ANOMALY DETECTION IN MANUFACTURING

ISEN 613 : ENGINEERING DATA ANALYSIS

NAME	UIN	INDIVIDUAL CONTRIBUTION
AMIT KUMAR SINGH	731008602	20%
ANWAY MAHENDRA PAWAR	933004339	20%
DYLAN LASRADO	233005743	20%
YASH NARMADAPRASAD JAISWAL	432001166	20%
RICHARD GERMANA	826004043	20%

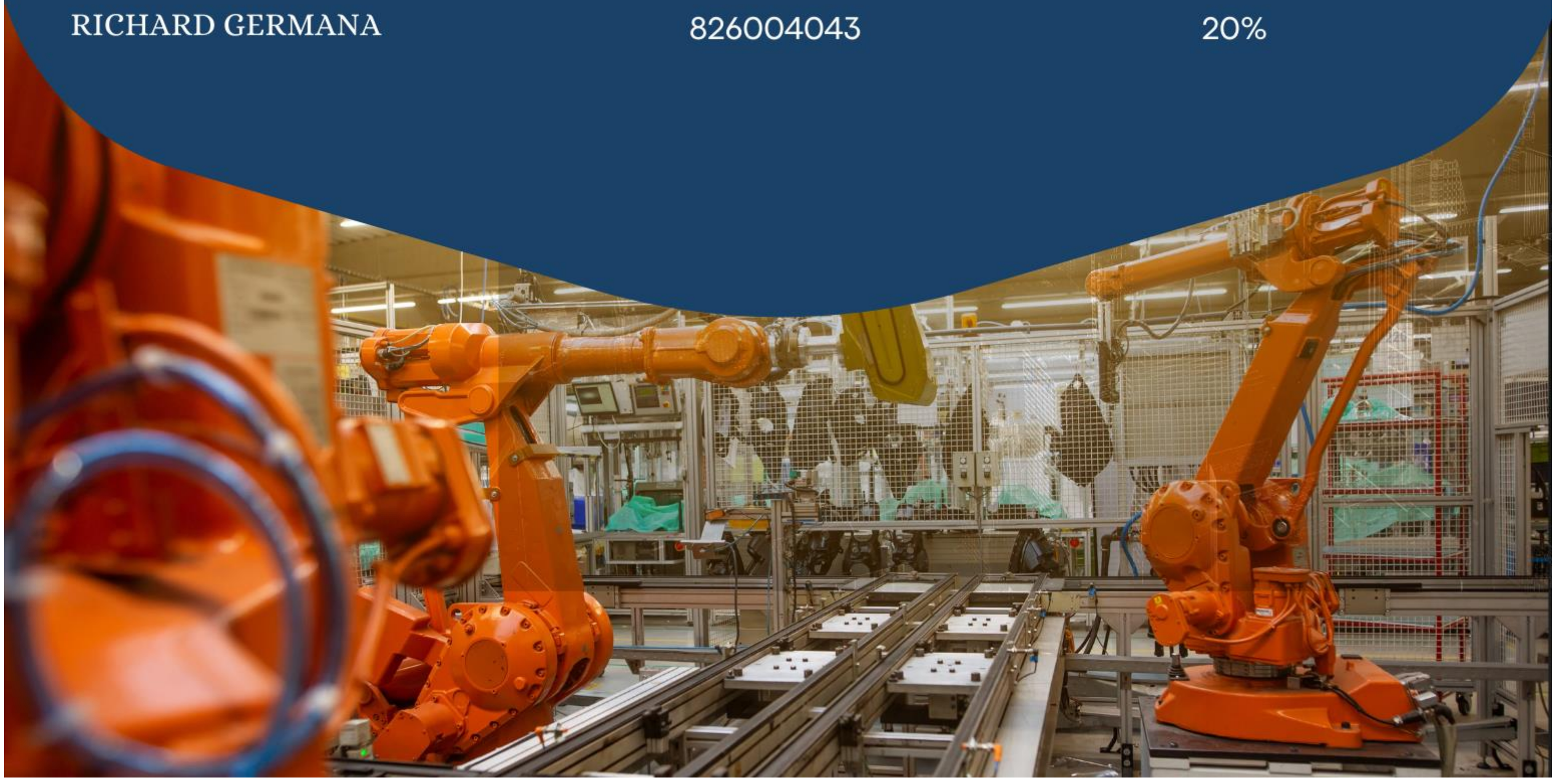


Table of Contents

1. Introduction

1.1 Importance of the problem

The manufacturing sector has a major contribution towards the economies of many developing countries across the globe. Manufacturing is estimated to have a contribution of about 16% towards the global GDP and accounts for 14% of global employment. [1] China, US, Japan and India are some of the countries which have the highest manufacturing output

Many manufacturing processes have been in use since medieval ages. Over the years these manufacturing processes have undergone major improvements. The introduction of concepts such as Six Sigma, Kaizen and Lean manufacturing have had a major impact on the manufacturing industry. The use of advanced modern machinery and computer aided programs has made the process of manufacturing complex parts relatively simple and more efficient. Mass production of various components has also been achieved. The focus has now shifted towards manufacturing processes that are agile, accurate, precise and can produce good quality products. [2]

Manufacturing firms lay special emphasis on quality inspection and specifically surface quality inspection is one of the most critical functions. Quality inspection is used to identify faults or defects in the finished product of the manufacturing process. Common manufacturing defects are surface scratches, pits, cold shuts, high roughness, porosity in the material, cracks, wrong dimensions, and improper paint finish on the surface. These defects occur due to a variety of reasons such as improper designs, unfavorable working conditions as well as due to production machine failure. These defects lead to wastage of raw material, decrease in productivity, increased production cost and a decrease in customer satisfaction thereby lowering the profits of the organization.

Porosity is one such defect that has gained the attention of researchers especially in the field of additive manufacturing over the last couple of years due to the devastating impacts it can have on the structural integrity when present in the final metal component. For instance, surface porosity enables the penetration of corrosive material or moisture which could lead to corrosion and compromises the structural integrity and is thus not desirable. Pores can also lead to spots of stress concentration which lead to the formation and propagation of cracks. This leads to a significant reduction in the mechanical properties of the sample. Porosity can occur due to a variety of reasons during the manufacturing process such as presence of moisture and other contaminants, etc. and can be present on the surface, sub surface or in the bulk interior of the material. Porous materials can also be less appealing. [3]

Some commonly used techniques to identify defects involve inspection techniques such as different receiver gauges, non-destructive and destructive testing, dye penetrant crack detection techniques and more. Ultrasonic testing is a nondestructive testing technique that is widely used for defect detection, especially in porosity detection. These techniques are time consuming, need manpower to perform and are not always accurate Advancements in the field of machine learning, computer vison and artificial intelligence have encouraged companies to replace traditional inspection techniques with modern and more accurate methods. Convolutional Neural networks and other deep learning models are widely being used for defect detection. With the use of machine learning and AI models, manufacturing firms are moving towards automating the quality inspection process. This has major advantages such as reduced need of manpower required for carrying out inspection and an enhanced accuracy. In most advanced techniques, cameras are used at the production line to capture the surface of manufactured workpiece. These photographs are then compared with images/ data of ideal workpieces to identify anomalies. The use of spectrograms has also gained importance in different manufacturing processes. The surface data of the manufacturing workpiece is captured using spectrogram to get the audio data and accelerometer data as well as surface images using cameras. This data can be used to identify possible defects in the workpiece. [4,5,6,7]

1.2 Objective

The objective of this project is to predict the porosity present in the material using different machine learning techniques. To do so, we wish to develop CNN and Linear regression based models which make use of the spectrogram data and the data in the excel sheet as input and predict the percentage area porosity as the output. We also aim to develop a Edge Detection model which uses the black and white images of sample 1 as input and can highlight the edges of the pore.

We also aim to compare the CNN models which have different architectures based on the predictions they produce and how accurate these predictions are when compared to the actual observed values. These models can further help us determine if a material is defective or not, such that defective material or workpiece can be removed from the production line.

1.2 Scope Of The Work

A) CNN BASED MODELS

- Modify the data in the excel sheet such that rows are rearranged in order to match the ordering of the tensor voxels. (MATLAB spectrogram data)
- Develop 2 CNN based models having different CNN architectures which use the spectrogram data and the percentage area of porosity from the excel sheet as the input and predict the percentage area porosity of untrained samples as the output.
- Compare the results and the architecture of these models.

B) SIMPLE REGRESSION MODEL

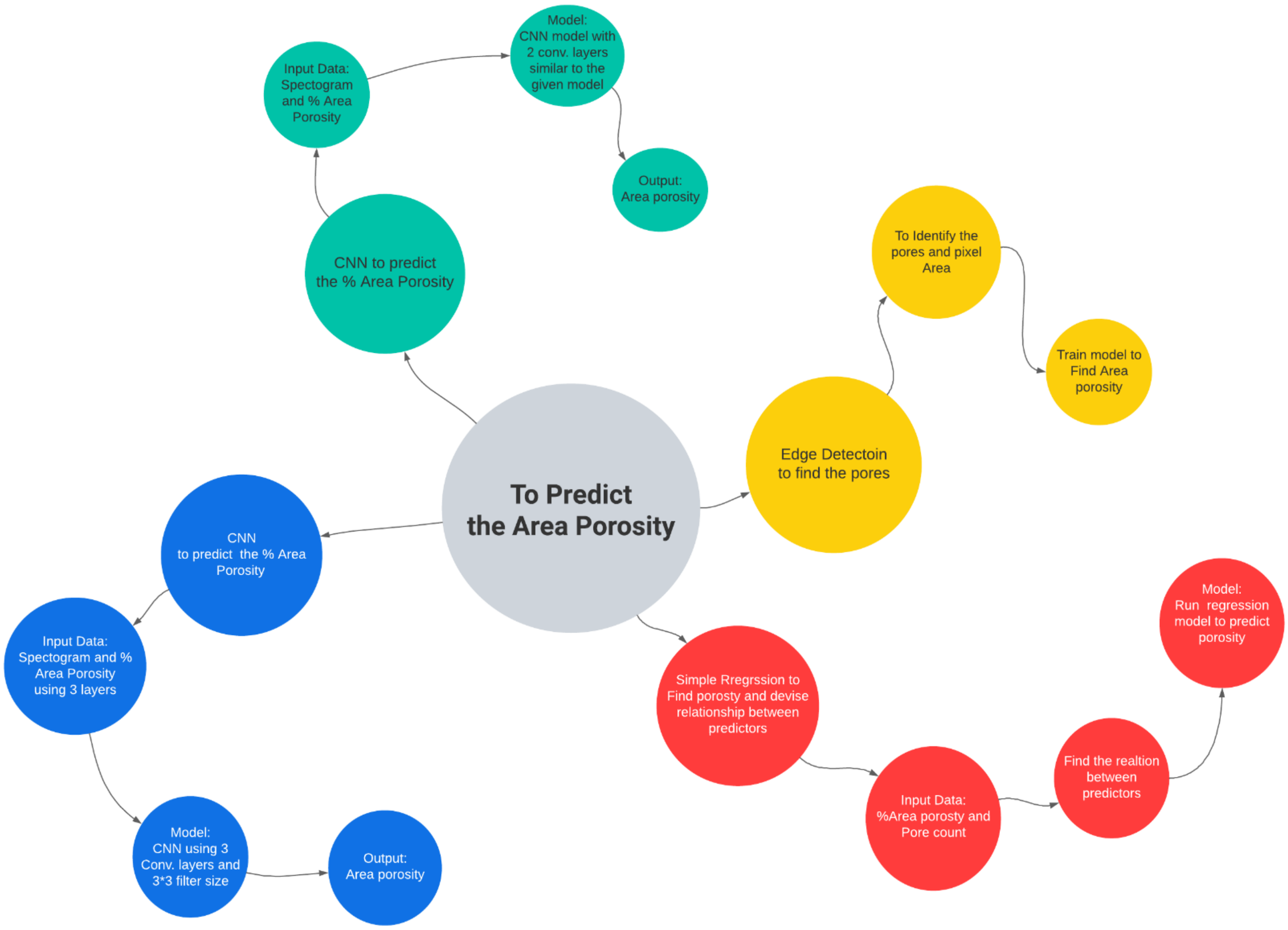
- Explore the relationship between the number of pores and the percentage area of porosity.
- Develop a Simple Linear Regression model which best explains the variation in the input data.
- Analyze the results and explore the drawbacks in using this model.

C) EDGE DETECTION MODEL

- Develop a Edge Detection model which uses the black and white images from sample 1 as the input and highlights the edges of the pores.

Provide an interpretation of the results of these models and a comparison between the different models.

Mind Map



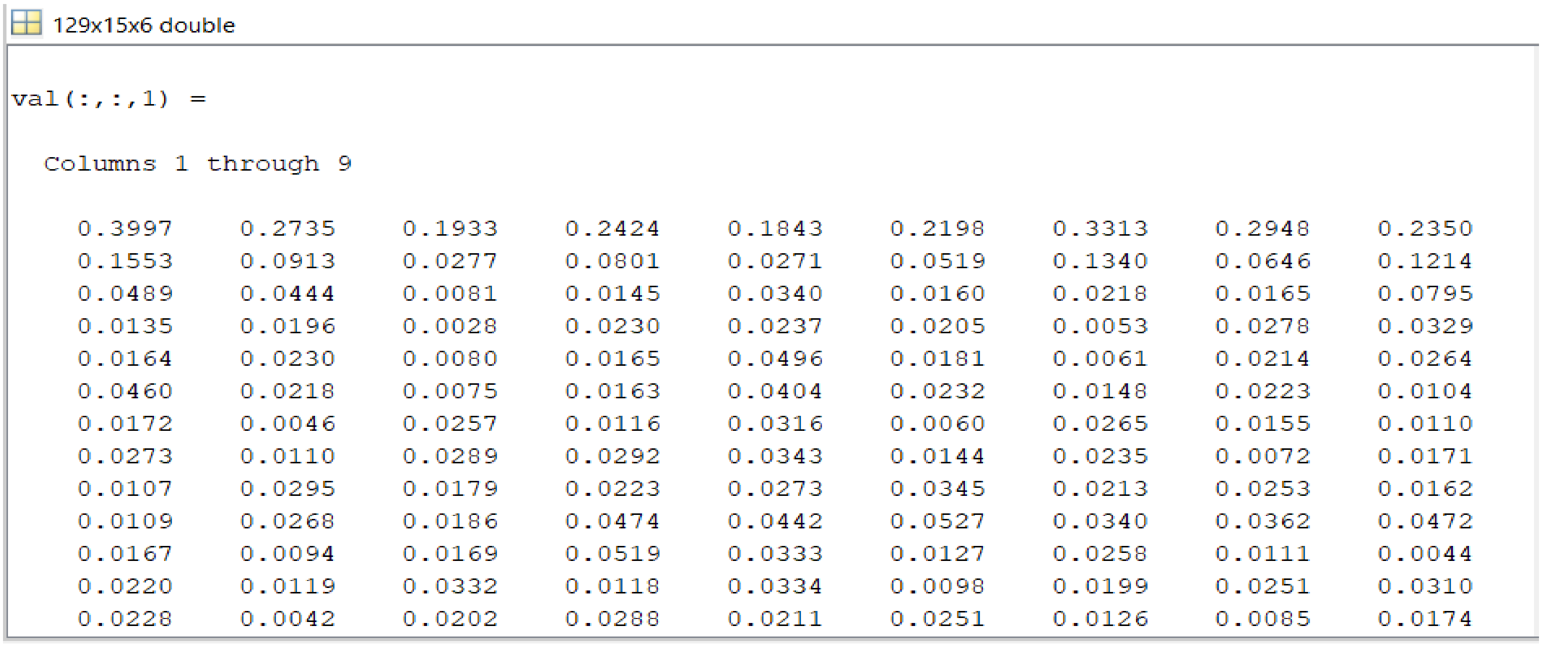
To Predict the Area Porosity
CNN to predict the % Area Porosity
Input Data: Spectogram and % Area Porosity
Model: CNN model with 2 conv. layers similar to the given model
Output: Area porosity
Edge Detectoin to find the pores
To Identify the pores and pixel Area
Train model to Find Area porosity
CNN to predict the % Area Porosity
Input Data: Spectogram and % Area Porosity using 3 layers
Model: CNN using 3 Conv. layers and 3*3 filter size
Output: Area porosity
Simple Rregrssion to Find porosty and devise relationship between predictors
Input Data: %Area porosty and Pore count
Find the realtion between predictors
Model: Run regression model to predict porosity

2. Project Approach

2.1 Data Description

A) SPECTROGRAM DATA-

- Spectrogram data contains 72 MATLAB files of tensor voxels corresponding to 72 samples. Each tensor voxel corresponds to dimension of 129 X 15 X 6 Spectrogram. Where 6 are the number of channels, 1st, 4th corresponds to milling and 2nd, 3rd, 5th and 6th corresponds to the printing.

