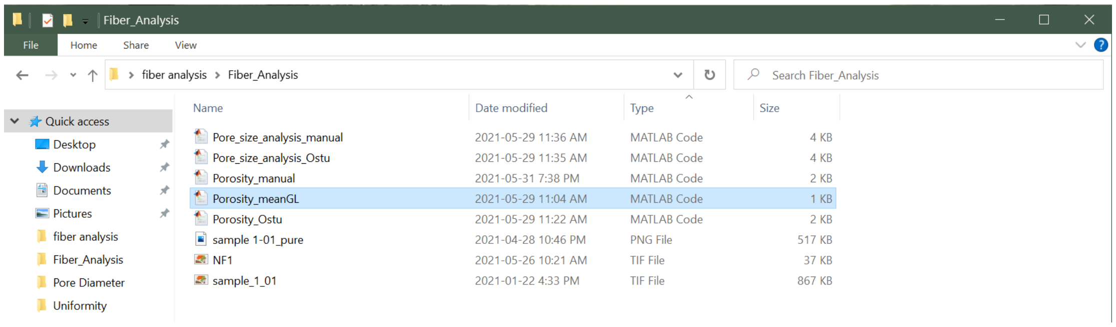
1.2 85% Average Grey Scale Thresholding Method

Put the SEM image in the same folder with the script



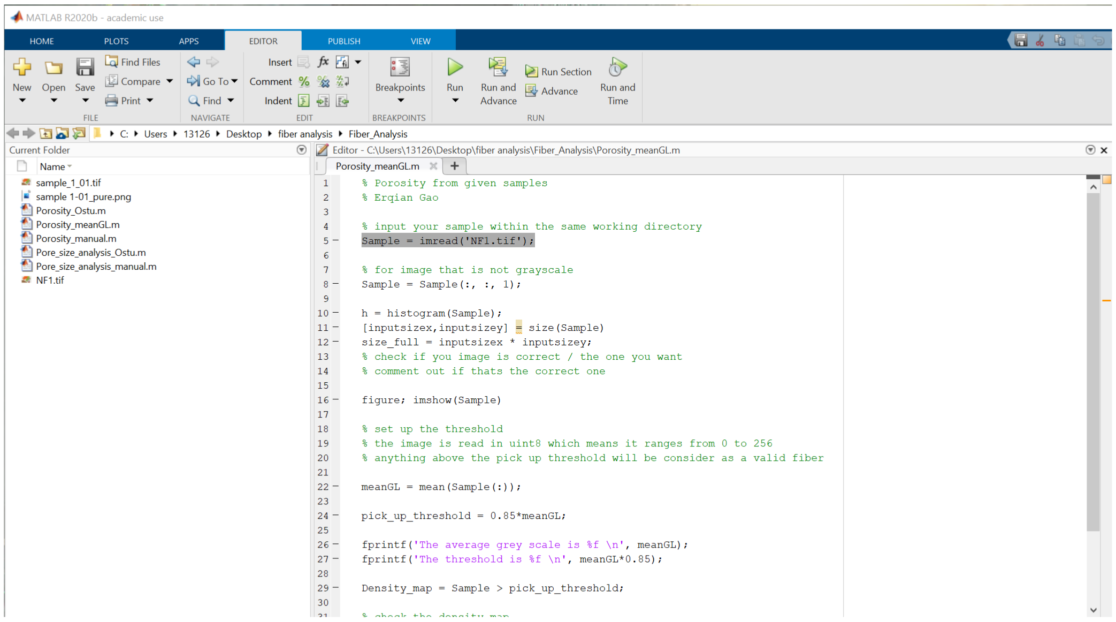
Name	Date modified	Type	Size
Pore_size_analysis_manual	2021-05-29 11:36 AM	MATLAB Code	4 KB
Pore_size_analysis_Ostu	2021-05-29 11:35 AM	MATLAB Code	4 KB
Porosity_manual	2021-05-29 11:17 AM	MATLAB Code	2 KB
Porosity_meanGL	2021-05-29 11:04 AM	MATLAB Code	1 KB
Porosity_Ostu	2021-05-29 11:22 AM	MATLAB Code	2 KB
NF1	2021-05-26 10:21 AM	TIF File	37 KB
sample 1-01	2021-01-22 4:33 PM	TIF File	867 KB

Open Porosity_meanGL file in the folder



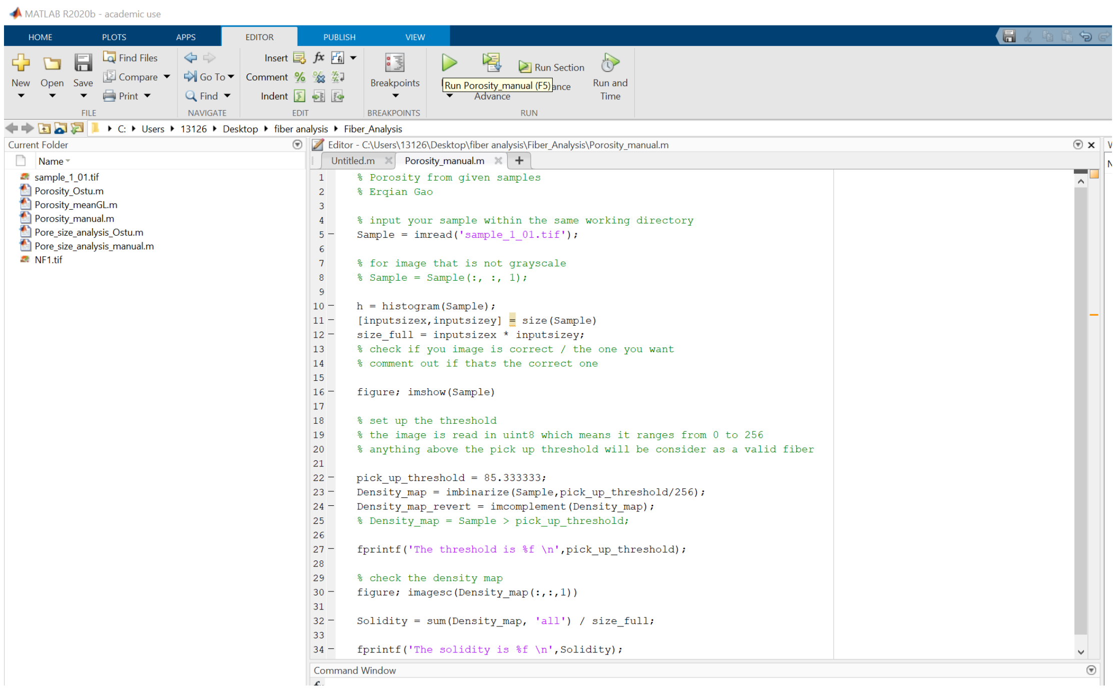
Name	Date modified	Type	Size
Pore_size_analysis_manual	2021-05-29 11:36 AM	MATLAB Code	4 KB
Pore_size_analysis_Ostu	2021-05-29 11:35 AM	MATLAB Code	4 KB
Porosity_manual	2021-05-31 7:38 PM	MATLAB Code	2 KB
Porosity_meanGL	2021-05-29 11:04 AM	MATLAB Code	1 KB
Porosity_Ostu	2021-05-29 11:22 AM	MATLAB Code	2 KB
sample 1-01_pure	2021-04-28 10:46 PM	PNG File	517 KB
NF1	2021-05-26 10:21 AM	TIF File	37 KB
sample_1_01	2021-01-22 4:33 PM	TIF File	867 KB

Change the highlighted line to the name of the SEM image file (In this example the file name is sample_1_01)


MATLAB R2020b - academic use
HOME PLOTS APPS EDITOR PUBLISH VIEW
New Open Save Find Files Compare Print FILE
Go To Find NAVIGATE
Insert Comment Indent EDIT
Breakpoints BREAKPOINTS
Run Run and Advance Run Section Advance Run and Time RUN
C: Users 13126 Desktop fiber analysis Fiber_Analysis
Current Folder
Name
sample_1_01.tif
sample 1-01_pure.png
Porosity_Ostu.m
Porosity_meanGL.m
Porosity_manual.m
Pore_size_analysis_Ostu.m
Pore_size_analysis_manual.m
NF1.tif
Editor - C:\Users\13126\Desktop\fiber analysis\Fiber_Analysis\Porosity_meanGL.m
Porosity_meanGL.m

```
% Porosity from given samples
% Erqian Gao

% input your sample within the same working directory
Sample = imread('NF1.tif');

% for image that is not grayscale
Sample = Sample(:, :, 1);

h = histogram(Sample);
[inputsizex,inputsizey] = size(Sample)
size_full = inputsizex * inputsizey;
% check if you image is correct / the one you want
% comment out if thats the correct one

figure; imshow(Sample)

% set up the threshold
% the image is read in uint8 which means it ranges from 0 to 256
% anything above the pick up threshold will be consider as a valid fiber

meanGL = mean(Sample(:));

pick_up_threshold = 0.85*meanGL;

fprintf('The average grey scale is %f \n', meanGL);
fprintf('The threshold is %f \n', meanGL*0.85);

Density_map = Sample > pick_up_threshold;
```

Run the file (hit the run button or press F5)


MATLAB R2020b - academic use
HOME PLOTS APPS EDITOR PUBLISH VIEW
New Open Save Find Files Compare Print FILE
Go To Find NAVIGATE
Insert Comment Indent EDIT
Breakpoints BREAKPOINTS
Run Porosity_manual (F5) Advance Run Section Run and Time RUN
C: Users 13126 Desktop fiber analysis Fiber_Analysis
Current Folder
Name
sample_1_01.tif
Porosity_Ostu.m
Porosity_meanGL.m
Porosity_manual.m
Pore_size_analysis_Ostu.m
Pore_size_analysis_manual.m
NF1.tif
Editor - C:\Users\13126\Desktop\fiber analysis\Fiber_Analysis\Porosity_manual.m
Untitled.m
Porosity_manual.m
Command Window

```
% Porosity from given samples
% Erqian Gao

% input your sample within the same working directory
Sample = imread('sample_1_01.tif');

% for image that is not grayscale
% Sample = Sample(:, :, 1);

h = histogram(Sample);
[inputsizex,inputsizey] = size(Sample)
size_full = inputsizex * inputsizey;
% check if you image is correct / the one you want
% comment out if thats the correct one

figure; imshow(Sample)

% set up the threshold
% the image is read in uint8 which means it ranges from 0 to 256
% anything above the pick up threshold will be consider as a valid fiber

pick_up_threshold = 85.333333;
Density_map = imbinarize(Sample,pick_up_threshold/256);
Density_map_revert = imcomplement(Density_map);
% Density_map = Sample > pick_up_threshold;

fprintf('The threshold is %f \n',pick_up_threshold);

% check the density map
figure; imagesc(Density_map(:,:,1))

Solidity = sum(Density_map, 'all') / size_full;

fprintf('The solidity is %f \n',Solidity);
```

Wait couple of minutes and get the results

Three figures will appear

Figure 1)

The histogram of greyscale value of every pixel in the image

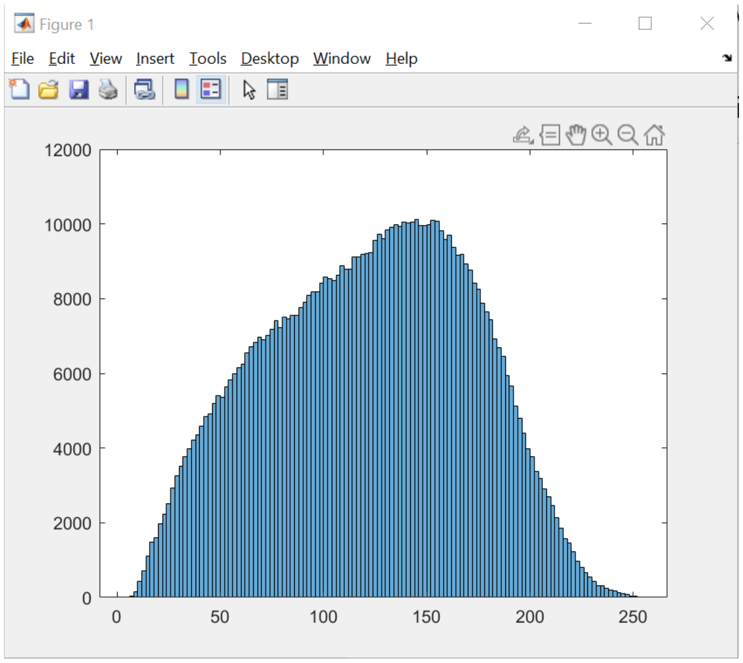


Figure 2)

The original picture

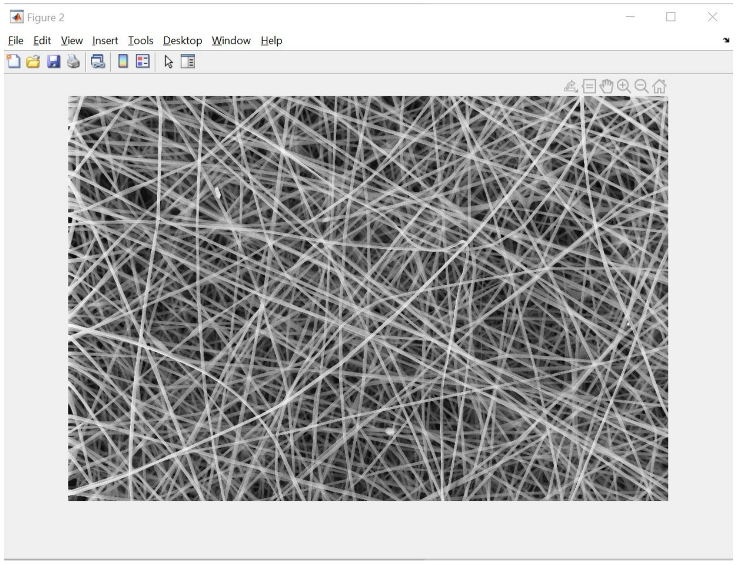
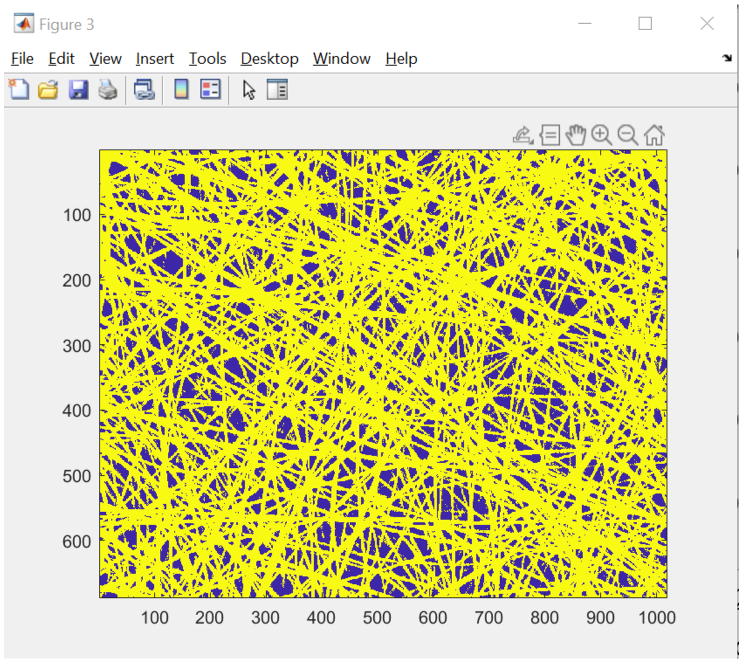
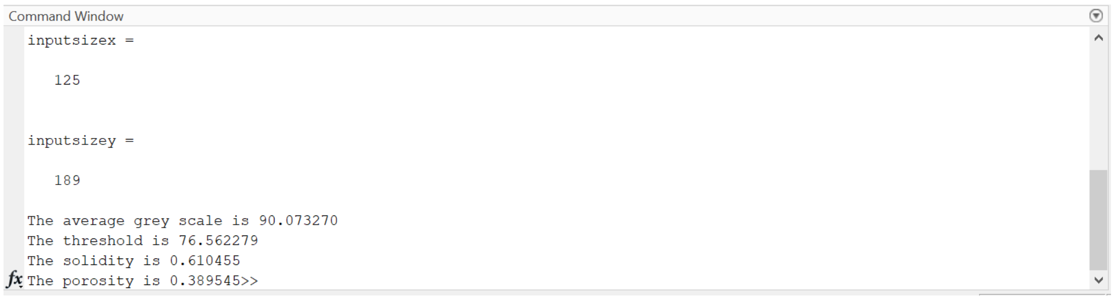


Figure 3)

