VisionAir - AQI estimation using a unified model that is trained on-device to ensure complete user privacy

***Abstract*—** Deteriorating Air Quality levels in the state of Delhi have had adverse effects on its residents. Thus arises the need for a Real-time and accessible AQI monitoring system that allows users to take protective measures against such toxic air by analysing the air around them.To achieve this, a diverse dataset is curated which consists of 4k images taken across the entire National Capital Region and this is used to train the VisionAir Model: A shallow neural net that uses image parameters extracted from HDR images, meteorological data, historical AQI data and time as the primary features to estimate the AQI from an image. To ensure that no private user data including the image the user clicks, the user's location, etc are shared with a remote server, we employ federated learning in our model to dynamically improve our model over time whilst ensuring complete user privacy.The model devised is computationally inexpensive and optimised for standalone On-Device training.

Keywords— federated learning, on-device training, Camera2API, HDR images, PM2.5 prediction, image-analysis, deep learning, application

Introduction:

Long-term health effects from air pollution include heart disease, lung cancer, severe respiratory diseases such as emphysema, damage to the nervous system, brain, kidneys, liver, and other organs. Existing methods of analysis include specialized sensors which are expensive, fragile are require high maintenance. Applications that claim to predict AQI often use interpolative methods which are inaccurate as AQI does not strictly follow any spatial-temporal trend.

To provide a cheap, accurate, and conveniently accessible method of this analysis, we use Smartphone cameras to obtain an HDR image from the user whose AQI is estimated using the VisionAir Model.This model provides estimates that are accurate and independent of the hardware that is used to click the image, otherwise, the same scene photographed by two different phones is found to give different results due to the post-processing techniques used by each phone.

Related Work:

In [1], Liu et. al uses a Linear Regression model to predict the PM 2.5 index by using image features such as transmission, entropy, RoI contrast, and Sky features.The model was trained on static images (same scene) taken across only 3 locations which dramatically reduces the generalization capacity of the model.

In [2], the authors use a stacked autoencoder model to extract representative spatiotemporal air quality features to predict the AQI.The model did not take into account the immediate surroundings of the user which significantly affect the AQI in a region due to its sensitivity to local factors.

In [3], inspired by [1], the authors use a Convolutional Neural Network and feed the Transmission Map along with contrast, entropy, wind speed, temperature, and humidity to predict the AQI. The location and the image captured by the user were sent to a server for evaluation which compromised the privacy of the users.

Proposed Work

A model that uses HDR image parameters mapped with meteorological and historical AQI data is proposed.As of now, there exists no meteorologically annotated dataset of images within Delhi. As a result, a dataset that constitutes both HDR and Non-HDR images taken across 80+ locations in Delhi across multiple smartphones is curated to train the model.

To ensure the allocation of scientifically valid labels to the images in the dataset, a three-way correlation between our model's prediction, a high-quality AQI

sensor: *AirVeda,* and data from the CPCB sensor for the same location.

System Design:

Dataset Methodology Design:

For collecting the dataset, we have developed an android application that will automatically click an HDR as well a Non-HDR image every 5 minutes. This application is ideally made to run for at least 48 hours in one cycle to exploit any temporal relation between the scene and the AQI levels in a region.

It was observed that if the same scene is captured using two different smartphones, due to the custom post-processing carried out by each phone, the model gave different predictions. Since most of the past work was carried out using a single camera device, the trained model performed poorly when fed with image parameters that had been extracted through an image from a different hardware device.

After researching the generic methods of image post-processing, it was concluded that HDR images would give a good estimate of the original information contained in the image (before any processing is carried out).The RAW mode of imaging was also explored for this purpose, but their large size and computationally expensive feature extraction process encouraged us to look into HDR viewing, thus also optimising the on-device computation.

Conceptually, HDR images are a combination of images at different exposure values:

Overexposed Image: This image is overexposed (more light enter the camera) than a normal image. The goal of this image is to get dark pixels of the image.

Underexposed Image: This image is underexposed (less light enters the camera) than a normal image. The goal of this image is to get a brighter pixel of the image.

Properly exposed Image: This is the regular image of the camera

Fusion of all these images will give a single image whose pixels will occupy a very vast range. So to bring them into a viewable range we tone map the images and obtain viewable HDRs [4]

Application Design

The final application is meant to train the model on the device after extracting appropriate image parameters from the HDR image and mapping that with the weather data, historical AQI data and time. The relevant image parameters are explained in detail in the Model section.