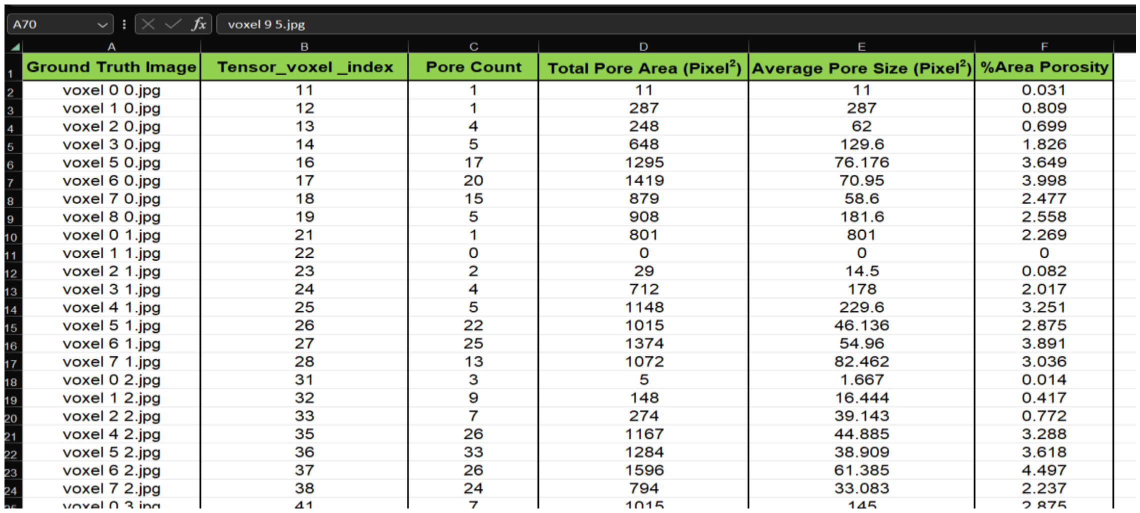


```
129x15x6 double

val(:,:,1) =

  Columns 1 through 9

    0.3997    0.2735    0.1933    0.2424    0.1843    0.2198    0.3313    0.2948    0.2350
    0.1553    0.0913    0.0277    0.0801    0.0271    0.0519    0.1340    0.0646    0.1214
    0.0489    0.0444    0.0081    0.0145    0.0340    0.0160    0.0218    0.0165    0.0795
    0.0135    0.0196    0.0028    0.0230    0.0237    0.0205    0.0053    0.0278    0.0329
    0.0164    0.0230    0.0080    0.0165    0.0496    0.0181    0.0061    0.0214    0.0264
    0.0460    0.0218    0.0075    0.0163    0.0404    0.0232    0.0148    0.0223    0.0104
    0.0172    0.0046    0.0257    0.0116    0.0316    0.0060    0.0265    0.0155    0.0110
    0.0273    0.0110    0.0289    0.0292    0.0343    0.0144    0.0235    0.0072    0.0171
    0.0107    0.0295    0.0179    0.0223    0.0273    0.0345    0.0213    0.0253    0.0162
    0.0109    0.0268    0.0186    0.0474    0.0442    0.0527    0.0340    0.0362    0.0472
    0.0167    0.0094    0.0169    0.0519    0.0333    0.0127    0.0258    0.0111    0.0044
    0.0220    0.0119    0.0332    0.0118    0.0334    0.0098    0.0199    0.0251    0.0310
    0.0228    0.0042    0.0202    0.0288    0.0211    0.0251    0.0126    0.0085    0.0174
```

Inputs from spectrogram

B) EXCEL DATA -

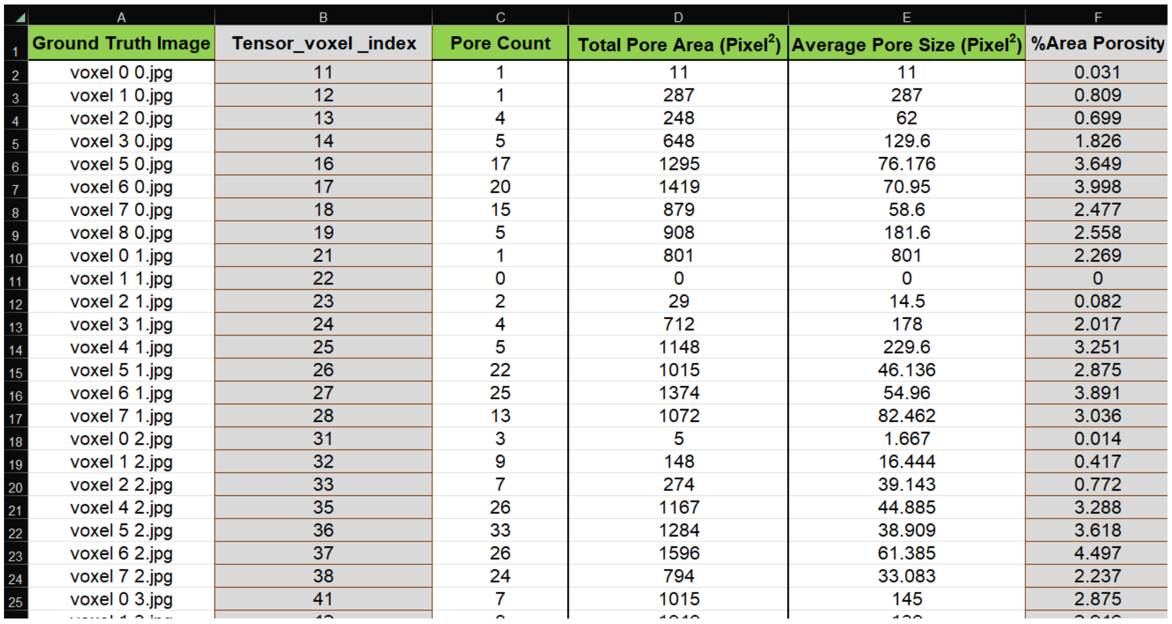
- The excel sheet consists of ground truth of sample 1. This gives the real values and information about the pore count, total pore area, average pore size and % area porosity.
- The given excel data was modified such that the 'Tensor_voxel_index' column of this data matches with the order of the tensor voxels from the (MATLAB data).



A70 | voxel 9 5.jpg

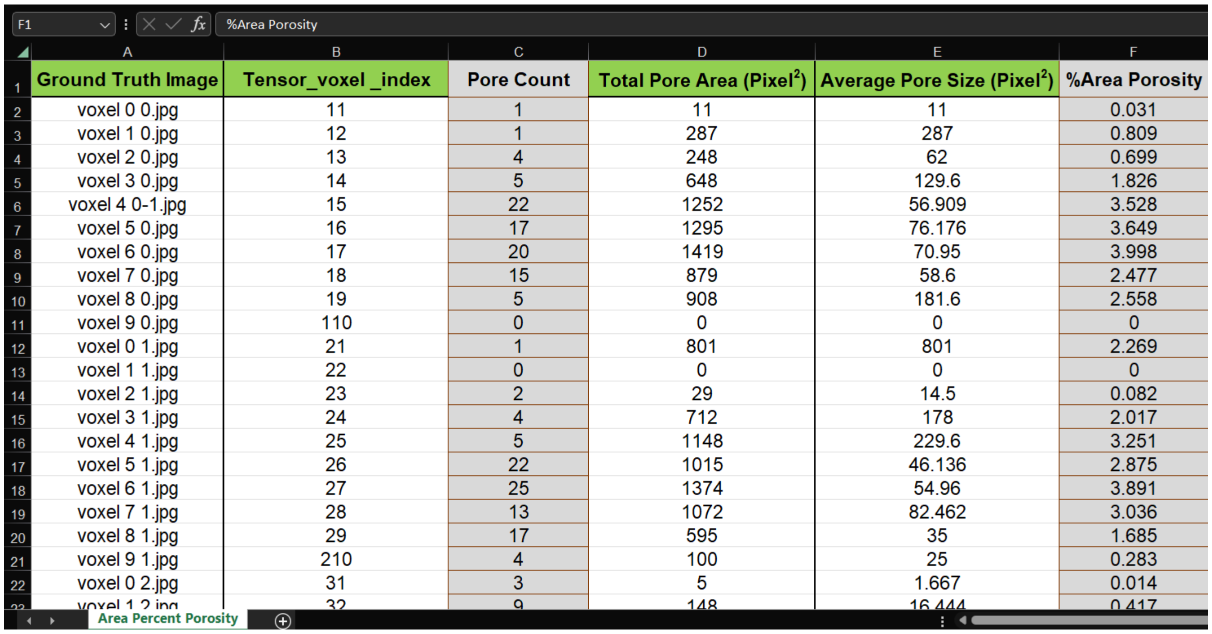
	A	B	C	D	E	F
1	Ground Truth Image	Tensor_voxel _index	Pore Count	Total Pore Area (Pixel^2)	Average Pore Size (Pixel^2)	%Area Porosity
2	voxel 0 0.jpg	11	1	11	11	0.031
3	voxel 1 0.jpg	12	1	287	287	0.809
4	voxel 2 0.jpg	13	4	248	62	0.699
5	voxel 3 0.jpg	14	5	648	129.6	1.826
6	voxel 5 0.jpg	16	17	1295	76.176	3.649
7	voxel 6 0.jpg	17	20	1419	70.95	3.998
8	voxel 7 0.jpg	18	15	879	58.6	2.477
9	voxel 8 0.jpg	19	5	908	181.6	2.558
10	voxel 0 1.jpg	21	1	801	801	2.269
11	voxel 1 1.jpg	22	0	0	0	0
12	voxel 2 1.jpg	23	2	29	14.5	0.082
13	voxel 3 1.jpg	24	4	712	178	2.017
14	voxel 4 1.jpg	25	5	1148	229.6	3.251
15	voxel 5 1.jpg	26	22	1015	46.136	2.875
16	voxel 6 1.jpg	27	25	1374	54.96	3.891
17	voxel 7 1.jpg	28	13	1072	82.462	3.036
18	voxel 0 2.jpg	31	3	5	1.667	0.014
19	voxel 1 2.jpg	32	9	148	16.444	0.417
20	voxel 2 2.jpg	33	7	274	39.143	0.772
21	voxel 4 2.jpg	35	26	1167	44.885	3.288
22	voxel 5 2.jpg	36	33	1284	38.909	3.618
23	voxel 6 2.jpg	37	26	1596	61.385	4.497
24	voxel 7 2.jpg	38	24	794	33.083	2.237

Modified Excel Data Sheet as Ground Truth



	A	B	C	D	E	F
1	Ground Truth Image	Tensor_voxel _index	Pore Count	Total Pore Area (Pixel^2)	Average Pore Size (Pixel^2)	%Area Porosity
2	voxel 0 0.jpg	11	1	11	11	0.031
3	voxel 1 0.jpg	12	1	287	287	0.809
4	voxel 2 0.jpg	13	4	248	62	0.699
5	voxel 3 0.jpg	14	5	648	129.6	1.826
6	voxel 5 0.jpg	16	17	1295	76.176	3.649
7	voxel 6 0.jpg	17	20	1419	70.95	3.998
8	voxel 7 0.jpg	18	15	879	58.6	2.477
9	voxel 8 0.jpg	19	5	908	181.6	2.558
10	voxel 0 1.jpg	21	1	801	801	2.269
11	voxel 1 1.jpg	22	0	0	0	0
12	voxel 2 1.jpg	23	2	29	14.5	0.082
13	voxel 3 1.jpg	24	4	712	178	2.017
14	voxel 4 1.jpg	25	5	1148	229.6	3.251
15	voxel 5 1.jpg	26	22	1015	46.136	2.875
16	voxel 6 1.jpg	27	25	1374	54.96	3.891
17	voxel 7 1.jpg	28	13	1072	82.462	3.036
18	voxel 0 2.jpg	31	3	5	1.667	0.014
19	voxel 1 2.jpg	32	9	148	16.444	0.417
20	voxel 2 2.jpg	33	7	274	39.143	0.772
21	voxel 4 2.jpg	35	26	1167	44.885	3.288
22	voxel 5 2.jpg	36	33	1284	38.909	3.618
23	voxel 6 2.jpg	37	26	1596	61.385	4.497
24	voxel 7 2.jpg	38	24	794	33.083	2.237
25	voxel 0 3.jpg	41	7	1015	145	2.875

Data input for CNN Models (Highlighted in grey)



F1 %Area Porosity

Ground Truth Image	Tensor_voxel _index	Pore Count	Total Pore Area (Pixel^2)	Average Pore Size (Pixel^2)	%Area Porosity
voxel 0 0.jpg	11	1	11	11	0.031
voxel 1 0.jpg	12	1	287	287	0.809
voxel 2 0.jpg	13	4	248	62	0.699
voxel 3 0.jpg	14	5	648	129.6	1.826
voxel 4 0-1.jpg	15	22	1252	56.909	3.528
voxel 5 0.jpg	16	17	1295	76.176	3.649
voxel 6 0.jpg	17	20	1419	70.95	3.998
voxel 7 0.jpg	18	15	879	58.6	2.477
voxel 8 0.jpg	19	5	908	181.6	2.558
voxel 9 0.jpg	110	0	0	0	0
voxel 0 1.jpg	21	1	801	801	2.269
voxel 1 1.jpg	22	0	0	0	0
voxel 2 1.jpg	23	2	29	14.5	0.082
voxel 3 1.jpg	24	4	712	178	2.017
voxel 4 1.jpg	25	5	1148	229.6	3.251
voxel 5 1.jpg	26	22	1015	46.136	2.875
voxel 6 1.jpg	27	25	1374	54.96	3.891
voxel 7 1.jpg	28	13	1072	82.462	3.036
voxel 8 1.jpg	29	17	595	35	1.685
voxel 9 1.jpg	210	4	100	25	0.283
voxel 0 2.jpg	31	3	5	1.667	0.014
voxel 1 2.jpg	32	9	148	16.444	0.417

Area Percent Porosity

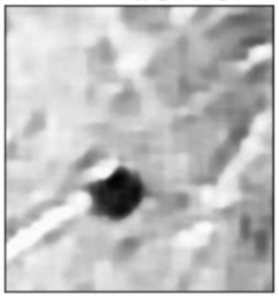
Data input for Simple Regression Model (Highlighted in grey)

