MACHINE LEARNING PROJECT
(2020-21)

AIR POLLUTION ESTIMATION

Institute of Engineering & Technology

Team Members

Nancy Varshney
(171500203)
Nikita Bansal
(171500206)
Purvi Malik
(171500245)

Supervised By
Mr Amir Khan
Technical Trainer

Department of Computer Engineering & Applications

Department of Computer Science Engineering and Applications
GLA University, Mathura

17 km. Stone NH-2, Mathura-Delhi Road, P.O.
Chaumuhan, Mathura – 281406

Declaration

We declare that this project is our own work and has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Sign: _____

Name of Candidate: Nancy Varshney

University Roll No: 171500203

Sign: _____

Name of Candidate: Nikita Bansal

University Roll No: 171500206

Sign: _____

Name of Candidate: Purvi Malik

University Roll No: 171500245

CERTIFICATE

This is to certify that the project made by the candidates is correct to the best of our knowledge on work titled "**Air Pollution Estimation**". Student of **GLA University,** Mathura (UP) has completed the project work successfully as a part of course curriculum.

Supervisor	**Program Co-ordinator**	**HOD**
Mr Amir Khan	Mr Shashi Shekhar	Prof. (Dr.) Anand Singh Jalal

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our guide **Mr Amir Khan, Technical Trainer, Dept. of CEA** for providing and guiding this project. We deeply respect professor for his vast knowledge, numerous suggestions, and strong passion to complete this project. Valuable discussions with him not only made our work smooth but also encouraged us to think more professionally in the field of research.

We are also thankful to **Prof. (Dr.) Anand Singh Jalal**, **Head of Dept. of CEA** for helping us to secure this project work.

We also thank all our teaching and non-teaching staff for their support and well wishes.

Finally, we would like to express our deepest gratitude to our parents and friends for their encouragement and support.

Sign: _____

Name of Candidate Nancy Varshney

University Roll No.: 1715002033

Sign: _____

Name of Candidate: Nikita Bansal

University Roll No.: 171500206

Sign: _____

Name of Candidate: Purvi Malik

University Roll No.: 171500245

Abstract

Our project is to monitor the pollution using image processing technology. Image processing obtains the polluted parts of the image using edge detection and depth estimation technique. Thus air pollution monitoring is done through the image processing.

It detects and quantifies PM pollution by extracting a combination of image features, including transmission, depth, RGB channel, local image contrast, and image entropy.

We further consider the time, date and weather condition of each photo, to determine the correlation between PM level and various factors. Based on these features, we build a regression model to predict PM level using photos collected.

Contents

Certificate **ii**
Declaration **ii**
ACKNOWLEDGEMENT **iii**

Chapter 1 Introduction **1**

1.1 Overview 1

1.2 Motivation 1

1.3 Project Planning 2

1.3.1 Objective 2

1.3.2 Scope 2

1.4 Proposed Method 2

1.4.1 Image Extraction 2

1.4.2 Linear Regression 3

Chapter 2 Software Requirement Analysis **5**

2.1 Feasibility Study 5

2.1.1 Operational Feasibility 5

2.1.2 Technical Feasibility 5

2.1.3 Scheduling Feasibility 5

2.2 Requirements Analysis 6

2.2.1 Hardware Requirement 6

2.2.2 Software Requirement 6

2.3 Analysis of Dataset 6

Chapter 3 Software Design **8**

3.1 Effective Parameters 8

3.2 Flow Chart 8

Chapter 4 Mathematical Modeling 9

Chapter 5 Training and Prediction 12

Chapter 6 Implementation Detail 14

6.1 Dataset Images(Beijing) 14

Chapter 9 Result 15

Chapter 10 Conclusion 16

References 17

Appendices 18

Chapter 1

Introduction

1.1 Overview

Air pollution has become an alarming environmental issue globally due to rapid urbanization and industrialization. Monitoring urban air quality is therefore required by municipalities and by the civil society.

Among different air pollutants, airborne particulate matter (PM) with diameters less than 2.5 micrometres (PM2.5) has significant harmful effects on the human body. Therefore, PM2.5 concentration has been used as a worldwide major air quality metric.

Currently, air quality monitoring methods are mainly based on monitoring stations, which are not available to the majority of regions because of the high setup cost and expensive sophisticated sensors.

In this project, we study image-based air quality analysis, in particular, the concentration estimation of particulate matter with diameters less than 2.5 micrometers (PM2.5). For example, Smartphone users can take photos and estimate the real-time local air quality by themselves.

The proposed method uses Regression model to classify natural images into different categories based on their PM2.5 concentrations. In order to evaluate the proposed method, we created a dataset that contains images taken from smart phones with corresponding PM2.5 concentrations. The experimental results demonstrate that our method is valid for image-based PM2.5 concentration estimation.