The thresholding result of the picture

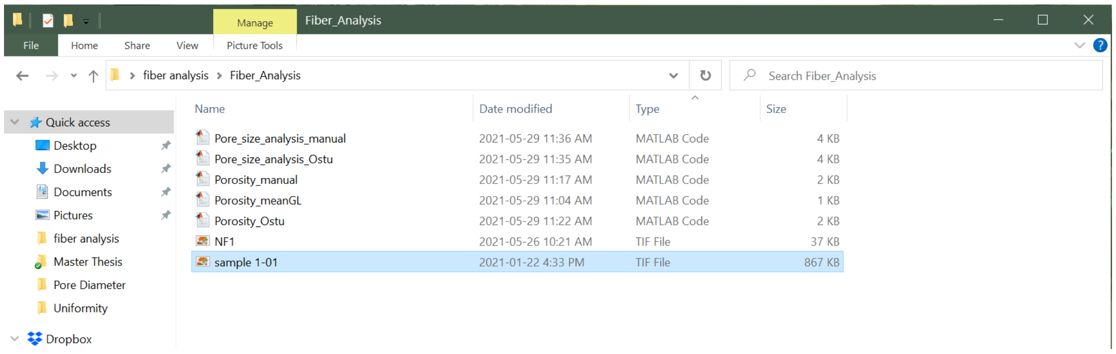


The porosity number will appear in the command window



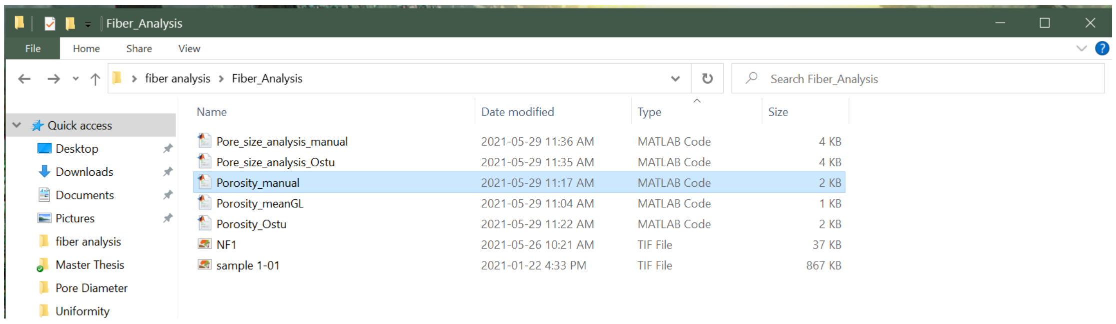
```
inputsizex =

   125


inputsizey =

   189

The average grey scale is 90.073270
The threshold is 76.562279
The solidity is 0.610455
fx The porosity is 0.389545>>
```

Put the SEM image in the same folder with the script



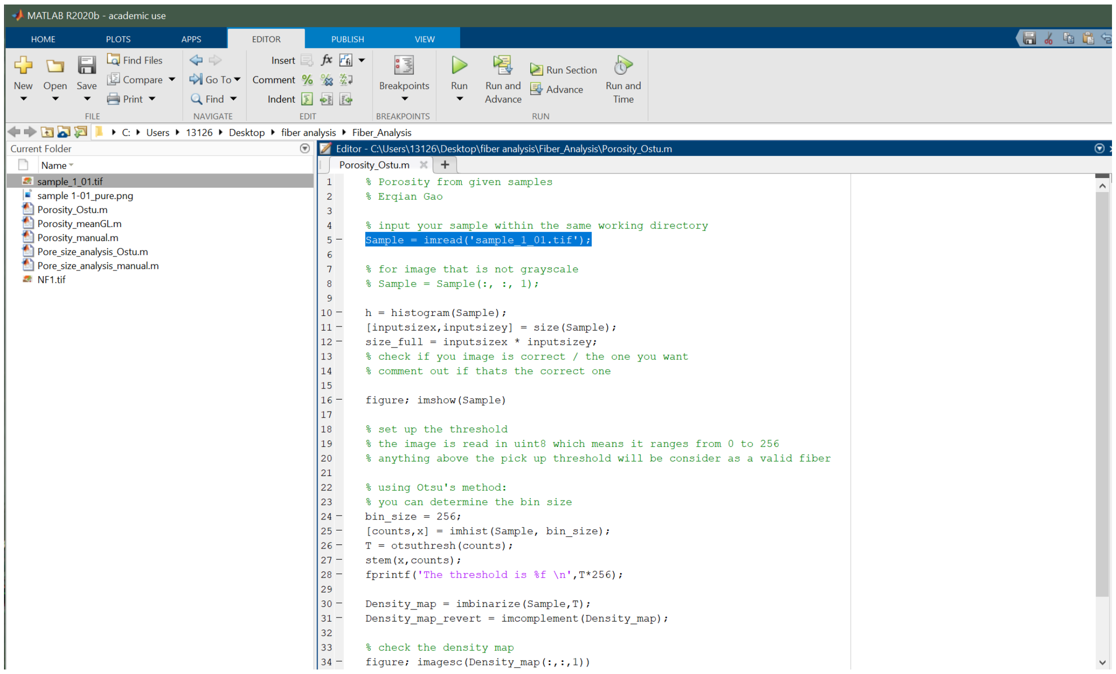
Name	Date modified	Type	Size
Pore_size_analysis_manual	2021-05-29 11:36 AM	MATLAB Code	4 KB
Pore_size_analysis_Ostu	2021-05-29 11:35 AM	MATLAB Code	4 KB
Porosity_manual	2021-05-29 11:17 AM	MATLAB Code	2 KB
Porosity_meanGL	2021-05-29 11:04 AM	MATLAB Code	1 KB
Porosity_Ostu	2021-05-29 11:22 AM	MATLAB Code	2 KB
NF1	2021-05-26 10:21 AM	TIF File	37 KB
sample 1-01	2021-01-22 4:33 PM	TIF File	867 KB

Open Porosity_Ostu file in the folder



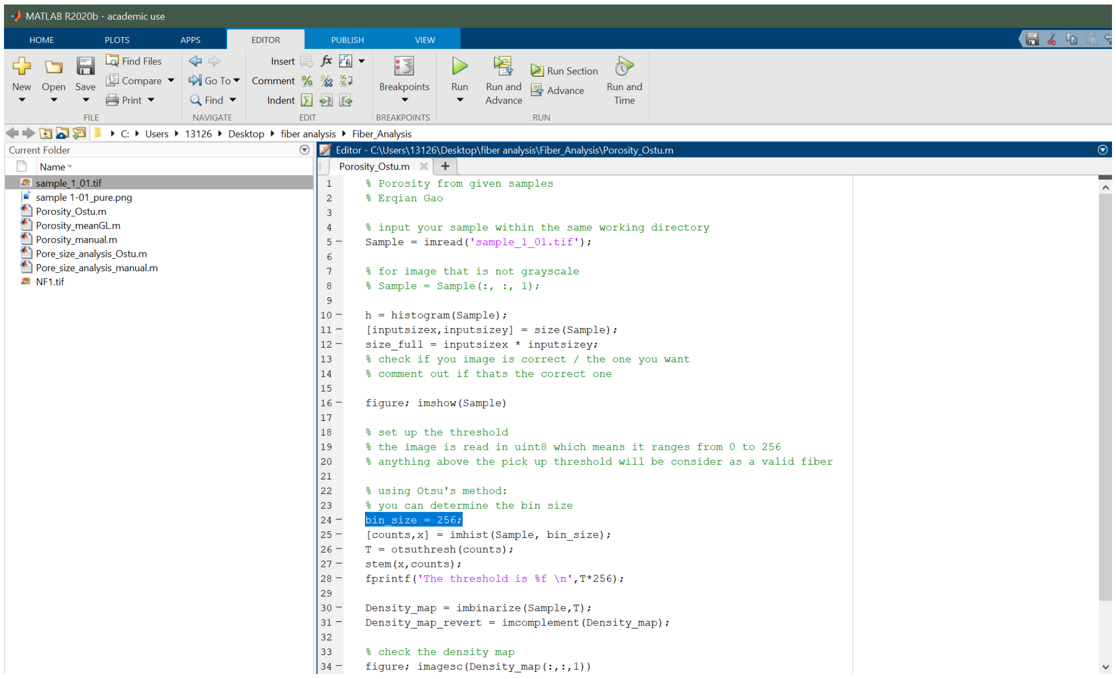
Name	Date modified	Type	Size
Pore_size_analysis_manual	2021-05-29 11:36 AM	MATLAB Code	4 KB
Pore_size_analysis_Ostu	2021-05-29 11:35 AM	MATLAB Code	4 KB
Porosity_manual	2021-05-29 11:17 AM	MATLAB Code	2 KB
Porosity_meanGL	2021-05-29 11:04 AM	MATLAB Code	1 KB
Porosity_Ostu	2021-05-29 11:22 AM	MATLAB Code	2 KB
NF1	2021-05-26 10:21 AM	TIF File	37 KB
sample 1-01	2021-01-22 4:33 PM	TIF File	867 KB

Change the highlighted line to the name of the SEM image file (In this example the file name is sample_1_01)



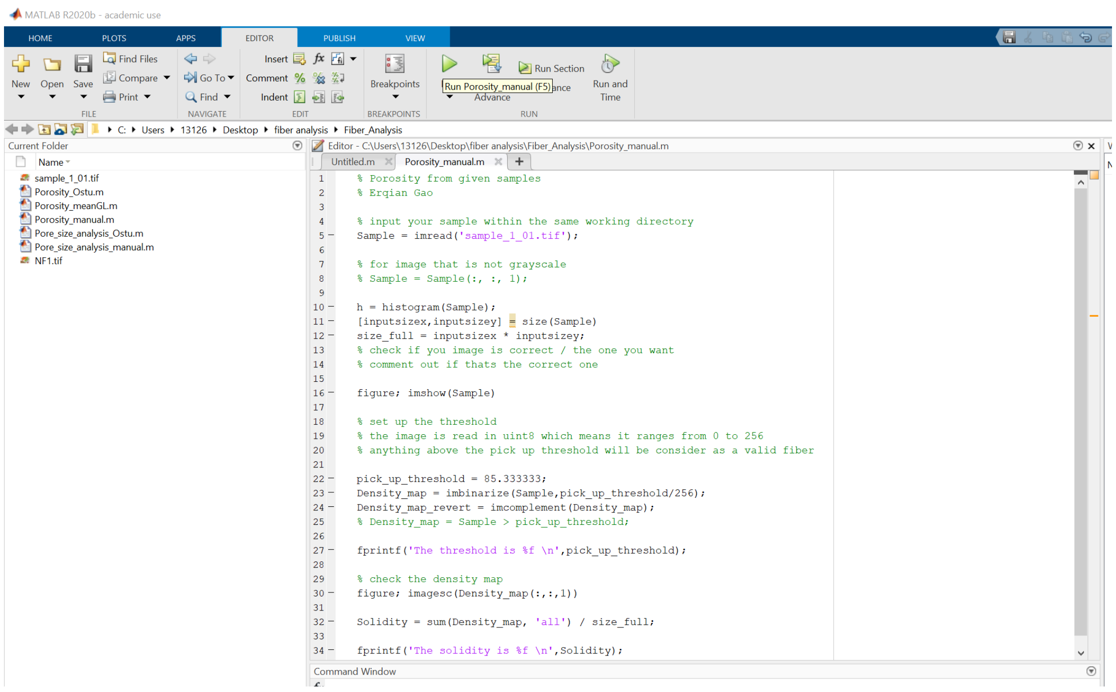
```
% Porosity from given samples
% Erqian Gao

% input your sample within the same working directory
Sample = imread('sample_1_01.tif');

% for image that is not grayscale
% Sample = Sample(:, :, 1);

h = histogram(Sample);
[inputsizex,inputsizey] = size(Sample);
size_full = inputsizex * inputsizey;
% check if you image is correct / the one you want
% comment out if thats the correct one

figure; imshow(Sample)

% set up the threshold
% the image is read in uint8 which means it ranges from 0 to 256
% anything above the pick up threshold will be consider as a valid fiber

% using Otsu's method:
% you can determine the bin size
bin_size = 256;
[counts,x] = imhist(Sample, bin_size);
T = otsuthresh(counts);
stem(x,counts);
fprintf('The threshold is %f \n',T*256);

Density_map = imbinarize(Sample,T);
Density_map_revert = imcomplement(Density_map);

% check the density map
figure; imagesc(Density_map(:,:,1))
```

Set up the bin size manually, the value should be between 16~ 256 and it will influence the binary result. More bins will result in a finer area division and usually a decrease in porosity.



```
% Porosity from given samples
% Erqian Gao

% input your sample within the same working directory
Sample = imread('sample_1_01.tif');

% for image that is not grayscale
% Sample = Sample(:, :, 1);

h = histogram(Sample);
[inputsizex,inputsizey] = size(Sample);
size_full = inputsizex * inputsizey;
% check if you image is correct / the one you want
% comment out if thats the correct one

figure; imshow(Sample)

% set up the threshold
% the image is read in uint8 which means it ranges from 0 to 256
% anything above the pick up threshold will be consider as a valid fiber

% using Otsu's method:
% you can determine the bin size
bin_size = 256;
[counts,x] = imhist(Sample, bin_size);
T = otsuthresh(counts);
stem(x,counts);
fprintf('The threshold is %f \n',T*256);

Density_map = imbinarize(Sample,T);
Density_map_revert = imcomplement(Density_map);

% check the density map
figure; imagesc(Density_map(:,:,1))
```

Run the file (hit the run button or press F5)



```
% Porosity from given samples
% Erqian Gao

% input your sample within the same working directory
Sample = imread('sample_1_01.tif');

% for image that is not grayscale
% Sample = Sample(:, :, 1);

h = histogram(Sample);
[inputsizex,inputsizey] = size(Sample)
size_full = inputsizex * inputsizey;
% check if you image is correct / the one you want
% comment out if thats the correct one

figure; imshow(Sample)

% set up the threshold
% the image is read in uint8 which means it ranges from 0 to 256
% anything above the pick up threshold will be consider as a valid fiber

pick_up_threshold = 85.333333;
Density_map = imbinarize(Sample,pick_up_threshold/256);
Density_map_revert = imcomplement(Density_map);
% Density_map = Sample > pick_up_threshold;

fprintf('The threshold is %f \n',pick_up_threshold);

% check the density map
figure; imagesc(Density_map(:,:,1))

Solidity = sum(Density_map, 'all') / size_full;

fprintf('The solidity is %f \n',Solidity);
```

Wait couple of minutes and get the results

Three figures will appear

Figure 1)

The histogram of greyscale value of every pixel in the image

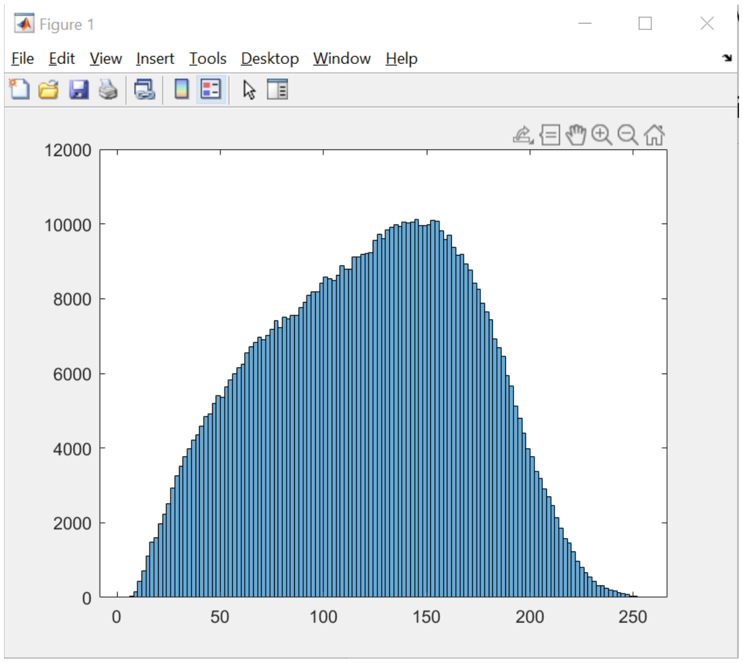


Figure 2)