The weather data is obtained by making API calls.To do this while ensuring user privacy, the geotag (the accurate location) of the user's location is rounded off to ensure that the user's location is not sent to the server. The API call returns all the necessary weather parameters which are concatenated with the image features, and the modified timestamp (giving the hour) as a part of the input features to the model.

For the historical AQI data, we use, a government API: "Real Time Air Quality Monitor". As this gives us real-time AQI readings, we store the previous hour's reading on a firebase database and fetch it as and when required.

Readings from the nearest CPCB center are used as the labels for the on-device training of the model.

Model:

Most of the papers implemented in the past – use regressors to predict AQI values using a set of given features. We, however, are using a shallow neural network for implementing our solution having observed consistently better generalization to new data. Since, the model was to be trained on-device, only a shallow network, consisting of 2 hidden layers has been used to derive the results mentioned ahead. To ensure that the neural network in consistently performs better than the regressor: Throughout the entire model evaluation – A linear regressor and a random forest regressor were trained and tested alongside the neural network.

Feature Selection for the Model:

We use vision to gauge the AQI conditions of the immediate surrounding of the user. Conceptually speaking, light scatters due to the presence of particulate matter (PM2.5, PM 10) in the air and we thus estimate a few features from the image that are directly affected by the presence of such light scattering particles. These features include:

1. Haze Degree
2. Contrast
3. Entropy

Haze Degree: A function for estimating the haze degree is developed for the automatic detection and quantification of smog in images.Building on the work of Mao et al [5], this function uses three constants that need to be determined iteratively using a few image parameters.

To do this, we curated a dataset consisting of 250 images from multiple locations; The constants were determined after minimising the loss over training data and then tested to be finally implemented in the actual function.

Sample Dataset for determining constants

Fig 1: Each Image is mapped to a value between 1-5

Haze Degree is predicted as:

$$\omega = exp\left\{-\frac{1}{2}(\mu x_1 + \nu x_2) + \sigma\right\},\ x_1 = \frac{A_0 - d}{A_0}, x2 = \frac{c}{A_0}$$

Entropy: Entropy is defined as corresponding states of intensity level which individual pixels can adapt.Entropy gives degree of randomness of an image.

$$entropy = -\sum_{i=1}^{M} p_i log_2 p_i$$

Contrast: Contrast is the difference in luminance or colour that makes an object (or its representation in an image or display) distinguishable. It gives an estimate of the visibility in an image.

$$RMS = \sqrt{\frac{1}{MN}\sum_{i=0}^{N}\sum_{j=1}^{M}(I_{ij} - avg(I))^2}$$

Refer to Fig[1] for a representation of the usefulness of these features.

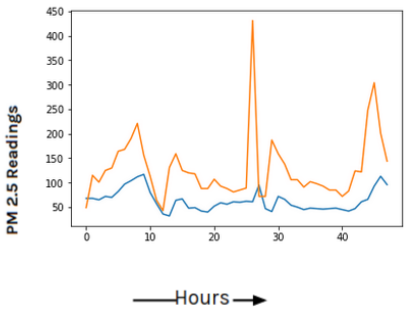
Weather Features: The wind speed and direction play a crucial role in determining the air dispersion which determines how well the particulate matter may be dispersed over a given region. Humidity, moisture and precipitation are observed to inversely affect the AQI in a region and hence they have also been added as relevant features.

Time of the Day: Several observations, one of which being, poorer AQI levels during peak traffic hours, can be observed through analysis of this feature over time. The previous hour's PM has also been added as an input feature to provide an initial estimation of the PM in the region.

Labels:

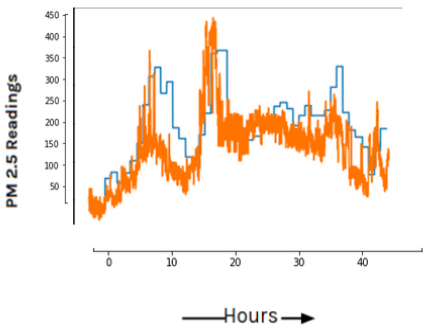
The labels for all the images have been obtained from two sources: The primary source of labels are provided by the nearest Central Pollution Control Board's (CPCB). At some locations, A High Quality AQI sensor – AirVeda has also been used to label the images. Before using the AirVeda as a valid ground truth source, it was calibrated and tested for 5 days by correlating it with a CPCB sensor's data placed within its 40m radius.

