AirCloud: A Cloud-based Air-Quality Monitoring System for Everyone

Yun Cheng*, Xiucheng Li*, Zhijun Li*, Shouxu Jiang*, Yilong Li†, Ji Jia†, Xiaofan Jiang
* Harbin Institute of Technology, Harbin, China
† Air Scientific, Beijing, China
{chengyun.hit, xiucheng90}@gmail.com, {lizhijun_os, jsx}@hit.edu.cn
{yilong.li, ji.jia}@air-scientific.com, fxjiang@gmail.com

Abstract

We present the design, implementation, and evaluation of AirCloud – a novel client-cloud system for pervasive and personal air-quality monitoring at low cost. At the front-end, we create two types of Internet-connected particulate matter ($PM_{2.5}$) monitors – AQM and miniAQM, with carefully designed mechanical structures for optimal air-flow. On the cloud-side, we create an air-quality analytics engine that learn and create models of air-quality based on a fusion of sensor data. This engine is used to calibrate AQMs and miniAQMs in real-time, and infer $PM_{2.5}$ concentrations. We evaluate AirCloud using 5 months of data and 2 month of continuous deployment, and show that AirCloud is able to achieve good accuracies at much lower cost than previous solutions. We also show three real applications built on top of AirCloud by 3rd party developers to further demonstrate the value of our system.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Experimentation, Measurement, Algorithm, Performance

Keywords

Air Quality, PM2.5, client-cloud calibration system

1 Introduction

Air is one of the most important shared resources on our planet. Unfortunately, the quality of air has deteriorated significantly over the past years, especially for metropolitan cities in developing countries, such as Beijing and New Delhi. Among the various dimensions of air quality, particulate matter (PM) with diameters less than 2.5 micron, or $PM_{2.5}$, has gained a lot of attention recently, partly because of its significant impact on our respiratory systems, and is the focus of this paper. Medical studies have shown that $PM_{2.5}$ can be easily absorbed by the lung, and high concentrations of $PM_{2.5}$ can lead to respiratory disease [14] or even blood diseases [25].

Figure 1. AQM and miniAQM – Internet-connected sensing front-end.

As a result, people are looking for better ways to monitor the quality of air in their immediate environment in order to take appropriate actions such as wearing masks or staying at home. While there are many smartphone applications that report publicly-available air quality data at the city or district-level, they cannot tell the actual air quality people breath-in, which is much more relevant and valuable. This is particularly important since we spend most of our time inside enclosed spaces such as homes and offices, where the air quality may deviate significantly from the outside.

Existing particulate matter monitors fall into roughly two categories: 1) microbalance PM monitoring stations that are very accurate, but are large and expensive (on the order of 50K-100K dollars) [8]; and 2) portable light-scattering based PM monitors with varying accuracies and costs between 300-10K dollars, which are still too expensive for most individuals or for deployment at scale. Over the past year, we start to see inexpensive monitors based on dust sensors. But they typically rely on static calibration curve, resulting in large

SenSys'14, November 3–5, 2014, Memphis, TN, USA.
Copyright 2014 ACM 978-1-4503-3143-2/14/11 ...$15.00
http://dx.doi.org/10.1145/2668332.2668346

errors.

Because of the intrinsic accuracy-cost tradeoff of PM sensors, it is challenging to be both accurate and affordable. In this project, AirCloud takes a novel cloud-based approach to this problem. Instead of using accurate but expensive standalone sensors, AirCloud is a client-cloud system consisting of custom-designed sensor front-ends and air quality analytics engine in the cloud, as described in more detail in Section 3. In addition to using data from our own sensors, we collect a fusion of data, such as meteorology data, location data, and etc., which are used by our calibration and inference algorithms in the analytics engine.

We created two versions of the sensor front-ends, **AQM** and **miniAQM**, as seen in Figure 1. **AQM** is the stationary version, and connects to our backend via Ethernet and GPRS. AQM can be placed inside offices or outside on lightpoles or other fixtures. We deployed a set of AQMs across regions of the city to provide baseline data for our air quality model. **miniAQM** is the portable version, which connects to the smartphone via Bluetooth 4.0. Our smartphone app relays raw dust sensor readings, plus the smartphone's own sensor readings such as GPS and IMU, to our cloud via 3G or WiFi. miniAQM can be easily carried around, used in cars, cubicles, or cafes, and enables us to collect mobile and crowd-sourced air quality data from large number of individuals. The mechanical structure and air-flow inside AQM and miniAQM are carefully designed based on 5 months of experimentation, as described in more detail in Section 4.

On the cloud side, we use a number of data analysis and machine learning techniques for signal conditioning, sensor calibration, and inference. Using data collected over 5 months, we built a analytics engine consisting of calibration and inference models. In this engine, we use Artificial Neural Network (ANN) for online sensor calibration; then we use Gaussian process to further improve the accuracy of sensors and infer the value at locations where sensors are not available. We consider different classes of sensors as our input, and model their accuracies, such that the more accurate classes of sensors have greater influence than the less accurate classes of sensors. We describe our cloud-side in more detail in Section 6.

By building a model of particulate matter using "big sensor data", crowd-sourcing data from sensing frond-ends, and by offloading analytics to the cloud, AirCloud is able to achieve good accuracies at much lower cost than any previous solutions. And in addition to accurate particulate measurement for end users, AirCloud also provides a set of APIs for the third party developers to create applications on top of our systems, as described in more detail in section 8.

We make the following contributions in this paper:

- Design, implementation, and evaluation of two types of Internet-connected $PM_{2.5}$ monitors—AQM and miniAQM with carefully designed mechanical structure for optimal air-flow.
- A cloud-based Air Quality Analytics Engine that takes in multi-dimensional sensor data, and uses several machine learning techniques to calibrate sensors, and accurately compute and infer $PM_{2.5}$ measurement.
- A real deployment over 2 months, which provides valuable training and testing data for our models, and help evaluate our overall system.
- Several useful air-quality related mobile applications building on top of our AirCloud platform.

2 Related Work

There are generally two approaches to measure $PM_{2.5}$—direct monitoring and indirect estimation, each with their advantages and disadvantages.

2.1 Direct Monitoring

Satellite remote sensing of surface air quality has been studied intensively in past decades [19]; this method can help people obtain a general idea of the air quality of the surface. However, this category of methods are easily influenced by clouds and would be sensitive to other factors, such as humidity, temperature, and location. In addition, the results inferred from satellite images only reflect the air quality of atmosphere rather than the ground air quality that people care more about.

To get a clear image of air quality, most countries nowadays deploy air quality stations. These official air-quality stations usually use TEOM (Tapered Element Oscillatory Microbalance) [21] to measure the air quality by weighting $PM_{2.5}$ concentrations accumulated on a filter over several hours. This type of devices are large and expensive (on the order of 300K dollars). A more affordable class of monitors are based on light-scattering method, and cost between 300-10K dollars. For example, `Dylos` [3] is a popular portable particle counter costing around 300 dollars while the TSI-3330 [10] is more accurate but costs around 5000 dollars. While 300 dollars is acceptable for some individuals, it is still too expensive for most people or for dense deployments. Over the past year or two, we start to see a number of dust-sensor based devices aimed at particulate monitoring. These devices typically cost around 100 to 200 dollars. However, by empirically evaluating these devices [4] [5], we found that they rely on a static calibration curve, together with poorly designed air-flow mechanisms, resulting in completely unusable readings under many conditions (with error as large as 300%).

2.2 Estimation and Inference

In the past of years, two major classical ways are mostly used to calculate the air quality of a location. One is interpolation using reports from nearby air quality monitor stations. This naive method is usually employed by public websites releasing the air quality index (AQI). As air quality varies in locations non-linearly, the inference accuracy is quite low. In [29], the author claims that, according to the statistics on the AQI recorded from Jan. 1, 2013 to Jan. 1, 2014, the average deviation between the maximum and minimum readings of $PM_{2.5}$ from the 22 official stations at the same timestamp can easily exceed 120. In addition, over 50% of time, the deviation is larger than 100. Since 100 almost denotes a two-level gap, when the air quality of a location is moderate, another one could be unhealthy. The other is classical dispersion models, such as Gaussian Plume models, Operational Street Canyon models, and Computational Fluid Dynamics. These models are in most cases a function of mete-

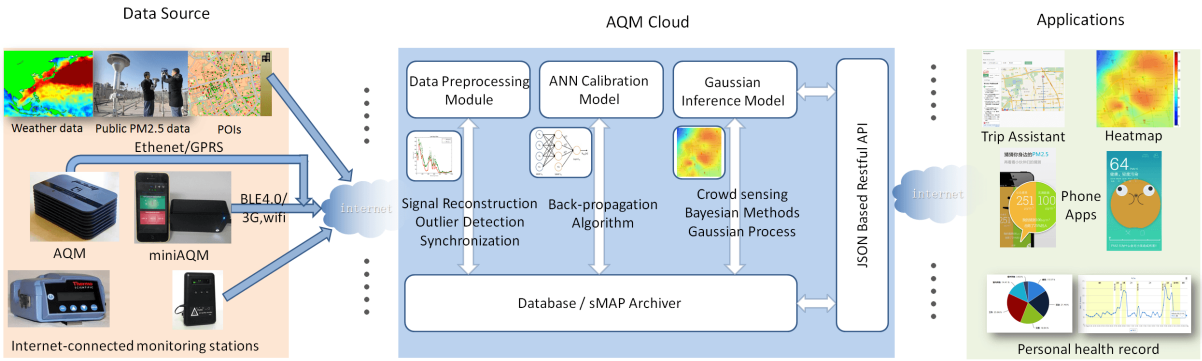


Figure 2. Architecture of the AirCloud system.

orology, street geometry, receptor locations, traffic volumes, and emission factors (e.g., g/km per single vehicle), based on a number of empirical assumptions and parameters that might not be applicable to all urban environments, however, these parameters are difficult to obtain precisely and the result is not very accurate either [27].

