Air Quality Estimation Using Images and Meteorological Data

Roop Choudhuri
roop.chouduri@iitgn.ac.in
Indian Institute of Technology Gandhinagar

Souvik Roy
souvik.roy@iitgn.ac.in
Indian Institute of Technology Gandhinagar

Soumita Kundu
soumita.kundu@iitgn.ac.in
Indian Institute of Technology Gandhinagar

Dhananjay Sonawane
dhananjay.sonawane@iitgn.ac.in
Indian Institute of Technology Gandhinagar

ABSTRACT

Air pollution is a major concern in today's world. In the year 2017, a staggering 1.24 million people died in India due to chronic respiratory diseases and asthma caused by severe air quality. And still, India only has 134 air quality sensors in 71 cities. Air quality estimation is an essential and fundamental problem in environmental protection. There have been several studies at estimating air quality using satellite images, in-situ air quality monitors, exploring features of images, etc. The problem is that deploying sensors at a large scale can be very expensive. Our goal is to design a reliable and inexpensive solution to estimate the concentration of $PM_{2.5}$ through real-time images clicked in open areas with visible skyline. We use a Convolutional Neural Network(CNN) and linear regression combined together with meteorological data for improving the discriminative ability of existing models using feature extraction. The model is trained on images extracted from a live stream in New Delhi and Aizawl. As a result, the linear regression model shows the least test RMSE.

KEYWORDS

feature extraction, air quality, machine learning, convolutional neural network, linear regression

1 INTRODUCTION

World Health Organization (WHO) reported that nine out of ten most polluted cities in the world are in India. Exposure to ambient particulate matter pollution (Particulate matter smaller than 2.5 μg or $PM_2.5$) contributes to heart disease, stroke, lung cancer, and respiratory ailments as it penetrates through the lungs. Researchers found that the annual population-weighted mean $PM_2.5$ in the air was highest in Delhi (209 $\mu g/m^3$), followed by Uttar Pradesh, Bihar, and Haryana (range: 125 to 174 $\mu g/m^3$). As technologists, we want to contribute to sustainable development by monitoring air pollution, so that necessary measures could be taken to control it. The first step would be to create an awareness amongst the people about the alarming levels of pollutants in the air they breathe in. This can be achieved by deploying an application that would provide real-time air quality estimation using a smartphone through camera images.

A polluted sky looks hazier than a clear sky in an image. We are estimating air quality through visual features of an image. (as shown in Figure 1) It is a supervised machine learning problem. We will train our model on images and data extracted from AQI India's live streams in New Delhi, Aizawl and Bangalore. The testing and validation will be performed on the images collected by us in Gandhinagar and various other cities. It is a regression problem as we will predict the continuous quantity output i.e. the concentration of $PM_2.5$ in the air.

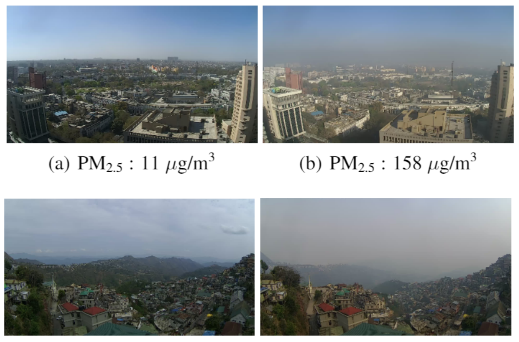

(a) $PM_{2.5}$: 11 $\mu g/m^3$ (b) $PM_{2.5}$: 158 $\mu g/m^3$ (c) $PM_{2.5}$: 22 $\mu g/m^3$ (d) $PM_{2.5}$: 206 $\mu g/m^3$

Figure 1: Images of Clear Sky and Polluted Sky of (a)|(b) New Delhi and (c)|(d) Aizawl respectively

2 RELATED WORK

Early from the year 2012, some researchers started working on estimating air quality through image processing techniques. [3] They analyzed the colour of the sky using image processing to determine the air quality of an area. Colour histograms of RGB were used to obtain data and analyze images.