1.2 Motivation

How good or bad is the air we breathe is known through monitoring and interpretation of data. It is an important need for present as well as for future of our planet. Inspired by the related works and considering human visual perception, we designed a simple but efficient algorithm for air quality evaluation from daily photos captured by mobile phones or digital cameras.

This project can further also help us in prediction of highly polluted areas which are unfit for staying and it can also help us to apply pollution control measures to the areas which are highly polluted.

1.3 Project Planning

1.3.1 Objective

The objective of our project is to monitor the pollution using image processing technology. Image processing obtains the polluted parts of the image using edge detection and depth estimation technique. Thus air pollution monitoring is done through the image processing.

It detects and quantifies PM pollution by extracting a combination of image features, including transmission, depth, RGB channel, local image contrast, and image entropy.

We further consider the time, date and weather condition of each photo, to determine the correlation between PM level and various factors. Based on these features, we build a regression model to predict PM level using photos collected.

1.3.2 Scope

The future scope of our project is to help the government in detecting highly polluted areas which are unfit for living. It will help the common people to relocate from a polluted area to safer environment.

After estimating polluted area, we can also apply pollution control techniques (like realising of O_2 & anti-hazing particles) in the air.

1.4 Proposed Method

1.4.1 Image Extraction

Generally, feature extraction is a key step for most image recognition algorithms. To imitate the ability of human visual perception, we choose some typical images with different AQI values to observe, as shown in Figure III. Given close observation, we find that the following results. Due to serious diffuse reflection caused by the large number of particles, the images with high AQI are much misty and bright, and the structural objects are blurred without clear boundaries. On the contrary, due to good air quality and light environment, the images with low AQI are much clear and have uniform brightness, and the objects also have clear structures. Therefore, we are inspired to evaluation the AQI based on the proportions of bright pixels and the edge pixels in the whole image.

Figure1: Different Images of Environment

1.4.2 Linear Regression

Simple linear regression is an approach for predicting a **response** using a **single feature**.
It is assumed that the two variables are linearly related. Hence, we try to find a linear function that predicts the response value(y) as accurately as possible as a function of the feature or independent variable(x).

Let us consider a dataset where we have a value of response y for every feature x:

x	0	1	2	3	4	5	6	7	8	9
y	1	3	2	5	7	8	8	9	10	12

Table1: X:feature, Y:Response

For generality, we define:

x as **feature vector**, i.e x = [x_1, x_2, …., x_n],

y as **response vector**, i.e y = [y_1, y_2, …., y_n]

for **n** observations (in above example, n=10).

A scatter plot of above dataset looks like:-

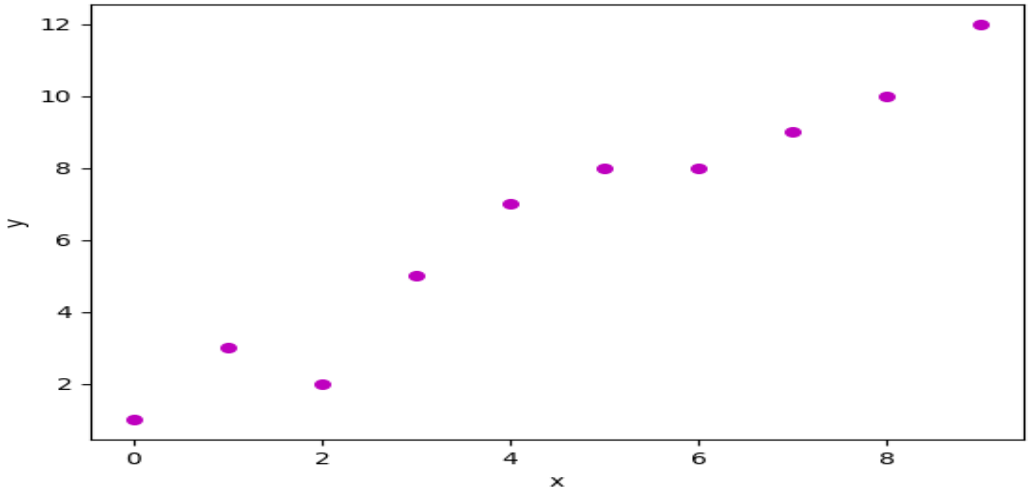

Figure2: Graph between response y and feature x

The equation of regression line is represented as:

Here,

- h(x_i) represents the **predicted response value** for ith observation.
- b_0 and b_1 are regression coefficients and represent **y-intercept** and **slope** of regression line respectively.

Chapter 2

Software Requirement Analysis

2.1 Feasibility Study

It is analysing the visibility of the Pollution Index. The idea for this project was thoroughly investigated while we constructed the different components and designed the dataset for the same. For an environment, pollution is the main asset which represents an environment is good or bad. It is used for pollution analysis, we will train the model by given data and test the model of real time image.