The original picture

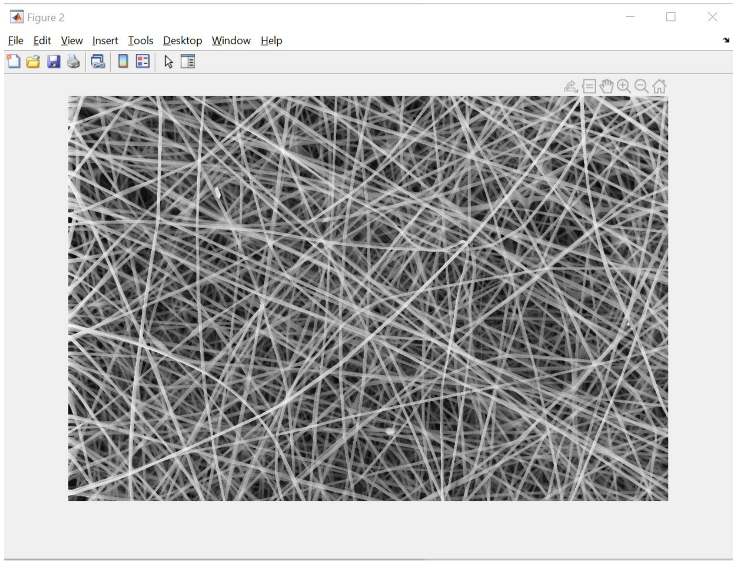
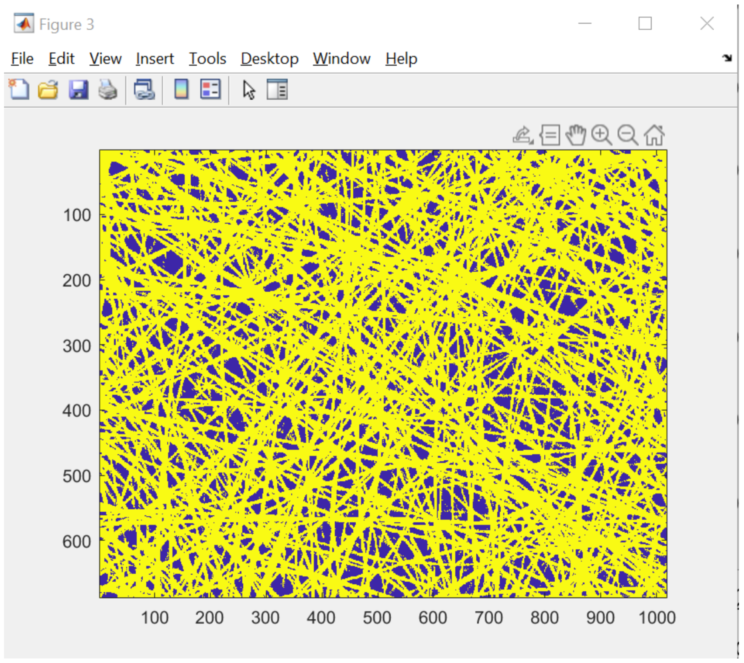
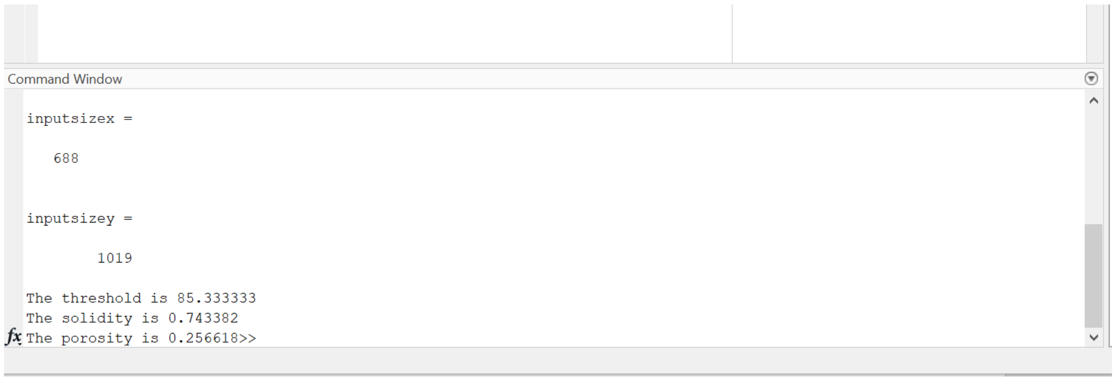


Figure 3)

The thresholding result of the picture



The porosity number will appear in the command window



```
inputsizex =

   688


inputsizey =

        1019

The threshold is 85.333333
The solidity is 0.743382
The porosity is 0.256618>>
```

1.4 FAQ

Do not to forget to cut the information bar before image processing

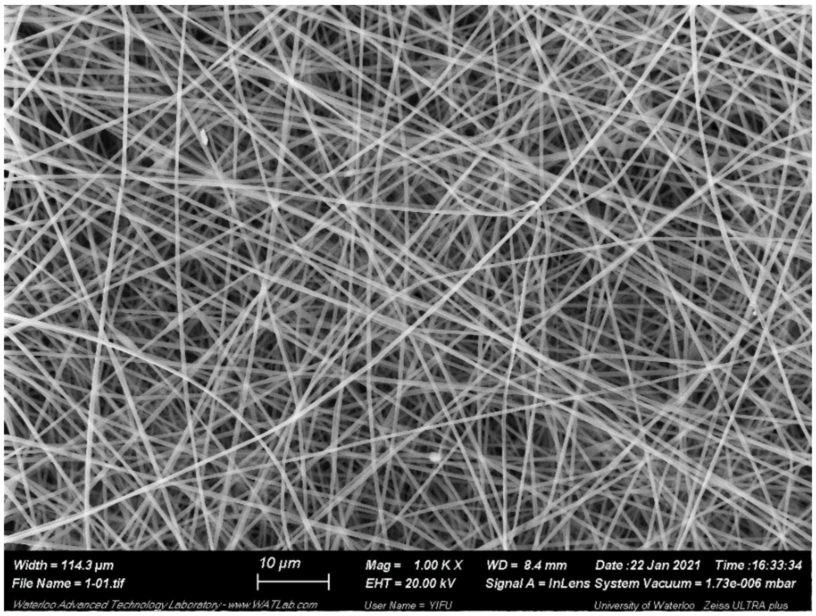


Image with information bar, cannot be used for this image processing

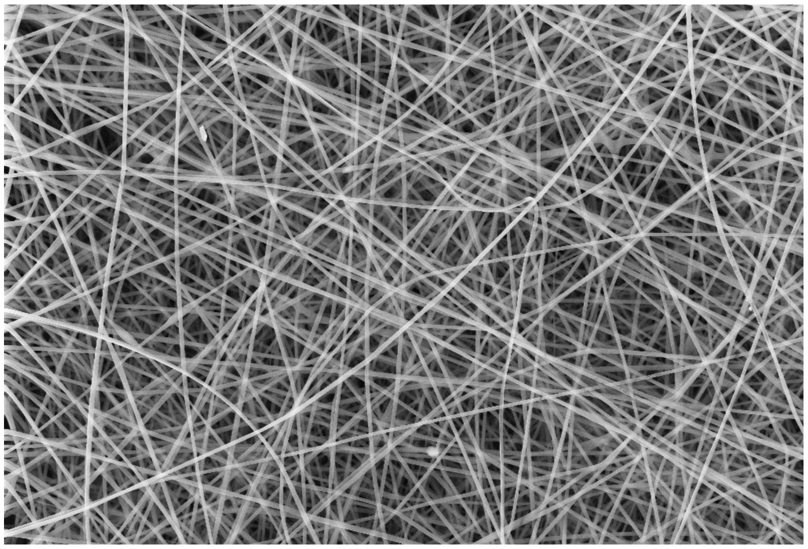
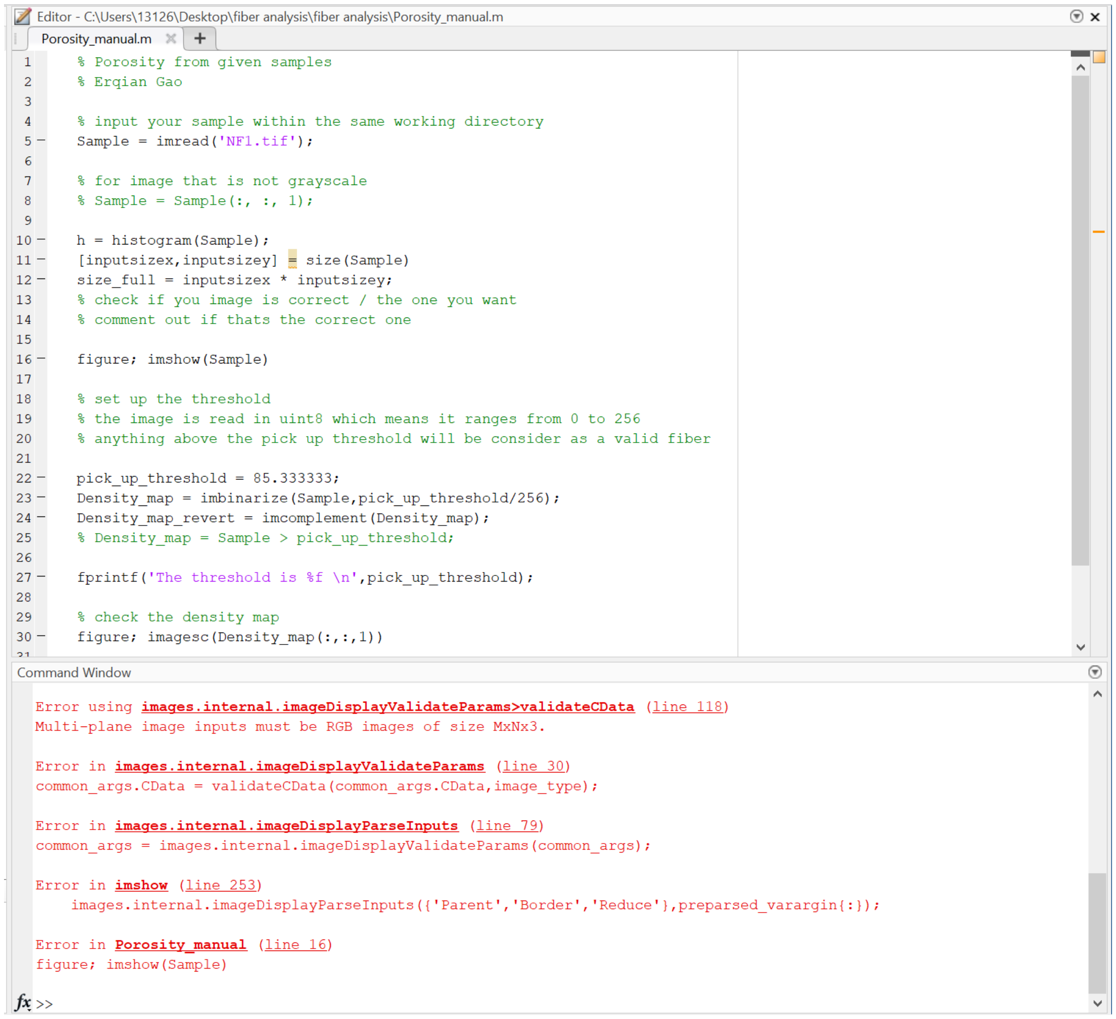


Image without information bar, can be used for this image processing

Sometimes the picture will be regarded as a color image instead of a grayscale image and result in error



Editor - C:\Users\13126\Desktop\fiber analysis\fiber analysis\Porosity_manual.m

Porosity_manual.m

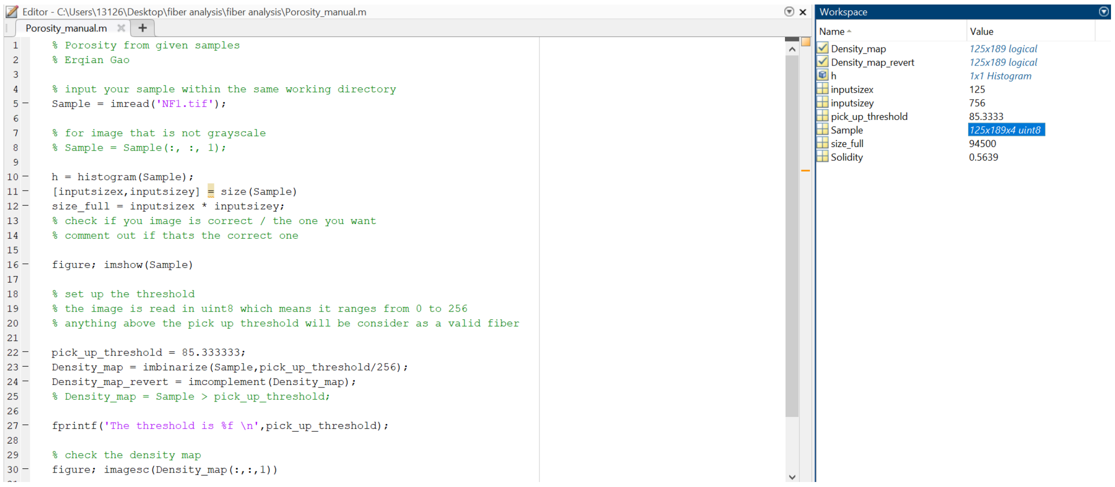
```
1    % Porosity from given samples
2    % Erqian Gao
3
4    % input your sample within the same working directory
5    Sample = imread('NF1.tif');
6
7    % for image that is not grayscale
8    % Sample = Sample(:, :, 1);
9
10   h = histogram(Sample);
11   [inputsizex,inputsizey] = size(Sample)
12   size_full = inputsizex * inputsizey;
13   % check if you image is correct / the one you want
14   % comment out if thats the correct one
15
16   figure; imshow(Sample)
17
18   % set up the threshold
19   % the image is read in uint8 which means it ranges from 0 to 256
20   % anything above the pick up threshold will be consider as a valid fiber
21
22   pick_up_threshold = 85.333333;
23   Density_map = imbinarize(Sample,pick_up_threshold/256);
24   Density_map_revert = imcomplement(Density_map);
25   % Density_map = Sample > pick_up_threshold;
26
27   fprintf('The threshold is %f \n',pick_up_threshold);
28
29   % check the density map
30   figure; imagesc(Density_map(:,:,1))
```

Command Window

```
Error using images.internal.imageDisplayValidateParams>validateCData (line 118)
Multi-plane image inputs must be RGB images of size MxNx3.

Error in images.internal.imageDisplayValidateParams (line 30)
common_args.CData = validateCData(common_args.CData,image_type);

Error in images.internal.imageDisplayParseInputs (line 79)
common_args = images.internal.imageDisplayValidateParams(common_args);

Error in imshow (line 253)
    images.internal.imageDisplayParseInputs({'Parent','Border','Reduce'},preparsed_varargin{:});

Error in Porosity_manual (line 16)
figure; imshow(Sample)

fx >>
```

When problems arise, the size of the 'Sample' file in the Workspace will be m*n*4

The last number 4 means R, G, B and transparency. It means the image is regarded as a color image.



Editor - C:\Users\13126\Desktop\fiber analysis\fiber analysis\Porosity_manual.m

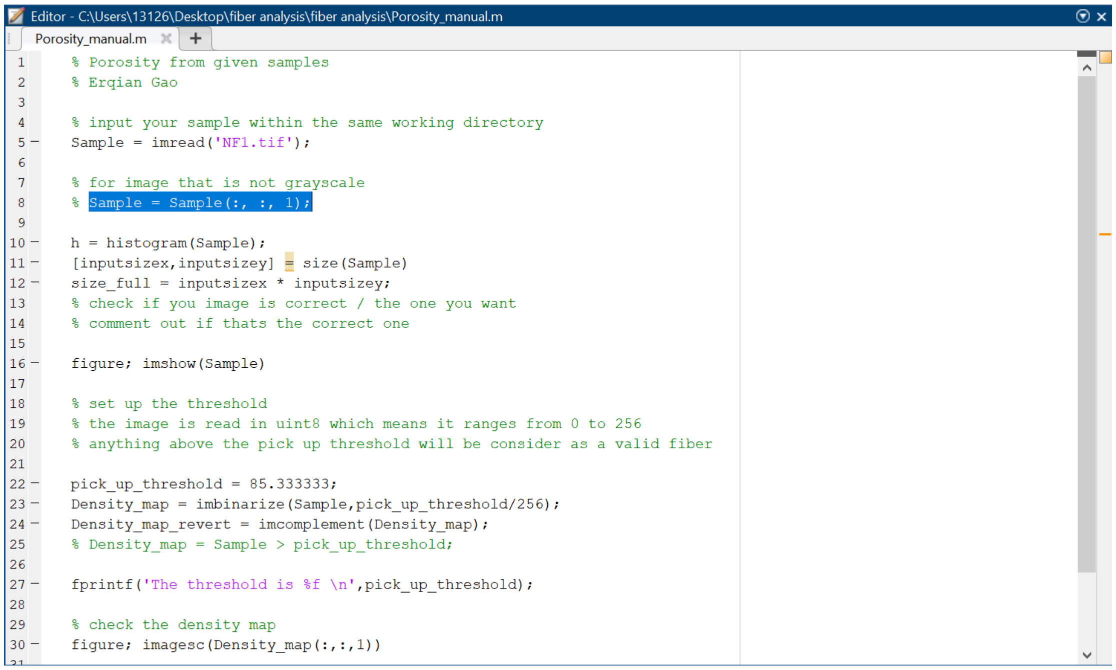
Porosity_manual.m

```
1    % Porosity from given samples
2    % Erqian Gao
3
4    % input your sample within the same working directory
5    Sample = imread('NF1.tif');
6
7    % for image that is not grayscale
8    % Sample = Sample(:, :, 1);
9
10   h = histogram(Sample);
11   [inputsizex,inputsizey] = size(Sample)
12   size_full = inputsizex * inputsizey;
13   % check if you image is correct / the one you want
14   % comment out if thats the correct one
15
16   figure; imshow(Sample)
17
18   % set up the threshold
19   % the image is read in uint8 which means it ranges from 0 to 256
20   % anything above the pick up threshold will be consider as a valid fiber
21
22   pick_up_threshold = 85.333333;
23   Density_map = imbinarize(Sample,pick_up_threshold/256);
24   Density_map_revert = imcomplement(Density_map);
25   % Density_map = Sample > pick_up_threshold;
26
27   fprintf('The threshold is %f \n',pick_up_threshold);
28
29   % check the density map
30   figure; imagesc(Density_map(:,:,1))
```

Workspace

Name	Value
Density_map	125x189 logical
Density_map_revert	125x189 logical
h	1x1 Histogram
inputsizex	125
inputsizey	756
pick_up_threshold	85.3333
Sample	125x189x4 uint8
size_full	94500
Solidity	0.5639

When that happens, enable the highlighted line and the problem will be solved.