In one of the papers on air quality estimation[4], they take a dataset that contains 591 images in Beijing with the corresponding $PM_2.5$ concentrations and use a deep Convolutional Neural Network (CNN) to classify natural images into different categories based on their $PM_2.5$ concentrations.

In another paper[8], the authors used an ensemble CNN. Three ordinal classifiers were devised in the last layer of each CNN to improve the ordinal discriminative ability of the model.

Another recent work was AirCognizer [1], they built an Android-based mobile application to provide local, real-time air quality estimation using smartphone camera images. They used TensorFlow Lite to train their model. It uses pictures taken from the phone's camera, processes them to provide an AQI estimate. They train two image-based machine learning models to build the application: the first model predicts the AQI using the features of the user-uploaded photo, and the second model filters out images where no skyline is present. The model predicts the AQI based on features extracted from the images. [6] The temporal model uses meteorological parameters to achieve higher inference accuracy and gives some results to the user while the image-based machine learning model is being trained,the image-based machine learning model customizes the model to the specific user thereby improving the inference accuracy by reducing the estimation error. The uniqueness of this approach was that they created custom models that collected images from each user. Then, two models were combined: imaged-based and temporal(meteorological parameters) which showed better performance than when they were deployed separately.

3 APPROACH

3.1 Dataset

A dataset was created with images extracted from AQI India's website which has live streaming cameras in New Delhi and Aizawl. The PM2.5 values and humidity values were collected from the live stream whenever they were updated. We only kept those corresponding images of the timestamp for which we had updated PM2.5 values and rest of them were discarded. Only daylight images were taken as it's difficult to extract features from a night sky.

3.2 Models

3.2.1 Multilayer Perceptron. At first, A naive approach was used to predict $PM_2.5$ using a neural network,i.e., Multilayer perceptron (MLP). An MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. MLP utilizes a supervised learning technique called back-propagation for training. All the input images were resized to 256*256*3. The MLP structure is shown in Table 1. Since each pixel of an input image was used as feature, input size to MLP = 256 * 256 * 3 = 196608 which is followed by 3 hidden layers with 16,8,8 nodes respectively. We trained MLP on Delhi dataset and tested on Delhi.

3.2.2 Convolutional Neural Network. Convolutional Neural Network (CNN) like neural networks, are made up of neurons with learnable weights and biases. Each neuron receives several inputs, takes a weighted sum over them, passes it through an activation function and responds with an output. The input to a convolutional layer is a m x m x r image where m is the height and width of the image and r is the number of channels, e.g. an RGB image has r=3. A CNN was trained with the structure given in Table 2 Both Delhi and Aizawl images were used for training separately. Kernel size for first and second layer is 5*5 and 3*3 respectively. CNN performs feature extraction using different filters. This extracted features acts as input to dense MLP network. We have used single hidden layer with 128 nodes.

3.2.3 Linear Regression. Linear Regression is a machine learning algorithm based on supervised learning. It is mostly used for finding out the relationship between variables and forecasting. Linear regression performs the task to predict a dependent variable value based on a given independent variables.

The feature set was extracted based on the reference of these two papers.[5] [7]

Table 1: MLP Structure

Layer(type)	Output Shape	Param #
dense_24 (Dense)	(None, 16)	3145744
dense_25 (Dense)	(None, 8)	136
activation_18 (Activation)	(None, 8)	0
dropout_7 (Dropout)	(None, 8)	0
dense_26 (Dense)	(None, 1)	9
activation_19 (Activation)	(None, 1)	0

Total params: 3,145,889
Trainable params: 3,145,889
Non-trainable params: 0

Table 2: Neural network structure

Layer(type)	Output Shape	Param #
conv2d_1 (conv2D)	(None, 256, 256, 32)	2432
max_pooling2d_1 (MaxPooling2)	(None, 128, 128, 32)	0
dropout_8 (Dropout)	(None, 128, 128, 32)	0
conv2d_2 (Conv2D)	(None, 128, 128, 64)	18496
max_pooling2d_2 (MaxPooling2)	(None, 64, 64, 64)	0
dropout_9 (Dropout)	(None, 64, 64, 64)	0
flatten_1 (Flatten)	(None, 262144)	0
dense_27 (Dense)	(None, 128)	33554560
dropout_10 (Dropout)	(None, 128)	0
dense_28 (Dense)	(None, 1)	129