2.1.1 Operational Feasibility

The performance of this project has been checked with the help of different data sets. By providing the information, we got to know about the efficiency of the project and the amount of services it can offer us flexibly. This functionality actually helped us determine the peak performance and the low points which helps us in further improvements.

2.1.2 Technical Feasibility

The technical feasibility for this project has been established by the working model of this project. We evaluate the overall feasibility of assessing the public health impact of air pollution reduction programs in the cities by linking projected emissions reductions from overall regulatory actions to estimated detectable health outcome changes. We began by identifying pollutants of interest, we focused on $PM_{2.5}$.We also try to identified health issue symptoms.

2.1.3 Scheduling Feasibility

The project aimed to be completed on time by the dead line we made and no extra time or delay will be there. Since, we assure to be one of the good models to detect the particulate matter accurately and with efficient results.

2.2 Requirements Analysis

2.2.1 Hardware Requirement

- **Processor:** Intel Pentium III or later
- **Main Memory:** RAM(256 MB)
- **Hard Disk:**160GB

2.2.2 Software Requirement

- **Language:** Python
- **Jupyter Notebook**
- **Python 3.7v**
- **Operating System:** Windows/Linux

2.3 Analysis of Dataset

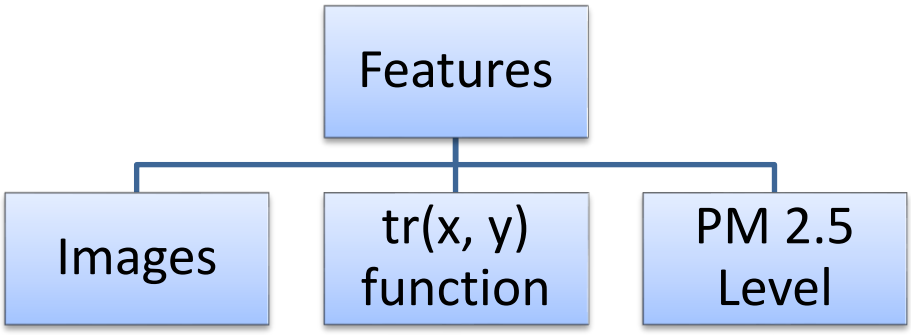

Figure 3: Features of Dataset

- The dataset consists of 327 images of a fixed scene, featuring Beijing Television Tower, captured at almost the same time every morning in 2014.
- The transmission function is calculated using above mathematical modeling.
- The last feature consists of PM level used for training and prediction.

We created the dataset using tr (x, y) values for each image captured and respective PM level for each image. As tr(x, y) is a matrix representation so we took the average of it and created a dataset. This dataset consists only of training data from which we trained our regression model and used it for prediction of PM level (pollution level) in the images captured at real time.

Our data is a excel file having data in 3 columns:

Image	tr(x, y)	PM conc.
1.jpg	0.057981	0.71
2.jpg	0.724369	0.45

Table2: Sample of Dataset

Based on this data we train our model.

An application is also created which captures image at real time and predicts its PM level.