C) B/W IMAGES OF SAMPLE-

- Images shown below have been used as the input for edge detection model

voxel 0 0.jpg image

voxel 0 1.jpg Image

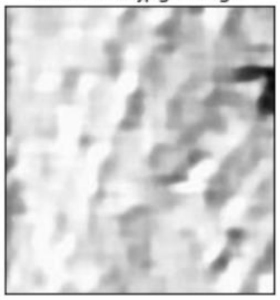


voxel 0 3.jpg Image

voxel 0 4.jpg Image

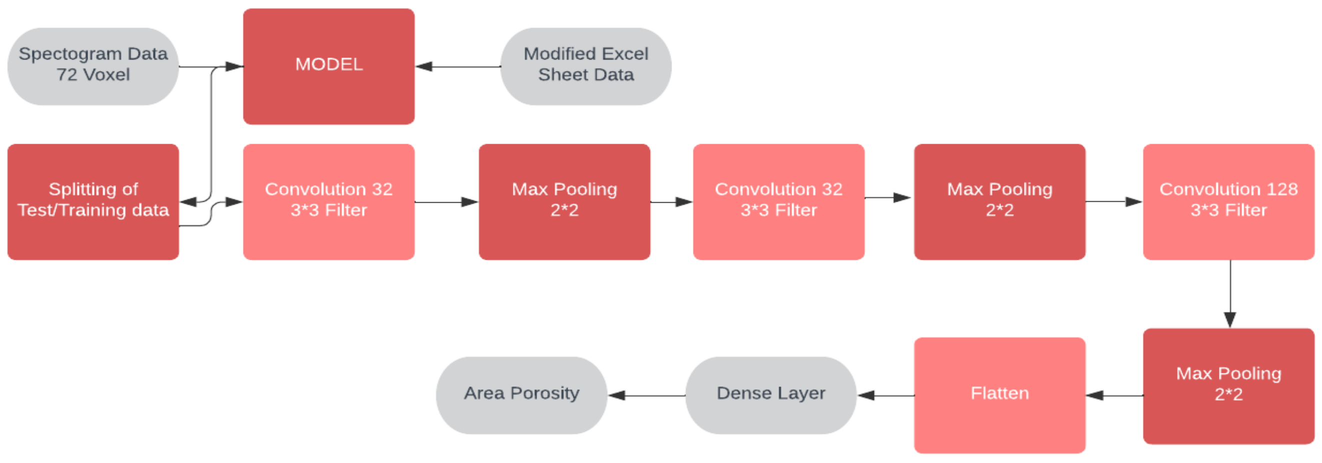
voxel 0 5.jpg Image

voxel 0 6.jpg Image

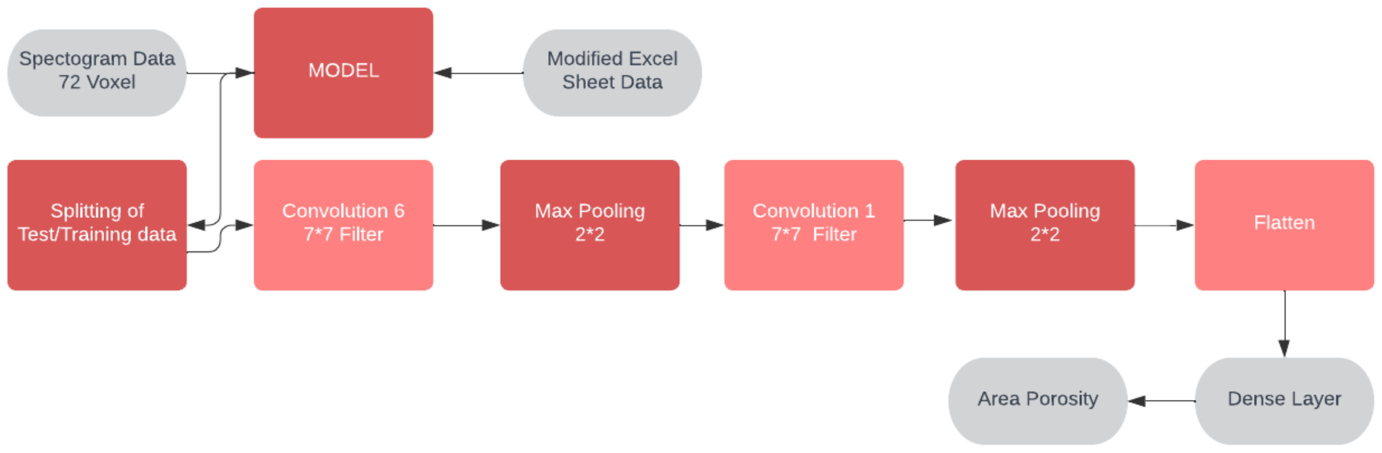


2.2 Flow Chart To Achieve The Objective

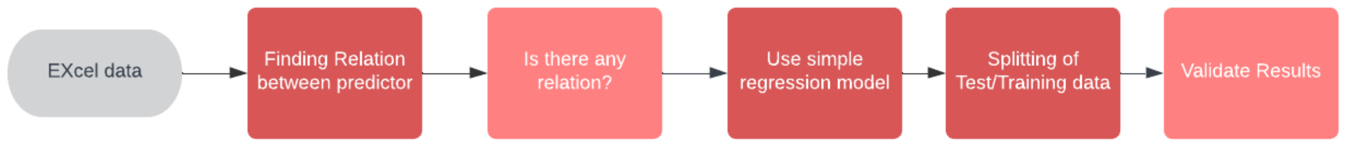
A) CNN MODEL WITH 3 CONVOLUTION LAYERS


Spectogram Data 72 Voxel
MODEL
Modified Excel Sheet Data
Splitting of Test/Training data
Convolution 32 3*3 Filter
Max Pooling 2*2
Convolution 32 3*3 Filter
Max Pooling 2*2
Convolution 128 3*3 Filter
Max Pooling 2*2
Flatten
Dense Layer
Area Porosity

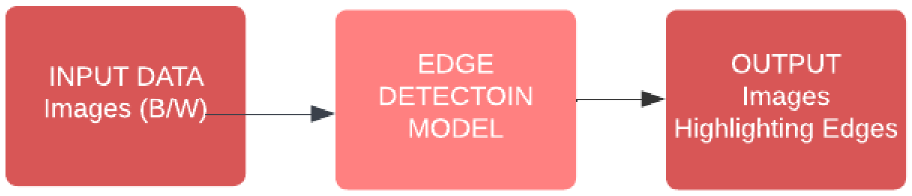
B) CNN MODEL WITH 2 CONVOLUTION LAYERS



C) SIMPLE LINEAR REGRESSION


EXcel data
Finding Relation between predictor
Is there any relation?
Use simple regression model
Splitting of Test/Training data
Validate Results

D) EDGE DETECTION MODEL



2.3 Reason For Our Approach

Traditional methods used in the detection of porosity suffer from various drawbacks. For instance, radioactive detection techniques such as X-rays are widely used for the detection of pores. However, these techniques are extremely expensive and can suffer from environmental interferences. Other testing techniques that are used for porosity detection can lead to the sample becoming unusable post application of such techniques

The reason we chose to use machine learning models is because of the various advantages associated with it. Deep learning machine learning models have the ability to learn massive amounts of data and detect patterns and other features from this data. Deep learning methods have outperformed many other machine learning methods in different domains. Among these methods Convolutional Neural networks is a widely used deep learning technique because of its ability to detect significant features without human intervention. [8] The reason we chose to develop CNN models is because they give us fairly accurately predictions on the percentage area of porosity. These models can be trained with spectrogram data and data from the excel sheet and have the ability to detect patterns and features from the input data and give out accurate predictions. We also wished to explore the impact that a CNN architecture can have on the prediction.

We also wish to explore if we can develop a Simple Linear Regression model which can predict the percentage area of porosity using either the spectrogram data or the data from the excel sheet as input.

Apart from these models we also wish to explore if machine learning models such as Edge Detection can be used to highlight the pores which can further be used with models that predict the porosity.

3. Implementation Details

3.1 CNN MODEL WITH 3 CONVOLUTION LAYERS (MODEL 1) –

A CNN model which predicts the percentage area porosity

STEP 1 - IMPORTING LIBRARIES AND PACKAGES

We first import the required libraries and packages in our code. A brief description of some of the important libraries used is provided below. [9]

I. **Sklearn-** The 'sklearn' library is one of the most widely used machine learning library in Python.

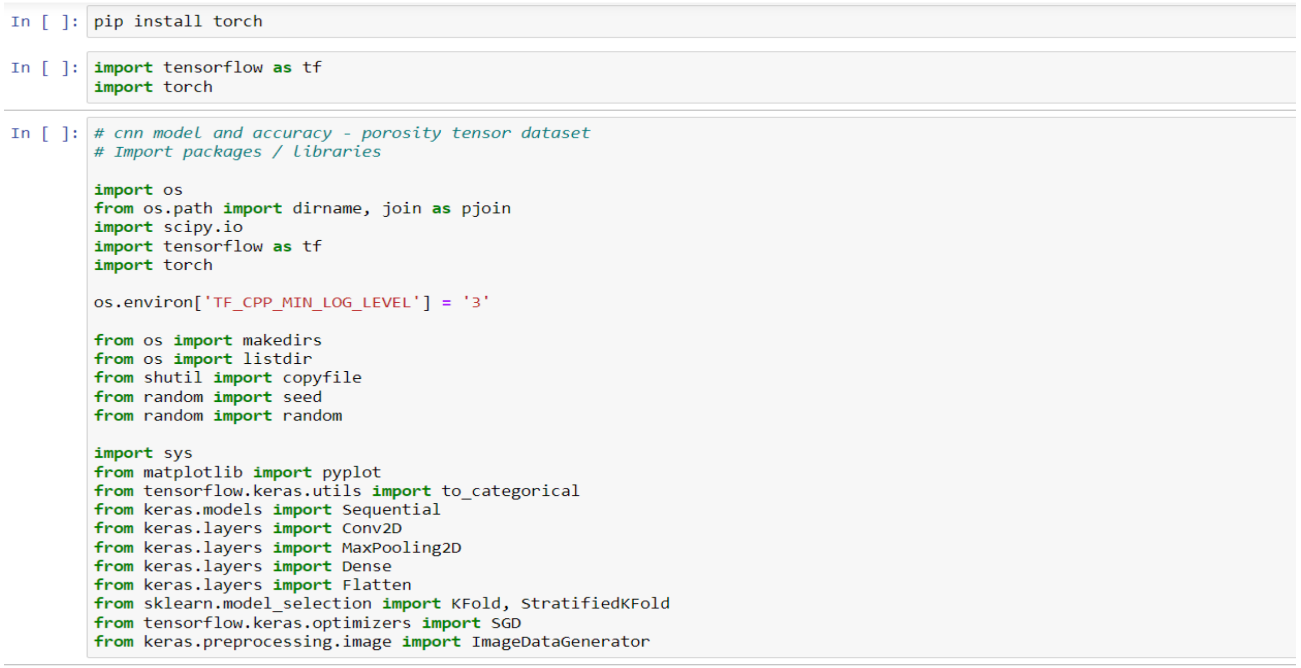
II. **Tensorflow -** The 'TensorFlow' library is an open-source library which is able to carry out fast numerical computation. It is often used to make neural networks faster and easier.

III. **Pandas -** The 'pandas' library is used for data manipulation and data analysis.

IV. **NumPy -** The 'NumPy' library is used so that we can work with large arrays and matrices.

V. **Torch -** The PyTorch library is used to develop and train deep learning models based on neural networks.

VI. **Matplotlib -** The Matplotlib library is used for plotting and data visualization. Using this library, a variety of graphs, plots and charts can be created.



```
In [ ]: pip install torch

In [ ]: import tensorflow as tf
        import torch

In [ ]: # cnn model and accuracy - porosity tensor dataset
        # Import packages / libraries

        import os
        from os.path import dirname, join as pjoin
        import scipy.io
        import tensorflow as tf
        import torch

        os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'

        from os import makedirs
        from os import listdir
        from shutil import copyfile
        from random import seed
        from random import random

        import sys
        from matplotlib import pyplot
        from tensorflow.keras.utils import to_categorical
        from keras.models import Sequential
        from keras.layers import Conv2D
        from keras.layers import MaxPooling2D
        from keras.layers import Dense
        from keras.layers import Flatten
        from sklearn.model_selection import KFold, StratifiedKFold
        from tensorflow.keras.optimizers import SGD
        from keras.preprocessing.image import ImageDataGenerator
```

```
In [ ]: pip install lime
```

```
In [ ]: pip install opencv-python
```

```
In [ ]: import numpy as np
        from sklearn.model_selection import train_test_split
        from matplotlib import pyplot as plt
        from skimage.color import gray2rgb
        from skimage.color import rgb2gray
        from tensorflow.keras.preprocessing import image
        from tensorflow.keras.applications import inception_v3 as inc_net
        from lime import lime_image
        import pandas as pd
        import glob
        from tensorflow.keras.models import load_model
        from sklearn.metrics import r2_score
        import cv2
        from keras.preprocessing.image import ImageDataGenerator
        from skimage.segmentation import mark_boundaries
```

```
In [ ]: pip install tensorflow-addons
```

```
In [ ]: pip install pydotplus
```

```
In [ ]: pip install graphviz
```

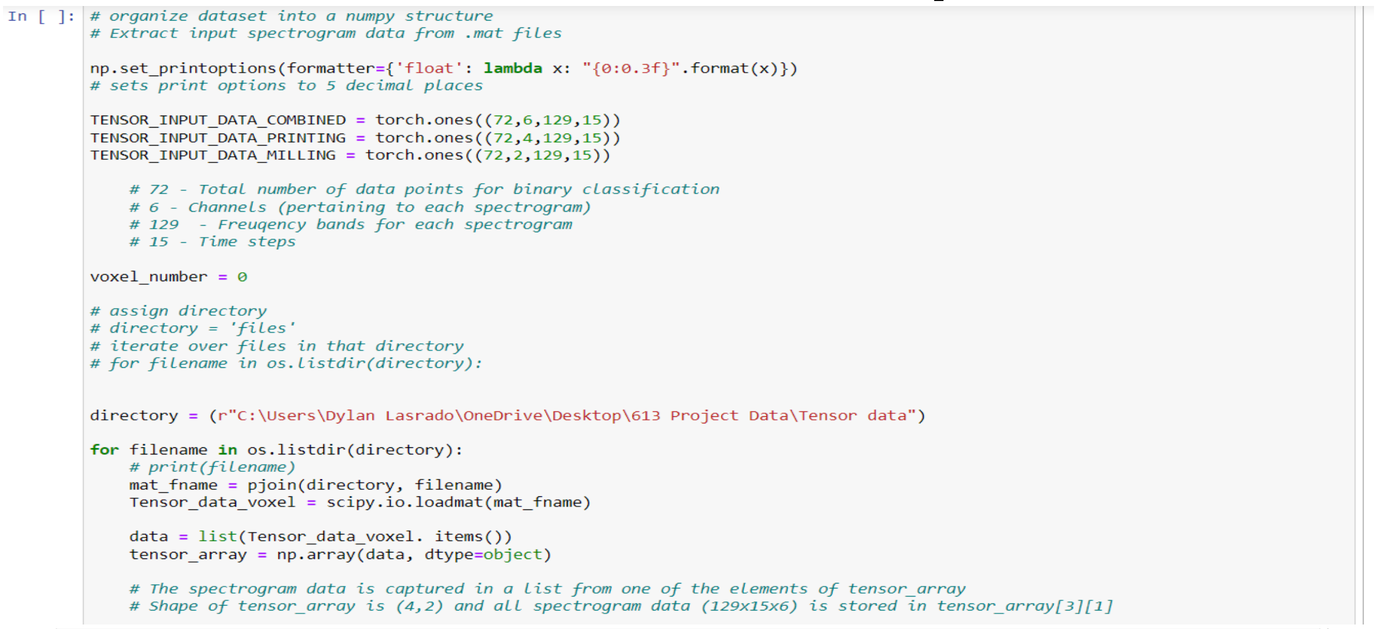
```
In [ ]: pip install pydot
```

```
In [ ]: from keras.layers import ELU, PReLU, LeakyReLU
        import matplotlib.pyplot as plt
```

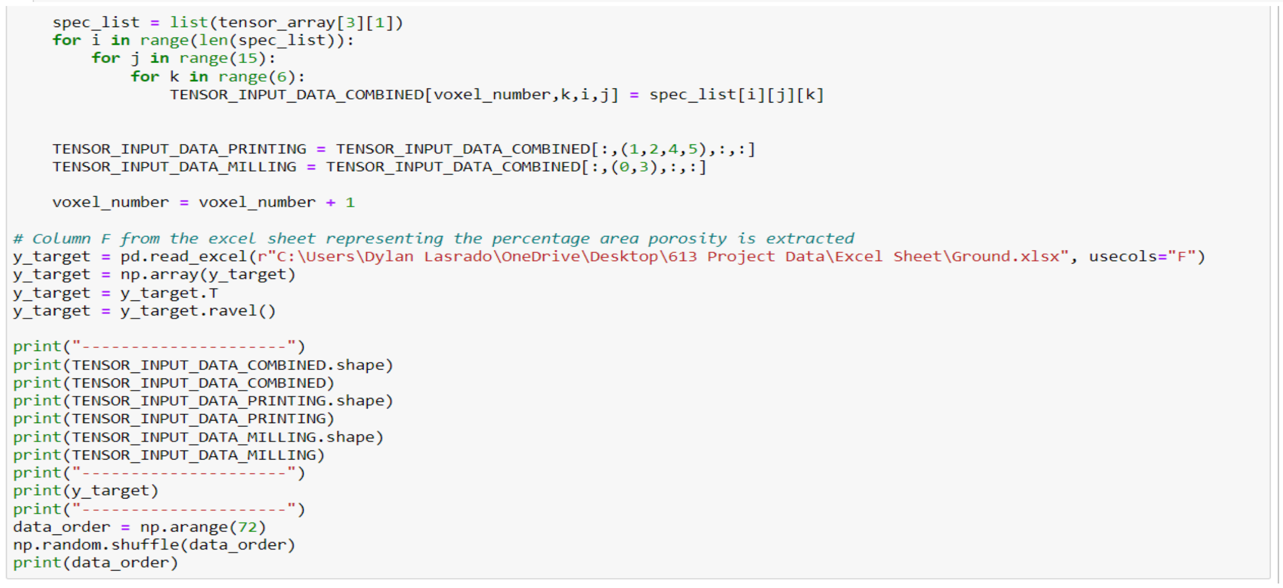
```
In [ ]: import numpy as np
        from tensorflow.keras import backend as K
        import matplotlib.pyplot as plt
        import math
        import scipy
        import scipy.io
        from PIL import Image
        from scipy import ndimage
        from tensorflow.keras.models import Sequential
        from tensorflow.keras.layers import Dense, Dropout, Flatten
        from tensorflow.keras.layers import Conv2D, MaxPooling2D
        from tensorflow.keras.utils import plot_model
        from tensorflow.keras.callbacks import ModelCheckpoint
        from tensorflow.keras.utils import to_categorical
        import tensorflow as tf
        # import tensorflow_addons as tfa
        import pydot
        import pydotplus
        import graphviz
        from tensorflow.keras.preprocessing.image import load_img
        from tensorflow.keras.preprocessing.image import img_to_array
        from tensorflow.keras.models import Model
        import random
        from keras.models import load_model
        import pickle
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import accuracy_score
        import glob
        import os
        import pandas as pd

        import types
        from lime.utils.generic_utils import has_arg
        from skimage.segmentation import felzenszwalb, slic, quickshift
        import copy
        from functools import partial

        import sklearn
        import sklearn.preprocessing
        from sklearn.utils import check_random_state
        from skimage.color import gray2rgb
        from tqdm.auto import tqdm

        import scipy.ndimage as ndi
        from skimage.segmentation._quickshift_cy import _quickshift_cython

        from lime import lime_base
        from lime.wrappers.scikit_image import SegmentationAlgorithm

        import skimage
        from matplotlib import colors
        from skimage.segmentation import mark_boundaries, find_boundaries
        from skimage.morphology import dilation,square
        from collections import Counter
```