Total params: 33,575,617
Trainable params: 33,575,617
Non-trainable params: 0

3.3 Features Used

The following features were extracted from the images using image processing techniques:

3.3.1 Transmission. This describes scene attenuation and the amount of light entering the phone camera after being reflected by air particles. It can be described from this equation:

$$I(x) = J(x) + A(1 - t(x)) \tag{1}$$

where, 'I' is the observed intensity, 'J' is the scene radiance, 'A' is the global atmospheric light, and 't' is the medium transmission describing the portion of the light that is not scattered and reaches the camera.

The first term J(x)t(x) is called direct attenuation, and the second term A(1 - t(x)) is called airlight.

Transmission for a single hazy image was found by using the concept of dark channel, which assumes the existence of some pixels with zero or very low intensity at least for one color channel in all the outdoor images. For a haze free image J, the dark channel of J is:

$$J^{dark}(x) = \min_{c \in \{r,g,b\}} (\min_{y \in \Omega(x)} (J^c(y))) \tag{2}$$

where, J^c is a color channel of J and $\omega(x)$ is a local patch centered at x. If J is a haze-free outdoor image, our observation says that except for the sky region, the intensity of J^{dark} is low and tends to be zero. We call the above statistical observation or knowledge, the dark channel prior.

The airlight can be estimated from the sky or the brightest region. So, the transmission can be obtained by:

$$\tilde{t}(x) = 1 - \min_{c}\left(\min_{y \in \Omega(x)} \left(\frac{I^c(y)}{A^c} \right)\right) \tag{3}$$

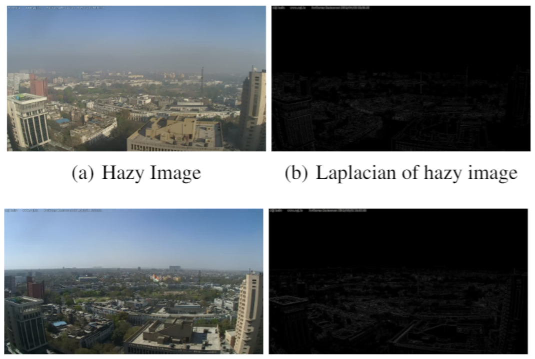
The second term on the right is the dark channel of the normalized haze image. It directly provides the estimation of the transmission.

3.3.2 *Atmospheric Light.*

This feature is somewhat similar to how we perceive a polluted day. If the sky is gray, it is taken as a polluted day. It was estimated using RGB splitting. The dark channel was used to improve this estimation. The top 0.1% brightest pixels were taken in the dark channel. These pixels are mostly haze-opaque. Among these pixels, the pixels with highest intensity in the input image is selected as the atmospheric light.

3.3.3 *Gradient of Sky.*

The sky could appear gray due to cloud cover, so to take this possibility into account this feature was incorporated. The gradient of a color is an excellent tool to calculate the edges of an image. It was calculated by making a mask of sky region and then calculating the Laplacian of the region.


(a) Hazy Image (b) Laplacian of hazy image

(c) Clear image (d) Laplacian of clear image

Figure 2: Laplacian of hazy and clear images

3.3.4 *RMS Contrast.*

Contrast is the amount of difference between color and brightness to which different objects in the image can be visually distinguished from one another. Root Mean Square (RMS) Contrast is the standard deviation of the pixel intensities. Following is the equation for RMS contrast:

$$RMSContrast = \sqrt{\frac{1}{MN} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} (I_{ij} - I')} \tag{4}$$

where, MN is the size of two dimension image, I_{ij} is intensity at (i,j) pixel of the image, and I‘ is the average intensity of all pixels in the image. Thus, Contrast has an inverse relation with $PM_2.5$ value.

3.3.5 *Entropy.*

Entropy of image is the amount of information that an image contains. Flat images have entropy zero and image containing heavy objects have high entropy value. The equation for Entropy is:

$$Entropy = -\sum_{i} P_i \log_2 P_i \tag{5}$$

where, P_i is the probability that the pixel intensity is equal to i and M is the maximum intensity of the image. As the PM concentration increases, the image increasingly loses its details, and the image entropy decreases. Thus it follows an inverse relation with $PM_2.5$ value.