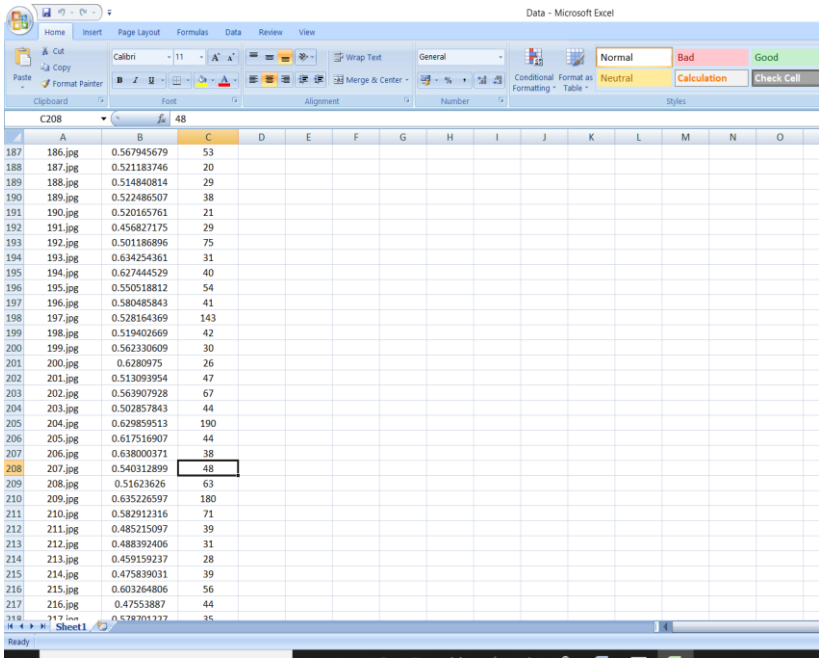

Figure 4: Real Time Dataset

Chapter 3

Software Design

3.1 Effective Parameters

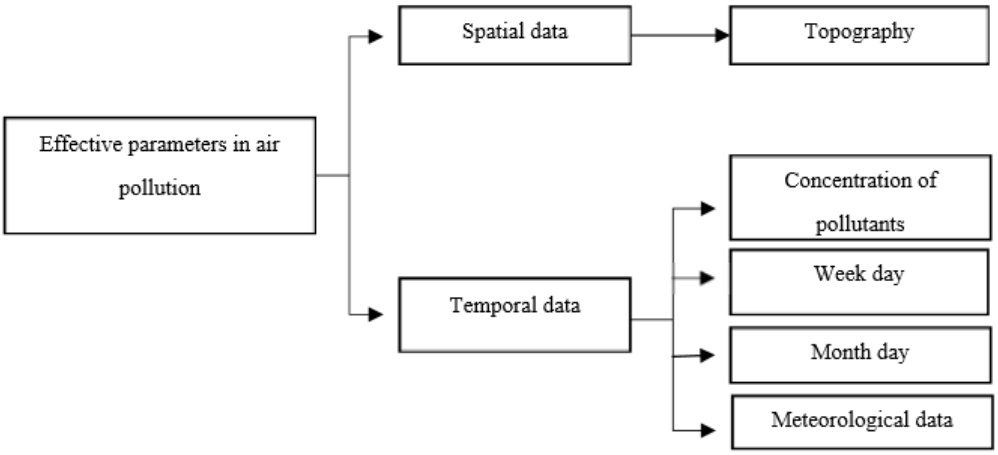

Figure 5: Effective Parameters

3.2 Flow Chart

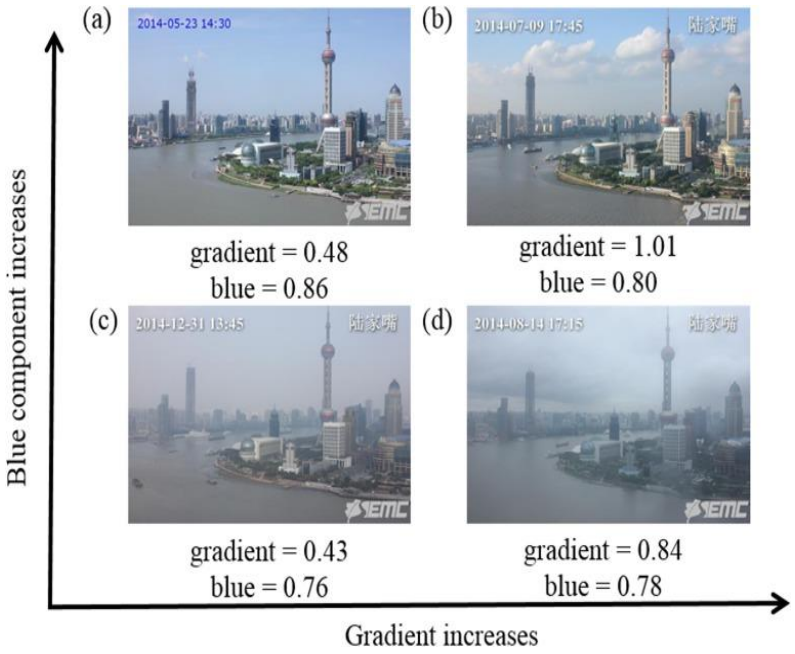

Figure 6: Flow Chart

Chapter 4

Mathematical Modeling

Here we take:

I = the observed hazy image

Our model is based on the principle equation:

I (x, y) = J (x, y) * tr (x , y) + A (1 – tr (x, y)) .. (1)

Where, tr (x, y) = Transmission from scene to camera

A = Air light colour vector

J(x, y) = Scene radiance

The first term of eq(1) is the direct transmission of the scene radiance into the camera, which is light reflected by the object surfaces in the scene and attenuated by air before entering the camera. The second term (1-t(x, y))A is called air light, which is the ambient light scattered by air molecules and PM into the camera.