STEP 2 - READING AND INTERPRETING THE INPUT DATA

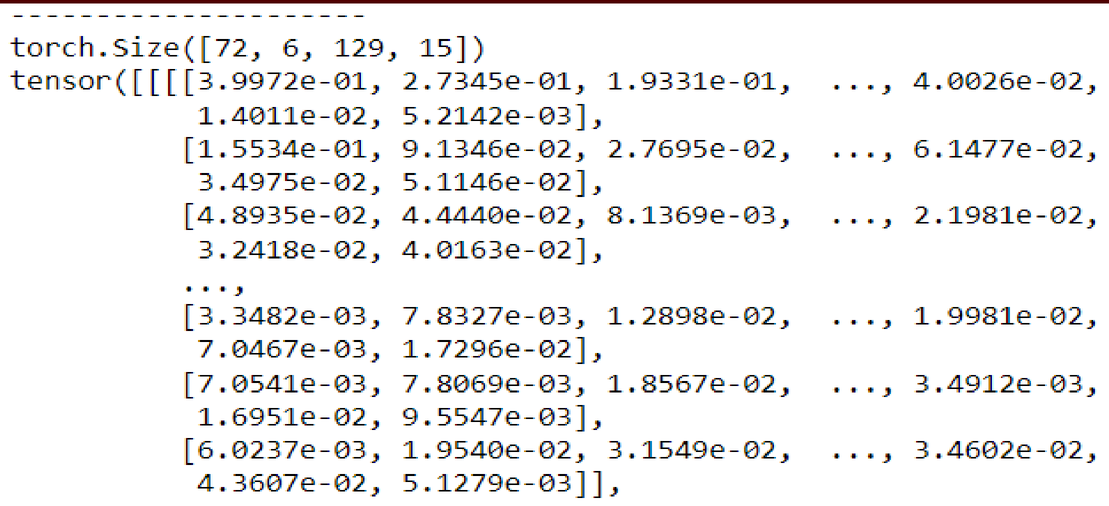
- The python code shown below extracts the spectrogram data of 72 voxels of sample 1 from the MATLAB files. This tensor voxel data is used as input for the CNN code.
- The percentage area porosity (column F of the excel sheet) is extracted from the excel sheet as input. This data is further converted into an array. The data in the excel sheet was rearranged and some rows of data were deleted such that each of the 72-tensor voxel corresponded to the percentage area porosity from the excel sheet. The excel sheet used by this model consists of 72 rows of data which is equal to the number of tensor voxel (spectrogram data) provided to us.
- The data is further shuffled, and the order of the data is printed out



```
In [ ]: # organize dataset into a numpy structure
        # Extract input spectrogram data from .mat files

        np.set_printoptions(formatter={'float': lambda x: "{0:0.3f}".format(x)})
        # sets print options to 5 decimal places

        TENSOR_INPUT_DATA_COMBINED = torch.ones((72,6,129,15))
        TENSOR_INPUT_DATA_PRINTING = torch.ones((72,4,129,15))
        TENSOR_INPUT_DATA_MILLING = torch.ones((72,2,129,15))

            # 72 - Total number of data points for binary classification
            # 6 - Channels (pertaining to each spectrogram)
            # 129  - Freuqency bands for each spectrogram
            # 15 - Time steps

        voxel_number = 0

        # assign directory
        # directory = 'files'
        # iterate over files in that directory
        # for filename in os.listdir(directory):


        directory = (r"C:\Users\Dylan Lasrado\OneDrive\Desktop\613 Project Data\Tensor data")

        for filename in os.listdir(directory):
            # print(filename)
            mat_fname = pjoin(directory, filename)
            Tensor_data_voxel = scipy.io.loadmat(mat_fname)

            data = list(Tensor_data_voxel. items())
            tensor_array = np.array(data, dtype=object)

            # The spectrogram data is captured in a list from one of the elements of tensor_array
            # Shape of tensor_array is (4,2) and all spectrogram data (129x15x6) is stored in tensor_array[3][1]
```

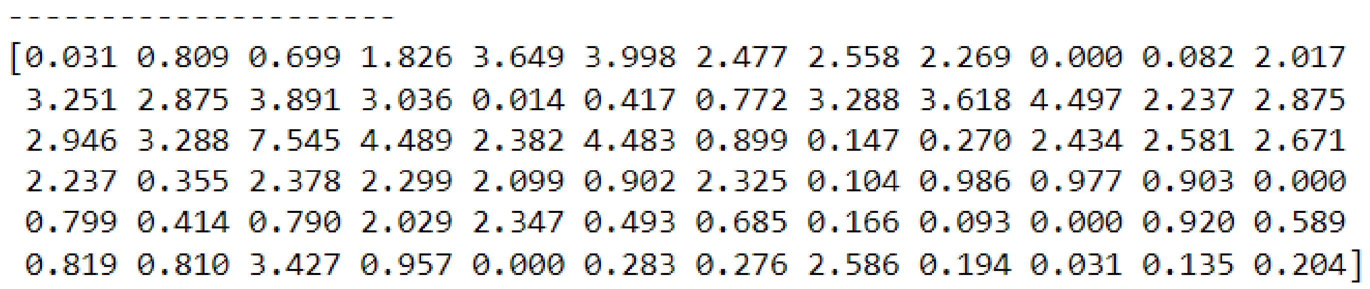


```
    spec_list = list(tensor_array[3][1])
    for i in range(len(spec_list)):
        for j in range(15):
            for k in range(6):
                TENSOR_INPUT_DATA_COMBINED[voxel_number,k,i,j] = spec_list[i][j][k]


    TENSOR_INPUT_DATA_PRINTING = TENSOR_INPUT_DATA_COMBINED[:,(1,2,4,5),:,:]
    TENSOR_INPUT_DATA_MILLING = TENSOR_INPUT_DATA_COMBINED[:,(0,3),:,:]

    voxel_number = voxel_number + 1

# Column F from the excel sheet representing the percentage area porosity is extracted
y_target = pd.read_excel(r"C:\Users\Dylan Lasrado\OneDrive\Desktop\613 Project Data\Excel Sheet\Ground.xlsx", usecols="F")
y_target = np.array(y_target)
y_target = y_target.T
y_target = y_target.ravel()

print("--------------------")
print(TENSOR_INPUT_DATA_COMBINED.shape)
print(TENSOR_INPUT_DATA_COMBINED)
print(TENSOR_INPUT_DATA_PRINTING.shape)
print(TENSOR_INPUT_DATA_PRINTING)
print(TENSOR_INPUT_DATA_MILLING.shape)
print(TENSOR_INPUT_DATA_MILLING)
print("--------------------")
print(y_target)
print("--------------------")
data_order = np.arange(72)
np.random.shuffle(data_order)
print(data_order)
```



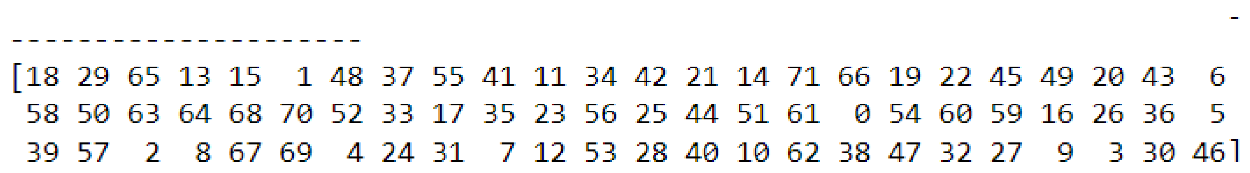
```
---------------------
torch.Size([72, 6, 129, 15])
tensor([[[[3.9972e-01, 2.7345e-01, 1.9331e-01,  ..., 4.0026e-02,
           1.4011e-02, 5.2142e-03],
          [1.5534e-01, 9.1346e-02, 2.7695e-02,  ..., 6.1477e-02,
           3.4975e-02, 5.1146e-02],
          [4.8935e-02, 4.4440e-02, 8.1369e-03,  ..., 2.1981e-02,
           3.2418e-02, 4.0163e-02],
          ...,
          [3.3482e-03, 7.8327e-03, 1.2898e-02,  ..., 1.9981e-02,
           7.0467e-03, 1.7296e-02],
          [7.0541e-03, 7.8069e-03, 1.8567e-02,  ..., 3.4912e-03,
           1.6951e-02, 9.5547e-03],
          [6.0237e-03, 1.9540e-02, 3.1549e-02,  ..., 3.4602e-02,
           4.3607e-02, 5.1279e-03]],
```

CONSOLE OUTPUT: The Spectrogram data form the MATLAB file



```
---------------------
[0.031 0.809 0.699 1.826 3.649 3.998 2.477 2.558 2.269 0.000 0.082 2.017
 3.251 2.875 3.891 3.036 0.014 0.417 0.772 3.288 3.618 4.497 2.237 2.875
 2.946 3.288 7.545 4.489 2.382 4.483 0.899 0.147 0.270 2.434 2.581 2.671
 2.237 0.355 2.378 2.299 2.099 0.902 2.325 0.104 0.986 0.977 0.903 0.000
 0.799 0.414 0.790 2.029 2.347 0.493 0.685 0.166 0.093 0.000 0.920 0.589
 0.819 0.810 3.427 0.957 0.000 0.283 0.276 2.586 0.194 0.031 0.135 0.204]
```

CONSOLE OUTPUT: The 'percentage area porosity' is extracted from the excel sheet as an input

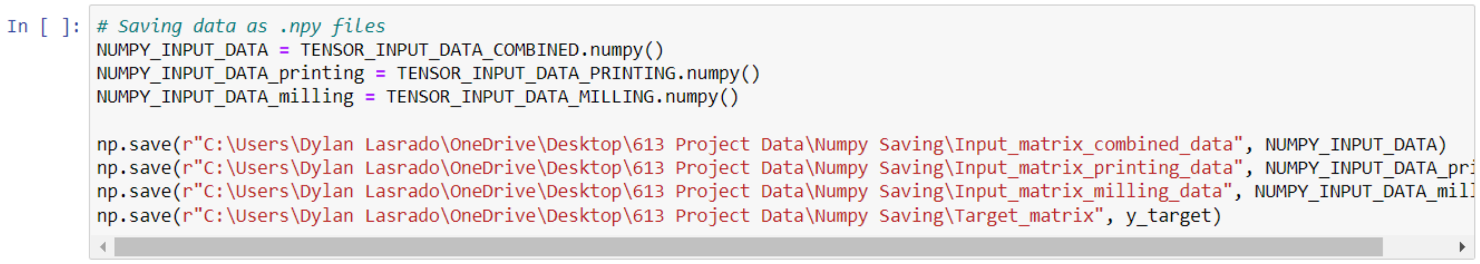


```
                                                                       -
---------------------
[18 29 65 13 15  1 48 37 55 41 11 34 42 21 14 71 66 19 22 45 49 20 43  6
 58 50 63 64 68 70 52 33 17 35 23 56 25 44 51 61  0 54 60 59 16 26 36  5
 39 57  2  8 67 69  4 24 31  7 12 53 28 40 10 62 38 47 32 27  9  3 30 46]
```

CONSOLE OUTPUT: The order of the data after shuffling

STEP 3 – SAVING THE INPUT AS. NPY FILES

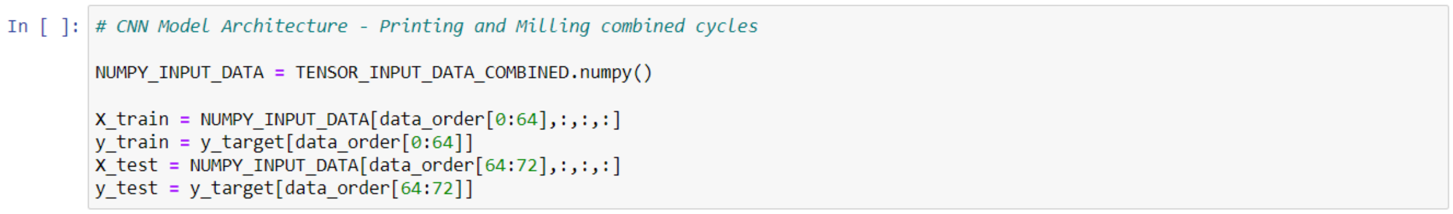
- The input data is stored as .npy files using the code shown below



```
In [ ]: # Saving data as .npy files
        NUMPY_INPUT_DATA = TENSOR_INPUT_DATA_COMBINED.numpy()
        NUMPY_INPUT_DATA_printing = TENSOR_INPUT_DATA_PRINTING.numpy()
        NUMPY_INPUT_DATA_milling = TENSOR_INPUT_DATA_MILLING.numpy()

        np.save(r"C:\Users\Dylan Lasrado\OneDrive\Desktop\613 Project Data\Numpy Saving\Input_matrix_combined_data", NUMPY_INPUT_DATA)
        np.save(r"C:\Users\Dylan Lasrado\OneDrive\Desktop\613 Project Data\Numpy Saving\Input_matrix_printing_data", NUMPY_INPUT_DATA_pri
        np.save(r"C:\Users\Dylan Lasrado\OneDrive\Desktop\613 Project Data\Numpy Saving\Input_matrix_milling_data", NUMPY_INPUT_DATA_mil
        np.save(r"C:\Users\Dylan Lasrado\OneDrive\Desktop\613 Project Data\Numpy Saving\Target_matrix", y_target)
```

STEP 4 – SPLITTING THE DATA INTO TRANING AND TESTING

The spectrogram data and the excel sheet data is split into training and testing data sets. The size of the training data set is 64 and that of the testing data set is 8.



```
In [ ]: # CNN Model Architecture - Printing and Milling combined cycles

        NUMPY_INPUT_DATA = TENSOR_INPUT_DATA_COMBINED.numpy()

        X_train = NUMPY_INPUT_DATA[data_order[0:64],:,:,:]
        y_train = y_target[data_order[0:64]]
        X_test = NUMPY_INPUT_DATA[data_order[64:72],:,:,:]
        y_test = y_target[data_order[64:72]]
```

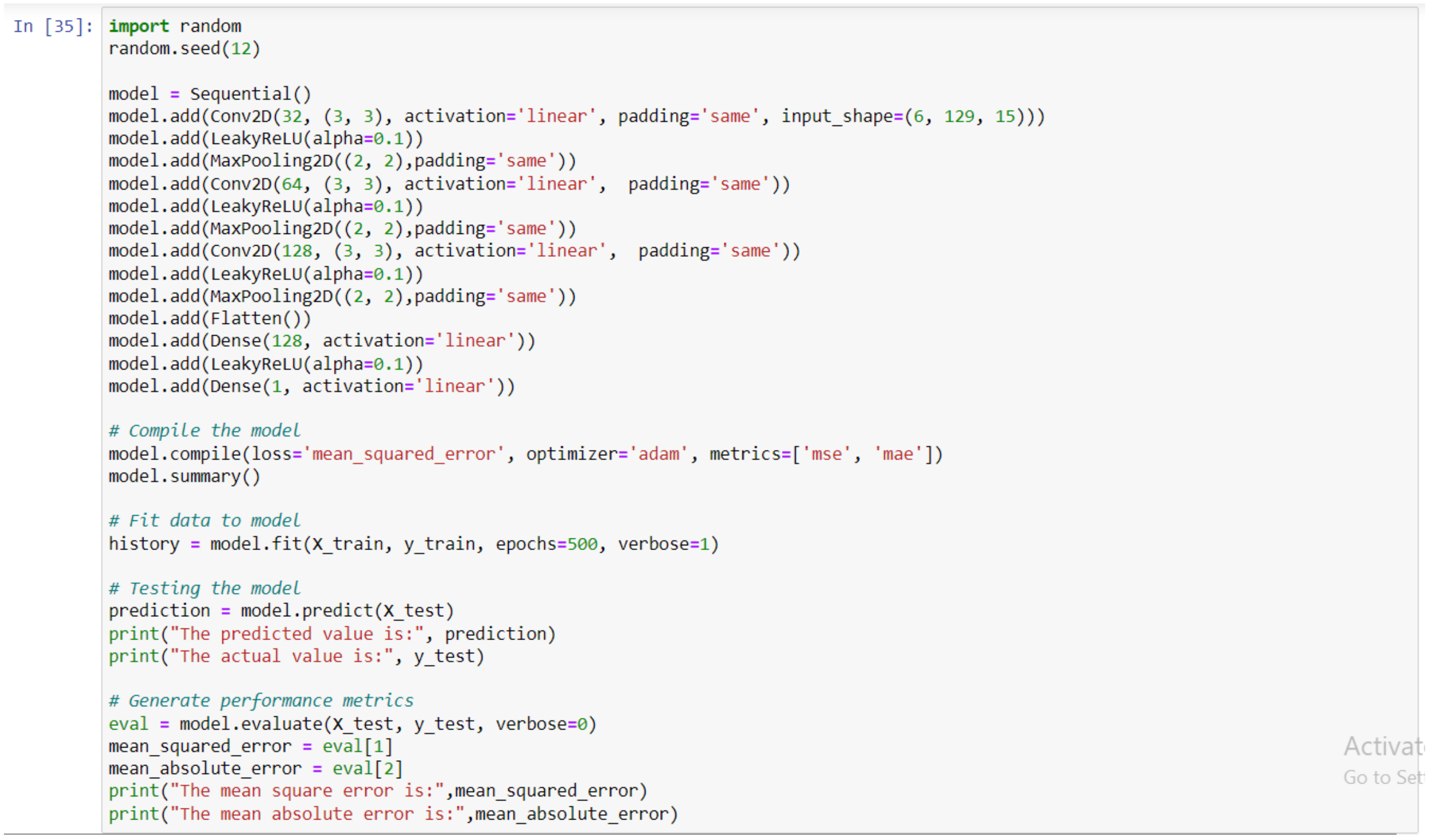
STEP 5 – DEVELOPING A CNN MODEL THAT PREDICTS THE PRECENTAGE AREA OF POROSITY

The CNN model that we developed is a Sequential model. Layers are added to the model using the 'add' function. The model has 3 Conv2D layers which make use of 2-dimensional input. These layers have 32, 64 and 128 nodes respectively. A filter size of 3x3 is used in each of these layers.