Before Calibration



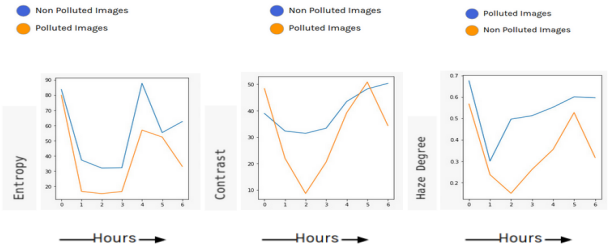
The Airveda was not reporting any readings above 100 ug/m^3 when compared to the CPCB readings thus forcing us to believe that it needed calibration.

After Calibration



The AirVeda readings started correlating strongly with the CPCB sensor readings after calibration.

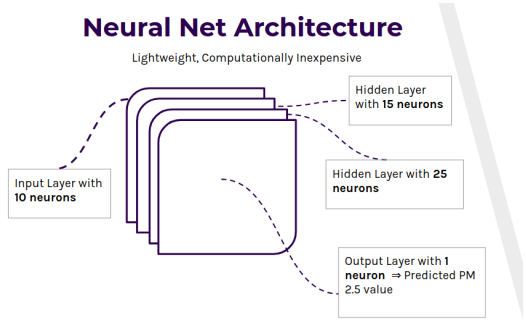
Fig [1]



Analysing Usefulness of Image Features:

Since Entropy, contrast and haze degree of polluted and non- polluted images are consistently different over time, they can be chosen as relevant features in the Model.

Model Architecture:



The neural network uses all the above mentioned features along with the labels as stated. They are fed into a shallow neural network with 2 hidden layers to predict the PM 2.5 value of a given image.

Model Evaluation

1. Model trained on Non-HDR images ⇒.
   RMSE ⇒ 9.05 ug/m^3
2. Model trained on HDR Images ⇒
   RMSE ⇒ 13.01 ug/m^3

The RMSE (Root Mean Square Error) is a metric of the maximum deviation between my predicted and actual labels.

Model Comparison:

Liu et al [1] use about static images taken across only 3 locations which restricts the generalisation capacity of the model. Third Eye [2] does all the computation on a remote server, thus compromising the privacy of the user.

Comparison Between Past implementations and the proposed model: VisionAir

Past Implementations	**RMSE**
1. Liu C. et. al	19.23
2.Third Eye	17.38
3.VisionAir	**9.05**

While this culminates the architecture of the global model, in order to iteratively enhance the power of the global mode we employ the concept of federated learning. This part of the application is currently in the testing phase and its implementation has been explained ahead.

Federated Learning

Since our application requires images to predict the AQI, many users had concerns with their private data such as images being uploaded to the server. As a result, a lot of people were reluctant to use the application. This also meant that fewer users were willing to share their data for training the model further. This resulted in a large amount of rich training data not being used.

Keeping up with the growing privacy concerns and to enable crowdsourcing, we implemented the concept of Federated Learning [6] in our application.

Each user device has local training data on which a few epochs are performed. The updated weights are then sent to the server which maintains the global model. The global model weights are updated to the average of the updated local weights.

User privacy was a major point of focus while building the application. We wanted to make sure that the data taken from the user could not be isolated and tracked back to the user. Thus, our application takes a huge step forward in setting a trend for Deep Learning models which rely heavily on users' data without breaching their client's privacy. Assuring a user's privacy leads to a relation of trust between us and our clients, which results in greater amount of data sharing and hence a better model.Also, this led to better crowdsourcing of data.

To ensure the privacy of the user and avoid sending private data to the server, On-Device training was employed. TensorFlow's API for Java was used for On-Device Inference and Training. The API required a frozen graph and a saved checkpoint which were generated while training the initial global model.

The features from the image were then extracted and fed into the model for prediction. The label was fetched from the nearest CPCB center. These features and the label were used to perform training on the device itself.