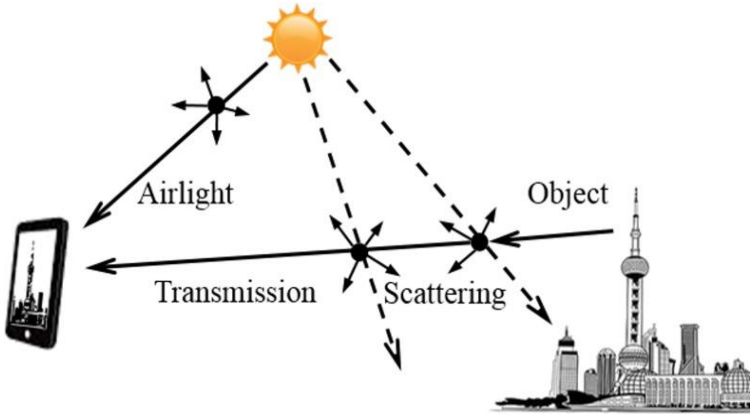

Figure 7: Appearance of real time object into an image

Assumptions:

- β=1 where β=scattering coefficient
- min($J(x, y)^c$) ≃ min[$J(x , y)^R$, $J(x, y)^G$, $J(x, y)^B$] ≃0
- A=1

Computations:

- tr (x, y)
- tr (x, y) as X

PM level as Y

- Y = mX + c
- Input Image = X

I (x, y) = J (x, y) * tr (x , y) + A (1 – tr (x, y)) ...eq(1)

As our main aim is to calculate tr(x, y) i.e. transmission function of our image pixels, so that we can calculate I(x, y) i.e. pollution level in our image. As value of tr(x, y) goes from 1 to 0, pollution level increases.

Light scattering causes an attenuation of light transmission in air, which can be expressed by the Beer-Lambert law,

$$\mathbf{tr\ (x, y)} = e^{-\beta d(x,y)}$$

(β = scattering coefficient)

(d(x, y) = depth of scene)

Here, if value of β=0 then tr(x, y) = 1. This will make our eq(1) as

I (x, y) = J (x, y) * 1 + 0

Hence,

I (x, y) = J (x, y)

Pollution function = function that radiant 0 pollution

Hence, our image has no pollution

But here we take our assumption 1 i.e. β=1:

So our tr(x, y) value becomes,

$$\mathbf{tr(x, y)= } e^{-d(x,y)}$$

As depth of an image is never zero, so,

$$d(x, y) \longrightarrow \infty$$

which makes value of tr(x, y) as,

$$\mathbf{tr(x, y)= 0}$$

This will make our eq(1) as

I (x, y) = 0 + A (1– 0)

$$\boxed{\mathbf{I\,(x, y) = A}}$$

Here, A is assumed to be 1 so,

I (x, y) = 1

i.e. highly polluted image.

As we all know there are 3 channels of color from which our image is made of i.e. Red, Green, Blue (RGB). So, we take the minimum of the I(x, y) function we calculated of all the 3 colours which together form the RGB channel.

Hence, $\mathbf{min(I(x, y)^c) = min(J(x, y)^c)tr(x, y)+A(1 - tr(x, y))}$

c € {RGB}

As min($J(x, y)^c$) is very small so it tends to 0:

$$\mathbf{min(I(x, y)^c) = A - A\ tr(x, y)}$$

Hence,

$$tr(x, y) = \frac{A - \min(I(x, y))}{A}$$

Chapter 5

Training and Prediction

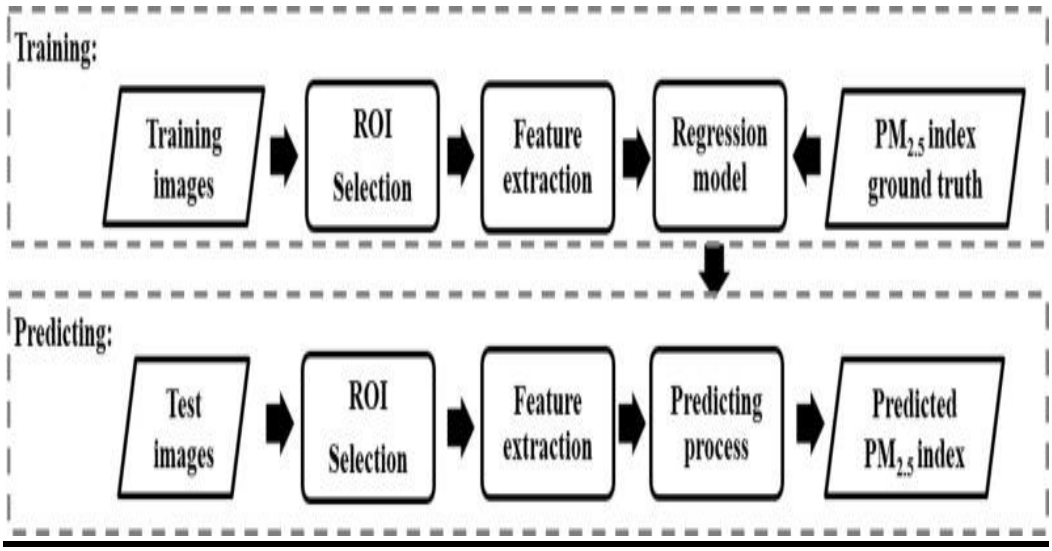

Figure 8: Analysis of training and testing

The first step is to remove the watermarks in these photos. The watermarks indicate the date and time stamp in our images, which appear in white characters in the first or last few rows.