Recently, big data reflecting city dynamics have become widely available and a group of researchers seek to infer the air quality using machine learning and data mining techniques. In the "U-Air" paper by Yu Zheng et al. [30], the authors infer air quality based on AQIs reported by a few public air quality stations and meteorological data, taxi trajectories, road networks, and POIs (Point of Interests). However, because they estimate AQI using a feature set based on historical data, their model cannot respond quickly enough to the change in $PM_{2.5}$ concentration, which often changes on an hourly basis, leading to large errors at times.

There are also crowdsourcing or participatory sensing approaches [17][18] to solve the air quality monitoring problem. This approach has only been applied into detecting some gasses like CO_2. As far as we know, no individuals or groups adopted this method in monitoring $PM_{2.5}$ so far as the devices designed in the past for sensing such kinds of air pollutants are not easily portable.

We approach this problem using a combination of direct monitoring, modeling and inference, and crowd-sourcing.

3 System Overview

AirCloud uses a heterogeneous set of data sources as inputs. These data are stored and analyzed by the **air quality analytics engine** in the cloud, to provide accurate device calibration and fine-granularity estimation based on GPS-location. AirCloud further provides a number of web services and **APIs** for third-party **applications** developers.

3.1 Data Sources

There are various types of data sources in our system. We collect public weather data to learn the relationships between weather information such as temperature, humidity, pressure, wind speed and air quality. In order to train the calibration and the inference model, the system also has several classes of air quality data sources. The public air quality stations use very expensive instruments and the data is the most accurate. `Thermo` [8] is less expensive but almost as accurate as the public stations. The system also has less accurate air quality sensors such as `Dylos`. `Thermo` is used to train the calibration model and predict the unknown values in the online calibration and inference algorithm, we also use it as the ground truth to evaluate the performance of the online calibration model. But to get a more dense coverage of $PM_{2.5}$ spatially, we design and implement our own $PM_{2.5}$ instruments (AQM and miniAQM). It is much less expensive and we can deploy them densely in our city. Using AirCloud crowd-sourcing and our calibration & inference algorithms, AirCloud provides accurate, calibrated results for AQMs and minAQMs.

3.2 Data Representation and Storage

To describe multiple kinds of data sources in an uniform format, and to store and query this data efficiently, we use sMAP [16] as the data representation and storage system. It's designed as the standard specification for physical sensor data. sMAP provides an efficient database for time-series data, plus a powerful query language.

We need to reform the calibration and inference online, however, we also have to ensure response speed and security. According to these specific needs, we add a backend asynchronous service and authentication module to the system.

3.3 Air Quality Analytics Engine

Air Quality Analytics Engine is the data processing and mining module, which contains the offline training model and online calibration and inference models, it takes the raw sensor data and computes the accurate calibration and inference results.

It can be observed that $PM_{2.5}$ is heavily influenced by meteorology factors [30]. This knowledge inspires us to exploit the dependencies between the sensor error and these meteorology factors. We collected 5 months of data which contains AQM sensor data, `Thermo` ground truth data and meteorology data, to train an ANN model. From this model, AirCloud learns the non-linear relationship between AQM sensor readings and `Thermo` readings.

As shown in Figure 2, the system first uses the data preprocessing module to reconstruct the raw sensor data. Next, using the offline model we have trained using historical data, AirCloud computes a calibrated $PM_{2.5}$ reading using real-time AQM data and weather data. Lastly, an online infer-

ence model based on Gaussian Process is used to infer and predict $PM_{2.5}$ concentrations at locations where direct sensors are not available, or to further calibrate AQM sensor. We describe the details in Section 6.

3.4 APIs and Applications

AirCloud provides several web services and RESTful APIs for developers. For example, `GPS-AIR` interface can help developers get accurate $PM_{2.5}$ estimation value if the GPS location data is valid; `Device-Calibration` interface can give back the device calibration result; `Data-Driver` interface can help developers easily add, get or search data from our database.

During two hackathons we have hosted, several smartphone applications have already been developed (some already online in the Apple App Store) that use our APIs to get calibrated $PM_{2.5}$ data. We describe some of them in Section 8.

4 AQM and miniAQM $PM_{2.5}$ Frontends

$PM_{2.5}$ concentrations vary significantly over space, especially for metropolitan cities where pollution sources are multi-faceted [30]. In addition, as we can observe from official $PM_{2.5}$ monitoring station data, $PM_{2.5}$ concentration changes at an hourly rate. As a result, direct monitoring is necessary.

However, we find that none of the existing monitors satisfy our needs, they are either overly expensive or not accurate enough. To solve this problem, we designed and built our own Internet-connected $PM_{2.5}$ monitors, AQM and miniAQM, as shown in Figure 1. The AQM costs about $60 at 10K quantity. We take a novel approach of using inexpensive sensors at the front-end, but rely on the analytic algorithm in the cloud to improve the accuracies.

There are three kind of data sources in all:

- **Public data:** weather data, public $PM_{2.5}$ data and POIs, which can be obtained using web crawler;
- **Internet-connected monitoring stations:** we use `Thermo` and `Dylos` connected with laptop as our highly accurate data sources;
- **AQM and miniAQM:** these are our designed low-cost monitoring devices

We will describe AQM and miniAQM in details in the following subsections.

4.1 Sensor Selection

Because of own cost constraints, sensors suitable for our front-ends are typically FIR-based light-reflecting sensor, commonly called dust sensor, which are used extensively inside air purifies as a rough indicator of air quality. Several choices are available, such as SHINYEI, SHARP, and etc. We evaluated most of the popular ones over various air conditions and found that SHINYEI `PPD42NJ` produced the most consistent results. We installed two `PPD42NJ` inside AQM, on opposite sides and facing opposite directions. This design choice enables AQM to self-calibrate, and increases the speed of converging to a usable value. This dust-sensor arrangement also enables us to automatically detect errors and sensor failures during long-term deployment. For miniAQMs, only one `PPD42NJ` is used to save space since portability is a priority for miniAQM.

4.2 Mechanical Design

Over the three months of designing AQM and miniAQM hardware, we found that the mechanical structure plays a significant role on the data accuracies of this type of sensors. In particular, the air flow determines how long and how fast the particles pass through the "detection window" of the sensor, and needs to be carefully controlled. Ideally, an air pump should be used to maintain constant air flow; but air pumps are too expensive. Instead, we experimented with many different designs, both passive and active air flow. Our first design completely exposes the sensors. This worked well most of the time, but does not cope well with high speed wind or if large particle become dominant. We then experimented enclosure holes and strain screen, in an attempt to decrease the effect of wind. But we found that this design often leads to a self-circulating local environment, and won't exchange air with the outside. Our final design is using a fan running at a low fixed speed to emulate the air pump, as shown in Figure 1. We install two `PPD42NJ` sensors inside AQM on opposite sides and facing opposite directions. In `PPD42NJ` sensor, air is self-aspirated with the current of air generation mechanism with a built-in heater, so if we place the fan around AQM device, the fan will affect the air flow, and we also do experiment to evaluate this hypothesis. Finally, we choose to place the fan on top of the device to ensure exchanging air with the outside to prevent a self-circulating local environment. The main evaluation criterion is whether the `PPD42NJ` sensor can respond quickly and have the correct trend.

4.3 Data Communication

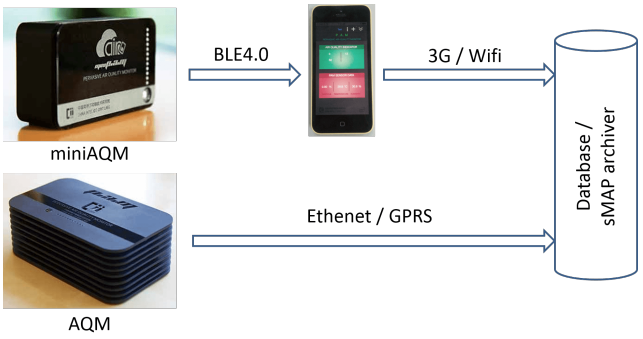


Figure 3. Communication between front-ends.

We use different communication approaches to connect AQM and miniAQM with cloud server. AQM, shown on the bottom of Figure 3, is the stationary version, that connects to the cloud via Ethernet or GPRS. miniAQM, shown on the top of Figure 3, is the portable version, which connects to the smartphone over Bluetooth 3.0 or 4.0 (both supported), and to the cloud via the mobile phone's data connection, such as 3G or WiFi.

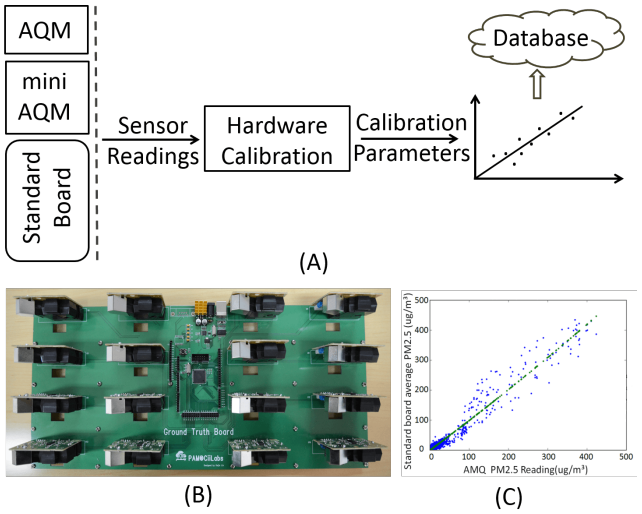


Figure 4. The hardware calibration process. (A) the hardware calibration procedure; (B) standard sensor board; (C) fitting result.