Editor - C:\Users\13126\Desktop\fiber analysis\fiber analysis\Porosity_manual.m

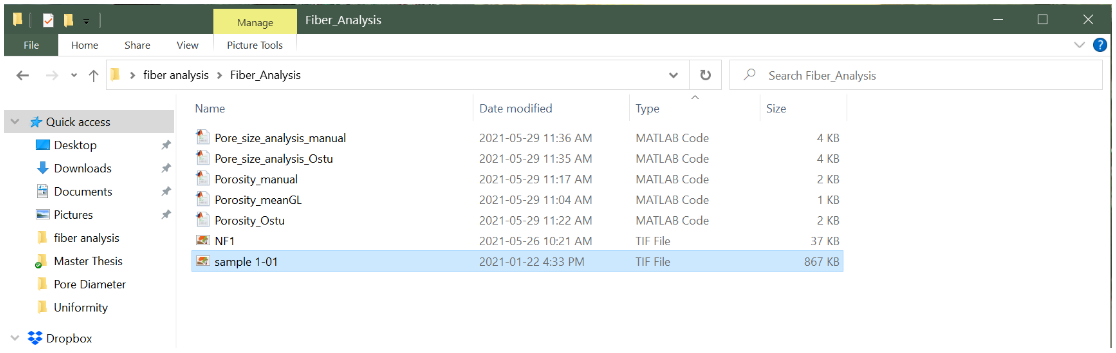
Porosity_manual.m

```matlab
% Porosity from given samples
% Erqian Gao

% input your sample within the same working directory
Sample = imread('NF1.tif');

% for image that is not grayscale
% Sample = Sample(:, :, 1);

h = histogram(Sample);
[inputsizex,inputsizey] = size(Sample)
size_full = inputsizex * inputsizey;
% check if you image is correct / the one you want
% comment out if thats the correct one

figure; imshow(Sample)

% set up the threshold
% the image is read in uint8 which means it ranges from 0 to 256
% anything above the pick up threshold will be consider as a valid fiber

pick_up_threshold = 85.333333;
Density_map = imbinarize(Sample,pick_up_threshold/256);
Density_map_revert = imcomplement(Density_map);
% Density_map = Sample > pick_up_threshold;

fprintf('The threshold is %f \n',pick_up_threshold);

% check the density map
figure; imagesc(Density_map(:,:,1))
```

2. Inter-fiber Pore Diameter Calculation

2.1 Inter-fiber Pore size analysis manual

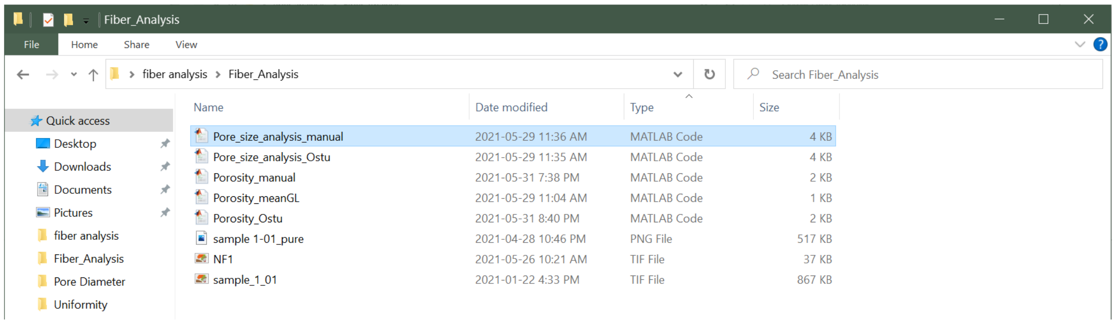
Put the SEM image in the same folder with the script



Fiber_Analysis — fiber analysis > Fiber_Analysis

Name	Date modified	Type	Size
Pore_size_analysis_manual	2021-05-29 11:36 AM	MATLAB Code	4 KB
Pore_size_analysis_Ostu	2021-05-29 11:35 AM	MATLAB Code	4 KB
Porosity_manual	2021-05-29 11:17 AM	MATLAB Code	2 KB
Porosity_meanGL	2021-05-29 11:04 AM	MATLAB Code	1 KB
Porosity_Ostu	2021-05-29 11:22 AM	MATLAB Code	2 KB
NF1	2021-05-26 10:21 AM	TIF File	37 KB
sample 1-01	2021-01-22 4:33 PM	TIF File	867 KB

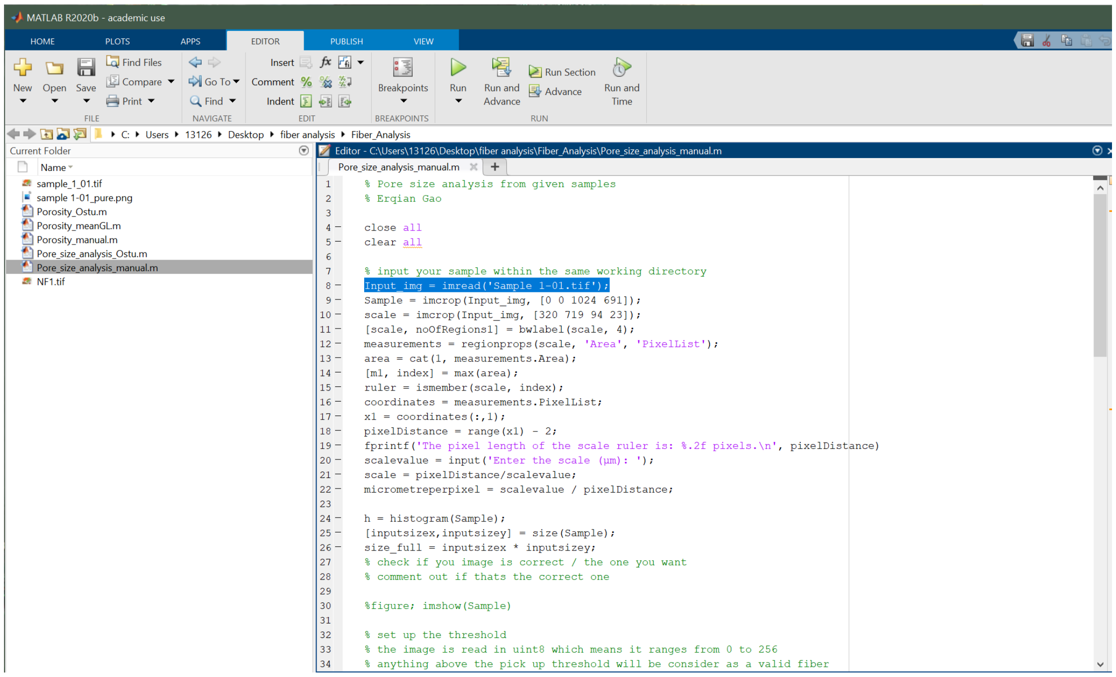
Open Pore_size_analysis_manual file in the folder



Fiber_Analysis — fiber analysis > Fiber_Analysis

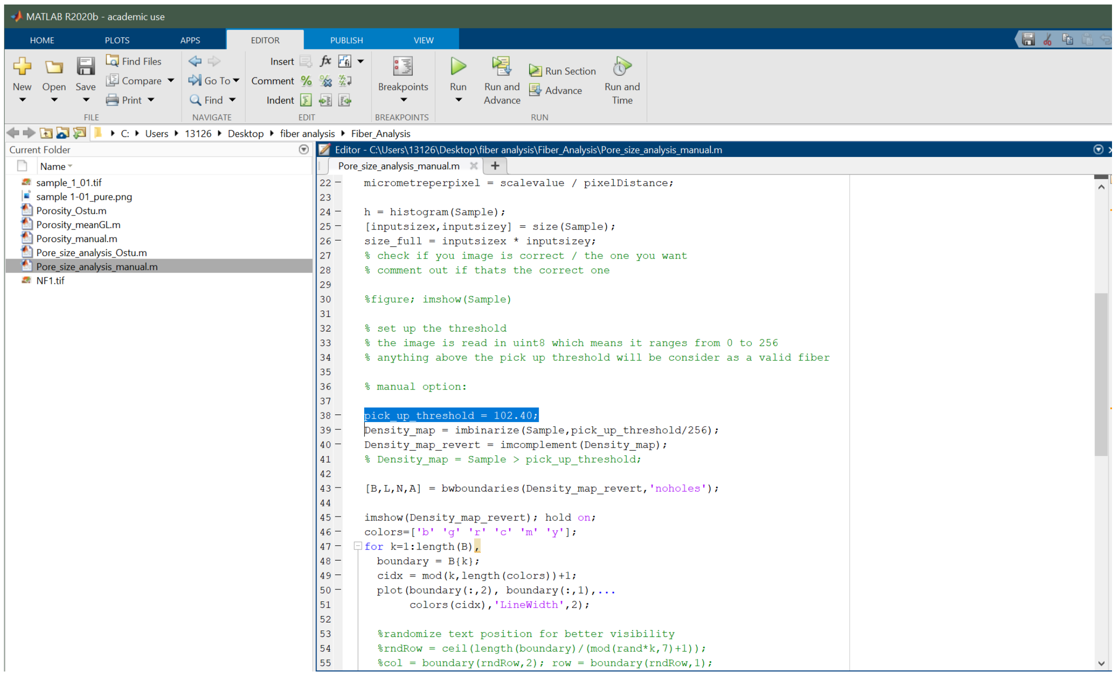
Name	Date modified	Type	Size
Pore_size_analysis_manual	2021-05-29 11:36 AM	MATLAB Code	4 KB
Pore_size_analysis_Ostu	2021-05-29 11:35 AM	MATLAB Code	4 KB
Porosity_manual	2021-05-31 7:38 PM	MATLAB Code	2 KB
Porosity_meanGL	2021-05-29 11:04 AM	MATLAB Code	1 KB
Porosity_Ostu	2021-05-31 8:40 PM	MATLAB Code	2 KB
sample 1-01_pure	2021-04-28 10:46 PM	PNG File	517 KB
NF1	2021-05-26 10:21 AM	TIF File	37 KB
sample_1_01	2021-01-22 4:33 PM	TIF File	867 KB

Change the highlighted line to the name of the SEM image file (In this example the file name is sample_1_01)



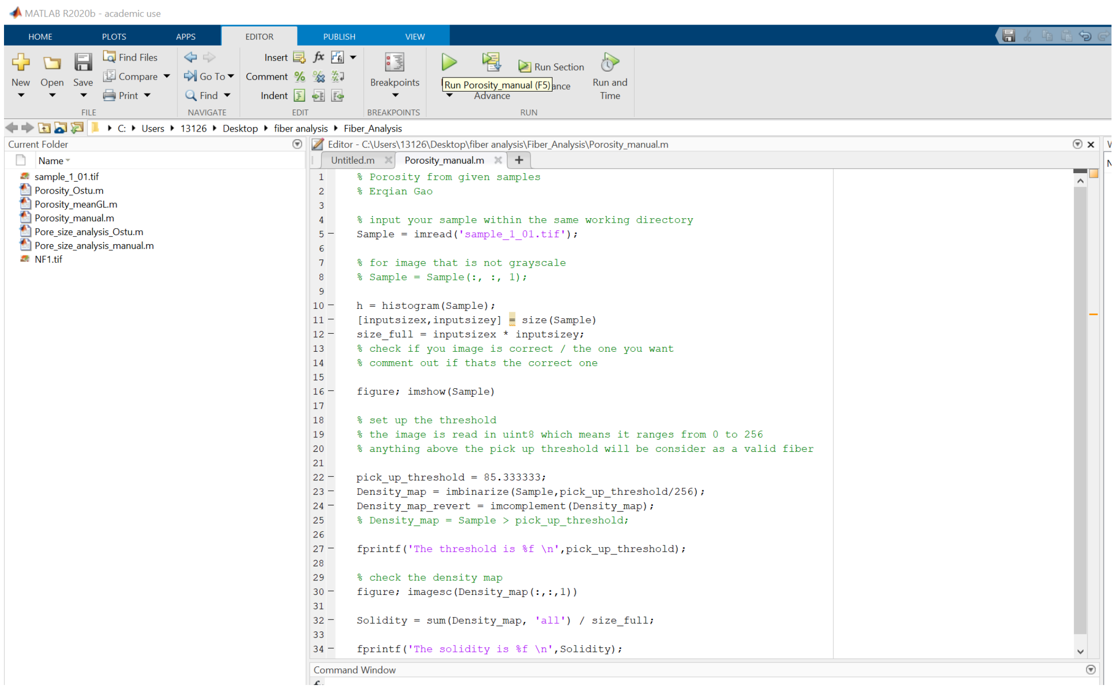
```
% Pore size analysis from given samples
% Erqian Gao

close all
clear all

% input your sample within the same working directory
Input_img = imread('Sample 1-01.tif');
Sample = imcrop(Input_img, [0 0 1024 691]);
scale = imcrop(Input_img, [320 719 94 23]);
[scale, noOfRegions1] = bwlabel(scale, 4);
measurements = regionprops(scale, 'Area', 'PixelList');
area = cat(1, measurements.Area);
[m1, index] = max(area);
ruler = ismember(scale, index);
coordinates = measurements.PixelList;
x1 = coordinates(:,1);
pixelDistance = range(x1) - 2;
fprintf('The pixel length of the scale ruler is: %.2f pixels.\n', pixelDistance)
scalevalue = input('Enter the scale (μm): ');
scale = pixelDistance/scalevalue;
micrometreperpixel = scalevalue / pixelDistance;

h = histogram(Sample);
[inputsizex,inputsizey] = size(Sample);
size_full = inputsizex * inputsizey;
% check if you image is correct / the one you want
% comment out if thats the correct one

%figure; imshow(Sample)

% set up the threshold
% the image is read in uint8 which means it ranges from 0 to 256
% anything above the pick up threshold will be consider as a valid fiber
```

Set up the threshold manually, the threshold value should be between 0~255



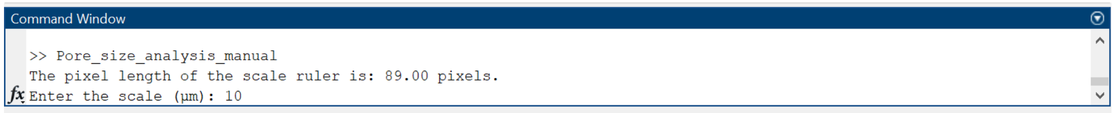
```
micrometreperpixel = scalevalue / pixelDistance;

h = histogram(Sample);
[inputsizex,inputsizey] = size(Sample);
size_full = inputsizex * inputsizey;
% check if you image is correct / the one you want
% comment out if thats the correct one

%figure; imshow(Sample)

% set up the threshold
% the image is read in uint8 which means it ranges from 0 to 256
% anything above the pick up threshold will be consider as a valid fiber

% manual option:

pick_up_threshold = 102.40;
Density_map = imbinarize(Sample,pick_up_threshold/256);
Density_map_revert = imcomplement(Density_map);
% Density_map = Sample > pick_up_threshold;

[B,L,N,A] = bwboundaries(Density_map_revert,'noholes');

imshow(Density_map_revert); hold on;
colors=['b' 'g' 'r' 'c' 'm' 'y'];
for k=1:length(B),
  boundary = B{k};
  cidx = mod(k,length(colors))+1;
  plot(boundary(:,2), boundary(:,1),...
       colors(cidx),'LineWidth',2);

  %randomize text position for better visibility
  %rndRow = ceil(length(boundary)/(mod(rand*k,7)+1));
  %col = boundary(rndRow,2); row = boundary(rndRow,1);
```