3.3.6 *Humidity.*

Air pollution is correlated with humidity because humidity adds moisture to the $PM_2.5$ particles, making them heavier and closer to the ground level. Thus, lowering the visibility. The current humidity value for a test image is extracted from OpenWeatherMap API.[2]

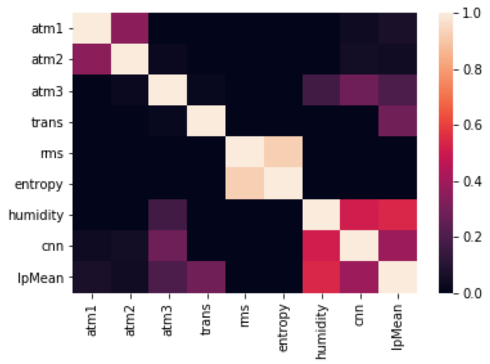


Figure 3: Correlation among the features

3.3.7 *Correlation between feature variables.*

Note that, features (predictor variables) are weakly correlated. Thus, multicollinearity is absent. The issue with multicollinearity is that small changes to the input data can lead to large changes in the model, even resulting in changes of sign of parameter estimates. Since multicollinearity causes imprecise estimates of coefficient values, the resulting out-of-sample predictions will also be imprecise. And if the pattern of multicollinearity in the new data differs from that in the data that was fitted, such extrapolation may introduce large errors in the predictions.

3.3.8 *Training using Linear Regression Model.*

The Linear regression model is trained to predict the value of PM2.5 for a given list of features. The model gets the best regression fit by finding the following intercept and coefficient values. The linear regression equation is given below:

$$\begin{aligned} PM2.5(predicted) = & -1106.22655364 - 8.24959072e^{-01} \times' atm1' + \\ & 5.30534230e^{+00} \times' atm2' - 5.87068577e^{-01} \times' atm3' \\ & - 1.39689835e^{-05} \times' trans' - 2.01123410e^{+00} \times' rmse' \\ & 4.62471193e^{+01} \times' entropy' 1.25389234e^{+00} \times' humidity' \\ & - 3.33084836e^{+02} \times' gradient' \end{aligned}$$

3.4 Method

The training involves the following steps:-

Step 1: Extract images, PM2.5 values, humidity and other information from AQI India's live stream.
Step 2: Train a CNN model.
Step 3: Train a linear regression model with the feature list and combine it with the CNN output.

The testing involves the following steps:-

Step 1: Input a test image and it's corresponding location.
Step 2: The model returns the predicted PM2.5 for the test image.

4 RESULT

Initially, for the Multi layer perceptron, The RMSE obtained was 9070324.45.
The CNN model was trained with automatic feature extraction. The number of images for training and testing are mentioned in Table 3. The following results were obtained:-

Table 3: CNN Results

Training Dataset	Testing Dataset	Test RMSE
Delhi (#Images=402)	Delhi (#Images=101)	203.42
Delhi (#Images=402)	Aizawl (#Images=585)	1513.65
Aizawl (#Images=468)	Aizawl (#Images=117)	197.52
Aizawl (#Images=468)	Delhi (#Images=503)	990.87

Due to less number of images, CNN had a high RMSE. That is why, the Linear Regression model was used with feature extraction. The final features were selected by retraining and testing on different features' subset.

Table 4: Linear Regression Results(trained on Delhi data)

Features used	Test RMSE(on Delhi data)	Test RMSE(on Aizawl data)
Transmission, atmospheric light, gradient, entropy, RMSE contrast, humidity	19	28
Transmission, atmospheric light, gradient, entropy, RMSE contrast, humidity, temperature	16	36
Transmission, atmospheric light, entropy, RMSE contrast, humidity, temperature	13	49

Table 5: Linear Regression Results(Trained on Aizawl data)

Features used	Test RMSE(on Aizawl data)	Test RMSE(on Delhi data)
Transmission, atmospheric light, gradient, entropy, RMSE contrast, humidity	20	51
Transmission, atmospheric light, gradient, entropy, RMSE contrast, humidity, temperature	17	64
Transmission, atmospheric light, entropy, RMSE contrast, humidity, temperature	18	69

The Linear regression model was trained on Delhi images (table 4). The features transmission, atmospheric light, gradient, entropy, RMSE contrast and humidity were used. The test RMSE on Delhi and test RMSE on Aizawl was 19 and 28 respectively.