Figure 5. The air chamber used for hardware calibration. Pollution source and air purifier are used to change $PM_{2.5}$ concentration, while air conditioning is used to change the temperature and humidity, we use the sensor readings across a wide range to do the hardware calibration.

4.4 Hardware Calibration

From our experimental data, we find that while the PPD42NJ sensors we used exhibit the same trends, there are still significant variations between them, and require calibrating them against a reference sensor, at various $PM_{2.5}$ concentrations. In order to calibrate them, we built an 10 m^3 air chamber with full internal climate control, as shown in Figure 5. Using this air chamber, we can manually vary the $PM_{2.5}$ concentrations across a wide range, and calibrate all the sensors together. Figure 4(a) shows the hardware calibration procedure. We use two order polynomial fit to calibrate error between different PPD42NJ sensors. Figure 4(c) shows the fitting result of one PPD42NJ sensor with the standard one. We use the average $PM_{2.5}$ concentration of a sensor board which contains 16 PPD42NJ sensors as the standard sensor reading, as shown in Figure 4(b). When we obtain the hardware calibration parameters, then it will be stored in a database. Before each deployment, we calibrate each PPD42NJ sensor to this standard board to remove initial hardware variations.

5 Cloud-side System Design

To store and query data efficiently, we use sMAP [16] to define the data format and store the data. In the following subsections, we will first describe sMAP, then details of our data exchange framework.

5.1 sMAP

sMAP [16] is a simple measurement and actuation profile for physical information, which enables the simple and efficient exchange of sensor data. It's a specification for transmitting physical data and describing its contents, which also provides tools for building, organizing, and querying large repositories of physical data. sMAP allows instruments and other producers of physical information to directly publish their data, and it provides powerful RESTful service to get the data from sMAP archiver. The repository gives a place for instruments to send their data. It supports the following features:

- Efficient storage and retrieval of time-series data
- Maintenance of metadata using structured key-value pairs
- Metadata querying using ArdQuery [7] language

The core object in sMAP is *timeseries*, a single progression of (time, value) tuples. Each *timeseries* in sMAP is identified by a UUID, and can be tagged with metadata; all grouping of *timeseries* occurs using these tags. These objects are exchanged among all components in this ecosystem.

sMAP has been online for at least three years and has proven to be robust; all the above features make it an ideal choice for our system.

5.2 Data Format, Authentication, Storage and Web Services

Figure 6 shows the system data exchange framework, which contains three parts: 1) data sources exchange framework. We define the data format according to sMAP specification and add authentication module; 2) system internal data exchange framework. We mainly use JSON format to exchange data among different system components; 3) API data exchange framework. To improve efficiency, we store result in RAM and use authentication module to verify the user. We will describe these in details in the following section.

We use sMAP as our storage scheme, so we define the source data format, as shown in Figure 7. According to sMAP specification, each *timeseries* is globally identified by a Universal Unique Identifier (UUID), which is a 128-bit name. Together with time-series readings, which contains UNIX timestamp and value, system can easily store the data. Time series are uniquely identified by UUID. However, these identifiers are unpleasant to use in practice. We add additional metadata to be attached as tags: structured key-value pairs, as shown in Figure 7. We add detailed and

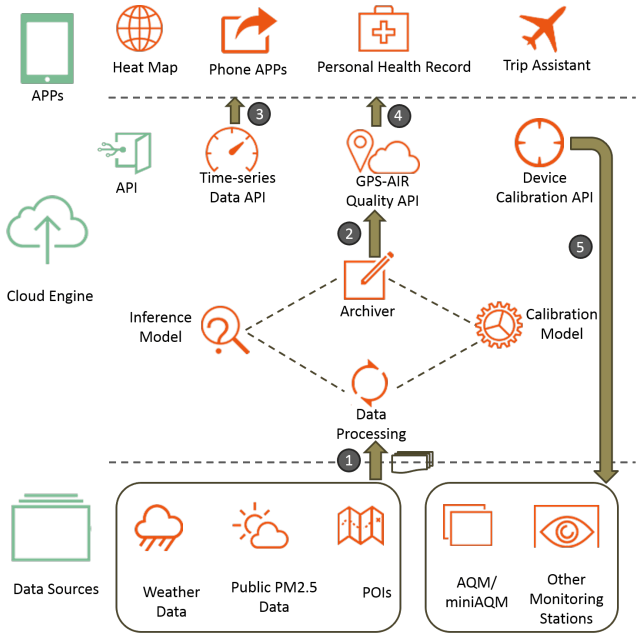


Figure 6. Data Exchange Framework.

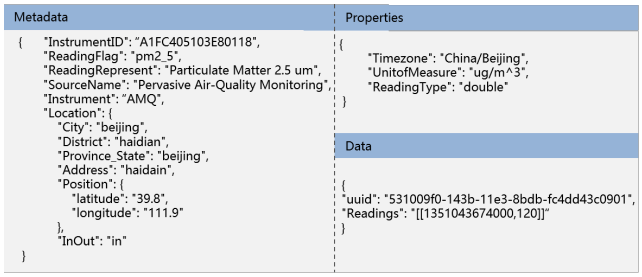


Figure 7. The Hardware Data Representation.

hierarchical metadata, such as *instrumentID* and location, to make it easier and natural to retrieve data in the future. The detailed location and *InOut* status also provide information to the backend algorithm and the visualization service. To make the system more secure and stable, we add authentication module and data schema check module in the data processing framework. In Figure 6, all uploading data in ① have the same data format and use specified key as authentication token.

InstrumentID definition(64 bits total; each letter below represent a hex digit)

RRAA BCCX DDDD EEFX

RR: Reserved / 8bits / 'A1' = default
AA: Manufacture ID / 8bits / 'FC' = Flex
B: Year counting from 2010 / 4bits / '4' = 2014
CC: Week of the year / 8bits / '05' = Jan. 23
X: First 4bits of CRC / 4bits
DDDD: Sequence number / 16bits / '03E8' = Start counting from 1000(DEC)
EE: Model / 8bits / '01' = AQM; '02' = miniAQM
F: Version / 4 bits / '1' = Revision 1
X: Second 4bits of CRC / 4bits

Figure 8. *InstrumentID* definition.

InstrumentID uniquely identifies an instrument, such as AQM or miniAQM, and can be associated with multiple UUIDs - an instrument may have more than one data stream. Because an user may manually enter the *InstrumentID* to associate the device to his/her account, we designed the format for *InstrumentID* in a way that enables (trained) humans to read out useful information visually, while preventing accidental input errors via CRC check, as shown in Figure 8.

When the backend server receives the data stream, it will use the cloud analytics engine to process the sensor data, produce the calibrated result and send to sMAP archiver. To improve the response speed, we choose asynchronous celery [1] workers to do the jobs in the backend and use tornado [9] web server as the frontend to deal with basic connection jobs. The system use JSON format to exchange data among various system components.

In Figure 6, ③, ④ and ⑤ represent the system API services. We have three different kinds of APIs:

- **Device Calibration API:** each AQM, miniAQM or other supported devices will use this API to get the calibration parameters. A key is required for authentication and JSON is the data format;
- **Time-series Data API:** developers can use this API to get time-series data. All data are presented in JSON format and use the authentication module;
- **GPS-AIR Quality API:** developers can use this API to get accurate $PM_{2.5}$ concentrations of the current location. To improve efficiency, we store the heatmap result in RAM once it's produced.

6 Air-Quality Analytics Engine

The cloud-based Air-Quality Analytics Engine mainly consists of three components: 1) a signal reconstruction module designed to denoise and smooth the corrupted original sensor signal; 2) an ANN based calibration model aiming to enhance the accuracy of AQM and miniAQM in real time; 3) an online inference model based on Gaussian Process that encompasses various source data to further improve the accuracies and to provide $PM_{2.5}$ estimation for places where sensors are not available. The overall framework is shown in Figure 9.

6.1 Signal Reconstruction

Due to the instability of the sensor itself and the extra noise added in transmission, the original signal is extremely unstable as shown in Figure 10. Therefore eliminating the noise and reconstructing the real signal is the first step . Formally, we represent the signal as $x \in R^n$ (n is the length of the signal) and assume that signal x is corrupted by an additive noise v:

$$x_{cor} = x + v$$

in which x_{cor} is the original corrupted signal uploaded by the sensor and we simply assume that the noise v is an unknown, small, and rapidly varying random variable. The goal here is to form an estimate $\hat{x}$ of the original signal x, given the corrupted signal x_{cor}.

The signal reconstruction could then be formulated as the bi-criterion problem [15]

$$\text{minimize } (w.r.t.\ R_+^2)\ (\|\hat{x} - x_{cor}\|_2, \phi(\hat{x})) \qquad (1)$$

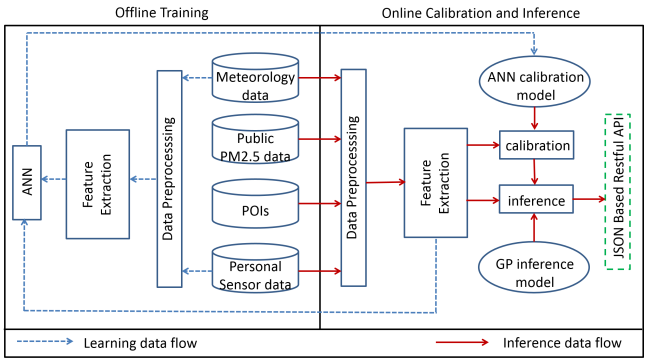


Figure 9. Framework of the Air-Quality Analytics Engine. The Data Preprocessing module will smooth the data flow before feeding it to ANN or GP inference model; ANN is trained offline and updated online; The GP model fuses multiple source data to make inference.