The second step is to build a mask of the sky region, which appears in the images of all the three cities.

The colour images were converted into gray scale images, and then further into binary images with the Otsu method. The Otsu method converts gray scale to binary images by selecting a threshold that minimizes the intra-class variance or maximizing the inter-class variance.

The third step is to draw the ROIs for the distant buildings manually which were used to examine the transmission difference at different distances and PM densities. The ROIs were selected in one image in each dataset and applied to the rest.

a) Photos captured at Beijing, Shanghai and Phoenix respectively.

b) Boundary lines (blue lines in b) between distant buildings and sky.

c) Selected ROIs (red boxes).

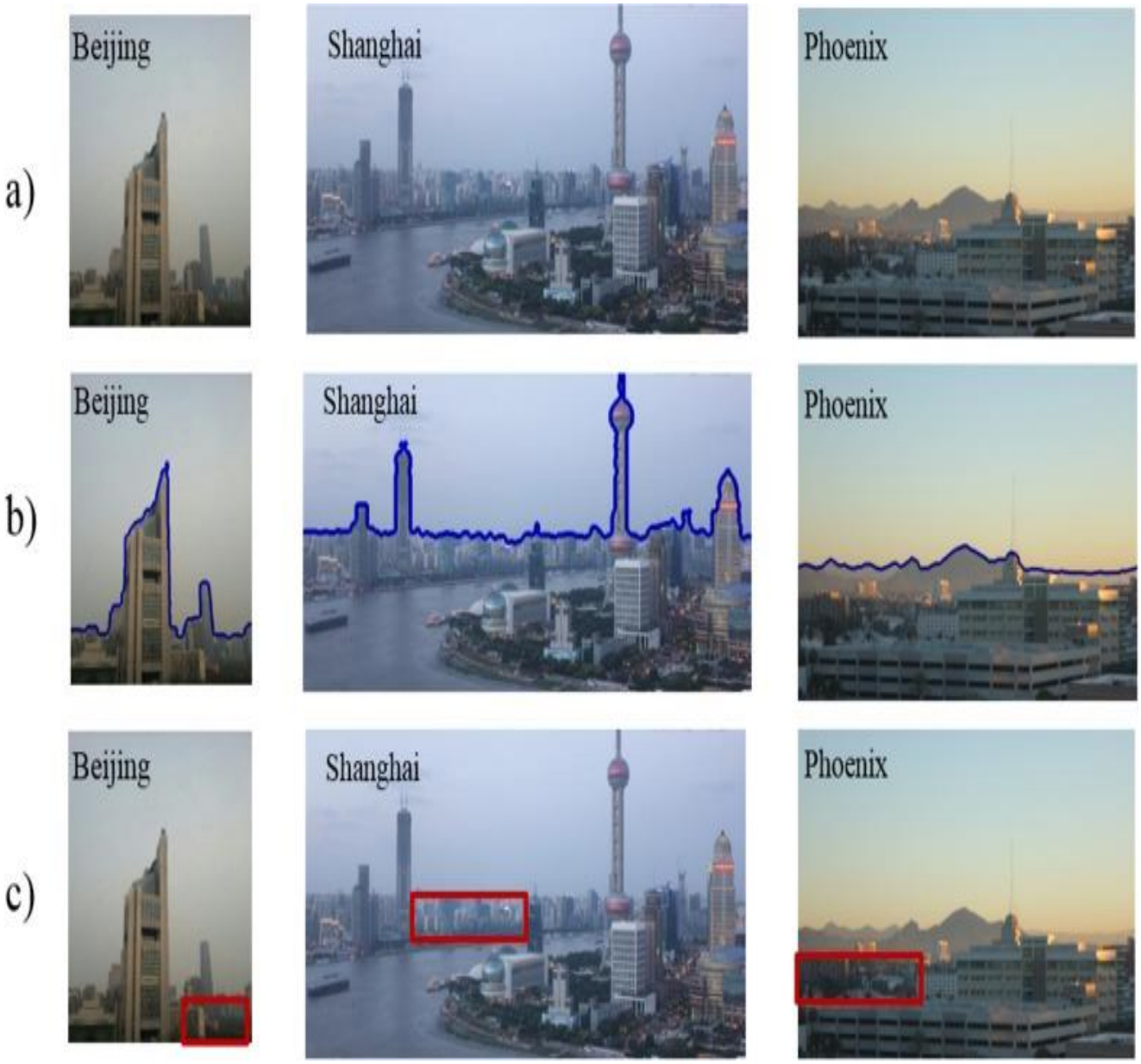

Figure9: Description of Dataset

Chapter 6

Implementation Detail

6.1 Dataset Images (Beijing)

Figure10: Dataset

- **Plotting of points in the graph (Regression Line)**
  (X-axis: tr, Y-axis: PM level)

