A PROJECT REPORT ON
AUTOMATIC DETECTION OF DEFECTS IN NANO FIBROUS MATERIALS USING CONVOLUTION NEURAL NETWORK.

Submitted By:

AKHAND PRATAP MISHRA
B-Tech Part III, Roll No: 15035005
DEPT. OF CERAMIC ENGINEERING

HARSHIT MITTAL
B-Tech Part III, Roll No: 15035015
DEPT. OF CERAMIC ENGINEERING

SOUJANYA MADASU
IDD Part III, Roll No: 15034014
DEPT. OF CERAMIC ENGINEERING

Introduction

The most common method used to produce Nano fibrous materials is the electro spinning. The production process of Nano fibres, due to several variables, such as applied voltage and polymer concentration, temperature and relative humidity, etc., may generate structural defects. Defects are usually beads or flattened areas, and have non-uniform distributions, such as alternation of big pores and dense material, and sparse fibre diameter distribution.

Defect detection in industry is performed in several ways:

(1) Manually by humans that monitor the entire production process.

(2) Automatically by a computer-based system that monitors, mostly with the help of digital cameras or sensors, the production process.

(3) Semi-automatically by humans that interact with a computer-based defect detection system.

In all ways, defect detection is performed in two different moments: during or at the end of the production process. To take the chance to correct the production process, defect detection should be performed during the production process in at time that should be less, or at most comparable with, the production time itself. This time constraint permits providing feedbacks or alarms that may be used to correct the production process.

Automatic detection of defects in Nano fibrous materials would reduce both the cost of production process and the time spent in quality inspection. Scanning Electron Microscope images have a resolution up to one nanometre and thus they may be used to detect and localize both fine- and coarse-grained defects. Figure 1b contains examples of SEM images without defects. Figure 1a contains examples of fine-grained defects: a small speck of dust at the centre and two beads (fibre clots) on the right and left side. Figure 1c contains an example of coarse grained defect: a thin layer of material among the fibres that is a film.