Run the file (hit the run button or press F5)



```matlab
% Porosity from given samples
% Erqian Gao

% input your sample within the same working directory
Sample = imread('sample_1_01.tif');

% for image that is not grayscale
% Sample = Sample(:, :, 1);

h = histogram(Sample);
[inputsizex,inputsizey] = size(Sample)
size_full = inputsizex * inputsizey;
% check if you image is correct / the one you want
% comment out if thats the correct one

figure; imshow(Sample)

% set up the threshold
% the image is read in uint8 which means it ranges from 0 to 256
% anything above the pick up threshold will be consider as a valid fiber

pick_up_threshold = 85.333333;
Density_map = imbinarize(Sample,pick_up_threshold/256);
Density_map_revert = imcomplement(Density_map);
% Density_map = Sample > pick_up_threshold;

fprintf('The threshold is %f \n',pick_up_threshold);

% check the density map
figure; imagesc(Density_map(:,:,1))

Solidity = sum(Density_map, 'all') / size_full;

fprintf('The solidity is %f \n',Solidity);
```

Enter the scale ruler in the command box (unit can be changed to nm if necessary) and press enter



```
>> Pore_size_analysis_manual
The pixel length of the scale ruler is: 89.00 pixels.
fx Enter the scale (μm): 10
```

Wait couple of minutes and get the results

Three figures will appear

Figure 1)

The figure of pore division

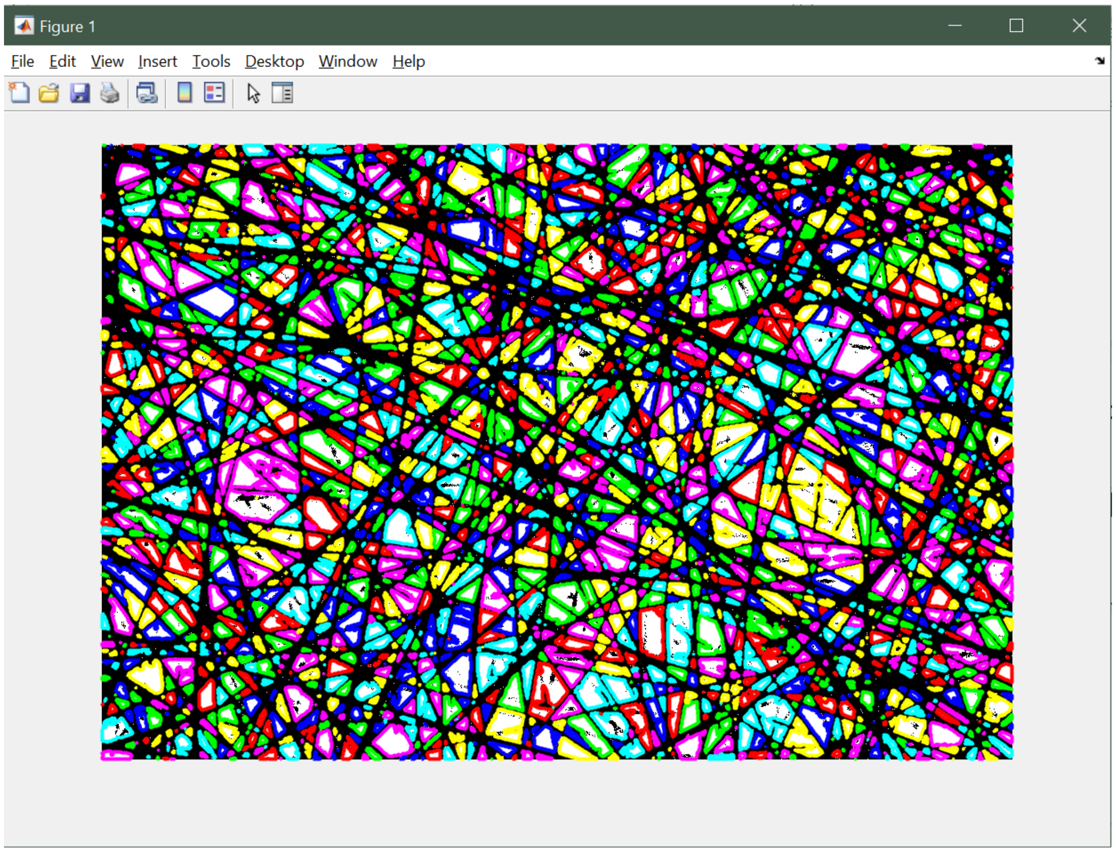


Figure 2)

The histogram of Pore diameter

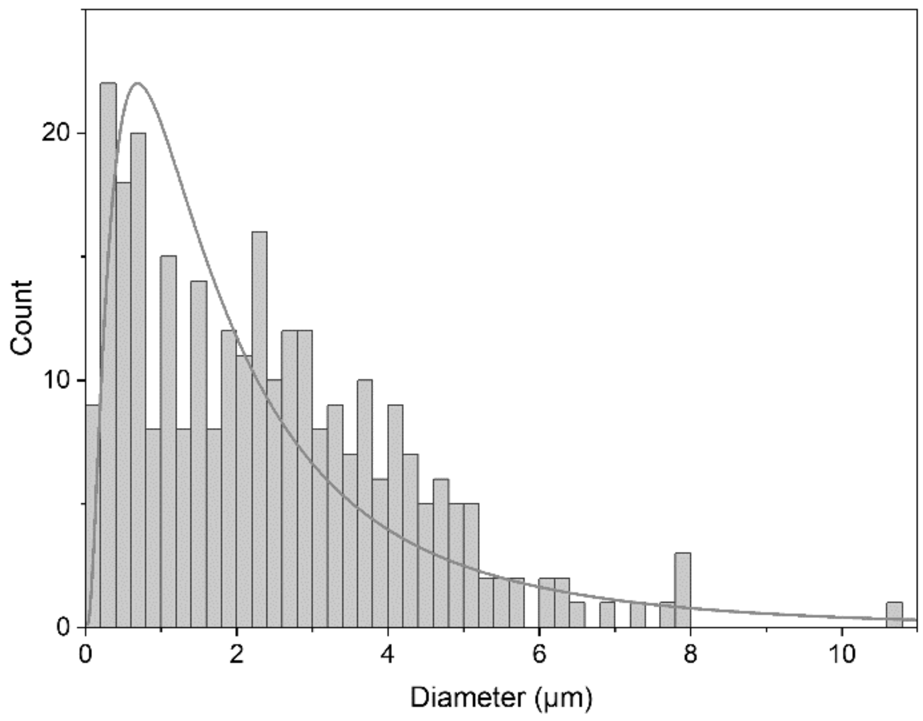
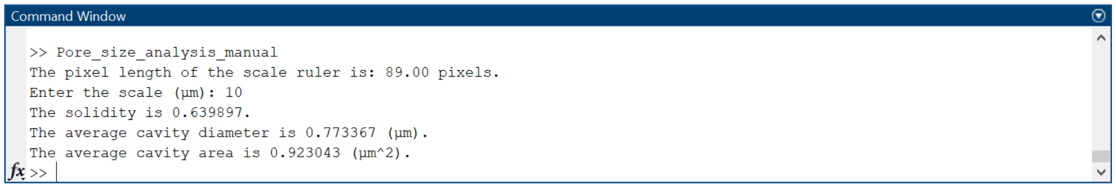


Figure 3)

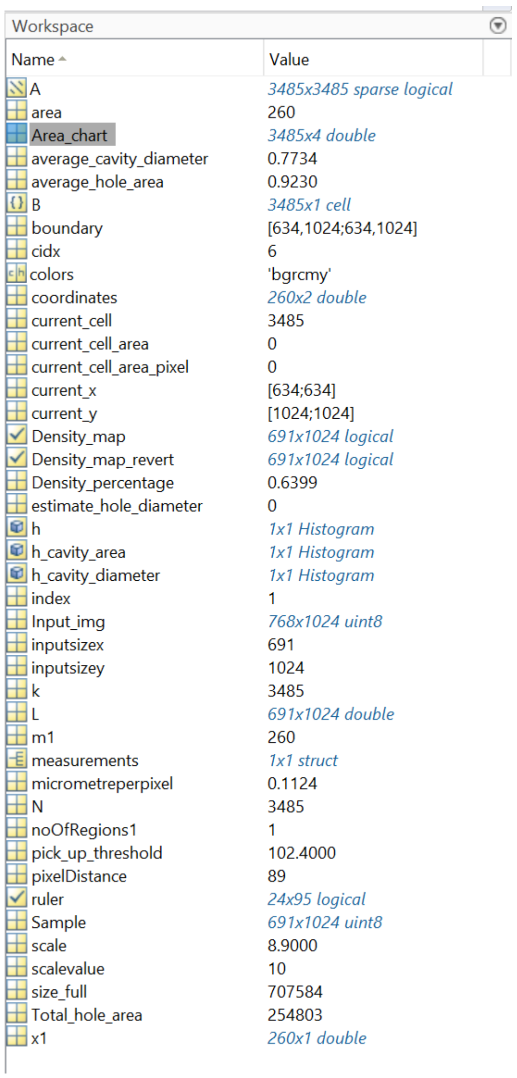
The histogram of Pore area

The result value will appear in the command window



```
>> Pore_size_analysis_manual
The pixel length of the scale ruler is: 89.00 pixels.
Enter the scale (μm): 10
The solidity is 0.639897.
The average cavity diameter is 0.773367 (μm).
The average cavity area is 0.923043 (μm^2).
fx >> 
```

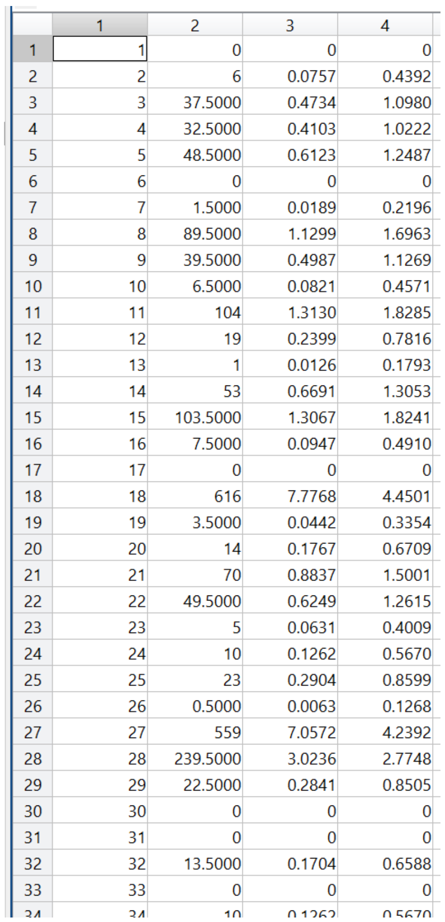
The raw data located in the Area chart



Workspace

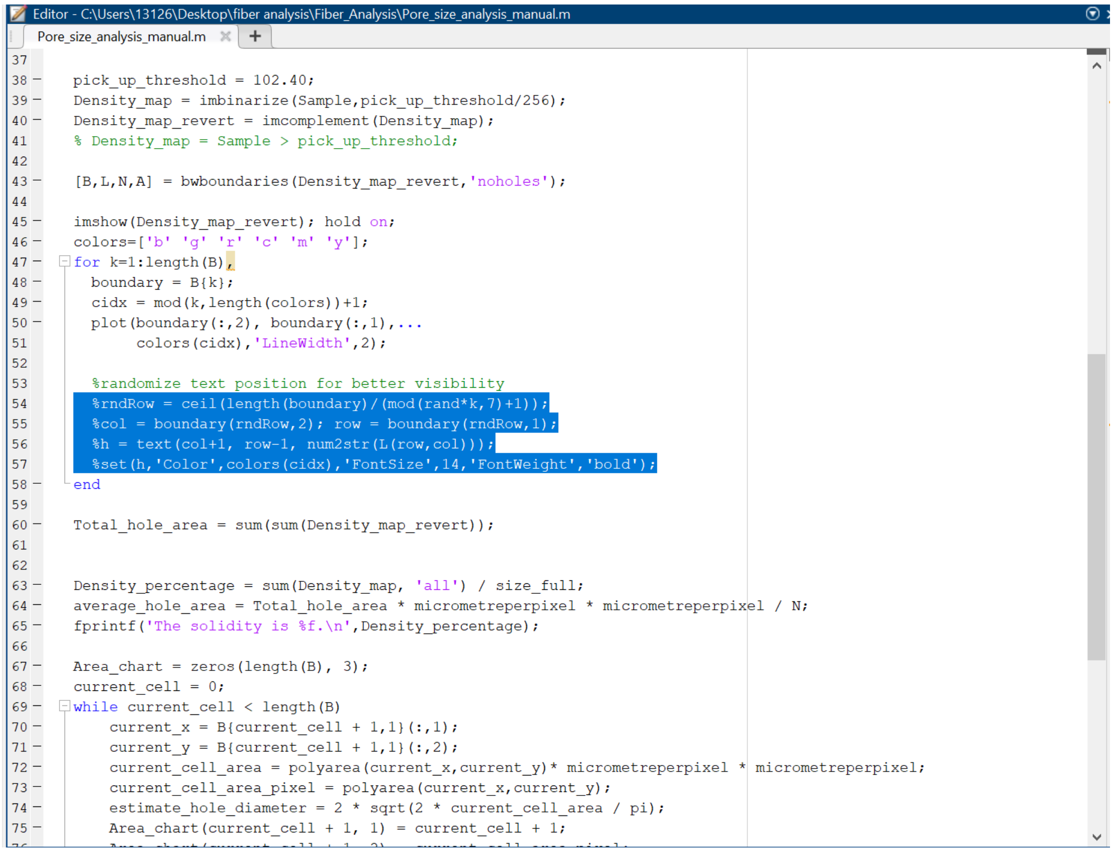
Name	Value
A	*3485x3485 sparse logical*
area	260
Area_chart	*3485x4 double*
average_cavity_diameter	0.7734
average_hole_area	0.9230
B	*3485x1 cell*
boundary	[634,1024;634,1024]
cidx	6
colors	'bgrcmy'
coordinates	*260x2 double*
current_cell	3485
current_cell_area	0
current_cell_area_pixel	0
current_x	[634;634]
current_y	[1024;1024]
Density_map	*691x1024 logical*
Density_map_revert	*691x1024 logical*
Density_percentage	0.6399
estimate_hole_diameter	0
h	*1x1 Histogram*
h_cavity_area	*1x1 Histogram*
h_cavity_diameter	*1x1 Histogram*
index	1
Input_img	*768x1024 uint8*
inputsizex	691
inputsizey	1024
k	3485
L	*691x1024 double*
m1	260
measurements	*1x1 struct*
micrometreperpixel	0.1124
N	3485
noOfRegions1	1
pick_up_threshold	102.4000
pixelDistance	89
ruler	*24x95 logical*
Sample	*691x1024 uint8*
scale	8.9000
scalevalue	10
size_full	707584
Total_hole_area	254803
x1	*260x1 double*

Double click area chart, an excel with four columns will appears. The first column is the pore number, the second column is the pixel number, the third column is the pore area, the forth column is the pore diameter. This data file can be exported for post-processing.