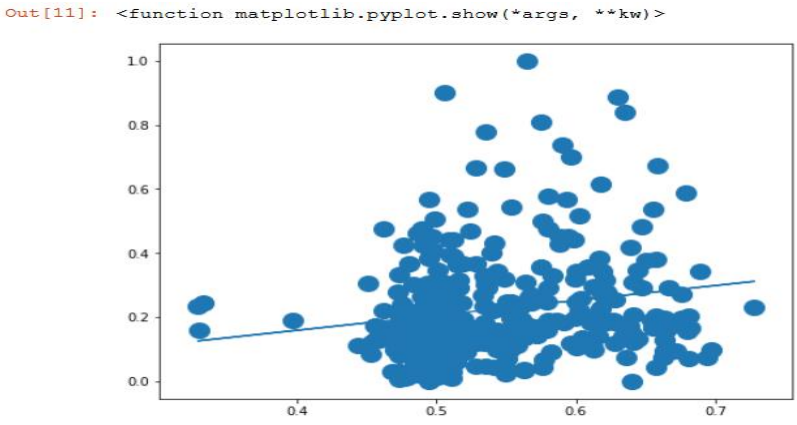

Figure11**: Plot Regression Line of trained data**

Chapter 9

Result

- **By Training the Data:**
  Slope is 0.4655242335374532
  Intercept is -0.027734934599438793
  Accuracy is 89.90825688073394
  294 correct predictions out of 327 predictions

- **By Testing the image:**
  Transmission value: 0.5254169139986724
  Slope is 0.4655242335374532
  Intercept is -0.027734934599438793
  Type of Tr <class 'numpy.float64'>
  Type of slope <class 'numpy.float64'>
  Type of intercept <class 'numpy.float64'>
  Output: The air quality level of the area is 47.19104614598772 PM 2.5
  So, in the testing image PM value shows good index.

Chapter 9

Conclusion

7.1 Proposed System Outcome

To validate the effectiveness of proposed algorithm, we compute the relative error for each testing image, and present the predicted air quality level (AQL) simultaneously, reports the predicted AQI and AQL values and prediction errors. As listed, the results show that our algorithm produced an average error in the aspect of AQI prediction, and achieved an accuracy for AQL prediction.

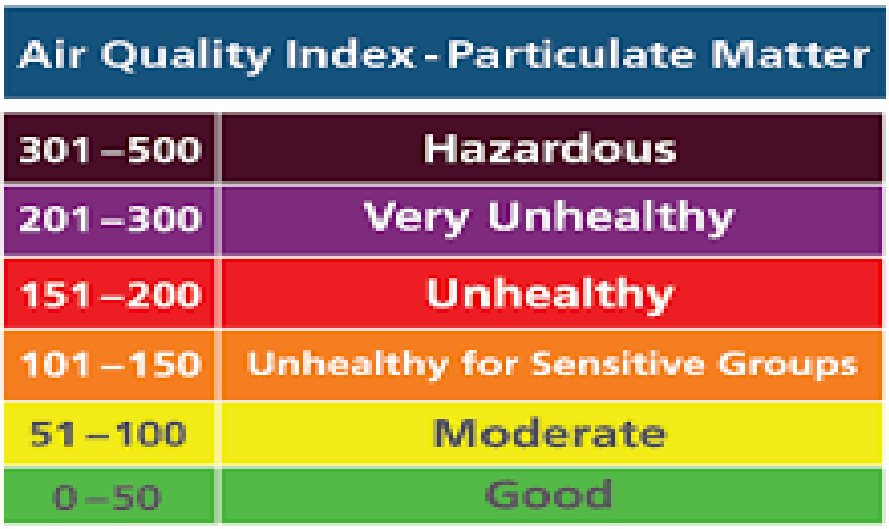

Figure12: APE Index

7.2 Limitations

- The sky part should be come in an image otherwise, wrong prediction of PM level will occur.
- Image captured in the dark(night time) also gives poor results.
- Model prepared is limited to location, that is Beijing but can be extended to the gla campus itself.

References

- **Websites:**

  WWW.Kaggle.com
  WWW.towardsdatascience.com

- **Books**

  Hands-on Machine Learning
  Fundamentals of Deep Learning

- **Faculty Guidelines**

  Prof.(Dr). Anand Singh Jalal
  Mr. Manish Raj

.