Each of the Convolution layers is followed by a Pooling layer. A Max Pooling layer is used which extracts the largest feature from the feature map. The Pooling layer reduces the size of the convolved feature map in order to reduce the computational costs. A Flatten layer which is added between the Conv2D layer, and the Dense layer serves as a connection between these 2 layers. The Dense layer is a commonly used layer for output.

The activation function plays a crucial role in our model in learning and understanding the relationship between the various variables in our model. Our model makes use of a ‘linear’ activation function.

Our model makes use of the regression loss function namely the ‘mean_squared_error’. This loss function determines the average of the squared differences between the actual values and the predicted values. The optimizer used is the Adam optimizer. The performance of the model is judged using 2 different metrics namely Mean squared error (MSE) and mean absolute error (MAE). [10,11,12,13]



```
In [35]: import random
         random.seed(12)

         model = Sequential()
         model.add(Conv2D(32, (3, 3), activation='linear', padding='same', input_shape=(6, 129, 15)))
         model.add(LeakyReLU(alpha=0.1))
         model.add(MaxPooling2D((2, 2),padding='same'))
         model.add(Conv2D(64, (3, 3), activation='linear',  padding='same'))
         model.add(LeakyReLU(alpha=0.1))
         model.add(MaxPooling2D((2, 2),padding='same'))
         model.add(Conv2D(128, (3, 3), activation='linear',  padding='same'))
         model.add(LeakyReLU(alpha=0.1))
         model.add(MaxPooling2D((2, 2),padding='same'))
         model.add(Flatten())
         model.add(Dense(128, activation='linear'))
         model.add(LeakyReLU(alpha=0.1))
         model.add(Dense(1, activation='linear'))

         # Compile the model
         model.compile(loss='mean_squared_error', optimizer='adam', metrics=['mse', 'mae'])
         model.summary()

         # Fit data to model
         history = model.fit(X_train, y_train, epochs=500, verbose=1)

         # Testing the model
         prediction = model.predict(X_test)
         print("The predicted value is:", prediction)
         print("The actual value is:", y_test)

         # Generate performance metrics
         eval = model.evaluate(X_test, y_test, verbose=0)
         mean_squared_error = eval[1]
         mean_absolute_error = eval[2]
         print("The mean square error is:",mean_squared_error)
         print("The mean absolute error is:",mean_absolute_error)
```

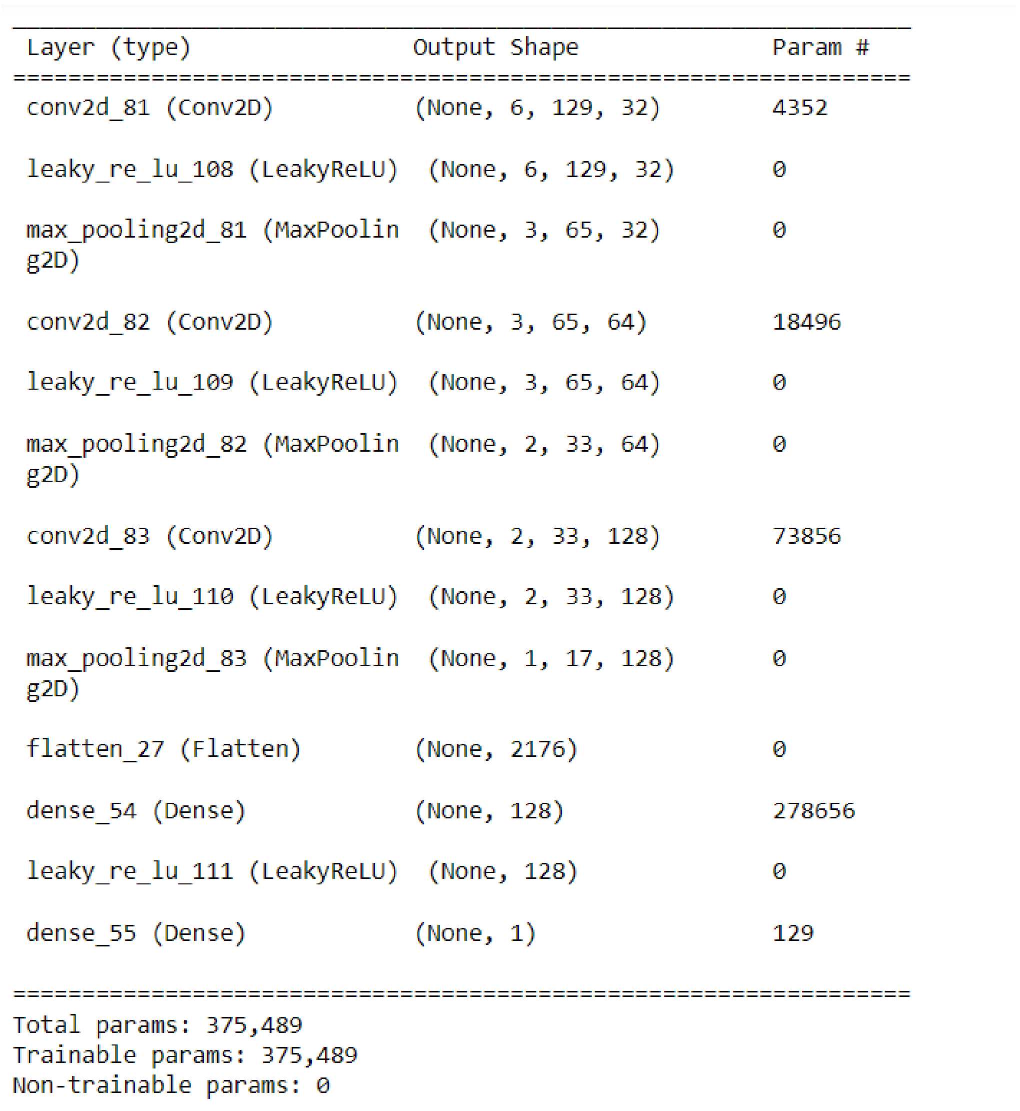
STEP 6 – INTERPRETING THE OUTPUT OF THE MODEL

The first Convolution layer of the model uses 32 filters of dimensions 3x3. The output has dimensions 6x129x32. A Max Pooling layer is then used which uses filters of dimensions 2x2 and the resulting output has dimensions 3x65x32.

The second Convolution layer of the model uses 64 filters of dimensions 3x3. The output has dimensions 3x65x64. A Max Pooling layer is then used which uses filters of dimensions 2x2 and the resulting output has dimensions 2x33x64.

The third Convolution layer of the model uses 128 filters of dimensions 3x3. The output has dimensions 2x33x128. A Max Pooling layer is then used which uses filters of dimensions 2x2 and the resulting output has dimensions 1x17x128. The next layer is a fully connected Convolutional network which makes use of 128 filters. The output has dimensions of 2176 (17 * 128). The final layer is a linear output layer.

A summary of the model is as shown below.

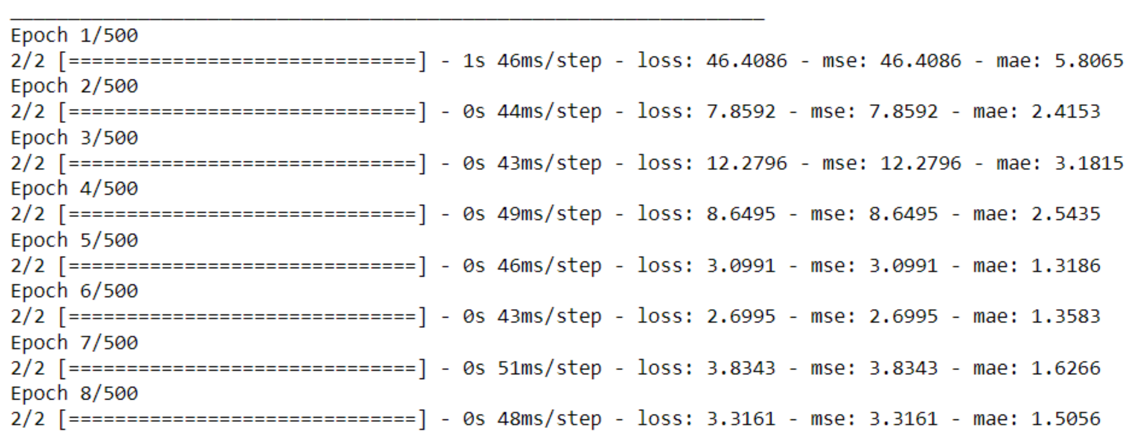


```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_81 (Conv2D)          (None, 6, 129, 32)        4352

 leaky_re_lu_108 (LeakyReLU)  (None, 6, 129, 32)       0

 max_pooling2d_81 (MaxPoolin  (None, 3, 65, 32)        0
 g2D)

 conv2d_82 (Conv2D)          (None, 3, 65, 64)         18496

 leaky_re_lu_109 (LeakyReLU)  (None, 3, 65, 64)        0

 max_pooling2d_82 (MaxPoolin  (None, 2, 33, 64)        0
 g2D)

 conv2d_83 (Conv2D)          (None, 2, 33, 128)        73856

 leaky_re_lu_110 (LeakyReLU)  (None, 2, 33, 128)       0

 max_pooling2d_83 (MaxPoolin  (None, 1, 17, 128)       0
 g2D)

 flatten_27 (Flatten)        (None, 2176)              0

 dense_54 (Dense)            (None, 128)               278656

 leaky_re_lu_111 (LeakyReLU)  (None, 128)              0

 dense_55 (Dense)            (None, 1)                 129

=================================================================
Total params: 375,489
Trainable params: 375,489
Non-trainable params: 0
```

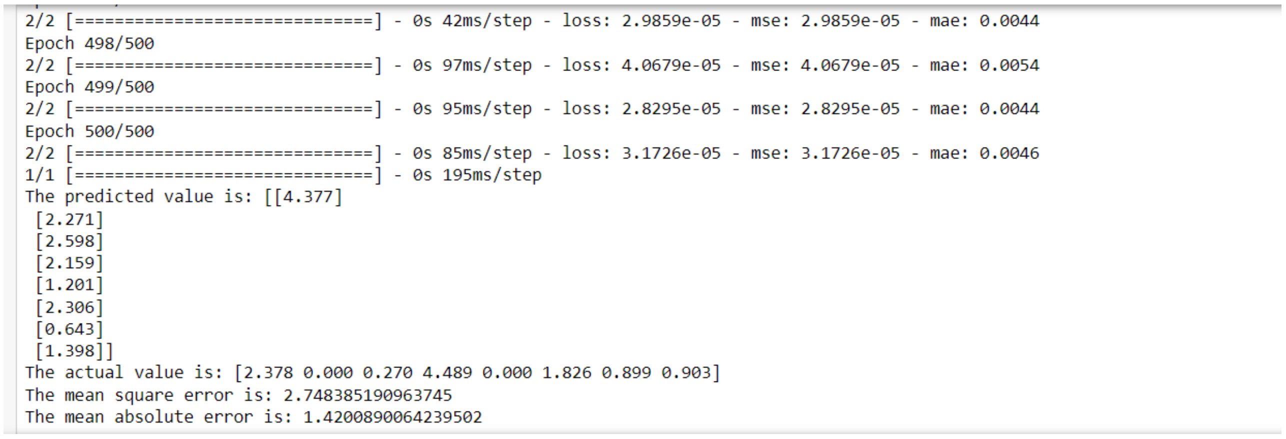
The model makes use of 500 epochs. It is observed that the MSE and the MAE values gradually decrease as the number of epochs increase.

The predicted values and the actual values of the percentage area porosity is shown below for the test data set which has 8 values. The mean square error for the model is 2.748 and the mean average error for the model is 1.42. The model is found to be moderately accurate and predicts close to the actual value. The predictions of this model can further be improved by increasing the size of the data set on which the model can be trained and tested.

Note: These values predicted be this model will change every time the model is run because the input data is shuffled every time, we run the model.



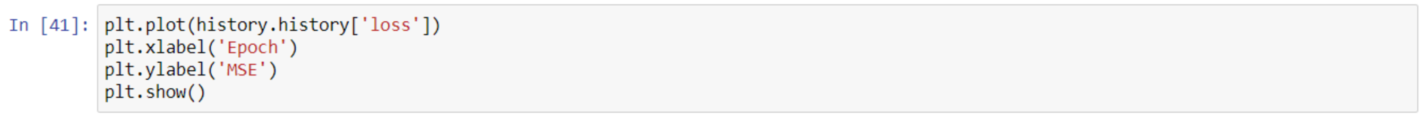
```
_________________________________________________________________
Epoch 1/500
2/2 [==============================] - 1s 46ms/step - loss: 46.4086 - mse: 46.4086 - mae: 5.8065
Epoch 2/500
2/2 [==============================] - 0s 44ms/step - loss: 7.8592 - mse: 7.8592 - mae: 2.4153
Epoch 3/500
2/2 [==============================] - 0s 43ms/step - loss: 12.2796 - mse: 12.2796 - mae: 3.1815
Epoch 4/500
2/2 [==============================] - 0s 49ms/step - loss: 8.6495 - mse: 8.6495 - mae: 2.5435
Epoch 5/500
2/2 [==============================] - 0s 46ms/step - loss: 3.0991 - mse: 3.0991 - mae: 1.3186
Epoch 6/500
2/2 [==============================] - 0s 43ms/step - loss: 2.6995 - mse: 2.6995 - mae: 1.3583
Epoch 7/500
2/2 [==============================] - 0s 51ms/step - loss: 3.8343 - mse: 3.8343 - mae: 1.6266
Epoch 8/500
2/2 [==============================] - 0s 48ms/step - loss: 3.3161 - mse: 3.3161 - mae: 1.5056
```



```
2/2 [==============================] - 0s 42ms/step - loss: 2.9859e-05 - mse: 2.9859e-05 - mae: 0.0044
Epoch 498/500
2/2 [==============================] - 0s 97ms/step - loss: 4.0679e-05 - mse: 4.0679e-05 - mae: 0.0054
Epoch 499/500
2/2 [==============================] - 0s 95ms/step - loss: 2.8295e-05 - mse: 2.8295e-05 - mae: 0.0044
Epoch 500/500
2/2 [==============================] - 0s 85ms/step - loss: 3.1726e-05 - mse: 3.1726e-05 - mae: 0.0046
1/1 [==============================] - 0s 195ms/step
The predicted value is: [[4.377]
 [2.271]
 [2.598]
 [2.159]
 [1.201]
 [2.306]
 [0.643]
 [1.398]]
The actual value is: [2.378 0.000 0.270 4.489 0.000 1.826 0.899 0.903]
The mean square error is: 2.748385190963745
The mean absolute error is: 1.4200890064239502
```

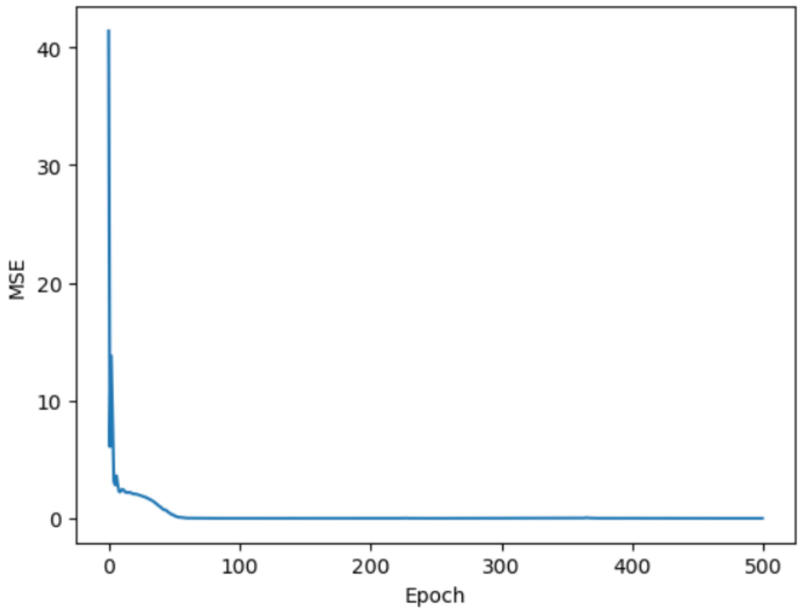
STEP 6 – SUMMARIZING THE PLOT

A plot of MSE vs the number of epochs was plotted.



```
In [41]: plt.plot(history.history['loss'])
         plt.xlabel('Epoch')
         plt.ylabel('MSE')
         plt.show()
```

From the plot it can be observed that the MSE values converge as the number of epochs increase.