	1	2	3	4
1	1	0	0	0
2	2	6	0.0757	0.4392
3	3	37.5000	0.4734	1.0980
4	4	32.5000	0.4103	1.0222
5	5	48.5000	0.6123	1.2487
6	6	0	0	0
7	7	1.5000	0.0189	0.2196
8	8	89.5000	1.1299	1.6963
9	9	39.5000	0.4987	1.1269
10	10	6.5000	0.0821	0.4571
11	11	104	1.3130	1.8285
12	12	19	0.2399	0.7816
13	13	1	0.0126	0.1793
14	14	53	0.6691	1.3053
15	15	103.5000	1.3067	1.8241
16	16	7.5000	0.0947	0.4910
17	17	0	0	0
18	18	616	7.7768	4.4501
19	19	3.5000	0.0442	0.3354
20	20	14	0.1767	0.6709
21	21	70	0.8837	1.5001
22	22	49.5000	0.6249	1.2615
23	23	5	0.0631	0.4009
24	24	10	0.1262	0.5670
25	25	23	0.2904	0.8599
26	26	0.5000	0.0063	0.1268
27	27	559	7.0572	4.2392
28	28	239.5000	3.0236	2.7748
29	29	22.5000	0.2841	0.8505
30	30	0	0	0
31	31	0	0	0
32	32	13.5000	0.1704	0.6588
33	33	0	0	0
34	34	10	0.1262	0.5670

Enable this part will make figure 1 pores has corresponding number on it

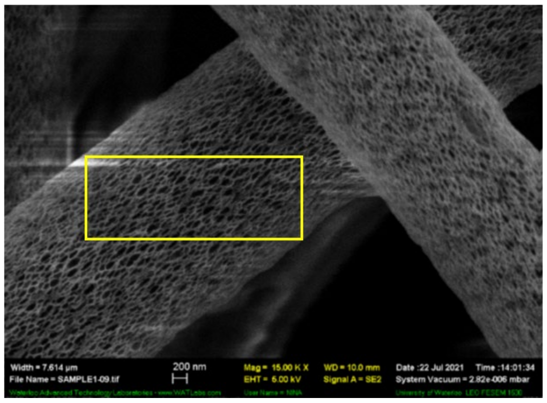

Editor - C:\Users\13126\Desktop\fiber analysis\Fiber_Analysis\Pore_size_analysis_manual.m
Pore_size_analysis_manual.m

```

pick_up_threshold = 102.40;
Density_map = imbinarize(Sample,pick_up_threshold/256);
Density_map_revert = imcomplement(Density_map);
% Density_map = Sample > pick_up_threshold;

[B,L,N,A] = bwboundaries(Density_map_revert,'noholes');

imshow(Density_map_revert); hold on;
colors=['b' 'g' 'r' 'c' 'm' 'y'];
for k=1:length(B),
  boundary = B{k};
  cidx = mod(k,length(colors))+1;
  plot(boundary(:,2), boundary(:,1),...
       colors(cidx),'LineWidth',2);

  %randomize text position for better visibility
  %rndRow = ceil(length(boundary)/(mod(rand*k,7)+1));
  %col = boundary(rndRow,2); row = boundary(rndRow,1);
  %h = text(col+1, row-1, num2str(L(row,col)));
  %set(h,'Color',colors(cidx),'FontSize',14,'FontWeight','bold');
end

Total_hole_area = sum(sum(Density_map_revert));


Density_percentage = sum(Density_map, 'all') / size_full;
average_hole_area = Total_hole_area * micrometreperpixel * micrometreperpixel / N;
fprintf('The solidity is %f.\n',Density_percentage);

Area_chart = zeros(length(B), 3);
current_cell = 0;
while current_cell < length(B)
    current_x = B{current_cell + 1,1}(:,1);
    current_y = B{current_cell + 1,1}(:,2);
    current_cell_area = polyarea(current_x,current_y)* micrometreperpixel * micrometreperpixel;
    current_cell_area_pixel = polyarea(current_x,current_y);
    estimate_hole_diameter = 2 * sqrt(2 * current_cell_area / pi);
    Area_chart(current_cell + 1, 1) = current_cell + 1;
```

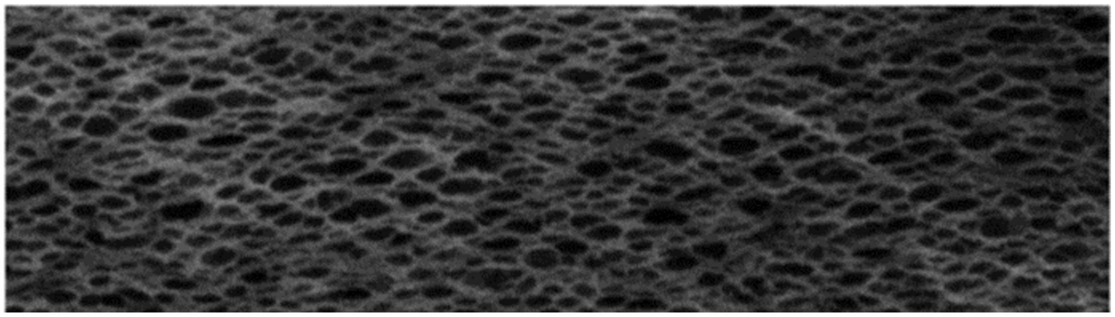
This procedure applies on all inter-fiber pores determination script

3. Surface intra-fiber pore diameter determination

Resize the image, only keep the porous fiber part.

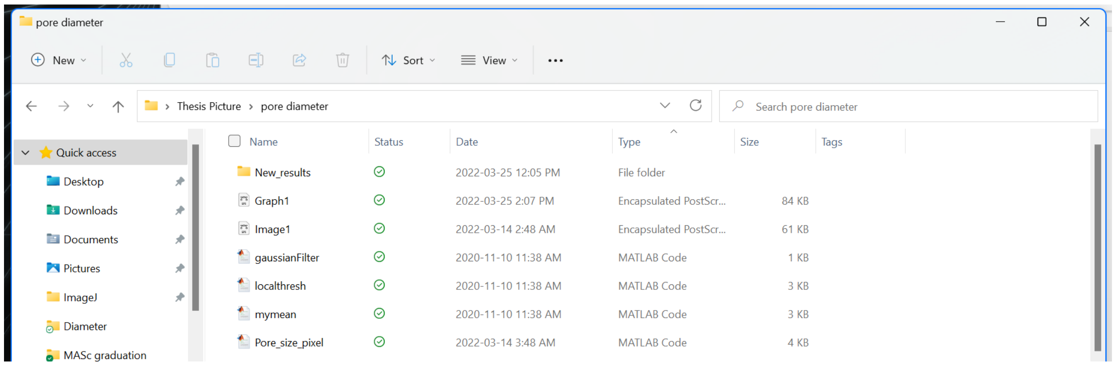


a) Original SEM image



b) Resized image

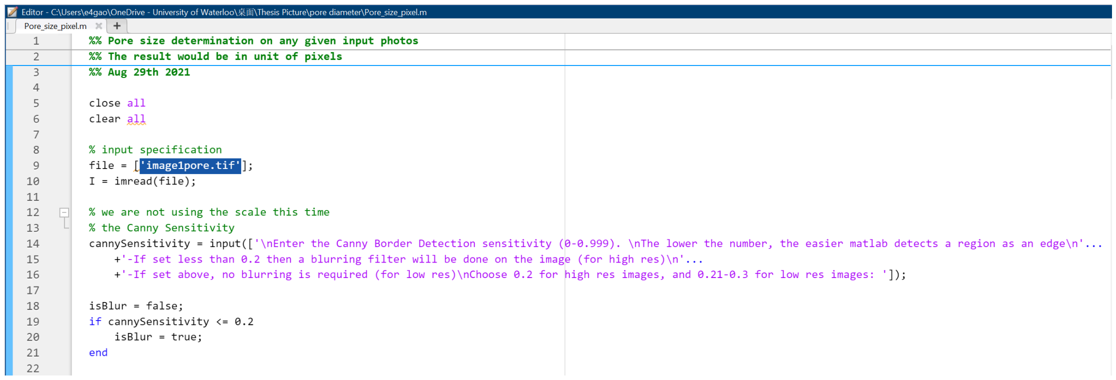
Put the SEM image in the same folder with the script



Name	Status	Date	Type	Size	Tags
New_results	✓	2022-03-25 12:05 PM	File folder		
Graph1	✓	2022-03-25 2:07 PM	Encapsulated PostScr...	84 KB	
Image1	✓	2022-03-14 2:48 AM	Encapsulated PostScr...	61 KB	
gaussianFilter	✓	2020-11-10 11:38 AM	MATLAB Code	1 KB	
localthresh	✓	2020-11-10 11:38 AM	MATLAB Code	3 KB	
mymean	✓	2020-11-10 11:38 AM	MATLAB Code	3 KB	
Pore_size_pixel	✓	2022-03-14 3:48 AM	MATLAB Code	4 KB	

Open Pore_size_pixel file in the folder

Change the highlighted line to the name of the SEM image file (In this example the file name is sample_1_01)

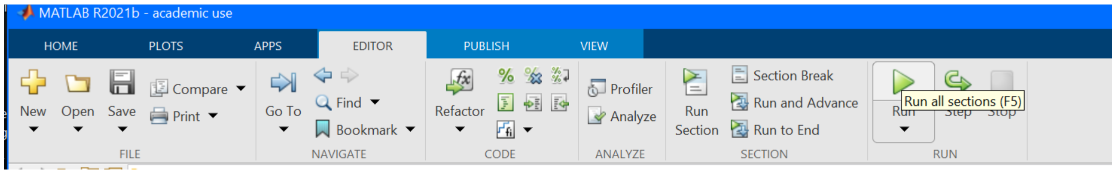


Editor - C:\Users\e4gao\OneDrive - University of Waterloo\桌面\Thesis Picture\pore diameter\Pore_size_pixel.m

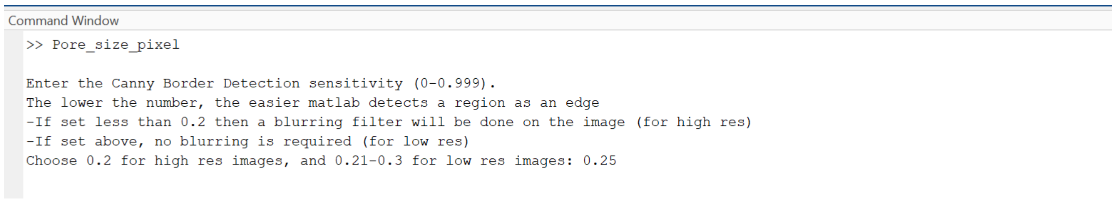
Pore_size_pixel.m

```
%% Pore size determination on any given input photos
%% The result would be in unit of pixels
%% Aug 29th 2021

close all
clear all

% input specification
file = ['image1pore.tif'];
I = imread(file);

% we are not using the scale this time
% the Canny Sensitivity
cannySensitivity = input(['\nEnter the Canny Border Detection sensitivity (0-0.999). \nThe lower the number, the easier matlab detects a region as an edge\n'...
    +'-If set less than 0.2 then a blurring filter will be done on the image (for high res)\n'...
    +'-If set above, no blurring is required (for low res)\nChoose 0.2 for high res images, and 0.21-0.3 for low res images: ']);

isBlur = false;
if cannySensitivity <= 0.2
    isBlur = true;
end

```

Run the file

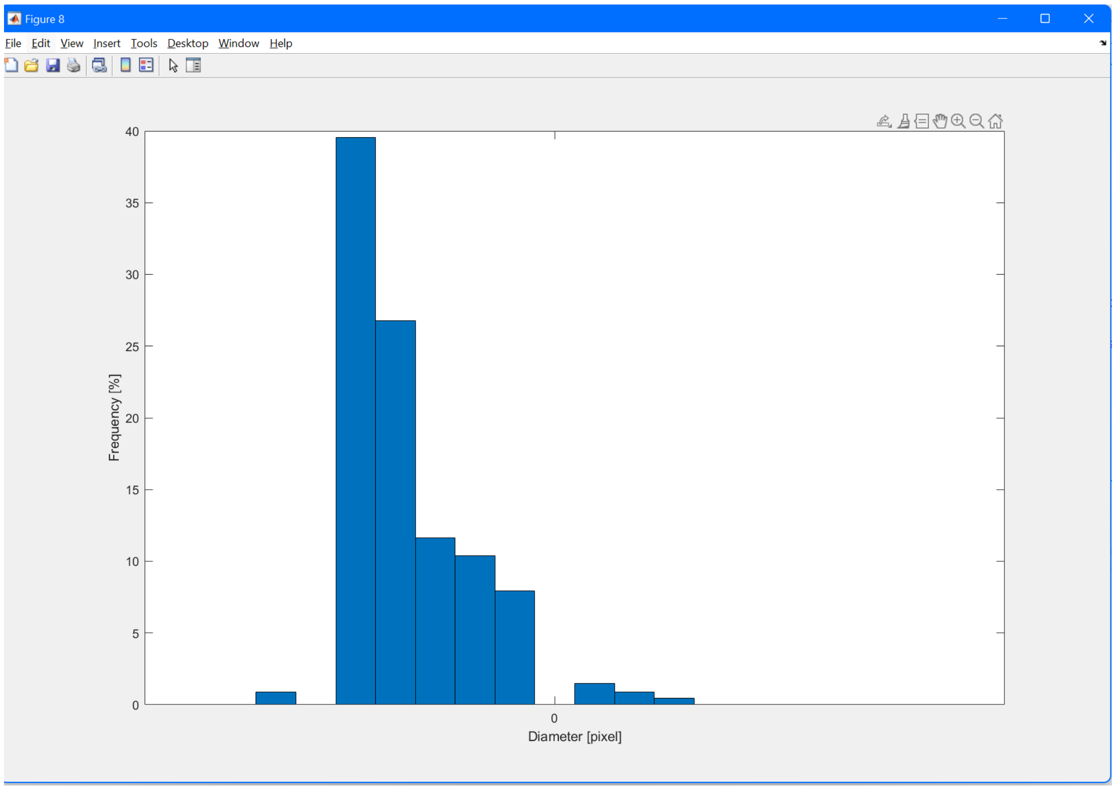


Enter the Canny Border Detection sensitivity. The value 0.25 is used in this example.

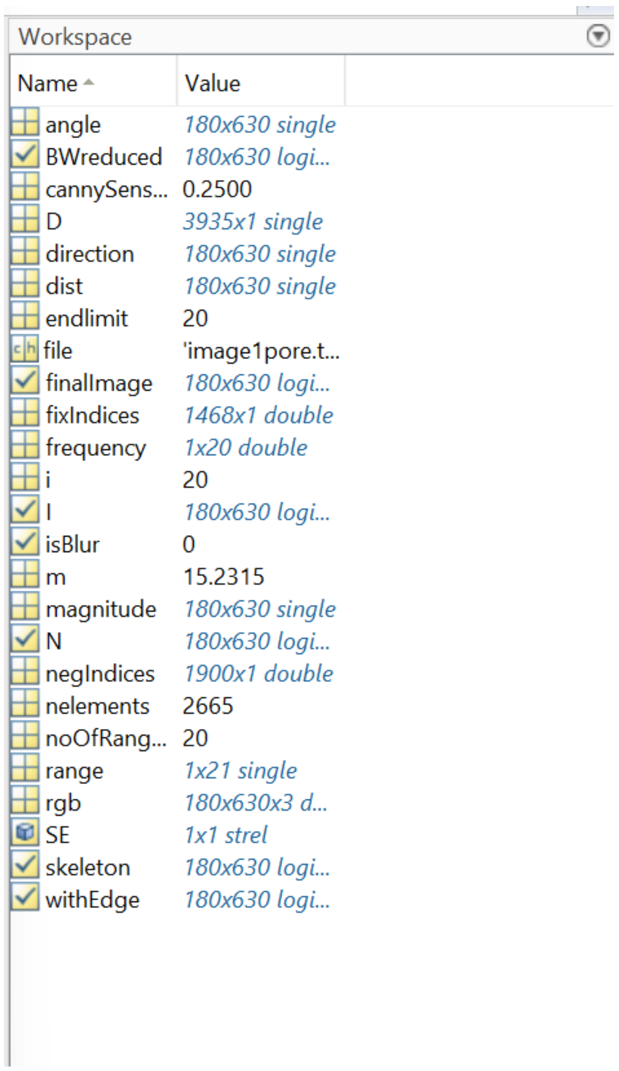


```
Command Window
>> Pore_size_pixel

Enter the Canny Border Detection sensitivity (0-0.999).
The lower the number, the easier matlab detects a region as an edge
-If set less than 0.2 then a blurring filter will be done on the image (for high res)
-If set above, no blurring is required (for low res)
Choose 0.2 for high res images, and 0.21-0.3 for low res images: 0.25
```

Get the result



The Diameter data can be found in a file named ‘D’ in the workspace.

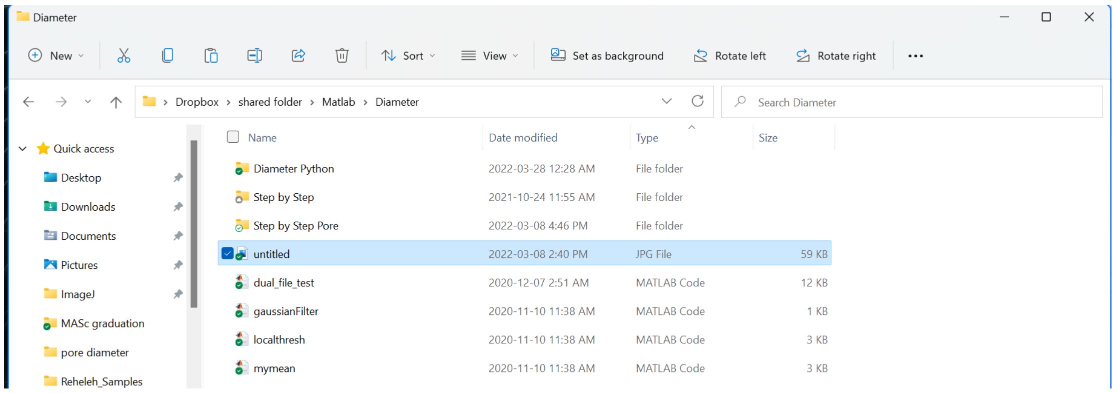


Workspace

Name	Value
angle	180x630 single
BWreduced	180x630 logi...
cannySens...	0.2500
D	3935x1 single
direction	180x630 single
dist	180x630 single
endlimit	20
file	'image1pore.t...
finalImage	180x630 logi...
fixIndices	1468x1 double
frequency	1x20 double
i	20
I	180x630 logi...
isBlur	0
m	15.2315
magnitude	180x630 single
N	180x630 logi...
negIndices	1900x1 double
nelements	2665
noOfRang...	20
range	1x21 single
rgb	180x630x3 d...
SE	1x1 strel
skeleton	180x630 logi...
withEdge	180x630 logi...