When temperature was included in this feature list, the test RMSE on Delhi decreased(16) but the test RMSE on Aizawl(36) increased.

When gradient (laplacian of an image) was removed from this feature list, the test RMSE on Delhi decreased(13) but the test RMSE on Aizawl(49) further increased.

So, the final features selected were: Transmission, atmospheric light, gradient, entropy, RMSE contrast, humidity.

Table 5 shows similar results when the model was trained on Aizawl images.

We propose a new model in which the output of the CNN is fed as one of the input features in training the linear model. Rest of the features remain the same. The result is summarized below:

Table 6: CNN+Linear regression results(trained on Delhi data)

Features used	Test RMSE(on Delhi data)	Test RMSE(on Aizawl data)
CNN output + Transmission, atmospheric light, gradient, entropy, RMSE contrast, humidity.	30.23	42.54

Table 7: CNN+Linear regression results(trained on Aizawl data)

Features used	Test RMSE(on Aizawl data)	Test RMSE(on Delhi data)
CNN output + Transmission, atmospheric light, gradient, entropy, RMSE contrast, humidity.	14.78	50.54

Combining CNN and linear regression did not show any improvement over the liner model. This can be accounted for by the RMSE of our training set and test set which increased due to less number of images used for training and the model being biased to a single location.

5 CONCLUSION AND FUTURE WORK

The high RMSE of the multilayer perceptron model is due to fact that it considers every pixel as an input. But, every pixel did not contribute to the haze content of an image. In fact, such pixels caused an increase in the error. CNN performed better than MLP as it extracted features from the images automatically using different filters. The features extracted by the CNN were unknown as we did not know what filter was being used by the CNN. Linear regression with feature extraction outperformed all the other models. The RMSE of the CNN can be reduced with a larger dataset with images from different locations. To improve the models, a larger dataset has to be collected over a longer period of time. Deployment of the models in real-time over multiple platforms like mobiles using efficient methods can further extend the work.

REFERENCES

[1] [n. d.]. Air Cognizer: Predicting Air Quality with TensorFlow Lite. https://medium.com/tensorflow/air-cognizer-predicting-air-quality-with-tensorflow-lite-942466b3d02e.

[2] [n. d.]. OpenWeatherMap API. https://openweathermap.org/api.

[3] [n. d.]. Using Image Processing Techniques to Estimate the Air Quality. https://digitalscholarship.unlv.edu/cgi/viewcontent.cgi?article=1019&context=mcnair_posters.

[4] A. Chakma, B. Vizena, T. Cao, J. Lin, and J. Zhang. 2017. Image-based air quality analysis using deep convolutional neural network. In *2017 IEEE International Conference on Image Processing (ICIP)*. 3949–3952. https://doi.org/10.1109/ICIP.2017.8297023

[5] Kaiming He, Jian Sun, and Xiaoou Tang. 2011. Single image haze removal using dark channel prior. *IEEE transactions on pattern analysis and machine intelligence* 33, 12 (2011), 2341–2353.

[6] Chenbin Liu, Francis Tsow, Yi Zou, and Nongjian Tao. 2016. Particle pollution estimation based on image analysis. *PloS one* 11, 2 (2016), e0145955.

[7] Gagan Rana, Rahul Saha, and Geetha Ganesan. 2016. Analysis of Relation between Entropy and Factors in Image Based CAPTCHAs. *International Journal of Engineering and Technology* 7 (01 2016), 2105–2108.

[8] Chao Zhang, Junchi Yan, Changsheng Li, Hao Wu, and Rongfang Bie. 2018. End-to-end learning for image-based air quality level estimation. *Machine Vision and Applications* 29, 4 (01 May 2018), 601–615. https://doi.org/10.1007/s00138-018-0919-x