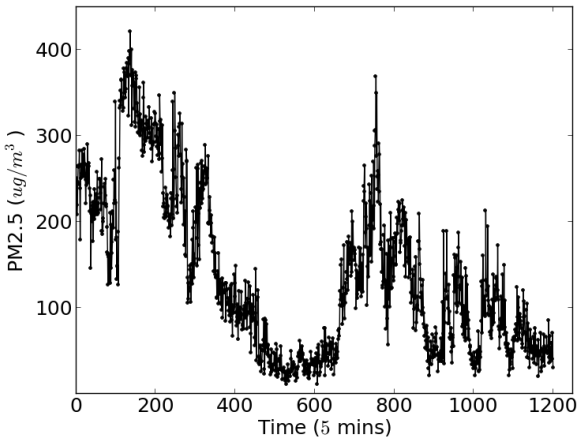


Figure 10. The original signal of `PPD42NJ` during about five days where the signal is sampled every 5 minutes.

where $\hat{x}$ is the optimization variable and x_{cor} is the parameter of the optimization problem. The function $\phi : R^n \to R$ is a convex function and can be treated as smoothing objective. It is normally meant to measure the roughness of the estimate $\hat{x}$. The optimization problem (1) seeks signal that are close to the corrupted signal, that are smooth, for which $\phi(\hat{x})$ is small.

There are multiple choices for the smooth objective function ϕ and the simplest one is a quadratic smoothing function

$$\phi(x) = \sum_{i=1}^{n-1} (x_{i+1} - x_i)^2 = \|Dx\|_2^2$$

where $D \in R^{(n-1)\times n}$ is a banded matrix (and n is the length of the signal)

$$D = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -1 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & -1 & 1 \end{bmatrix}$$

The optimal estimate $\hat{x}$ can be obtained by solving a convex optimization problem [15]

$$\text{minimize } \|\hat{x} - x_{cor}\|_2^2 + \delta\|D\hat{x}\|_2^2 \qquad (2)$$

where $\delta > 0$ is a trade-off parameter trading between $\|\hat{x} - x_{cor}\|_2^2$ and $\delta\|D\hat{x}\|_2^2$. A large δ typically implies a high penalty in $\delta\|D\hat{x}\|_2^2$. Therefore a larger δ normally leads to a smoother estimate $\hat{x}$ while a smaller δ makes $\hat{x}$ more consistent to the observed signal x_{cor}.

The solution of this quadratic problem can be obtained analytically as

$$\hat{x} = (I + \delta D^T D)^{-1} x_{cor}$$

in which I is an identity matrix and $(I + \delta D^T D)$ is tridiagonal. Therefore the solution can be computed very efficiently. Actually, since $(I + \delta D^T D)$ is symmetric and positive definite it can be factorized via Cholesky factorization, the complexity of computing $\hat{x}$ is $\Theta(nk^2 + 4nk)$ [15] where k is the bandwidth of $(I + \delta D^T D)$, k is 3 in our case, and n is the length of the signal we handle in response to each request.

In theory, a larger n tends to provide a smoother reconstruction signal $\hat{x}$ but requires much more computing resource. In our scenario the sensor data is coming in real time and we have to handle them dynamically. Therefore the choice of n is a compromise between the smoothness and efficiency. The effectiveness of the signal reconstruction will be shown in Section 7.

6.2 Calibration Model Based on Neural Network

In the Air-Quality Analytics Engine, the prediction is made based on the data acquired by the sensors. Thus the precision of sensors is critical to the prediction accuracy. Unfortunately, there is usually an error in the data acquired by AQM due to the inherent randomness in particulates through the sensor's focusing window and noises in the devices. Figure 11 shows the concentration of $PM_{2.5}$ given by AQM (`PPD42NJ`, after signal reconstruction) and the standard device (`Thermo`) at the same place, and there are obvious errors between them. Therefore, it is necessary to eliminate such errors to improve the accuracy of our engine, i.e., to estimate each $h(x)$ from its observation $\hat{h}(x)$.

There are two observations from Figure 11 that inspire us to design the calibrator. 1) Each `PPD42NJ` reading may correspond to multiple `Thermo` readings, and thus more information is required to calibrate `PPD42NJ`. According to previous work [30], the concentration of $PM_{2.5}$ is heavily influenced by meteorology factors, such as temperature, humidity, pressure, and etc. This empirical knowledge inspires us to exploit these meteorology factors to help calibrate the AQM (`PPD42NJ`); 2) The relationship between

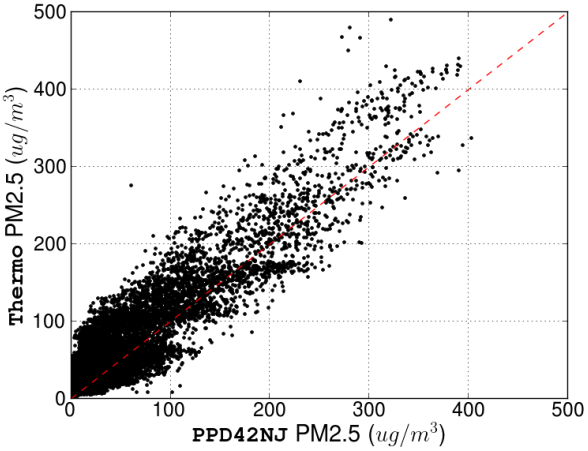


Figure 11. Thermo VS PPD42NJ. A large deviation exists between PPD42NJ's readings and Thermo's readings which we take as ground truth.

PPD42NJ readings and Thermo readings is complex and nonlinear, which makes it difficult to design the calibrator directly, and thus we resort to the learning based method.

Based on these observations, we select to simulate calibrator with a neural network which takes the readings of PPD42NJ with co-located temperature and humidity readings as inputs and the groundtruth value (given by Thermo) as output since the neural network is capable of well fitting an arbitrary function. Many types of neural networks have been investigated and extensively used in practice, among which the deep network architectures are becoming increasingly prevalent recently [12]. But in our application the widely-used back-propagation (BP) network is chosen due to its simplicity and generality [13]. Denoting the readings of PPD42NJ and Thermo as $\hat{h}(x)$ and $h(x)$ respectively, the calibration function can be formulated as $h(x) = g(z(x))$, where $z(x) = [\hat{h}(x),\ H,\ T]^T$, and H, T represents the humidity and temperature respectively.

In the training process, we adopt the stochastic gradient descent algorithm [28] rather than the batch methods like L-BFGS for the following two reasons: 1) computing the cost and gradient for the entire training set is very slow; 2) stochastic gradient descent allows us to incorporate new training data online as the relationship between the input and output is not constantly invariable due to the aging of device and the change of environment. In our Analytics Engine, we keep updating the calibrator by retraining it using online data collected at locations where both a AQM and a Thermo exist.

Once the parameters of the BP network are learned, the prediction can be obtained in $O(1)$ which takes only some simple calculations. Using this model, we effectively calibrated the AQM, which is shown in Section 7.4.

6.3 Online Inference Model

The Neural Network calibration model attempts to improve the accuracy of the sensor by using only local point features. In the online inference model, we seek to incorporate various data sources with different confidence levels, such as the official public monitor station data, Thermo data and Dylos data, to further improve the accuracy of AQM and miniAQM in open areas spatially. And meanwhile it could also provide $PM_{2.5}$ estimation values for the places where the sensors are unavailable.

In [30], Yu Zheng et al. proposed a framework named U-Air to infer the real-time and fine-grained air quality in urban area. The U-Air model takes as input the data with same confidence from a small number of existing monitor stations and is based on co-training learning which is designed to resolve the data sparsity issue. In our scenario, there are a variety of source data with different confidence levels, and as a dense deployment, we can collect a relatively large amount of data for meaningful inference. Therefore, it is not wise to directly apply the U-Air model. In contrast, as [30] has pointed out, the urban quality varies by location non-linearly and depends on multiple factors, such as meteorology, land use, and urban structures. We turn to one of widely used Bayesian methods-Gaussian Process, which can naturally bring these prior information and adaptively integrate the multiple source data into the model.

Gaussian Process: A Gaussian Process is a stochastic process such that any finite subcollection of random variables follow a multivariate Gaussian distribution [24]. Formally, a collection of random variables $\{h(x) : x \in R^n\}$ is said to be drawn from a Gaussian Process with mean function $m(\cdot)$ and covariance function $k(\cdot,\cdot)$ if for any finite set of elements $x^{(1)}, x^{(2)}, \ldots, x^{(m)} \in R^n$, the associated finite set of random variables $h(x^{(1)}), h(x^{(2)}), \ldots, h(x^{(m)})$ follow the distribution,

$$\begin{bmatrix} h(x^{(1)}) \\ \vdots \\ h(x^{(m)}) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(x^{(1)}) \\ \vdots \\ m(x^{(m)}) \end{bmatrix}, \begin{bmatrix} k(x^{(1)},x^{(1)}) & \cdots & k(x^{(1)},x^{(m)}) \\ \vdots & \ddots & \vdots \\ k(x^{(m)},x^{(1)}) & \cdots & k(x^{(m)},x^{(m)}) \end{bmatrix} \right)$$

For simplicity, we denote this as:

$$h(\cdot) = \mathcal{GP}(m(\cdot), k(\cdot,\cdot))$$

$m(\cdot)$ is the mean function which can be any real-valued function and $k(\cdot,\cdot)$ is the kernel function which must satisfy the Mercer's condition. In our implementation, squared exponential function is chosen as the kernel function which is defined as

$$k_{SE}(x,x') = \exp\left(-\sum_{i=1}^{n} \frac{\|x_i - x_i'\|^2)}{2\omega_i^2}\right.$$

in which $x, x' \in R^n$, x_i is the i-th dimension of feature vector x, n is the number of dimension and ω_i is used to control the importance of i-th feature, the kernel function can be treated as a measurement of how similar two feature vectors are, and hence ω_i plays a significantly important role in characterizing the similarity of two feature vectors.