The process of Federated Averaging [4] required a server for computation. So, a server using Flask and REST API was created to facilitate the global model. The server used Firebase Database and Storage to temporarily save the weights for averaging. Also, the server hosted the checkpoint files for the initial model. The server has different routes to enable upload and download of weights.

The weights of the trained client model are sent to the server. The weights are then averaged at the server every 24 hours and the model is updated with these new weights and a checkpoint file for this model is sent back to the client devices. The stored weights are then deleted.

For evaluation, each time the model is updated, it is evaluated on a constant test set and the metric (Root Mean Squared Error) is stored in the Firebase Database.

Results:

While the performance of the global model has already been defined in the section discussing the model, the entire federated model is currently in its testing phase.

We have distributed the android application to 45 active users and the global model is being updated every 24 hours depending upon the inputs received from the crowdsourced application.

The final android application will be released on the Google Play Store soon after the conceptual working of federated learning is validated by our work.

Deliverables include:

1. Application Demo: **Final Application Demo**
2. Website consisting of all datasets and labels, Tutorial on implementation of federated learning with android: **http://vision-air.github.io**
3. Codes to all the implementations in the report - **https://github.com/Vision-Air**

Conclusion:

The developed application uses a novel method to achieve high-accuracy predictions without compromising user data. Once validated, the application presents a working demonstration of the concept of federated learning, a fairly new yet powerful concept, especially with regards to crowdsensing applications.The concept of developing a generalised model, having analysed the behaviour of HDR images is also of utmost importance to individuals who wish to work related domains.The proposed solution also enables researchers in this field to use the dataset to further their work so as to encourage the development of a better solution to solve the issue at hand.

Acknowledgments:

We would like to express sincere gratitude to our mentor, Dr. Aakanksha Chowdhery whose constant guidance and critical feedback encouraged us to develop a thorough understanding and scientifically valid implementation of our project. We would also like to offer a special thanks to Dr. Brejesh Lall for

providing us with invaluable suggestions that greatly enhanced the quality of our project. We are also particularly grateful to our mentors at the Indian Institute of Technology, Delhi; Dr. Karuna Sunami, Dr. Vinay Kaushik and Dr. Prerana Mukherjee for their constant support throughout the entire duration of the project. Our special thanks are extended to the Staff of the Indian Institute of Technology, Delhi for providing us resources and a venue for our internship. Finally, We would also like to thank the Celestini Program, under Google and The Marconi Society for giving us such a valuable learning opportunity.

References:

1. Liu C, Tsow F, Zou Y, Tao N (2016), *Particle Pollution Estimation Based on Image Analysis*, PLoS ONE 11(2): e0145955. DOI: https://doi.org/10.1371/journal.pone.0145955

2. Zhongang Q., Tianchun W., Guojie S., Weisong H., Xi L., Zhongfei (Mark) Z., *Deep Air Learning: Interpolation, Prediction, and Feature Analysis of Fine-grained Air Quality,* IEEE Transactions on Knowledge and Data Engineering. DOI: 10.1109/TKDE.2018.2823740

3. L. Liu, W. Liu, Y. Zheng ,H. Ma, C. Zhang, *Third-Eye: A Mobilephone-Enabled Crowdsensing System for Air Quality Monitoring*, Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies archive Volume 2 Issue 1, March 2018. DOI: 10.1145/3191752

4. Paul E. Debevec, Jitendra Malik, *Recovering High Dynamic Range Radiance Maps from Photographs*, http://www.pauldebevec.com/Research/HDR/debevec-siggraph97.pdf

5. J. Mao, U. Phommasak , S. Watanabe and H. Shioya, *Detecting Foggy Images and Estimating the Haze Degree Factor,* Journal of Computer Science & Systems Biology. https://www.omicsonline.org/open-access/detecting-foggy-images-and-estimating-the-haze-degree-factor-jcsb.1000226.pdf

6. H. Brendan McMahan, E. Moore, D. Ramage, S. Hampson, B. Agüera y Arcas, *Communication-Efficient Learning of Deep Networks from Decentralized Data*. arXiv:1602.05629v3

7. S. Mallick, *High Dynamic Range (HDR) Imaging using OpenCV (C++/Python),* https://www.learnopencv.com/high-dynamic-range-hdr-imaging-using-opencv-cpp-python/