Public Scene Recognition Using Mobile Phone Sensors

Shuang Liang and Xiaojiang Du
Dept. of Computer and Information Science
Temple University,
Philadelphia, PA 19122, USA
{shuang.liang2012, dux}@temple.edu

Ping Dong
School of Electronic Information and Engineering
Beijing Jiao Tong University
P.R. China
pdong@bjtu.edu.cn

***Abstract*—AbstractSmartphones evolve rapidly and become more powerful in computing capabilities. More importantly, they are becoming smarter as more sensors such as the accelerometer, gyroscope, compass and the camera have been embedded on the digital board. In this paper, we propose a novel framework to recognize public scenes based on the sensors embedded in mobile phones. We build individual models for audio, light, wifi and bluetooth first, then integrate these sub-models using dynamically-weighted majority voting. We consider two factors when deciding the voting weight. One factor is the recognition rate of each sub-model and the other factor is recognition precision of the sub-model in specific scenes. We build the data-collecting app on the Android phone and implement the recognition algorithm on a Linux server. Evaluation of the data collected in the bar, cafe, elevator, library, subway station and the office shows that the ensemble recognition model is more accurate and robust than each individual sub-models. We achieved 83.33% (13.33% higher than audio sub-model) recognition accuracy when we evaluated the ensemble model with test dataset.**

Keywords*—mobile sensing; scene recognition; ensemble learning

I. Introduction

According to the data published by Gartner [1], sales of smartphones accounted for 53.6 percent of overall mobile phone sales in 2013. Smartphones have a number of built in sensors such as light sensor, proximity, accelerometer, gyroscope, bluetooth, wifi, camera, GPS, microphone and so on. The rich set of sensors on smartphones makes it possible for them to get the ambient context information and become more intelligent. Context-aware apps also benefit a lot from the built-in sensors. Location-based apps can present customized content and service based on your current location obtained by GPS. Yelp lets you find nearby restaurants and the upcoming events around your current location. Motion control games such as Abduction, Radio Ball 3D and Asphalt 6 exploit mobile sensors to control the game.

Scene recognition tells the type of location you are in, for example whether you are in a cafe or in a bar. There has been research on scene classification. These research papers are based on tagged camera images [1], [2], [3]. Some research papers such as [4], [5] are based on only audio features to identify and differentiate scenes. However, the results are biased and not accurate. Is there any way to just use the sensors on smartphones to recognize the scenes accurately and robustly? To answer this question, we propose using only audio, light, wifi and bluetooth features to recognize public scenes. We trained individual models for each os these sensors. From experimental results we found the strengths and weaknesses of sub-models when recognizing certain scenes. We combined the sub-models using ensemble learning. To make the model accurate and robust, we considered both the overall recognition rate of the sub-models and the precision on specific scenes of each sub-model in the ensemble model.

The contributions of our work are as follows:

- We designed and implemented the public scene recognition system on a real smartphone.
- We proposed using only mobile phone sensors for scene classification.
- We proposed an ensemble learning method based on dynamically-weighted majority learning to build a accurate and robust scene recognition model.
- We collected amounts of datasets of audio, light, wifi, bluetooth, which could be published for evaluation of similar systems.

The rest of this paper is organized into the following. In section II, we will discuss related works. We will introduce the system architecture and the data models in Section III. Later in Section IV, we will discuss in detail about the implementation of the system and the recognition algorithm. Evaluations of the system are given in Section V and we draw our conclusions in Section VI.

II. Related Work

Public scene classification has been researched for a long time, and is still active in some areas such as image classification and robot scene recognition. Very few scene recognition research papsers are about resource-constrained mobile platforms. Among them, only one or two are about using mobile phone sensors.

There are papers on using audio features to recognize everyday activities and identify human voices. Chen et al.

monitored activities which occurred within a bathroom based on the sound [6]. Wang et al. presented a sound recognition system for home automation [7]. They evaluated their system using six sound classes: doorbell, glass breaking, knocking, telephone ringing, cough and human speech. Their system can also recognize speech if the sound was identified as human speech. Mesaros et al. presented a system to detect events in real life recordings [8]. They detected up to 61 event classes with an accuracy of 24%. All of the research exploits audio features to detect environment or life activities and events, which could also be used to detect scenes.

To our knowledge, only a few researchers focused on using mobile phone sensors to recognize scenes. AmbientSense [9] is such a system which was implemented on smartphones. Based on audio features collected by smartphones, AmbientSense could recognize 23 daily life ambient sound classes. The overall recognition rate is 58.45% for the system. The performance of AmbientSense differs for recognizing different sound classes.

Instead of only counting on individual sensors on mobile platforms, we propose a novel framework which exploits four different sensors on smartphones to recognize public scenes . The features we use include audio, light, wifi and bluetooth, and the recognition can be done almost immediately within two minutes. Our recongnition model is more accurate and robust compared with systems that just use individual sensor data for classification.