Gaussian Process regression: Given a training set of i.i.d.(independent and identically distributed) examples, $S = \{(x^{(i)}, y^{(i)})\}_{i=1}^N$, from some unobserved distribution. The Gaussian Process regression model can be written as

$$y^{(i)} = h(x^{(i)}) + \varepsilon^{(i)}, i = 1, \ldots, N$$

where the $\varepsilon^{(i)}$ are i.i.d. noise variables (which might be generated by our sensors) with independent $\mathcal{N}(0,\delta_i^2)$ distributions.

The GP representation is very powerful. Given a set of test points, $T = \{(x_*^{(i)}, y_*^{(i)})\}_{i=1}^{N_*}$, drawn from the same unknown distribution as S. We could then derive equation to compute the posterior predictive distribution over the testing outputs y_* by using the properties of multivariate Gaussian distribution and Bayesian approach as [20]

$$\begin{bmatrix} y \\ y_* \end{bmatrix} \Bigg| X, X_* = \begin{bmatrix} h \\ h_* \end{bmatrix} + \begin{bmatrix} \varepsilon \\ \varepsilon_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K_y & K_* \\ K_*^T & K_{**} \end{bmatrix}\right)$$

where $K_y = K(X,X) + diag(\delta_i^2)$ is $N \times N$, $K_* = K(X,X_*)$ is $N \times N_*$, and $K_{**} = K(X_*,X_*) + diag(\delta_i^2)$ is $N_* \times N_*$. Each entry of this matrix is computed as

$$K(X,X)_{ij} = k_{SE}(x^{(i)}, x^{(j)}), K(X,X_*)_{ij} = k_{SE}(x^{(i)}, x_*^{(j)})$$

By the standard rules of conditional Gaussians, the posterior distribution has the following form $(y_*|y,X,X_*) \sim \mathcal{N}(\mu^*, \Sigma^*)$

$$\mu_* = K_*^T K_y^{-1} y \tag{3}$$

$$\Sigma_* = K_{**} - K_*^T K_y^{-1} K_* \tag{4}$$

The Gaussian Process is non-parametric which is able to model essentially arbitrary functions of the input points, and thus if we take the distribution of $PM_{2.5}$ value of a city as an random distribution, the Gaussian Process would enable us to model this distribution at any given time dynamically. Equation (3) guides us to compute the predictions of $PM_{2.5}$ value in a specific place and Equation (4) indicates the uncertainty of the predictions.

In the spatial inference model, we divide the region into grids and collect air quality related features for each grid. The features consist of the GPS coordinates, location-related humidity, temperature, POI (points of interest, Table 1 indicates the category of POIs which we studied in our experiment). Intuitively, a grid tends to show similar air quality with its neighbors and the grids, just as [30] has pointed out that the similar meteorology and POI normally take the same air quality level. However, since these features do not have the same impact on the $PM_{2.5}$ treating them equally is not a smart way, and it is necessary to select different ω_i in the kernel function to distinguish these features. Here, we consider an empirical Bayes approach, which allows us to use continuous optimization methods. In particular, we try to maximize the marginal likelihood function [20]

$$\log p(y|X) = -\frac{1}{2} y K_y^{-1} y - \frac{1}{2}\log|K_y| - \frac{N}{2}\log(2\pi) \tag{5}$$

The kernel parameters learned by maximizing the marginal likelihood could adaptively adjust their magnitude which indicates the importance of each feature and also reflect the influence of the urban structure to $PM_{2.5}$.

Considering that the various data sources reported by different kinds of devices have different confidence levels, this reflects the different noise levels of the observed data. Specifically, the more reliable the device is, the lower noise it owns. Therefore we use a diagonal matrix $diag(\delta_i^2)$ in Equation (3) (contained in K_y term) to represent the noise. In this fashion, the ultimate value of a device depends on not only itself but the values of its nearby devices.

Table 1. Category of POIs

C1: Culture & education	C5: Shopping malls and Supermarkets
C2: Parks	C6: Entertainment
C3: Sports	C7: Decoration and furniture markets
C4: Hotels	C8: Vehicle Services(pas station, repair)

Since both computing μ_* in Equation (3) and maximizing the marginal likelihood function in Equation (5) involve computing K_y^{-1}, the complexity of our inference model is $\Theta(N^3)$ [24]. For the present scale of our deployment, the number of sensors N is targeted within 1000, so the exact computation is still our best choice but if N is larger than 1000, the approximation methods would be more appropriate. In an effort to overcome scaling problems, a range of sparse Gaussian Process approximations have been proposed [26] [22] [23]. The paper [23] stated that these approximation methods can give a satisfactory performance while only takes $O(M^2N)$ time, where M is a user-specifiable parameter. The approximation approaches will be considered in our future work.

7 Evaluation and Results

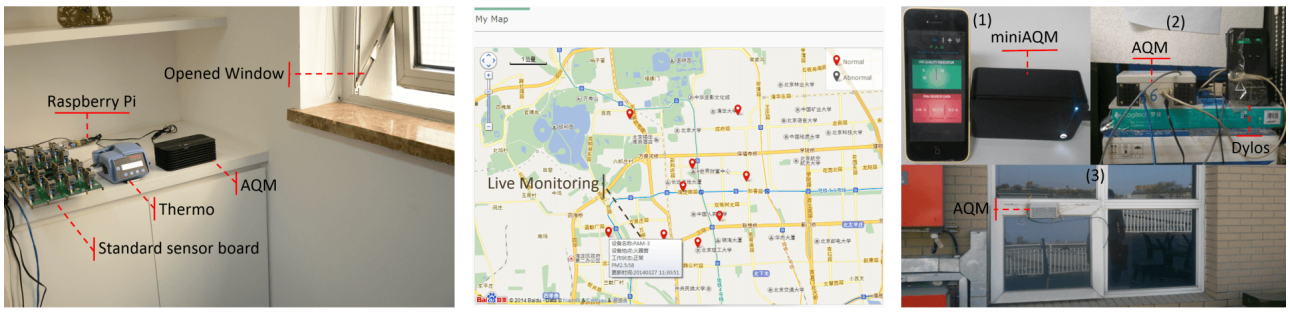
In this section, we present our evaluation methodology and experimental results. First, we describe the deployment setup and the definition for $PM_{2.5}$ levels. Next, we evaluate the effectiveness and performance of the signal reconstruction and the neural network calibration model using historical dataset collected over 5 months. Finally, the online inference model and the overall online analytics engine are evaluated through a real deployment over two months, followed by a detailed heatmap of our deployment.

7.1 Experiment Setup and Datasets

To collect sufficient data to train our ANN calibration model, we set up a data collection module. Figure 12(a) shows the training data collection setup, `Thermo` and several `PPD42NJ` sensors are placed at the same location near an opened window. `Thermo` is our most accurate device, and is used as ground truth. `Thermo` readings, `PPD42NJ` readings, temperature and humidity data are recorded every minute over 5 months. This historical dataset is used to train and evaluate the neural network calibration model.

A real deployment of 10 AQMs, 2 miniAQMs, 3 `Dylos` and 2 `Thermo` over two months is used to evaluate the performance of the calibration and inference algorithms. We select an area of 4km $\times$ 4km which covers all kinds of environments. Figure 12(b) and Figure 12(c) show the locations we selected to deploy and the photos of the deployment environments.

We try to deploy AQM stations evenly inside the deployment area, but also include a wide range of environments, such as school, park, residential area, and commercial area. Each location corresponds to a particular GPS coordinate, POI, and sensor readings, which will be detailed in the following subsections. Apart from the fixed AQM sensors, we



(A) (B) (C)

Figure 12. AQM experimental environment and deployment. (A) is the setup for collecting training data. We use Raspberry Pi to collect and transfer data of standard sensor board and Thermo, while AQM use internal GPRS module to interact with backend server. They are all placed near opened window. (B) is map of the deployment, it's an web service, we can minitor each device's condition in real-time. (C) includes photo of the deployment-(1) shows miniAQM with accompaning mobile app; (2) shows how we transfer Dylos data using AQM's serial port; (3) is a real deployment environment.

also have 3 mobile miniAQM sensors. minAQMs are carried by humans and transmit sensor readings together with real-time GPS information from the paired smartphone.

In summary, the collected dataset can be divided into two parts:

- Historical dataset: This dataset is collected over 5 months (from August 1, 2013 to December 31, 2013) in the laboratory for the purpose of offline training and verifying the neural network calibration model. It is composed of more than 210,000 tuples (we sample once a minute over four months). Each tuple includes the value of `PPD42NJ`, `Thermo`, humidity and temperature. We divided it into training dataset (70%) and testing dataset (30%).
- The deployment dataset: This dataset is composed of continuous data from 10 AQMs, 2 miniAQMs, 3 `Dylos` and 2 `Thermo` data for about two months (from January 31, 2014 to March 31, 2014).

7.2 Definition of the $PM_{2.5}$ Levels

For end users, dividing the range of $PM_{2.5}$ values into discrete levels may be more useful, as indicated by the official definition of $PM_{2.5}$ levels [6]. Discrete levels also enable better ways for evaluation. However, since the official definition has levels spanning wide range on some levels, we define $PM_{2.5}$ levels in the following ways (level 1-4 is the same as the official one; 5 and 6 equal to official level 5; 7-8 equal to official level 6). We show $PM_{2.5}$ levels in real time on AQM and miniAQM devices via rows of LEDs.