3.2 CNN MODEL WITH 2 CONVOLUTION LAYERS (MODEL2)

A CNN model which predicts the percentage area porosity. This model is similar to the classification-based CNN model provided to us initially with some modifications.

STEP 1 TO 4 – which involve invoking the libraries and extracting the input from the MATLAB files and the excel sheet are the same for this model when compared to the previous model. Hence these steps have not been shown again
.

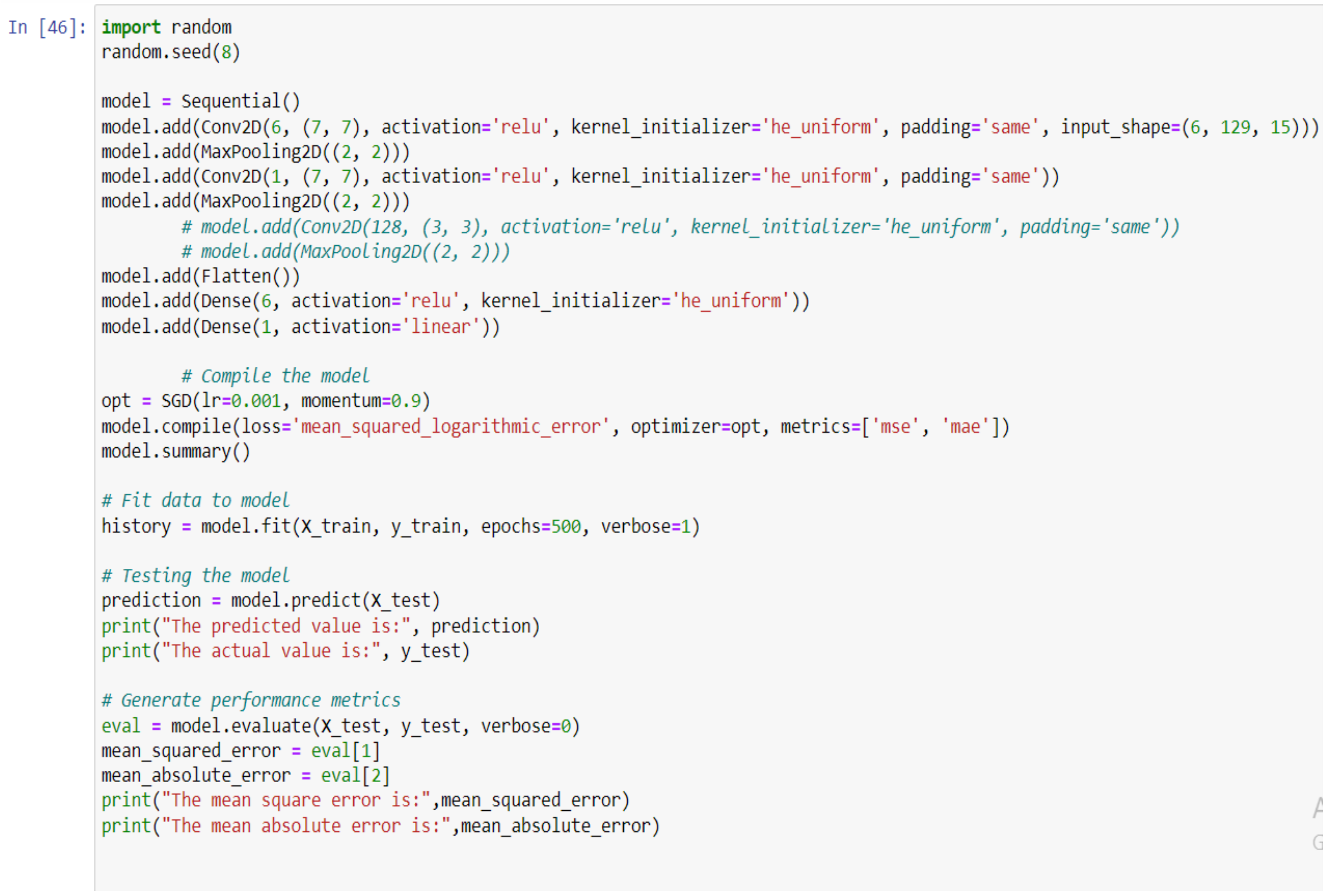
STEP 5 - DEVELOPING A CNN MODEL THAT PREDICTS THE PERCENTAGE AREA OF POROSITY

The CNN model that we developed is a Sequential model. Layers are added to the model using the ‘add’ function. The model has 2 Conv2D layers which make use of 2-dimensional input. These layers have 6 and 1 nodes respectively.   A filter size of 2x2 is used in each of these layers.

Each of the Convolution layers is followed by a Pooling layer. A Max Pooling layer is used which extracts the largest feature from the feature map. The Pooling layer reduces the size of the convolved feature map in order to reduce the computational costs. A Flatten layer which is added between the Conv2D layer, and the Dense layer serves as a connection between these 2 layers. The Dense layer is a commonly used layer for output.

The activation function plays a crucial role in our model in learning and understanding the relationship between the various variables in our model. Our model makes use of a ‘linear’ activation function.

This model makes use of the regression loss function namely the ‘mean_squared_logarithmic_error’. The optimizer used is the Stochastic gradient decent. The performance of the model is judged using 2 different metrics namely Mean squared error (MSE) and mean absolute error (MAE).



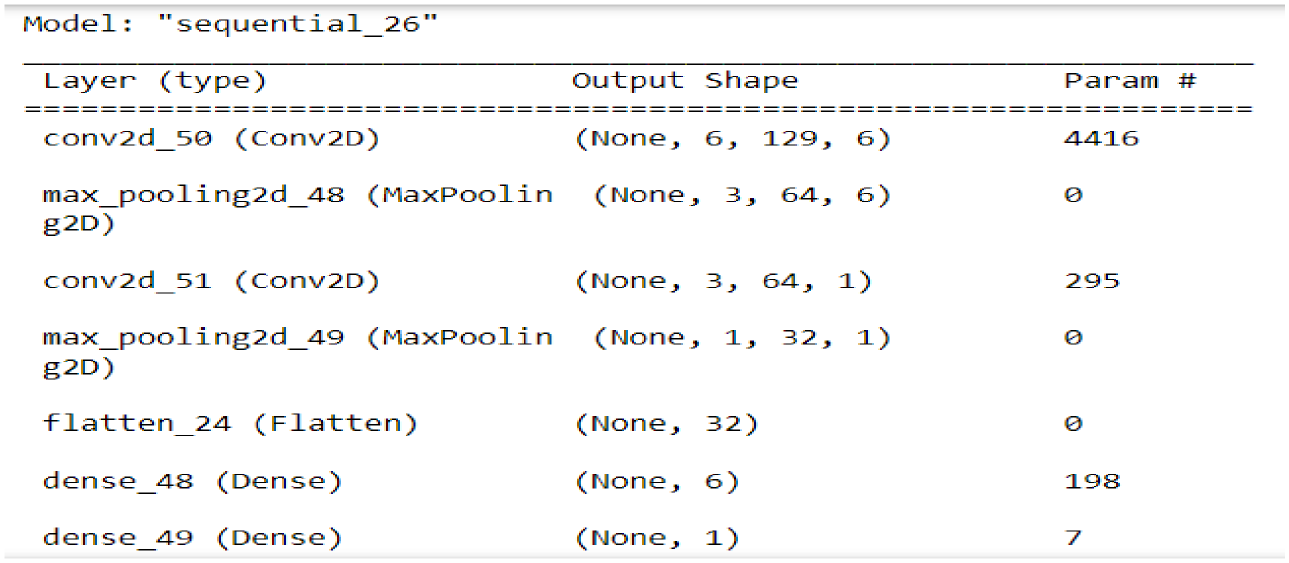
```
In [46]: import random
         random.seed(8)

         model = Sequential()
         model.add(Conv2D(6, (7, 7), activation='relu', kernel_initializer='he_uniform', padding='same', input_shape=(6, 129, 15)))
         model.add(MaxPooling2D((2, 2)))
         model.add(Conv2D(1, (7, 7), activation='relu', kernel_initializer='he_uniform', padding='same'))
         model.add(MaxPooling2D((2, 2)))
                 # model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
                 # model.add(MaxPooling2D((2, 2)))
         model.add(Flatten())
         model.add(Dense(6, activation='relu', kernel_initializer='he_uniform'))
         model.add(Dense(1, activation='linear'))

                 # Compile the model
         opt = SGD(lr=0.001, momentum=0.9)
         model.compile(loss='mean_squared_logarithmic_error', optimizer=opt, metrics=['mse', 'mae'])
         model.summary()

         # Fit data to model
         history = model.fit(X_train, y_train, epochs=500, verbose=1)

         # Testing the model
         prediction = model.predict(X_test)
         print("The predicted value is:", prediction)
         print("The actual value is:", y_test)

         # Generate performance metrics
         eval = model.evaluate(X_test, y_test, verbose=0)
         mean_squared_error = eval[1]
         mean_absolute_error = eval[2]
         print("The mean square error is:",mean_squared_error)
         print("The mean absolute error is:",mean_absolute_error)
```

STEP 6 – INTERPRETING THE OUTOUT THE MODEL

The first Convolution layer of the model uses 32 filters of dimensions 7x7. The output has dimensions 6x129x6. A Max Pooling layer is then used which uses filters of dimensions 2x2 and the resulting output has dimensions 3x64x6.

The second Convolution layer of the model uses 1 filter of dimensions 7x7. The output has dimensions 3x64x1. A Max Pooling layer is then used which uses filters of dimensions 2x2 and the resulting output has dimensions 1x32x1. next layer is a fully connected Convolutional network while the final layer is a linear output layer.

A summary of the model is as shown below.

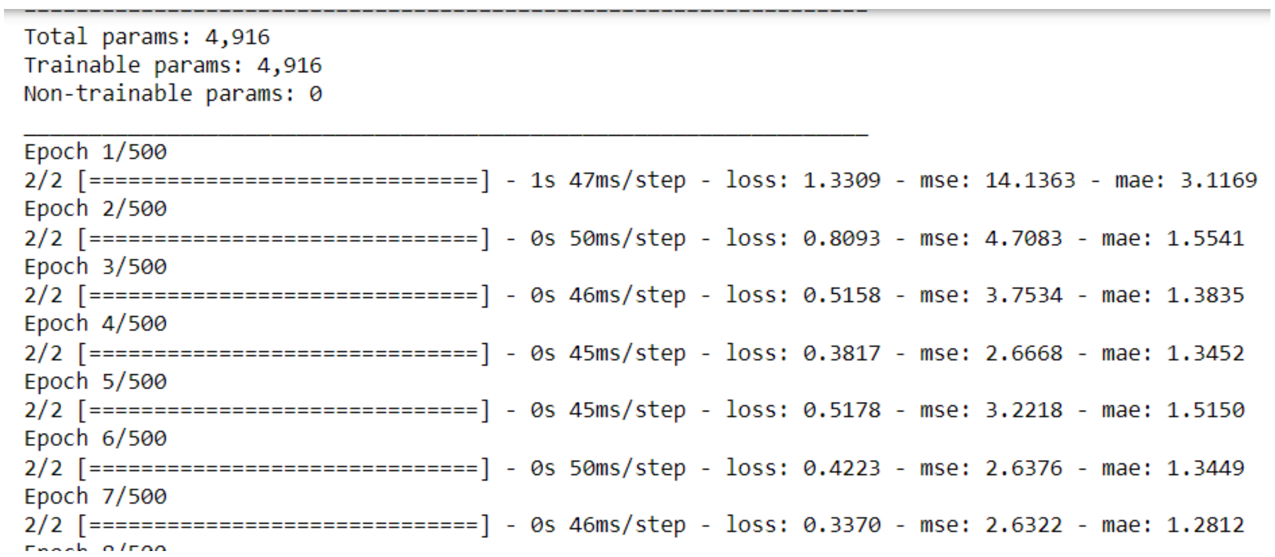


```
Model: "sequential_26"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_50 (Conv2D)          (None, 6, 129, 6)         4416

 max_pooling2d_48 (MaxPoolin  (None, 3, 64, 6)         0
 g2D)

 conv2d_51 (Conv2D)          (None, 3, 64, 1)          295

 max_pooling2d_49 (MaxPoolin  (None, 1, 32, 1)         0
 g2D)

 flatten_24 (Flatten)        (None, 32)                0

 dense_48 (Dense)            (None, 6)                 198

 dense_49 (Dense)            (None, 1)                 7
```

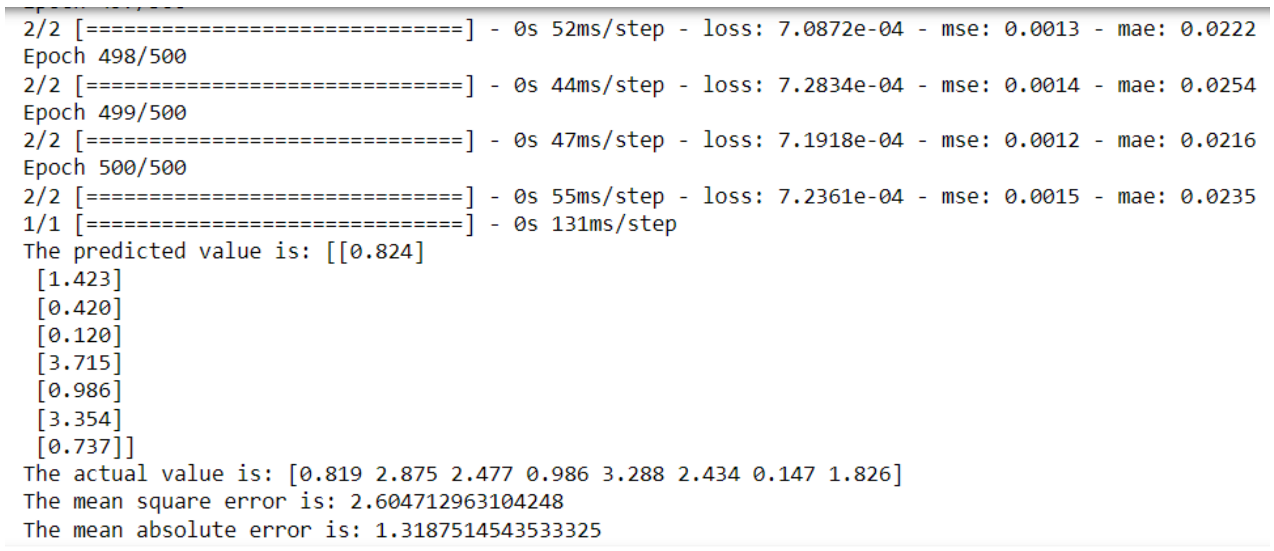
The model makes use of 500 epochs. It is observed that the MSE and the MAE values gradually decrease as the number of epochs increase. The predicted values and the actual values of the percentage area porosity is shown below for the test data set which has 8 values. The mean square error for the model is 2.6 and the mean average error for the model is 1.31.

The model is found to be moderately accurate and predicts close to the actual value. The predictions of this model can further be improved by increasing the size of the data set on which the model can be trained and tested.

Note: These values predicted be this model will change every time the model is run because the input data is shuffled every time, we run the model.