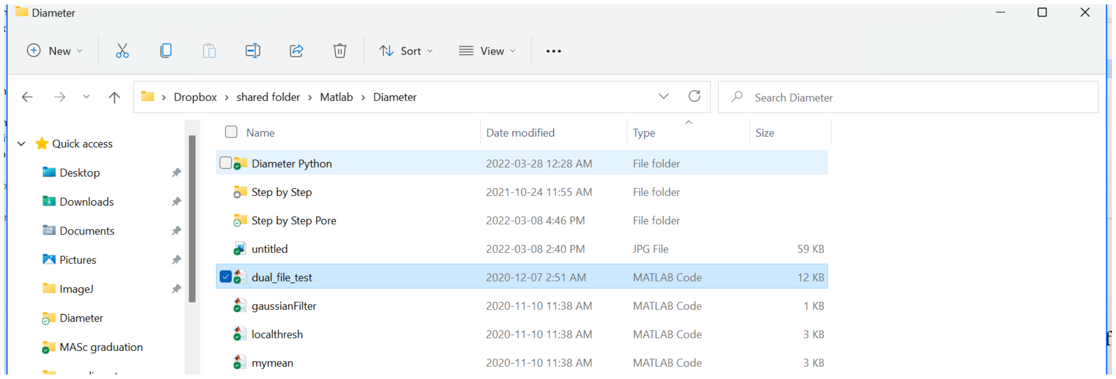
4. Diameter determination tool

Put the SEM image in the same folder with the script



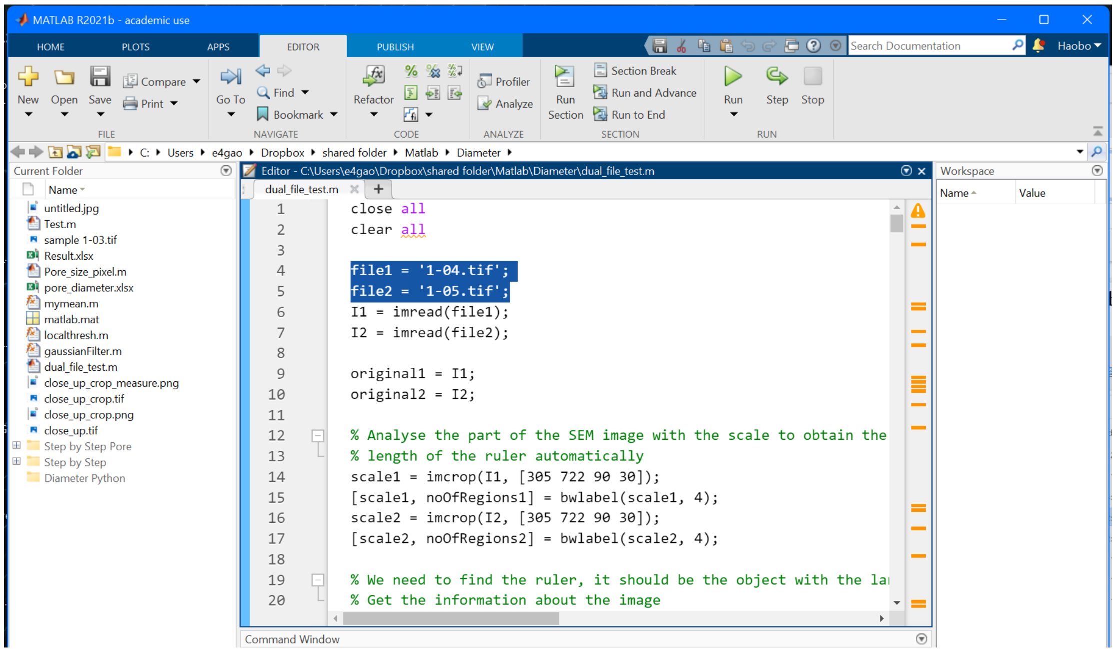
Name	Date modified	Type	Size
Diameter Python	2022-03-28 12:28 AM	File folder	
Step by Step	2021-10-24 11:55 AM	File folder	
Step by Step Pore	2022-03-08 4:46 PM	File folder	
untitled	2022-03-08 2:40 PM	JPG File	59 KB
dual_file_test	2020-12-07 2:51 AM	MATLAB Code	12 KB
gaussianFilter	2020-11-10 11:38 AM	MATLAB Code	1 KB
localthresh	2020-11-10 11:38 AM	MATLAB Code	3 KB
mymean	2020-11-10 11:38 AM	MATLAB Code	3 KB

Open dual_file_test file in the SEM Analysis folder:



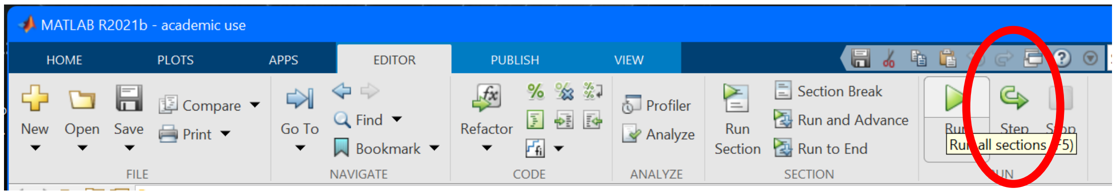
Name	Date modified	Type	Size
Diameter Python	2022-03-28 12:28 AM	File folder	
Step by Step	2021-10-24 11:55 AM	File folder	
Step by Step Pore	2022-03-08 4:46 PM	File folder	
untitled	2022-03-08 2:40 PM	JPG File	59 KB
dual_file_test	2020-12-07 2:51 AM	MATLAB Code	12 KB
gaussianFilter	2020-11-10 11:38 AM	MATLAB Code	1 KB
localthresh	2020-11-10 11:38 AM	MATLAB Code	3 KB
mymean	2020-11-10 11:38 AM	MATLAB Code	3 KB

Change the highlighted line to the name of the SEM image file (In this example the file name is ‘1-04.tif’, ‘1-05.tif’).

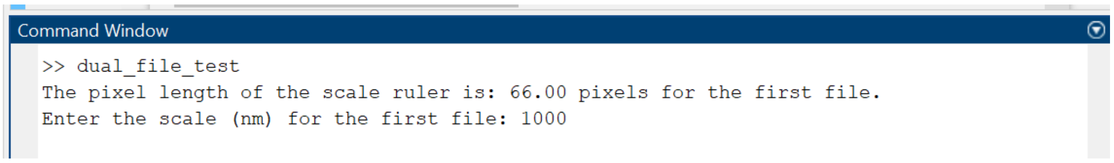


```
close all
clear all

file1 = '1-04.tif';
file2 = '1-05.tif';
I1 = imread(file1);
I2 = imread(file2);

original1 = I1;
original2 = I2;

% Analyse the part of the SEM image with the scale to obtain the
% length of the ruler automatically
scale1 = imcrop(I1, [305 722 90 30]);
[scale1, noOfRegions1] = bwlabel(scale1, 4);
scale2 = imcrop(I2, [305 722 90 30]);
[scale2, noOfRegions2] = bwlabel(scale2, 4);

% We need to find the ruler, it should be the object with the la
% Get the information about the image
```

Run the file:

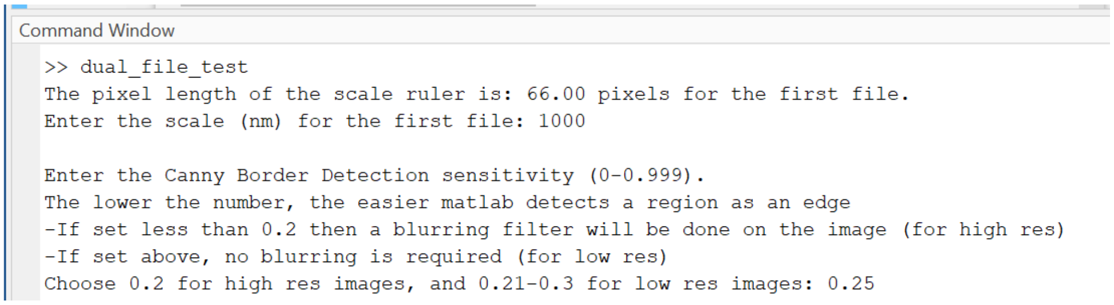


Enter the scale of the first images in the command window.



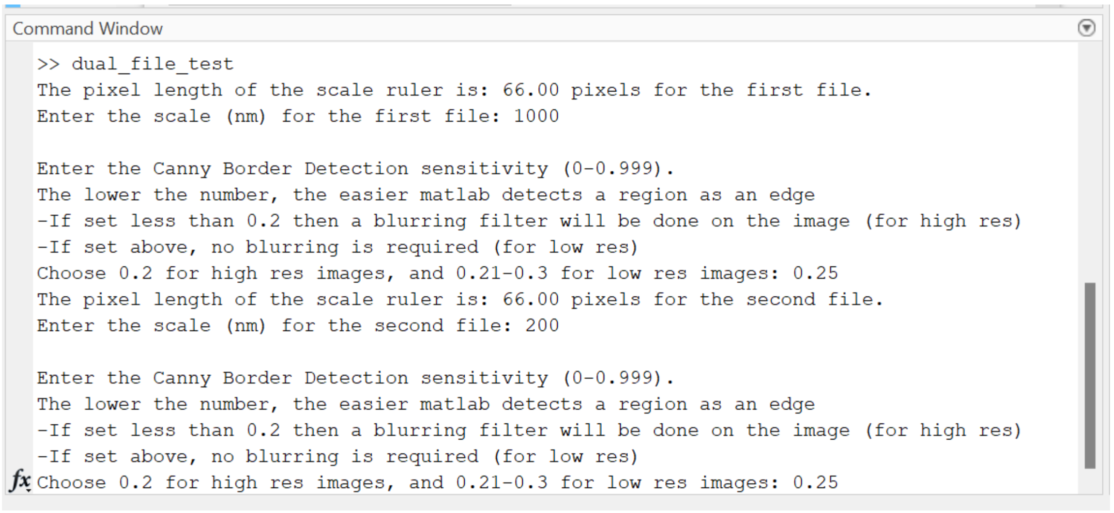
```
>> dual_file_test
The pixel length of the scale ruler is: 66.00 pixels for the first file.
Enter the scale (nm) for the first file: 1000
```

Enter the Canny Border Detection sensitivity (0-0.999) for the first file



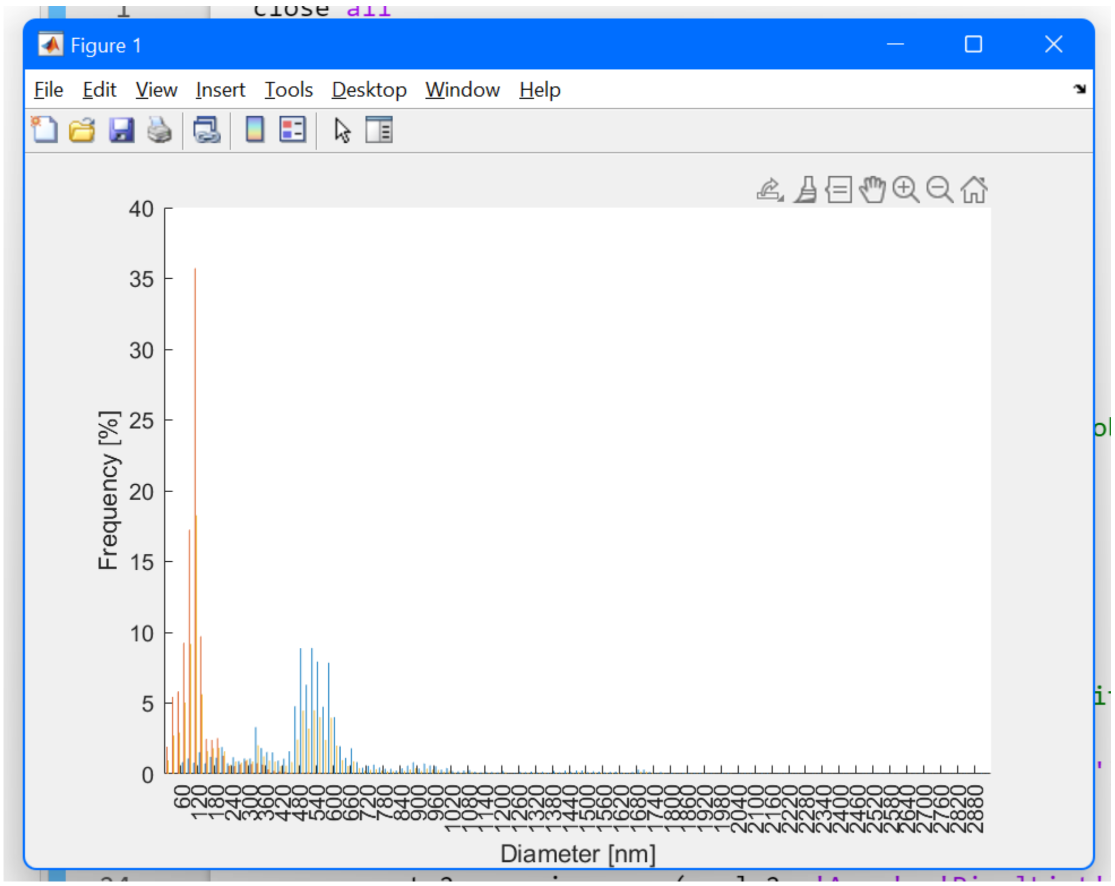
```
>> dual_file_test
The pixel length of the scale ruler is: 66.00 pixels for the first file.
Enter the scale (nm) for the first file: 1000

Enter the Canny Border Detection sensitivity (0-0.999).
The lower the number, the easier matlab detects a region as an edge
-If set less than 0.2 then a blurring filter will be done on the image (for high res)
-If set above, no blurring is required (for low res)
Choose 0.2 for high res images, and 0.21-0.3 for low res images: 0.25
```

Repeat the procedure for the second image

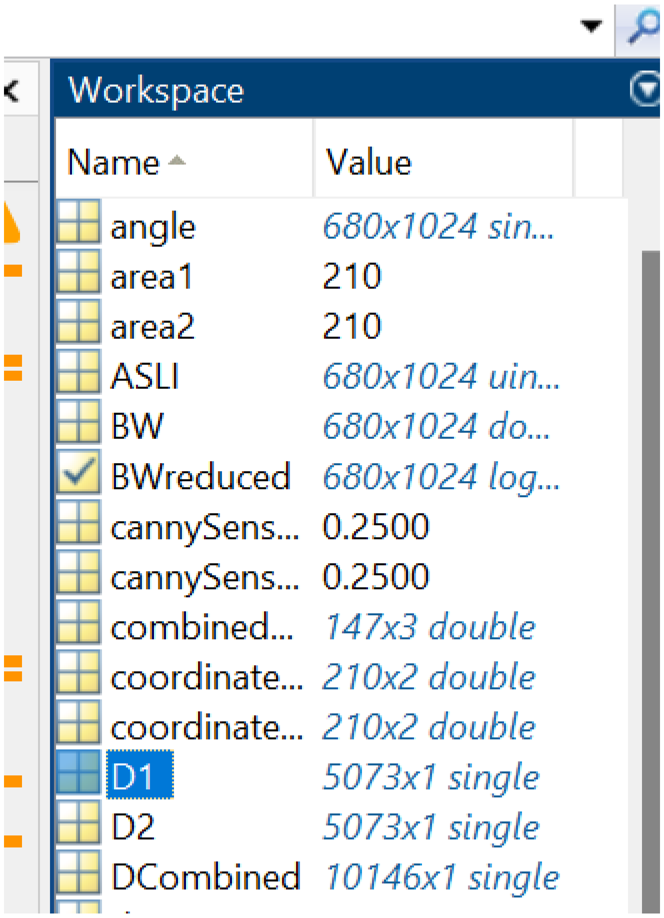


```
>> dual_file_test
The pixel length of the scale ruler is: 66.00 pixels for the first file.
Enter the scale (nm) for the first file: 1000

Enter the Canny Border Detection sensitivity (0-0.999).
The lower the number, the easier matlab detects a region as an edge
-If set less than 0.2 then a blurring filter will be done on the image (for high res)
-If set above, no blurring is required (for low res)
Choose 0.2 for high res images, and 0.21-0.3 for low res images: 0.25
The pixel length of the scale ruler is: 66.00 pixels for the second file.
Enter the scale (nm) for the second file: 200

Enter the Canny Border Detection sensitivity (0-0.999).
The lower the number, the easier matlab detects a region as an edge
-If set less than 0.2 then a blurring filter will be done on the image (for high res)
-If set above, no blurring is required (for low res)
fx Choose 0.2 for high res images, and 0.21-0.3 for low res images: 0.25
```

Wait a few seconds and obtain the result.



The Diameter data for image 1 can be found in a file named ‘D1’ in the workspace. ‘D2’ for image2 and ‘DCombined’ for combined diameter.