7.3 Evaluation of the Signal Reconstruction Module

The signal reconstruction module aims at recovering a smooth signal $\hat{x}$ from the original noisy signal x_{cor}. As we stated in Subsection 6.1, a larger δ in equation (2) tends to generate a smoother signal while a smaller one gives a relatively rougher signal. If we take `Thermo`'s value as ground truth x with length n, the performance of the signal reconstruction can be evaluated using the Root Mean Square Error,

$$RMSE = \sqrt{\frac{\|x-\hat{x}\|_2^2}{n}}$$

Table 2. $PM_{2.5}$ levels definition

$PM_{2.5}(ug/m^3)$	Levels	Values Levels of Health Concern
0-35	1	Good
35-75	2	Moderate
75-115	3	Unhealthy for sensitive groups
115-150	4	Unhealthy
150-200	5	Highly unhealthy
200-250	6	Very unhealthy
250-350	7	Hazardous
>350	8	Severely Hazardous

Figure 13 presents a piece of signal with length $n = 600$ and the smooth effectiveness of different choice of δ. The *RMSE* of the original with $\delta = 0$ is as large as 208.59, when the δ increases to 25 the *RMSE* falls down to 96.98 and the *RMSE* ceases to decrease largely even if we continue increasing δ. We also evaluate the signal reconstruction via confusion matrix in section 7.4 using historical data. When mapping the numerical data into the $PM_{2.5}$ levels defined in 2, we could obtain the corresponding confusion matrix. Table 3 shows the raw data confusion matrix and Table 4 shows the reconstructed data confusion matrix. The total accuracy of the former is 0.558 while the latter achieve an accuracy with 0.591, a slight increase of 5.9% .

7.4 Evaluation of the ANN Calibration Model

The ANN calibration model is estimated using historical dataset collected in our laboratory. Before we train our model, we firstly reconstruct the signal over the whole dataset. The reconstructed sensor data and `Thermo`'s value (ground truth) are used to fit the neural network as input and output respectively. The calibration model is trained by using 70 percent of the historical dataset; a regularization technique is used to regularize the weight W and improve the ability of generalization.

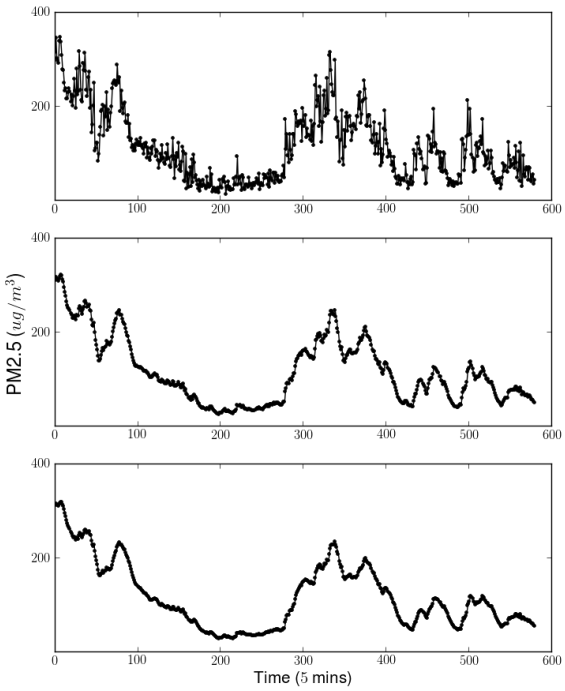


Figure 13. Three reconstructed signal(sampled every five minutes) $\hat{x}$. The top one corresponds to $\delta = 0$ with $RMSE = 208.59$, the middle one to $\delta = 25$ with $RMSE = 96.98$, and the bottom one to $\delta = 50$ with $RMSE = 96.69$.

After the signal reconstruction and mapping the $PM_{2.5}$ readings in testing dataset into the eight $PM_{2.5}$ levels as defined in Table 2, we obtain a reconstructed data confusion matrix [2] shown in Table 4. We then feed the reconstructed data into the calibration model and map the calibrated output data into the same eight $PM_{2.5}$ levels, a calibrated data confusion matrix can be generated, as shown in Table 5. The overall accuracy after signal reconstruction is 0.591; after the calibration model, the overall accuracy is 0.639 - an increase of 7.9% over reconstructed signal, or 14.5% over raw data.

It can be observed that both the reconstructed data confusion matrix and the calibrated data confusion matrix are banded which implies that the data is most likely misclassified among neighbouring levels. The precision of **Level 2, Level 3 and Level 4** in Table 4 improved only slightly from Table 3. This indicates that the effectiveness of signal reconstruction is very limited. On the other hand, the precision of **Level 2, Level 3 and Level 4** in Table 5 shows a significant improvement from Table 4. This indicates that by incorporating temperature and humidity, our ANN-based calibration is able to meaningfully improve the accuracy. However, the accuracy numbers are far from ideal, motivating the need for an online inference based calibration, described in more detail in next section.

Table 3. Raw data confusion matrix

Ground Truth	Predictions								Recall
	Level 1	Level 2	Level 3	Level 4	Level 5	Level 6	Level 7	Level 8	
Level 1	11013	2730	858	179	0	0	0	0	0.75
Level 2	2761	1331	1337	391	117	2	0	0	0.22
Level 3	47	887	357	197	74	0	0	0	0.23
Level 4	2	8	342	299	28	3	1	0	0.39
Level 5	0	3	37	103	393	84	55	2	0.58
Level 6	0	0	13	23	180	170	90	5	0.35
Level 7	0	0	0	13	26	170	151	33	0.38
Level 8	0	0	0	3	19	12	11	67	0.60
	0.80	0.27	0.12	0.23	0.45	0.39	0.49	0.63	
	Precision								

Table 4. Reconstructed data confusion matrix

Ground Truth	Predictions								Recall
	Level 1	Level 2	Level 3	Level 4	Level 5	Level 6	Level 7	Level 8	
Level 1	11228	2993	559	0	0	0	0	0	0.76
Level 2	2787	1647	1212	291	2	0	0	0	0.28
Level 3	34	908	433	187	0	0	0	0	0.28
Level 4	0	11	281	362	29	0	0	0	0.53
Level 5	0	0	30	133	423	73	18	0	0.62
Level 6	0	0	0	0	211	190	80	0	0.40
Level 7	0	0	0	0	0	150	204	39	0.52
Level 8	0	0	0	0	0	0	31	81	0.72
	0.80	0.30	0.14	0.23	0.64	0.46	0.61	0.68	
	Precision								

Table 5. Calibrated data confusion matrix

Ground Truth	Predictions								Recall
	Level 1	Level 2	Level 3	Level 4	Level 5	Level 6	Level 7	Level 8	
Level 1	9500	3996	1278	6	0	0	0	0	0.64
Level 2	1683	3463	100	91	0	0	0	0	0.65
Level 3	34	497	865	165	0	1	0	0	0.55
Level 4	0	2	202	391	86	2	0	0	0.57
Level 5	0	0	12	85	452	126	0	2	0.67
Level 6	0	0	1	10	86	332	43	3	0.70
Level 7	0	0	0	0	35	85	249	29	0.63
Level 8	0	0	0	0	0	0	11	101	0.90
	0.85	0.44	0.35	0.52	0.69	0.61	0.82	0.75	
	Precision								

7.5 Evaluation of the Online Inference Model

In this section, the overall performance of the analytics engine is evaluated over the two months' deployment. We first describe the setting of the parameters in the online inference model. Then the performance of the analytics engine is presented in an intuitive way—$PM_{2.5}$ data through each calibration step in time domain, as shown in Figure 14. In addition, we visualized the confusion matrix after each calibration step, as shown in Figure 15. Finally, the numerical accuracy numbers are shown in the end.

As described in Section 6.3, we associate different devices with different noise parameters δ_i^2 to distinguish the different confidence level of the devices. Generally, the higher confidence a device takes the smaller noise parameter it will be bound. Additionally, we assume that the public monitor station and `Thermo` stay in the same confidence level since both of them are high accuracy devices. In our experiment, the noise parameter δ^2 associated with `Thermo`, public monitor station, `Dylos`, AQM and miniAQM are 0.5, 0.5, 1.0, 1.5 and 2.0 respectively.

We present three days' $PM_{2.5}$ data in Figure 14. Figure 14(a) shows the raw data of AQM compared with `Thermo`; it can be seen that `PPD42NJ` not only deviates from the `Thermo` significantly, but is also noisy. And after the signal reconstruction, it becomes much more stable as Figure 14(b) shows. This data is then calibrated by the neural network model shown in Figure 14(c). It shows that the ANN calibration model is able to recover the real $PM_{2.5}$ value to

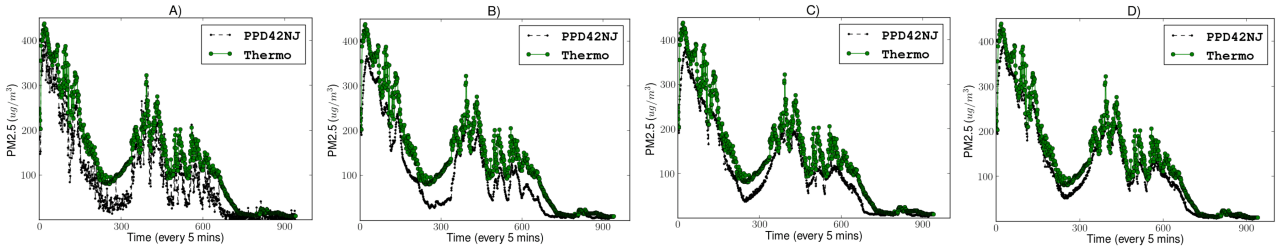


Figure 14. AQM $PM_{2.5}$ in time domain. A) Raw data; B) Reconstructed data; C) Calibrated by ANN; D) Calibrated by GP inference model.