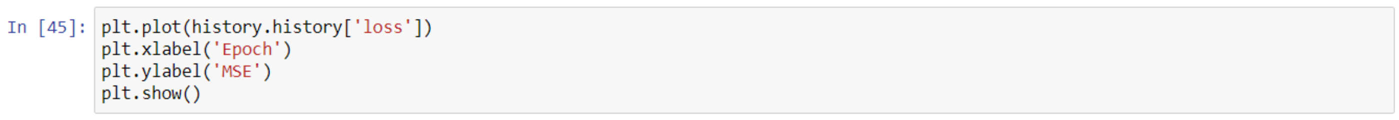
```
Total params: 4,916
Trainable params: 4,916
Non-trainable params: 0

_________________________________________________________________
Epoch 1/500
2/2 [==============================] - 1s 47ms/step - loss: 1.3309 - mse: 14.1363 - mae: 3.1169
Epoch 2/500
2/2 [==============================] - 0s 50ms/step - loss: 0.8093 - mse: 4.7083 - mae: 1.5541
Epoch 3/500
2/2 [==============================] - 0s 46ms/step - loss: 0.5158 - mse: 3.7534 - mae: 1.3835
Epoch 4/500
2/2 [==============================] - 0s 45ms/step - loss: 0.3817 - mse: 2.6668 - mae: 1.3452
Epoch 5/500
2/2 [==============================] - 0s 45ms/step - loss: 0.5178 - mse: 3.2218 - mae: 1.5150
Epoch 6/500
2/2 [==============================] - 0s 50ms/step - loss: 0.4223 - mse: 2.6376 - mae: 1.3449
Epoch 7/500
2/2 [==============================] - 0s 46ms/step - loss: 0.3370 - mse: 2.6322 - mae: 1.2812
```



```
2/2 [==============================] - 0s 52ms/step - loss: 7.0872e-04 - mse: 0.0013 - mae: 0.0222
Epoch 498/500
2/2 [==============================] - 0s 44ms/step - loss: 7.2834e-04 - mse: 0.0014 - mae: 0.0254
Epoch 499/500
2/2 [==============================] - 0s 47ms/step - loss: 7.1918e-04 - mse: 0.0012 - mae: 0.0216
Epoch 500/500
2/2 [==============================] - 0s 55ms/step - loss: 7.2361e-04 - mse: 0.0015 - mae: 0.0235
1/1 [==============================] - 0s 131ms/step
The predicted value is: [[0.824]
 [1.423]
 [0.420]
 [0.120]
 [3.715]
 [0.986]
 [3.354]
 [0.737]]
The actual value is: [0.819 2.875 2.477 0.986 3.288 2.434 0.147 1.826]
The mean square error is: 2.604712963104248
The mean absolute error is: 1.3187514543533325
```

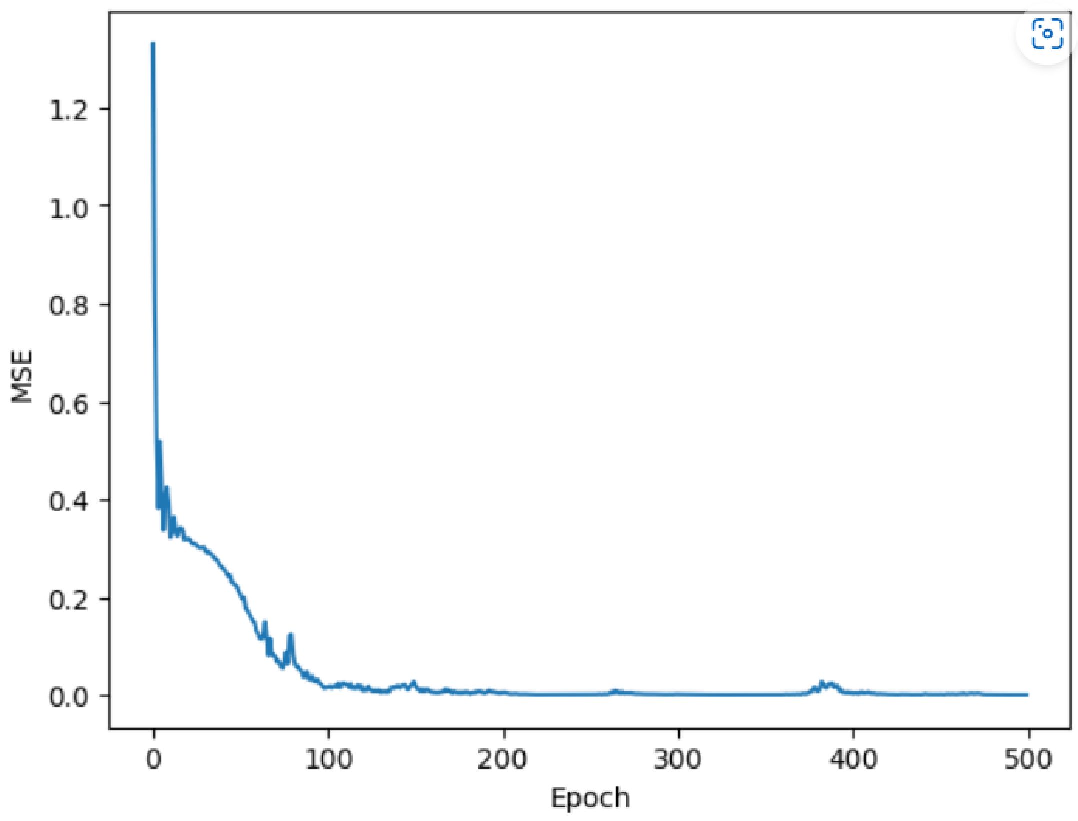
STEP 6 – SUMMARIZING THE PLOT

A plot of MSE vs the number of epochs was plotted.



```
In [45]: plt.plot(history.history['loss'])
         plt.xlabel('Epoch')
         plt.ylabel('MSE')
         plt.show()
```

From the plot it can be observed that the MSE values converge as the number of epochs increase.

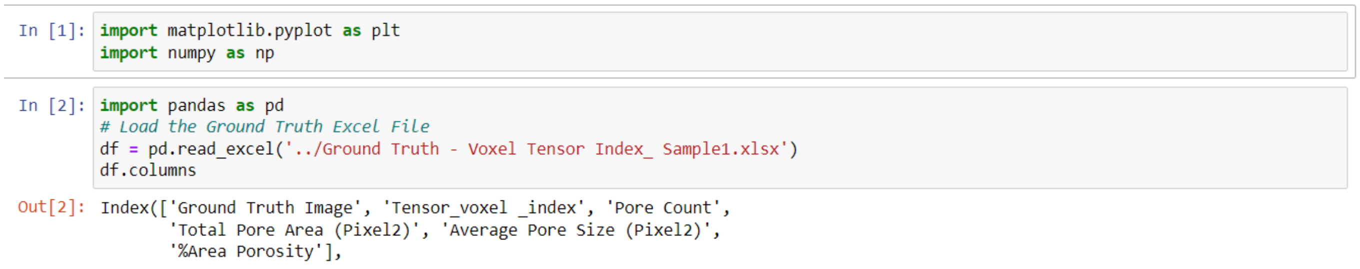


3.3 SIMPLE LINEAR REGRESSION (MODEL 3)

Simple Linear Regression model to predict the percentage area of porosity

STEP 1 – IMPORTING REQUIRED LIBRARIES AND DATA SET

We first import the required libraries and packages in our code.



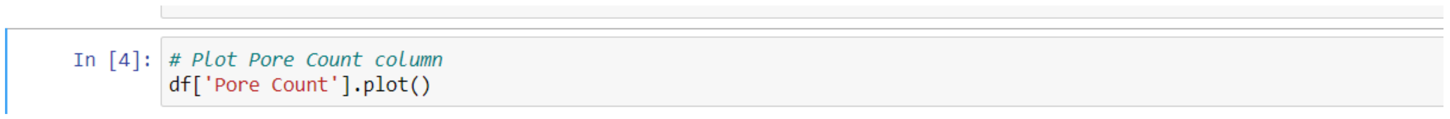
```
In [1]: import matplotlib.pyplot as plt
        import numpy as np
```

```
In [2]: import pandas as pd
        # Load the Ground Truth Excel File
        df = pd.read_excel('../Ground Truth - Voxel Tensor Index_ Sample1.xlsx')
        df.columns
```

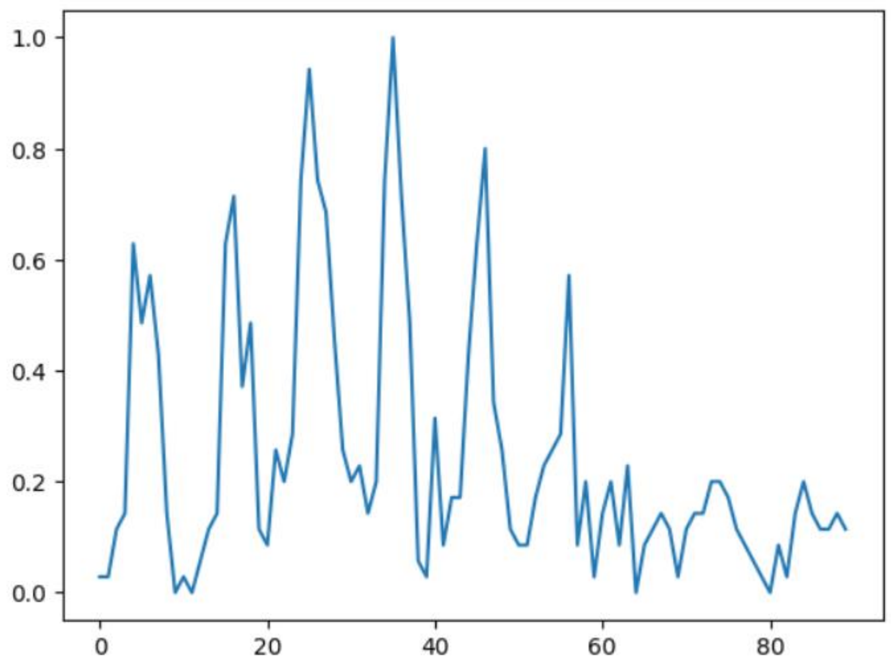
```
Out[2]: Index(['Ground Truth Image', 'Tensor_voxel _index', 'Pore Count',
               'Total Pore Area (Pixel2)', 'Average Pore Size (Pixel2)',
               '%Area Porosity'],
```

STEP 2 – VISUALIZING THE VARIATION OF THE PORE COUNT WITH DIFFERENT VOXEL

A plot for the pore count for different voxels is as shown below.

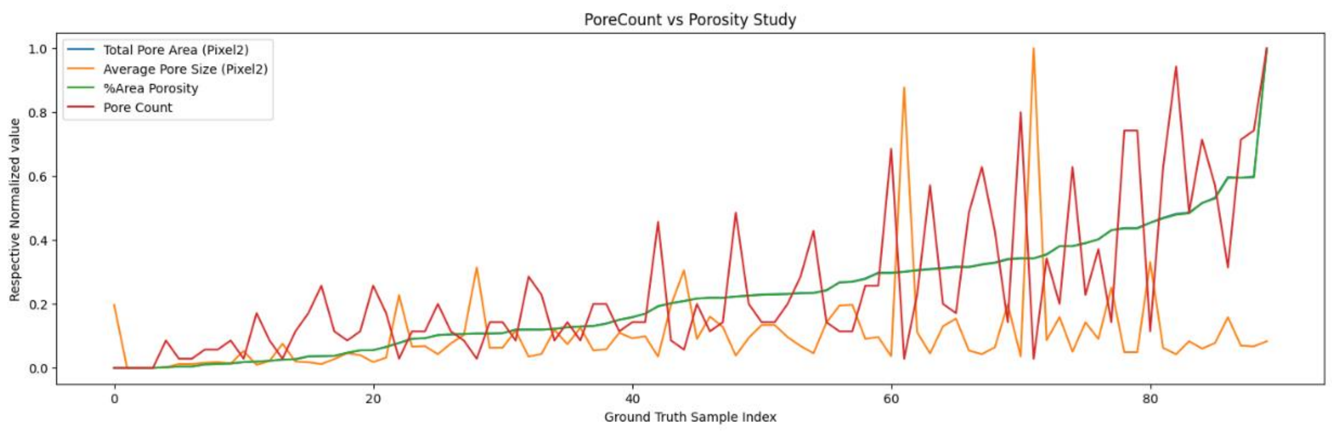


```
In [4]: # Plot Pore Count column
        df['Pore Count'].plot()
```

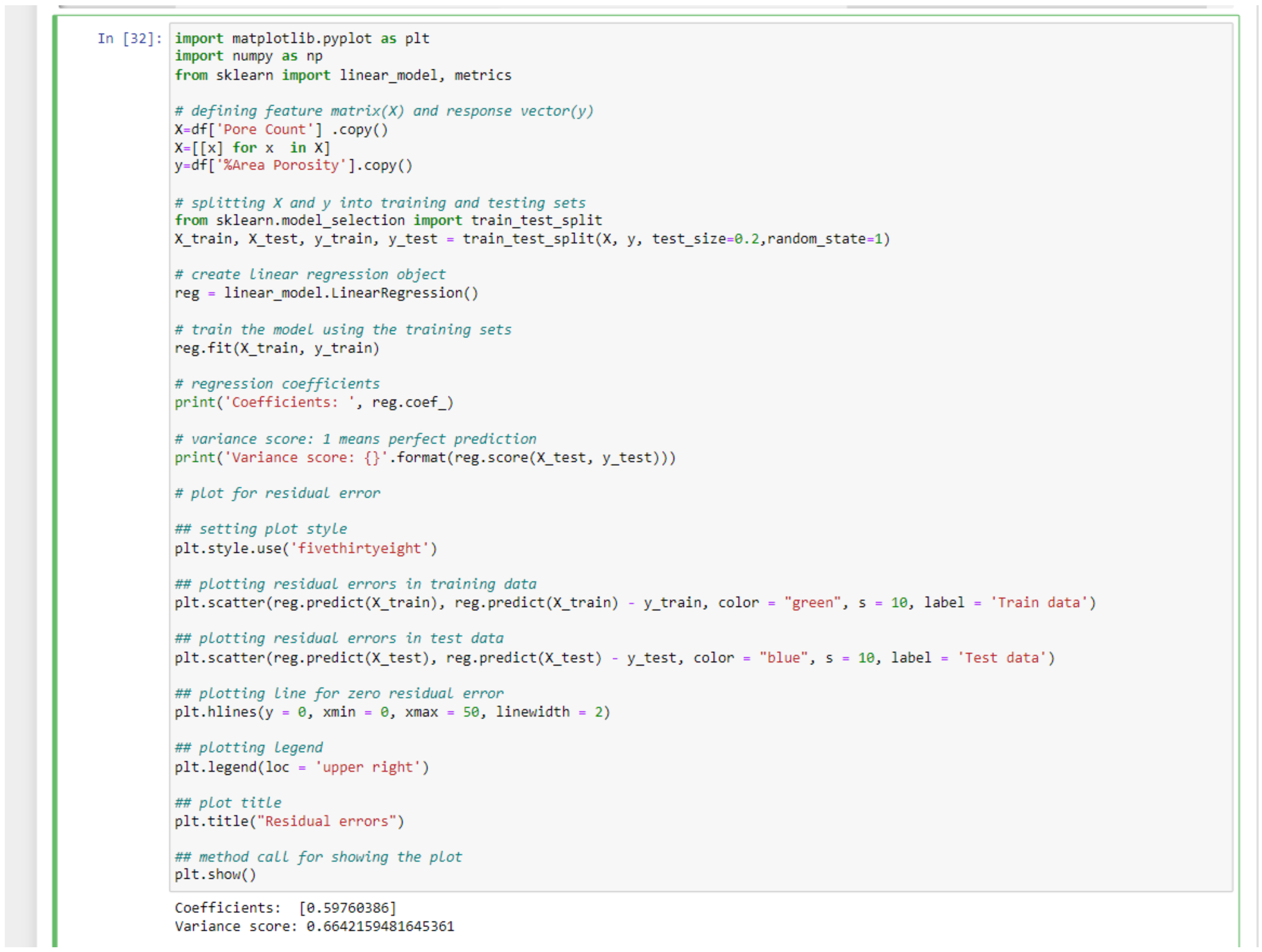


STEP 3 – FINDING THE RELATION BETWEEN THE PORE COUNT AND % AREA POROSITY

From the image we can observe that as the pore count is increasing the % area porosity is increasing with the sample index. There seems to be a slight relationship between pore count and the Average pore size (pixel)^2. No other relationship with any other parameters is observed through the plots.