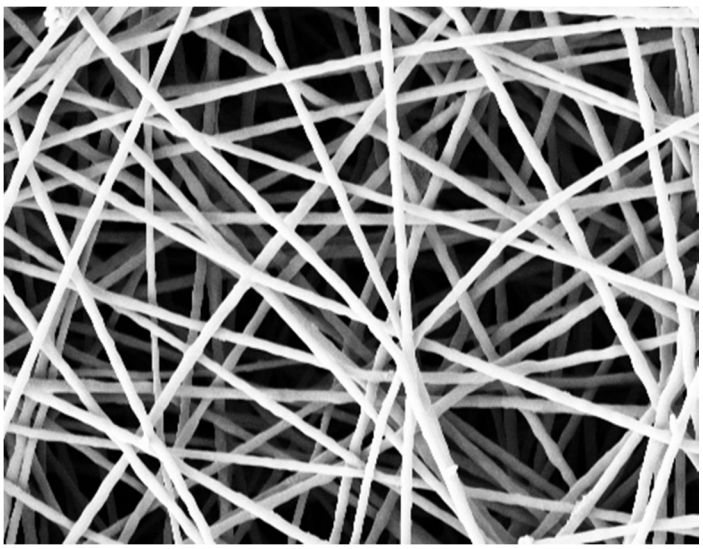

Figure 1a

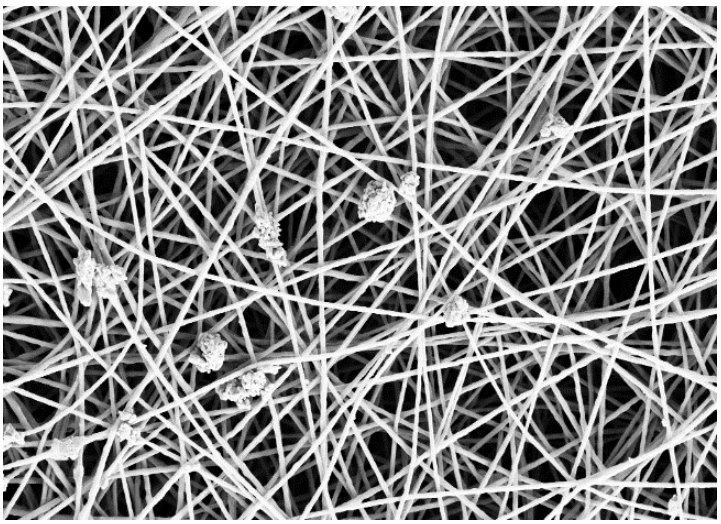

Figure 1b

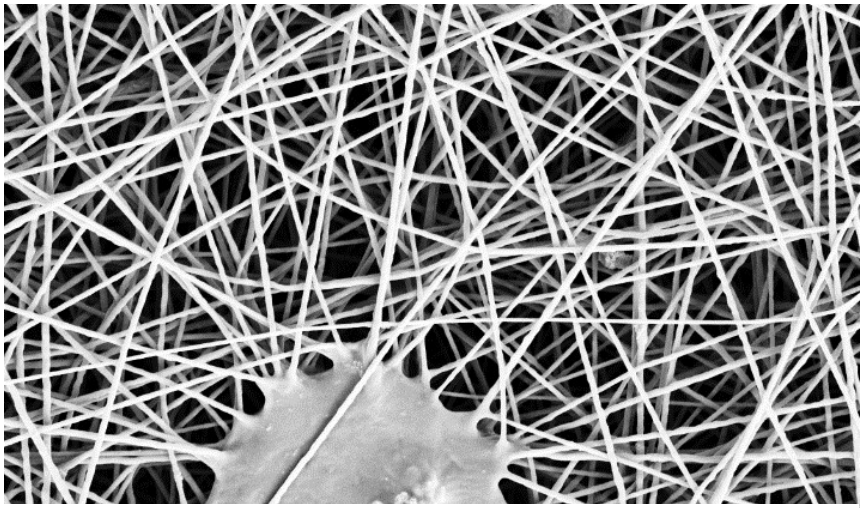

Figure 1c

Here we have taken 61 images of SEM, in which 40 images have defects and 21 images are normal. So our objective is to make the model which classify the materials on the basis of defects.

Data Augmentation

Obviously, it would not be a great idea to make the model with such less amount of data. As, that model would lead to under fitting and produces insignificant results in future. So, for increasing the count of data, augmentation was done on the basis of rotation and translation.

Number of images after data augmentation:

a) Images Generated by spatial translation:

We have taken 4 strips horizontally of width 60, 120, 180 and 240 pixels from one side and translated them on the other and vice-versa from another side. Similarly, 3 strips vertically of width 60, 120 and 180 from one side and translated them on other and vice-versa for other side. So total number of images generated from one image is 14 (8 horizontal translation, 6 vertical translation)

Resulted No of Images:

Images Having Defects: 40*(8 + 6 + 1) = 600
Normal Images: 21*(8 + 6 + 1) = 315

b) Due to Image rotation:

After spatial translation we have rotated images with defects by 10 degree and normal images by 5 degree. We are generating 36 images from 1 defect image and 72 from 1 normal image.

Resulted No of Images:

Having Defect: 600*36 = 21,600
Normal Images: 315*72 = 22,680

Data Pre-processing

Now, we are concerned in labelling the normal and defective images of material followed by splitting the dataset into train and test (with test ratio: 20%)

Labelling:

1) Label Normal Images by: 1
Label Defected Image by: 0

2) Randomly split the labelled images in the ratio of 80:20 and store them in:
x_train = Resulted to all the images that will going to use for training.
y_train = Label of all the images that will going to use for training.
x_test = Resulted to all the images that will going to use for testing.
y_test = Label of all the images that will going to use for testing.

Followed by storing all of them in dictionary and pickling them.

CNN Model

We have built a CNN model using tensorflow to classify the images into normal and defective.

The first layer of this model is convolution layer which convolute on our image matrix having dimensions 32x32x1, this layers contains 50 filters of each having dimensions 5x5x1, the padding is same and strides is 1. The result of this layer is of dimension 32x32x50.

After this layer we have added a max pooling layer of size 2x2x1 with stride 2. The output of this layer is of dimension 16x16x50.

The next layer is another convolution layer with 100 filters of dimension 5x5x50, the padding of this layer is also same. The output of this layer is o dimension 16x16x100.

After this layer there is another max pooling layer of size 2x2x1 with stride 2. The output of this layer is of dimension 8x8x100.

The next layer is the last convolution layer of the model. This layer has 200 filters each of dimension 5x5x100. The output of this layer is of dimension 8x8x200.

After the last convolution layer there is last ma pooling layer of size 2x2x1 with strides 2. The result of this layer is of dimension 4x4x200.

The output of this layer is then reshaped into a single vector of dimension 1x4*4*200, i.e. 1x3200.

The next layer is fully connected layer with input layer of dimension 1x3200, three hidden layer, first, which takes input from input layer and give output a vector of 1600 neurons. The second hidden layer takes input from the previous layer and give output a vector of 800 neurons. The third hidden layer converts the input of dimension 1x800 into 1x2, which is the result of our classifier.

The loss function that we used was **Cross entropy,** the activation function used was **ReLu** and **Softmax** for probabilities estimation. The optimiser used was **AdamOptimiser** with learning rate 0.001.

Result

We trained the model for 50 epochs on Floyd-hub. We were able to achieve the accuracy of 90.05% on training set and 88.93% on test set.

Thanks

Submitted by:

Harshit Mittal 15035015 B-Tech Ceramic Engineering

Akhand Pratap Mishra 15035005 B-Tech Ceramic Engineering

Soujanya Madasu 15034014 IDD Ceramic Engineering