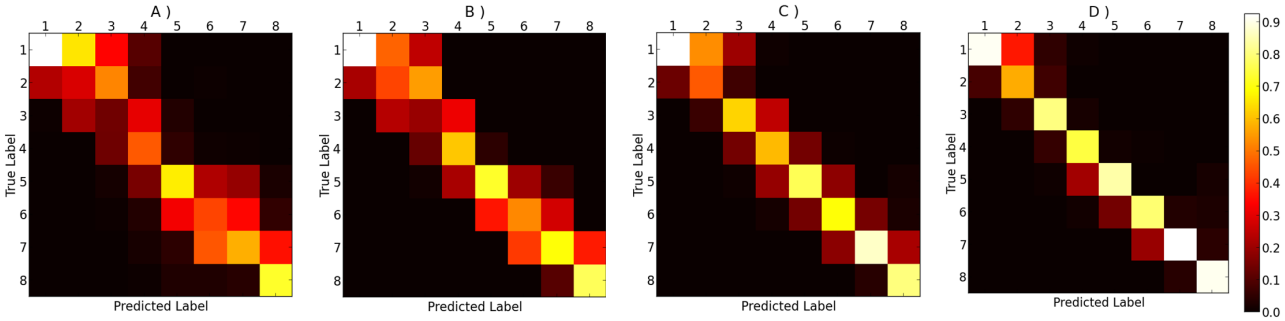


Figure 15. Confusion matrix of the deployment dataset. A) Raw data; B) Reconstructed data; C) Calibrated by ANN; D) Calibrated by GP inference model.

certain extent from the associated temperature and humidity. The final calibrated data given by the online inference model is shown in Figure 14(d).

The confusion matrix of the two months' deployment dataset is displayed in Figure 15. Figure 15(a) shows that the predicted results of raw data have large variances. After this data is processed by the online based GP inference model, the predicted levels are much batter aligned with the true levels, as shown in Figure 15(d).

The overall accuracy improvement from each step of the analytics engine are shown in Figure 16. The prediction accuracy achieved by RAW (raw data), SR (signal reconstruction), ANN, and GP (inference) are 0.532, 0.603, 0.641, and 0.817 respectively. The improvement results from the previous step for SR, ANN, and GP are 13.4%, 6.3%, and 27.5% respectively. These results from the deployment dataset show that 1) raw AQM data contains significant noise and can meaningfully benefit from signal reconstruction; 2) while ANN-based calibration is effective for historical dataset, it's less useful for real deployment; and 3) AQM data can benefit significantly from an online inference-based calibration model. This result corroborated with our hypothesis that certain features such as distance and POI can help establish the relative effects between data sources of varying (and same) confidence levels, thus enabling data sources to improve each other's accuracies. As shown in Figure 16, the combined improvement using SR, ANN, and Inference is 53.6% in real deployment.

The long-term errors caused by hardware shift/aging are an important issue, we use the following methods to deal with this problem: 1) For the products of the same batch, we will use one unique ANN calibration model to do the single-point calibration. As time progresses, more data will be added to the ANN training model and the parameters will get updated, and the errors caused by hardware aging are removed; 2) In the real deployment, we use Gaussian Process Inference model to futher improve the accuracy of the AQM devices, we will refer to all the devices in the area, such as accurate public monitoring stations, `Thermo` and less accurate `Dylos` or AQM to improve the accuracy and reduce the error caused by hardware aging; 3) we want to solve the hardware shift/aging problem by a real-time calibration technique in the future. Specifically, we continuously calibrate devices utilizing the real-time data from nearby public monitoring stations, which are considered to be always precise.

7.6 Visualization of the Spatial Inference in Heatmap

AirCloud can infer the air quality at places where sensors are unavailable and further improve accuracies of low confidence AQM or miniAQM sensors. Figure 17 shows the estimation of air quality generated by the GP inference model in our deployment area using a heatmap. It demonstrates that the $PM_{2.5}$ concentration deviation can be as large as 100 even in a small area of 4km$\times$4km. The heatmap is valuable for a variety of applications such as pollution tracking, trip planning, and locating pollution sources.

8 API and Applications

8.1 RESTful APIs

AirCloud currently supports two types of API—**time-series** based access and **GPS-based** access, both in RESTful style and use JSON as the data format.

The time-series based APIs inherit the powerful querying capabilities from sMAP and support direct data access using

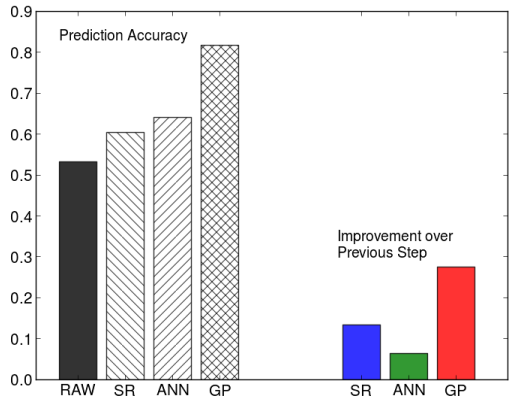


Figure 16. Overall prediction results of each step in the analytics engine. The left part is the prediction accuracy of RAW (raw data), SR (signal reconstruction), ANN, and GP (inference); the right part shows the relative improvements over previous step.

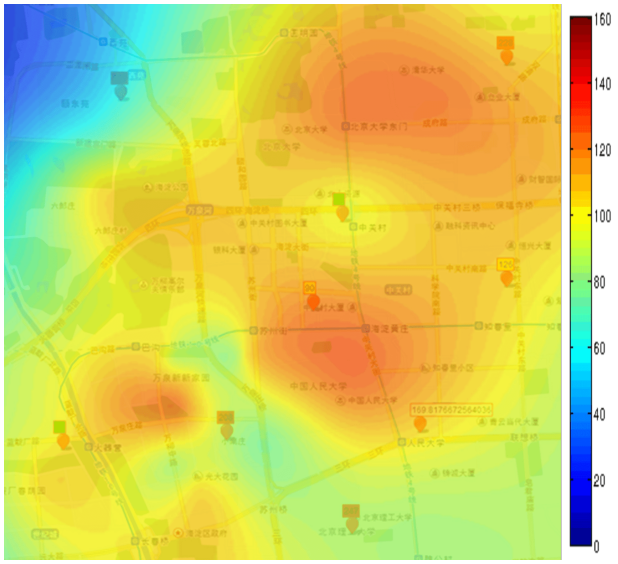


Figure 17. The spatial inference result of our deployment area in heatmap.

a SQL-like language, or directly via device IDs or meta-data lookups (please refer to [7] for examples).

The GPS-based API provides access to one of the build-in services of AirCloud, which compute an estimate of air-quality given an GPS coordinate. This service essentially export the inference functionality of the Air Quality Analytics Engine.

8.2 Applications

In addition to the Air Quality Analytics Engine, AirCloud is an open cloud platform for storing, accessing and sharing of air-quality related data. The APIs we provide enable third-party developers to innovate on top of AirCloud.

Because air quality is an important social issue, we have attracted many developers in a short two month period. To date, there are around eight applications running on our platform - two are web applications developed in-house, and six are smartphone applications developed by others.

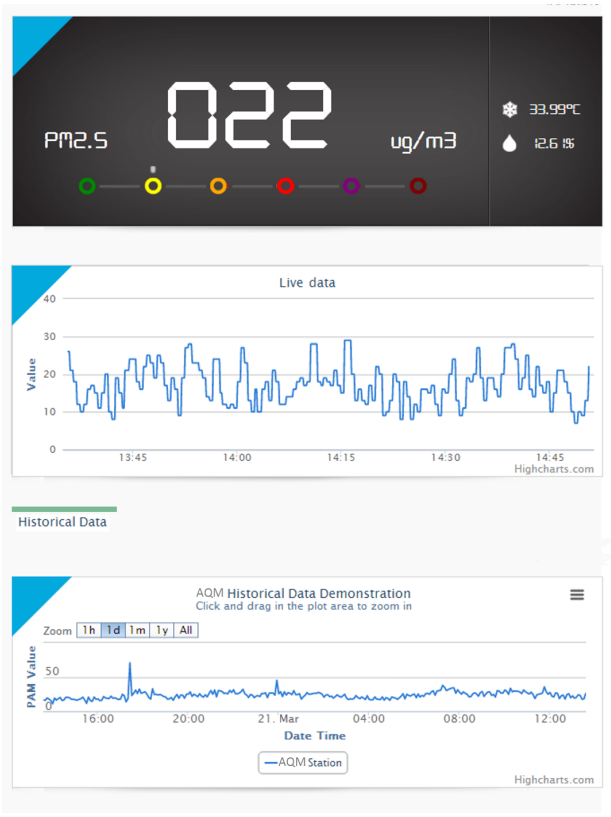


Figure 18. Visualization web app to view time-series data.