Appendices

- **Air Pollution Estimation using Images(code)**

 # Create dataset:

Step1: To find Transmission value

```
from PIL import Image
import numpy
import os
import pandas as pd
tr=[]
for i in range(1,328):
photo=Image.open('C:\\Users\\Lenovo2019\\Desktop\\Ml\\Beijing\\Beijing\\{}.jpg'.format(i))
    width,height=photo.size
    pix_val=list(photo.getdata())
    pix_val=numpy.array(pix_val).reshape((width,height,3))
    photo=photo.convert('RGB')
    width=photo.size[0]
    height=photo.size[1]
    R1=[]
    G1=[]
    B1=[]
    for y in range(0,height):
        row=""
        for x in range(0,width):
            RGB=photo.getpixel((x,y))
            R,G,B=RGB
```

```
            R1.append(R)
            G1.append(G)
            B1.append(B)
    R1=numpy.array(R1).reshape((width,height))
    G1=numpy.array(G1).reshape((width,height))
    B1=numpy.array(B1).reshape((width,height))
    x=numpy.minimum(G1,B1)
    x=numpy.minimum(x,R1)
    t=1-(x/numpy.max(x))/1
    avg=numpy.mean(t)
  tr.append(avg)
    i+=1
tr
```

Step2: To Create Workbbook

```
import xlsxwriter
workbook = xlsxwriter.Workbook('C:\\Users\\Lenovo2019\\Desktop\\Ml\\Mlp.xlsx')
worksheet = workbook.add_worksheet()
bold = workbook.add_format({'bold': True})
# Write some data headers.
worksheet.write('A1', 'Images', bold)
worksheet.write('B1', 'Tr(x, y)', bold)
worksheet.write('C1', 'PM level', bold)
# Start from the first cell below the headers.
row = 1
col = 1
for item in (tr):
    worksheet.write(row, col, item)
    row += 1
```

```
r=1
c=0
for i in range(1,328):
    worksheet.write(r, c, '{}.jpg'.format(i))
    r += 1
workbook.close()
```

#Training Code for train the data

```
from statistics import mean
import numpy as np
import pandas as pd
import random
import math
import matplotlib.pyplot as plt
color='#003F72'
air=pd.read_csv('C:\\Users\\Lenovo2019\\Desktop\\Ml\\Mlp.csv')
x=air['Tr(x, y)']
y=air['PM level']
# print(y)
# type(y)
a=min(y)
b=max(y)
d=b-a
print(a,b,d)
y1=[]
len(y)
#print(y)
for i in range(0,327):
    y1.append((y[i]-a)/d)
```

```
print(y1)
mn=mean(y1)
print(mn)
def best_fit_slope_and_intercept(x,y1):
    slope=((mean(x)*mean(y1))-mean(x*y1))/((mean(x)**2)-mean(x**2))
    intercept=mean(y1)-slope*mean(x)
    return slope,intercept
m,c=best_fit_slope_and_intercept(x,y1)
print(m,c)
plt.figure(figsize=(8,6))
plt.scatter(x,y1,s=200)
regression_line=[]
for xi in x:
    regression_line.append((m*xi)+c)
print(regression_line)
plt.figure(figsize=(8,6))
#plt.plot(x,y,color='#004F72')
plt.plot(x,regression_line)
plt.scatter(x,y1,s=200)
plt.show
correct=0
for i in range(len(x)):
    predict=(m*x[i])+c
    if  abs(y1[i]-predict)<mn:
        correct+=1
accuracy=float(correct)/float(len(x))*100
print (accuracy)
```

 # Testing Code for test the model

```
from PIL import Image
import numpy
import os
photo=Image.open("C:\\Users\\Lenovo2019\\Desktop\\Ml\\ima.jpg")
width,height=photo.size
pix_val=list(photo.getdata())
pix_val=numpy.array(pix_val).reshape((width,height,3))
photo=photo.convert('RGB')
width=photo.size[0]
height=photo.size[1]
R1=[]
G1=[]
B1=[]
for y in range(0,height):
    row=""
    for x in range(0,width):
        RGB=photo.getpixel((x,y))
        R,G,B=RGB
        R1.append(R)
        G1.append(G)
        B1.append(B)
R1=numpy.array(R1).reshape((width,height))
G1=numpy.array(G1).reshape((width,height))
```

```
B1=numpy.array(B1).reshape((width,height))

x=numpy.minimum(G1,B1)

x=numpy.minimum(x,R1)

t=1-(x/numpy.max(x))/1

avg=numpy.mean(t)

tr=avg

print(tr)

x=tr

print(m,c)

print(type(tr))

print(type(m))

print(type(c))

print(mn)

y = (m*x)+c

y=(y*d) + a

print("The air quality level of the area is",y,"PM 2.5 ")
```