STEP 4 – DEVELOPING A SIMPLE REGRESSION MODEL

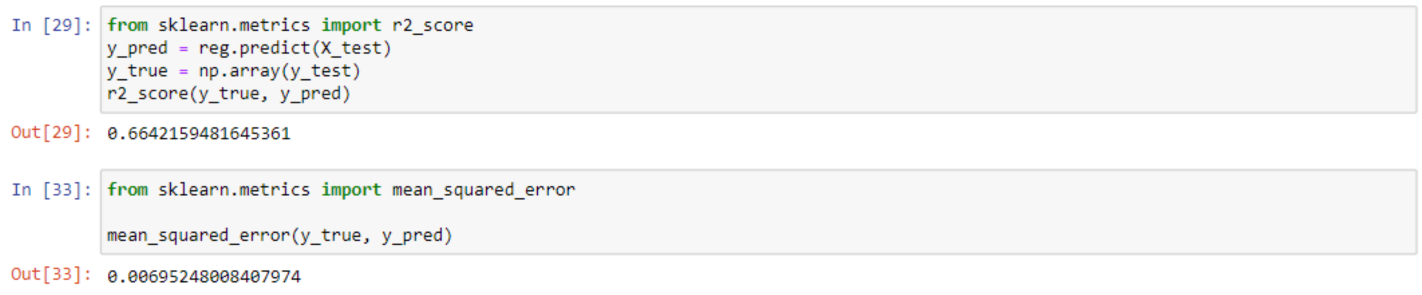


```
In [32]: import matplotlib.pyplot as plt
         import numpy as np
         from sklearn import linear_model, metrics

         # defining feature matrix(X) and response vector(y)
         X=df['Pore Count'] .copy()
         X=[[x] for x  in X]
         y=df['%Area Porosity'].copy()

         # splitting X and y into training and testing sets
         from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_state=1)

         # create linear regression object
         reg = linear_model.LinearRegression()

         # train the model using the training sets
         reg.fit(X_train, y_train)

         # regression coefficients
         print('Coefficients: ', reg.coef_)

         # variance score: 1 means perfect prediction
         print('Variance score: {}'.format(reg.score(X_test, y_test)))

         # plot for residual error

         ## setting plot style
         plt.style.use('fivethirtyeight')

         ## plotting residual errors in training data
         plt.scatter(reg.predict(X_train), reg.predict(X_train) - y_train, color = "green", s = 10, label = 'Train data')

         ## plotting residual errors in test data
         plt.scatter(reg.predict(X_test), reg.predict(X_test) - y_test, color = "blue", s = 10, label = 'Test data')

         ## plotting line for zero residual error
         plt.hlines(y = 0, xmin = 0, xmax = 50, linewidth = 2)

         ## plotting legend
         plt.legend(loc = 'upper right')

         ## plot title
         plt.title("Residual errors")

         ## method call for showing the plot
         plt.show()

         Coefficients:  [0.59760386]
         Variance score: 0.6642159481645361
```

A simple linear regression model is developed using the 'pore count' as the predictor in order to predict the % area porosity. The spectrogram data was not used as input for this model.

STEP 5 – COMPUTING MEAN SQUARE AND R- SQUARE VALUES

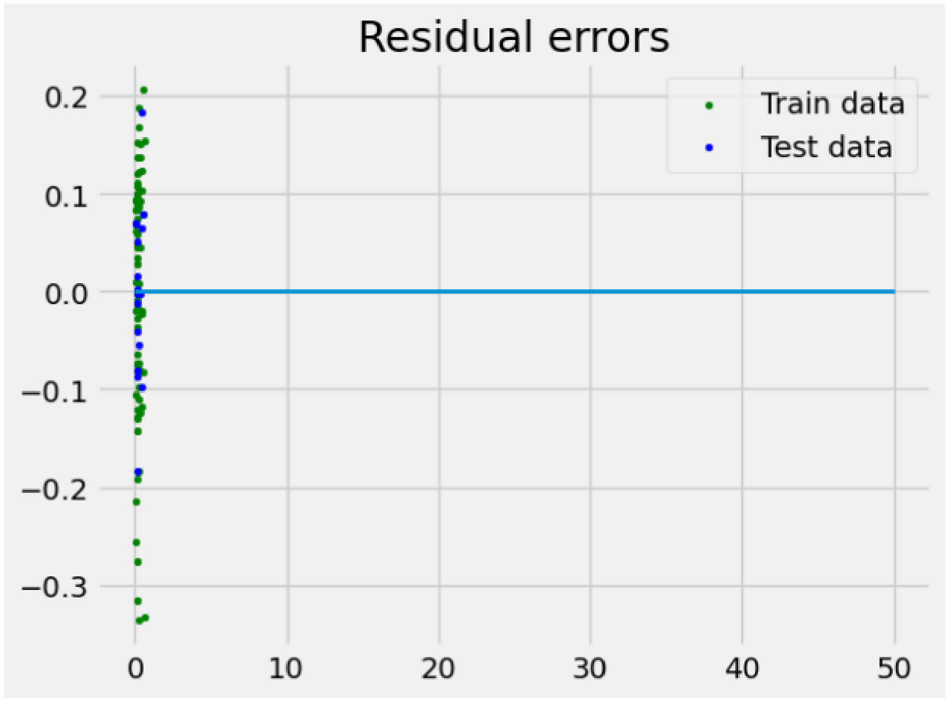
The Mean Square Error for the model is found to be 0.006962 and the R –square value is found to be 0.664.



```
In [29]: from sklearn.metrics import r2_score
         y_pred = reg.predict(X_test)
         y_true = np.array(y_test)
         r2_score(y_true, y_pred)

Out[29]: 0.6642159481645361

In [33]: from sklearn.metrics import mean_squared_error

         mean_squared_error(y_true, y_pred)

Out[33]: 0.00695248008407974
```

STEP 6 – SUMMARIZING THE PLOT AND INTREPRETING THE RESULTS

Plotted below are the Residual errors for the training and the testing data.



From the above results we can infer that the model has pretty good accuracy but the R- Square is fairly low. The model is able to explain about 66% of the variation in the data. From the plot of residual error, we observe that our data is highly scattered and have high outliers.

DRAW BACKS USING THE SIMPLE LINEAR REGRESSION MODEL

There are certain drawbacks with using the Simple Linear Regression model. This model uses only one predictor (number of pores) to predict the percentage area of porosity in the sample. This model does not use spectrogram data as input.

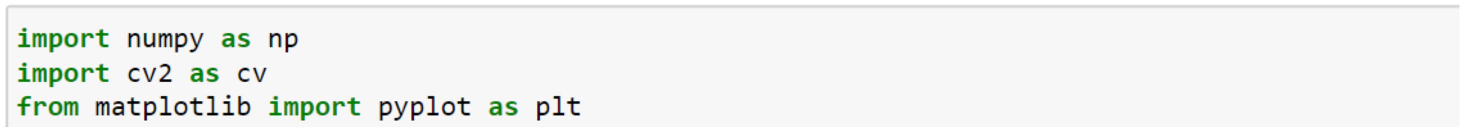
As every workpiece is unique, we can't rely only on the pore count to determine the % area of porosity. The number of pores will also have to be manually measured for each sample and input into the model for it to be able to predict the percentage area of porosity.

3.4 EDGE DETECTION MODEL

Edge Detection model which highlights the edges of pores

STEP 1 – IMPORT REQUIRED LIBRAIRES

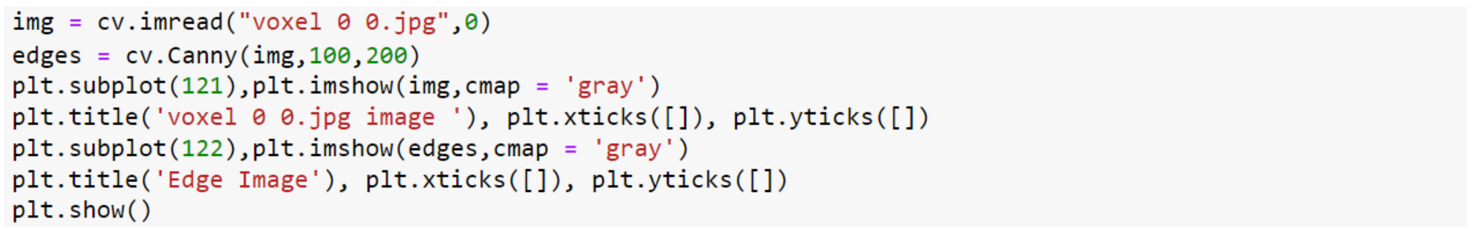
We first import the required libraries and packages in our code.



```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
```

STEP 2 –EDGE DETECTION ALGORITHM USING B/W IMAGES

We developed a machine learning model which imports one image at a time and then uses the Edge detection model to highlight the edges of the pores.



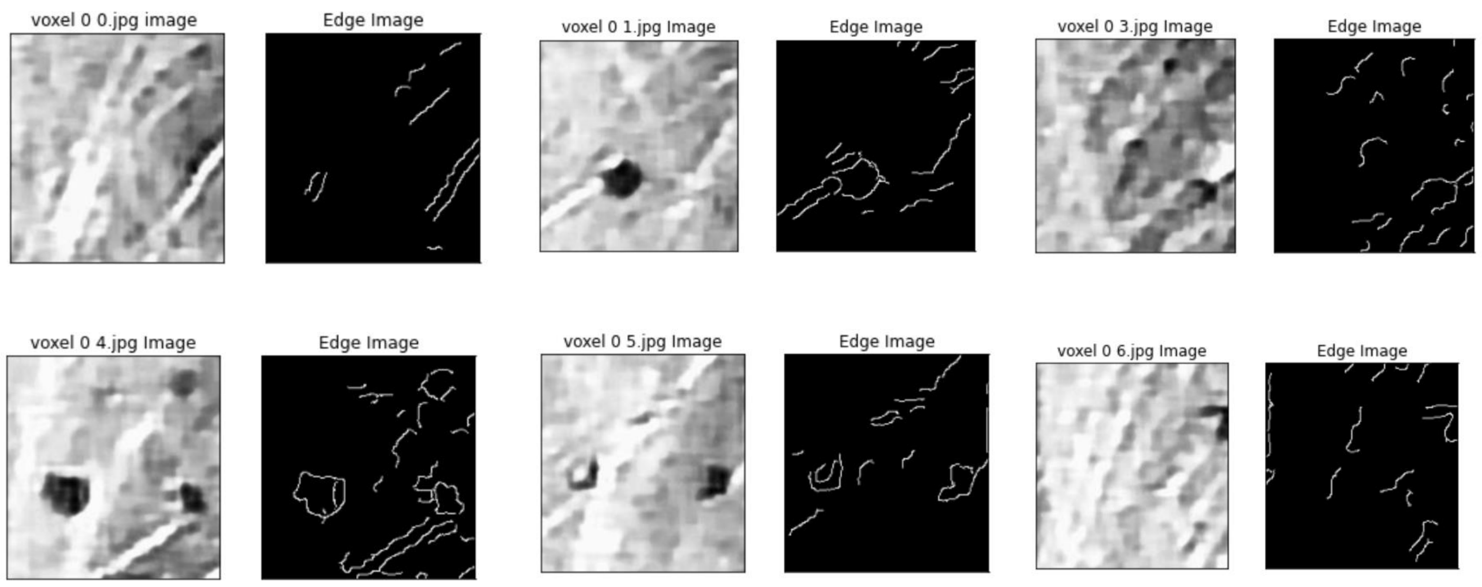
```
img = cv.imread("voxel 0 0.jpg",0)
edges = cv.Canny(img,100,200)
plt.subplot(121),plt.imshow(img,cmap = 'gray')
plt.title('voxel 0 0.jpg image '), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(edges,cmap = 'gray')
plt.title('Edge Image'), plt.xticks([]), plt.yticks([])
plt.show()
```

STEP 3 – INTREPRETING THE RESULTS

From the below images it is evident that model is not very successful identifying the pores as resolution of the images is very less. Surfaces which are uneven or even have a small gradient change are highlighted as an edge which makes it difficult to differentiate other defects with the pore. If we decide to move forward with this model to devise the % Area Porosity, there are high chances of deceptive results.

Hence it is important to consider the spectrogram data and use other machine learning techniques that would give us more promising results.

The results that we obtained on using the edge detection model are as shown below.



4. Comparison Of Models

PRAMETER/MODELS	CNN MODEL 1	CNN MODEL 2	SIMPLE LINEAR REGRESSION MODE
INPUT USED	SPECTROGRAM DATA AND % AREA POROSITY	SPECTOGRAM DATA AND % AREA POROSITY	PORE COUNT (EXCEL SHEET)
NUMBER OF CONVOLUTIONAL LAYERS	3	2	N/A
FILTER SIZE OF CONVOLUTION LAYERS	3*3	7*7	N/A
LOSS FUNCTION	MEAN SQUARE ERROR	MEAN SQUARED LOGRITHMIC ERROR	N/A
OPTIMIZER	ADAM	STOCHASTIC GRADIENT DESCENT	N/A
MEAN SQUARE ERROR	2.74	2.6	0.06962
MEAN ABSOLUTE ERROR	1.42	1.31	N/A

The 2 CNN models that we developed made use of similar inputs. (Spectrogram data and % area porosity from the excel sheet) However, both these models had different architectures. Model 1 was developed using 3 Convolutional layers with a filter size of 3x3. The loss function used for this model was Mean Square Error which is a regression-based loss function. The optimizer used for this model is ‘Adam’.
On the other hand, Model 2 was developed using 2 Convolutional layers with a filter size of 7x7. The loss function used for this model was Mean Square Logarithmic Error which is a regression-based loss function. The optimizer used for this model is ‘Stochastic gradient descent’. This model is very similar to the CNN model provided to us which performs classification. However, this model was modified such that it predicted the percentage porosity of area.

It is observed that both these models have similar predictions of percentage porosity of area as well as similar mean square error and mean absolute error values. In comparison the simple linear regression model was developed using the pore count as an input in order to predict the percentage area porosity. Though the model is fairly accurate it has a few drawbacks when compared to the CNN model.

5. Executive Summary

This project focuses on percentage area of porosity detection and prediction using different deep learning techniques such as CNN, Edge Detection, and simple linear regression using the provided spectrogram data, black and white images of the sample and Excel data as input. Porosity detection is important because of the dangers associated with poor structural integrity. It is known that pores can become a region of stress concentration which could lead to the formation and propagation of cracks. Industries such as manufacturing, aerospace and midstream oil and gas have been expanding their capabilities to detect and prevent corrosion to ensure the safety of the workers and the quality of their products. Components with pores can have significantly lower mechanical properties and can undergo premature failure. The techniques used in this project have the potential to further improve detection and prediction of possible defects in the operating material. The CNN models that we have developed are fairly accurate in predicting the percentage area of porosity.

In general CNN models are used for classification-based problems, age detection problems and time series modelling. What we have done differently when compared to other models is that we have developed a CNN based model which used spectrograms and excel based tabular data as input in order to predict porosity. In comparison, CNN based models that have been developed perform classification by using images or tabular data as input. We have also attempted to use Simple Linear Regression and Edge detection to try and predict the percentage area of porosity.

The Edge detection that we developed to predict the edges of the pores was found to be lacking in some respects. Apart from the edges of the pores, gradients on the surface and other features were highlighted as well and hence the output of this model cannot be considered to be accurate.

Similarly, the Simple linear regression model that we developed does predict the area of porosity fairly well but has some drawbacks associated with it. This model uses only the pore count to predict the percentage area porosity. This implies that the number of pores will either have to be manually counted or detected with the use of another model and can then be used along with this model to accurately predict the percentage area of porosity.

The results of the CNN models were encouraging. The models that we developed show that they are able to predict the percent area porosity to a moderate degree of accuracy even though they had different architectures. The two-layer model currently shows slightly better results. The three-layer model has an MSE of around 2.75 with an MAE around 1.4, while the two-layer model has an MSE of around 2.60 and an MAE around 1.32.

6. Conclusion

A major takeaway from this project is the accuracy of the predictions of percentage area of porosity of the CNN models. Even though both the CNN models had different architectures, their predictions were fairly close. This can also be reflected in the MSE and MAE values obtained using these models. Although the simple linear regression model was accurate it has certain drawbacks associated with its use.

The CNN models that we developed can be used to detect other anomalies and surface defects apart from porosity. These models can also be trained on larger data sets in order to make them more accurate. The relationship between the number of convolution layers and the accuracy with which the model predicts can also be explored.

Similarly, a linear regression model can be developed which uses the spectrogram data as input rather than only the number of pores from the excel sheet. Though these models may not be accurate as there is no linear relationship between the predictors and the percentage area of porosity it is still worth exploring.

The edge detection technique can be used on images that have better clarity. In such a case, the model would be able to predict the surface defects more accurately. By building on this project, we could develop machine learning models that can predict more than just pores on the surface. Deep learning techniques have great potential which can be explored in the field of defect detection.

7. References

(1) https://www.mckinsey.com/capabilities/operations/our-insights/the-future-of-manufacturing

(2) https://en.wikipedia.org/wiki/Manufacturing

(3) https://www.clintonaluminum.com/what-are-the-most-common-and-some-uncommon-reasons-for-welding-porosity/

(4) https://www.mdpi.com/1996-1944/13/24/5755

(5) https://www.mdpi.com/1996-1944/13/24/5755

(6) https://link.springer.com/article/10.1007/s00170-022-08995-7

(7) https://blog.gramener.com/manufacturing-defect-detection-with-computer-vision/

(8) Alzubaidi, L., Zhang, J., Humaidi, A.J. et al. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. J Big Data 8, 53 (2021) https://doi.org/10.1186/s40537-021-00444-8

(9) https://towardsdatascience.com/introduction-to-py-torch-13189fb30cb3

(10) https://towardsdatascience.com/building-a-convolutional-neural-network-cnn-in-keras-329fbbadc5f5

(11) https://youtu.be/2yhLEx2FKoY

(12) https://www.datacamp.com/tutorial/convolutional-neural-networks-python

(13) https://www.upgrad.com/blog/basic-cnn-architecture/