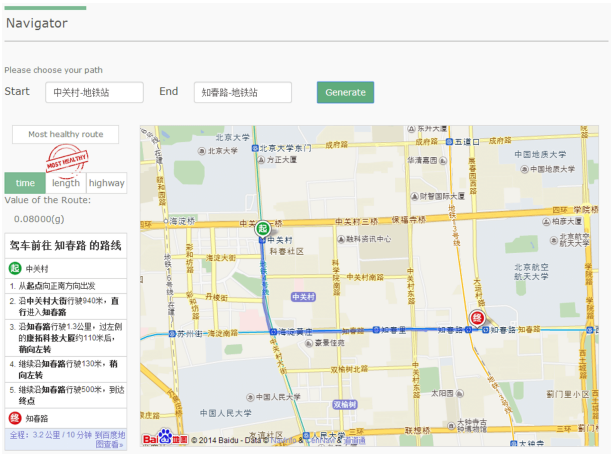


Figure 19. Trip planning web app - select the most healthy route.

Figure 18 is a visualization app to view time-series data stored on AirCloud, we can see the real time data or historical data of each AQM or miniAQM device. Figure 19 is a trip planning app that plans a route between A and B based on the least $PM_{2.5}$ intake instead of shortest distance, which

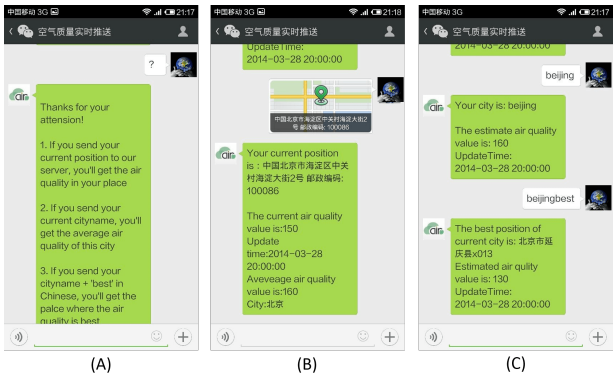


Figure 20. The wechat app.

will give users the most healthy route. Figure 20 is a service using the WeChat [11] official account platform. Users can subscribe to the channel and revive system message or use our system interactively. Figure 20(a) is the service menu; Figure 20(b) shows getting the $PM_{2.5}$ when given the current location; Figure 20(c) shows two other services: finding the city's average $PM_{2.5}$ and displaying the city's best $PM_{2.5}$ location.

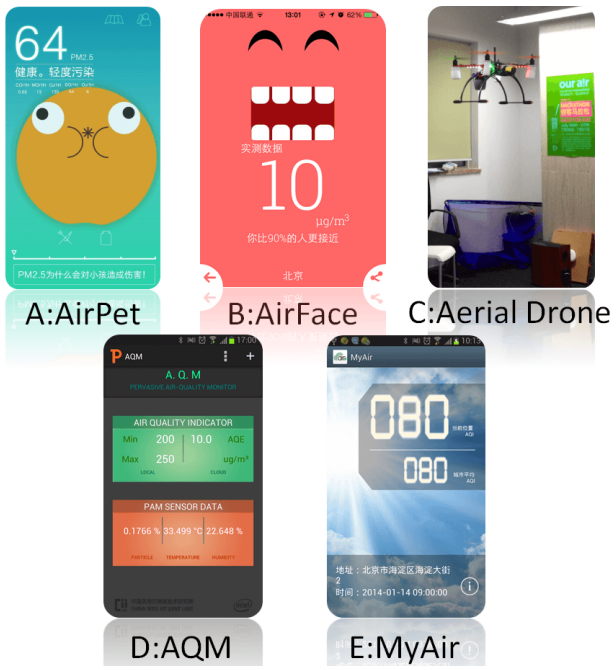


Figure 21. A and B are iOS games, while C is an air-quality aware aerial drone, all using AirCloud APIs, and by third-party developers. D is AQM phone app used to connect to the miniAQM device and E is MyAir app used to show $PM_{2.5}$ information in details, which are developed by ourselves.

Figure 21(a) is an iOS game called "AirPet". AirPet is a virtual pet that lives in your iPhone. She will grow up health and happy if fed at locations with good air (using the GPS API) or unhealthy and sad if fed at locations with bad air. This game encourages the player to go to places with good air. Figure 21(b) is a crowd-sourcing based iOS game called AirFace that asks the user to guess the current $PM_{2.5}$ concentration, with animated smile faces and sounds that correspond to different levels of air quality. AirFace then compares your guess with others in your area and tells you the actual $PM_{2.5}$ value, again, using our GPS API, Figure 21(c) is a quadcopter that flies around the city and "warns" citizens with a loud siren sound if the air quality is bad at that GPS location. Figure 21(d) is the AQM phone app used to connect to the miniAQM device; it will show the local and calibrated $PM_{2.5}$ concentrations. Figure 21(e) is MyAir app; it shows the $PM_{2.5}$ concentrations at your current location, and gives visualized warnings.

In the future, we will provide more applications and services for users, and more real-time data APIs for developers.

9 Conclusion

We approach the challenging problem of accurate and affordable $PM_{2.5}$ monitoring from a novel cloud-based data analytics perspective. By carefully designing and building our own $PM_{2.5}$ monitors - AQM and miniAQM - we are able to obtain reasonably accurate $PM_{2.5}$ measurement in real-time and at low cost. And by aggregating their data, plus other types of data at the cloud, we are able to learn and create model for particulate matter, which in turn helps us calibrate AQMs & miniAQMs, and infer $PM_{2.5}$ concentrations. We show a combined improvement of 53.6% in a real deployment using our cloud-based Air-Quality Analytics Engine. Together, AirCloud is able to achieve good accuracies at much lower cost than previous solutions. As a result, we are able to deploy a large number of AQMs throughout a city, providing dense coverage spatially.

At the time of this writing, 500 AQMs and 500 miniAQMs have been manufactured, and will be deployed across a metropolitan city with a population of 20 million. As future work, we hope to use this $PM_{2.5}$ sensor network to further improve our model in the analytic engine, and to solve a number of environment problems, such as identifying pollution sources and air-quality prediction. At the same time, we hope to build an ecosystem where people can create novel applications using the wealth of air-quality related data in AirCloud.

10 Acknowledgments

We would like to thank to our shepherd, Thomas Schmid, Jun Luo, and the anonymous reviewers for their insight and detailed feedback. Special thanks to Peipei Yang for his participation and thoughtful commentary that vastly improved the quality of this work.

11 References

[1] Celery: Distributed task queue. http://www.celeryproject.org/.
[2] confusion matrix. http://en.wikipedia.org/wiki/Confusion_matrix.
[3] Dylos. http://www.dylosproducts.com.
[4] Green Welcome. http://www.green-welcome.com.
[5] Peric. http://www.peric.cn.
[6] PM2.5. http://www.dwz.cn/cnnRO.
[7] sMAP archiver query language. https://code.google.com/p/smap-data/wiki/ArdQuery.
[8] Thermo. http://www.thermoscientific.com.

[9] Tornado web server. http://www.tornadoweb.org/.

[10] TSI. http://www.tsi.com/Optical-Particle-Sizer-3330.

[11] WeChat official account platform. https://mp.weixin.qq.com/.

[12] Y. Bengio. Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[13] C. M. Bishop et al. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

[14] E. Boldo, S. Medina, A. Le Tertre, F. Hurley, H.-G. Mücke, F. Ballester, and I. Aguilera. Apheis: Health impact assessment of long-term exposure to PM2. 5 in 23 European cities. *European journal of epidemiology*, 21(6):449–458, 2006.

[15] S. P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[16] S. Dawson-Haggerty, X. Jiang, G. Tolle, J. Ortiz, and D. Culler. sMAP: a simple measurement and actuation profile for physical information. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, pages 197–210. ACM, 2010.

[17] D. Hasenfratz, O. Saukh, S. Sturzenegger, and L. Thiele. Participatory air pollution monitoring using smartphones. 2012.

[18] Y. Jiang, K. Li, L. Tian, R. Piedrahita, X. Yun, O. Mansata, Q. Lv, R. P. Dick, M. Hannigan, and L. Shang. MAQS: a personalized mobile sensing system for indoor air quality monitoring. In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 271–280. ACM, 2011.

[19] R. V. Martin. Satellite remote sensing of surface air quality. *Atmospheric Environment*, 42(34):7823–7843, 2008.

[20] K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT Press, 2012.

[21] H. Patashnick, M. Meyer, and B. Rogers. Tapered element oscillating microbalance technology.

[22] J. Quiñonero-Candela and C. E. Rasmussen. A unifying view of sparse approximate Gaussian process regression. *The Journal of Machine Learning Research*, 6:1939–1959, 2005.

[23] J. Quinonero-Candela, C. E. Rasmussen, and C. K. Williams. Approximation methods for gaussian process regression. *Large-scale kernel machines*, pages 203–223, 2007.

[24] C. E. Rasmussen. Gaussian processes for machine learning. 2006.

[25] M. Sørensen, B. Daneshvar, M. Hansen, L. O. Dragsted, O. Hertel, L. Knudsen, and S. Loft. Personal PM2. 5 exposure and markers of oxidative stress in blood. *Environmental Health Perspectives*, 111(2):161, 2003.

[26] M. E. Tipping. Sparse Bayesian learning and the relevance vector machine. *The journal of machine learning research*, 1:211–244, 2001.

[27] S. Vardoulakis, B. E. Fisher, K. Pericleous, and N. Gonzalez-Flesca. Modelling air quality in street canyons: a review. *Atmospheric environment*, 37(2):155–182, 2003.

[28] T. Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the twenty-first international conference on Machine learning*, page 116. ACM, 2004.

[29] Y. Zheng, X. Chen, Q. Jin, Y. Chen, X. Qu, X. Liu, E. Chang, W.-Y. Ma, Y. Rui, and W. Sun. A cloud-based knowledge discovery system for monitoring fine-grained air quality.

[30] Y. Zheng, F. Liu, and H.-P. Hsieh. U-Air: when urban air quality inference meets big data. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1436–1444. ACM, 2013.