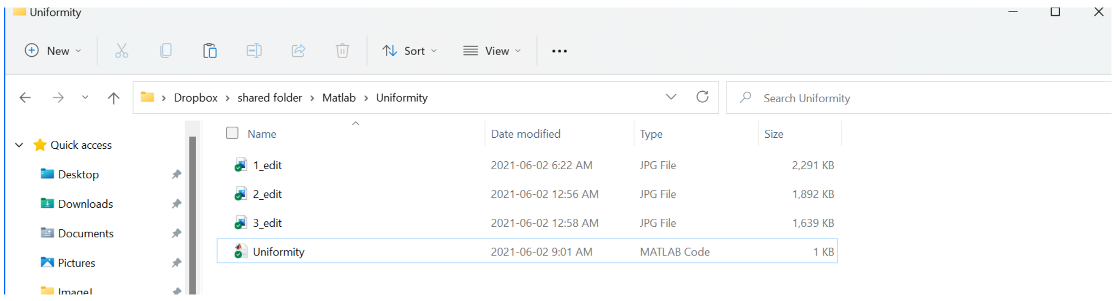


Workspace

Name	Value
angle	680x1024 sin...
area1	210
area2	210
ASLI	680x1024 uin...
BW	680x1024 do...
BWreduced	680x1024 log...
cannySens...	0.2500
cannySens...	0.2500
combined...	147x3 double
coordinate...	210x2 double
coordinate...	210x2 double
D1	5073x1 single
D2	5073x1 single
DCombined	10146x1 single

5. Surface homogeneity determination

Put the SEM image in the same folder with the script



Uniformity

New Sort View

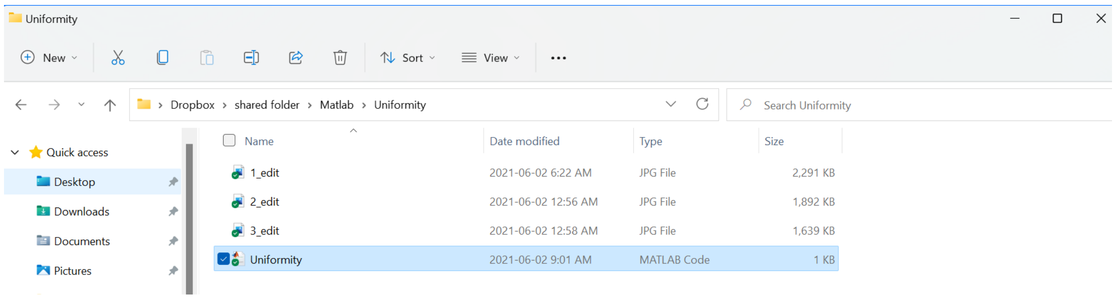
Dropbox › shared folder › Matlab › Uniformity

Search Uniformity

Quick access
Desktop
Downloads
Documents
Pictures
ImageI

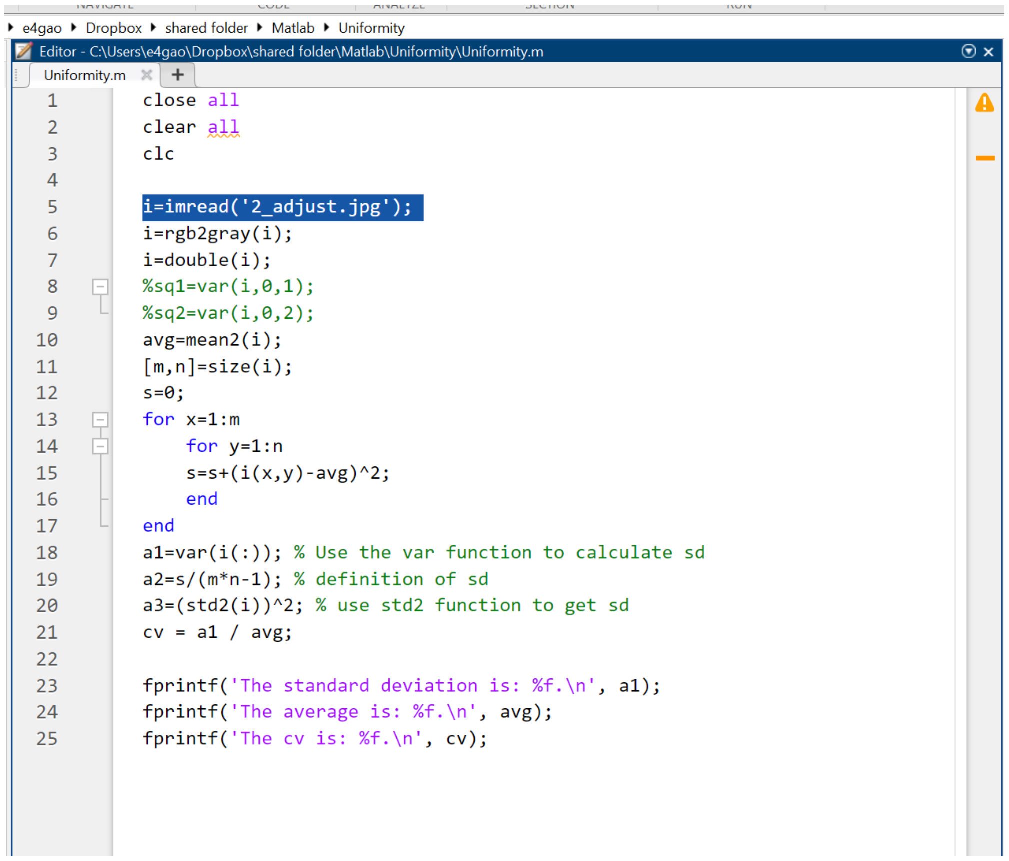
Name	Date modified	Type	Size
1_edit	2021-06-02 6:22 AM	JPG File	2,291 KB
2_edit	2021-06-02 12:56 AM	JPG File	1,892 KB
3_edit	2021-06-02 12:58 AM	JPG File	1,639 KB
Uniformity	2021-06-02 9:01 AM	MATLAB Code	1 KB

Open Uniformity file in the folder:



Name	Date modified	Type	Size
1_edit	2021-06-02 6:22 AM	JPG File	2,291 KB
2_edit	2021-06-02 12:56 AM	JPG File	1,892 KB
3_edit	2021-06-02 12:58 AM	JPG File	1,639 KB
Uniformity	2021-06-02 9:01 AM	MATLAB Code	1 KB

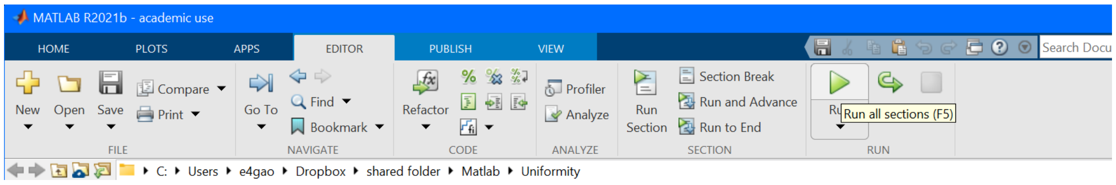
Change the highlighted line to the name of the image file name



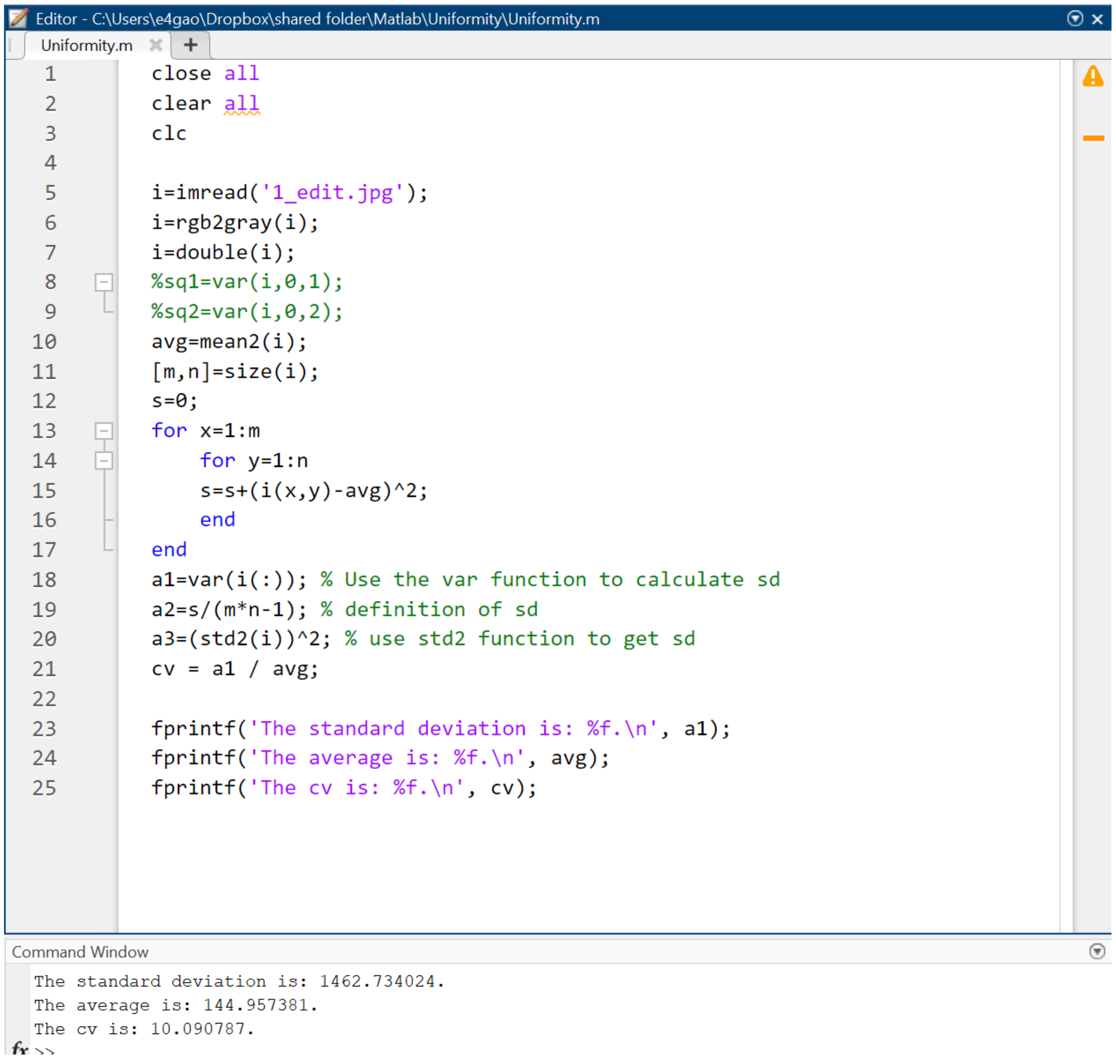
e4gao ▸ Dropbox ▸ shared folder ▸ Matlab ▸ Uniformity
Editor - C:\Users\e4gao\Dropbox\shared folder\Matlab\Uniformity\Uniformity.m
Uniformity.m

```
close all
clear all
clc

i=imread('2_adjust.jpg');
i=rgb2gray(i);
i=double(i);
%sq1=var(i,0,1);
%sq2=var(i,0,2);
avg=mean2(i);
[m,n]=size(i);
s=0;
for x=1:m
    for y=1:n
    s=s+(i(x,y)-avg)^2;
    end
end
a1=var(i(:)); % Use the var function to calculate sd
a2=s/(m*n-1); % definition of sd
a3=(std2(i))^2; % use std2 function to get sd
cv = a1 / avg;

fprintf('The standard deviation is: %f.\n', a1);
fprintf('The average is: %f.\n', avg);
fprintf('The cv is: %f.\n', cv);
```

Run the script



Obtained the result

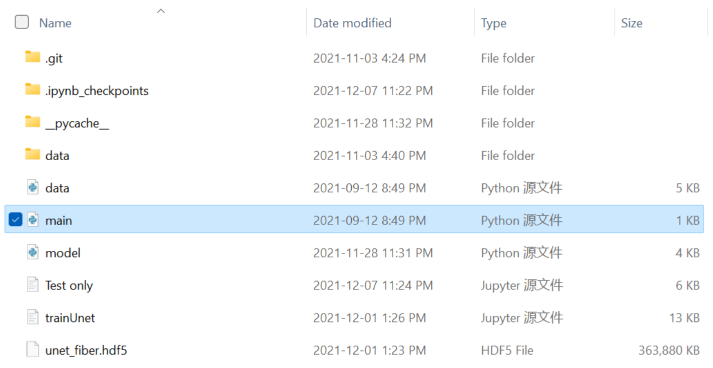

Editor - C:\Users\e4gao\Dropbox\shared folder\Matlab\Uniformity\Uniformity.m
Uniformity.m
Command Window

```matlab
close all
clear all
clc

i=imread('1_edit.jpg');
i=rgb2gray(i);
i=double(i);
%sq1=var(i,0,1);
%sq2=var(i,0,2);
avg=mean2(i);
[m,n]=size(i);
s=0;
for x=1:m
    for y=1:n
    s=s+(i(x,y)-avg)^2;
    end
end
a1=var(i(:)); % Use the var function to calculate sd
a2=s/(m*n-1); % definition of sd
a3=(std2(i))^2; % use std2 function to get sd
cv = a1 / avg;

fprintf('The standard deviation is: %f.\n', a1);
fprintf('The average is: %f.\n', avg);
fprintf('The cv is: %f.\n', cv);
```

```
The standard deviation is: 1462.734024.
The average is: 144.957381.
The cv is: 10.090787.
fx >>
```

6. UNet

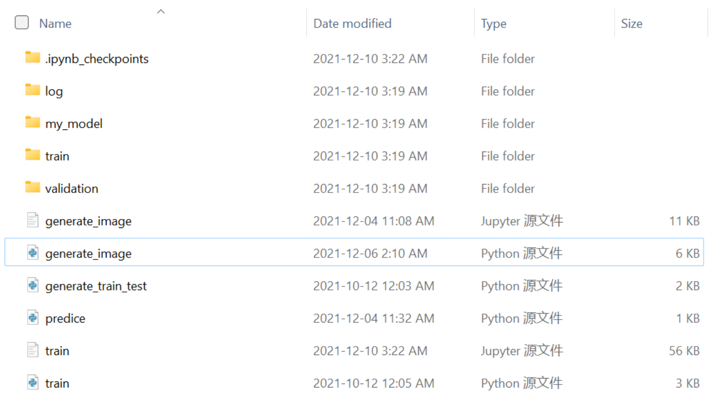
1) Libraries needed
   Tensorflow
   Keras greater than 1.0
   According to the original coder this code should be compatible with Python versions 2.7-3.5.
2) Run main.py to train the model
   With proper hyperparameter, the accuracy can reach 90% within 10 epochs.
3) Files
   Main.py is the main script.
   Data.py is the data augmentation script.
   Model.py is the construction of UNet model.
   unet_fiber.hdf5 is the trained model.
   Put the training data into the 'Data' folder.



Name	Date modified	Type	Size
.git	2021-11-03 4:24 PM	File folder	
.ipynb_checkpoints	2021-12-07 11:22 PM	File folder	
__pycache__	2021-11-28 11:32 PM	File folder	
data	2021-11-03 4:40 PM	File folder	
data	2021-09-12 8:49 PM	Python 源文件	5 KB
main	2021-09-12 8:49 PM	Python 源文件	1 KB
model	2021-11-28 11:31 PM	Python 源文件	4 KB
Test only	2021-12-07 11:24 PM	Jupyter 源文件	6 KB
trainUnet	2021-12-01 1:26 PM	Jupyter 源文件	13 KB
unet_fiber.hdf5	2021-12-01 1:23 PM	HDF5 File	363,880 KB

7. Resnet

1) Libraries needed
   Tensorflow
   Keras greater than 1.0
   OpenCV
   Numpy
   Time Module
   The code is written with Python versions 3.7.
2) Run train.py to train the model
3) Files
   Generate_image.py is the script for image generation
   Generate_train_test.py is the script for dataset division.
   Predice.py is the script for verify the result.
   Train.py is the script for model training.



Name	Date modified	Type	Size
.ipynb_checkpoints	2021-12-10 3:22 AM	File folder	
log	2021-12-10 3:19 AM	File folder	
my_model	2021-12-10 3:19 AM	File folder	
train	2021-12-10 3:19 AM	File folder	
validation	2021-12-10 3:19 AM	File folder	
generate_image	2021-12-04 11:08 AM	Jupyter 源文件	11 KB
generate_image	2021-12-06 2:10 AM	Python 源文件	6 KB
generate_train_test	2021-10-12 12:03 AM	Python 源文件	2 KB
predice	2021-12-04 11:32 AM	Python 源文件	1 KB
train	2021-12-10 3:22 AM	Jupyter 源文件	56 KB
train	2021-10-12 12:05 AM	Python 源文件	3 KB