III. System Architecture

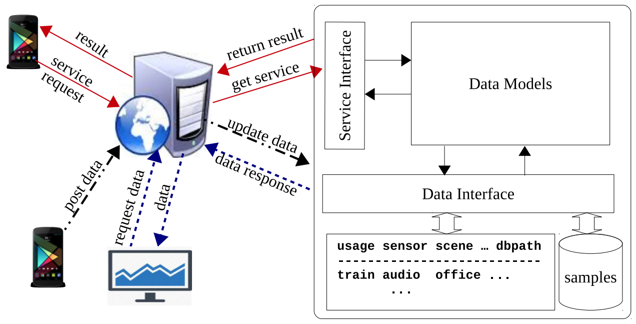


Fig. 1. System architecture

Figure 1 shows the system architecture and how each component interacted with the other components. Basically the system is a client / server structue. The client devices could be a mobile device or a desktop computer. The mobile device posts the collected sensor data to the server. The server puts the data into the database through the Data Interface on the server. And the Data Models on the server build the recognition model using the collected data. Additionally, the desktop client can request and browse the collected data on the server by sending requests to the server.

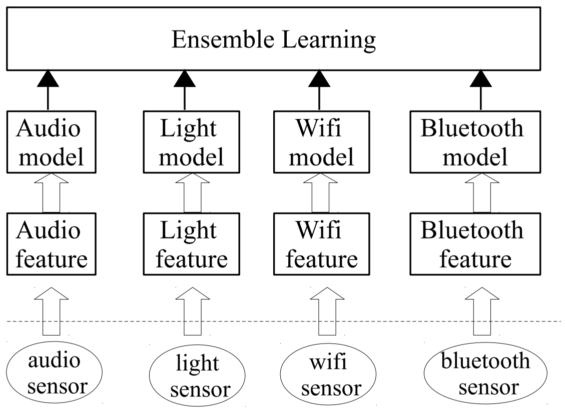


Fig. 2. Data model hierarchy

A. Data Models

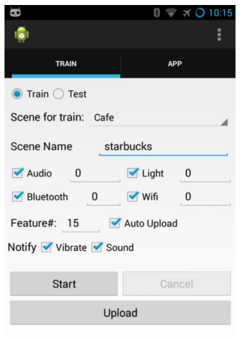
Figure 2 shows the structure of the data models. At the lowest level are the sensors we use to recognize the scene. We extract features from these underlying sensors and pass the features data to the corresponding classifier model. Then we integrate individual models to generate the final classifier. Besides having data models as the primary part, the system also includes interfaces to data and service. The data interface unit receives data from mobile devices, stores the data and keeps track of the data in the server. It also monitors data requests from the controller and responds with the data records. When the data model needs training and testing datasets, it sends requests to the data interface and the data interface will extract training and testing data from the stored dataset. For the service interface, it analyzes the service request information from the mobile device and responds with the classification result from the data models.

IV. Implementation

The public scene recognition system implementation involves data sensing and collection on the mobile platform, data storage and management on the server and data classification models based on the collected data. In this section, we will review the details about each component's implementation.

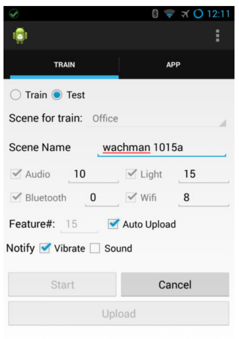
A. Mobile Data Sensing

The data sensing and collection components of the system were built on the Android 4.4.2 system. We exploited the built-in sensing probes of the FUNF open sensing framework [10] to obtain the ambient sensor data of audio, light, wifi, bluetooth. Figure 3 shows the Android app for collecting training and testing data. Using this app, users can set scene type, name of the location, the type of sensor to use and the number of features to collect. The app can work both in offline mode and online mode. In offline data, the collected data samples are stored in the SD card and users can upload data when network is available. The user can also check the *”Auto Upload”* box to upload directly, without storing data on mobile phone external storage. Both vibration and sound notifications are provided to notify the user when data collection and uploading have completed.



(●) Train ( ) Test
Scene for train: Cafe
Scene Name starbucks
[✓] Audio 0 [✓] Light 0
[✓] Bluetooth 0 [✓] Wifi 0
Feature#: 15 [✓] Auto Upload
Notify [✓] Vibrate [✓] Sound

(a) Collecting training data



( ) Train (●) Test
Scene for train: Office
Scene Name wachman 1015a
[✓] Audio 10 [✓] Light 15
[✓] Bluetooth 0 [✓] Wifi 8
Feature#: 15 [✓] Auto Upload
Notify [✓] Vibrate [ ] Sound

(b) Collecting testing data

Fig. 3. Mobile data collection app

The app stops collecting data when all selected sensors finish. For audio and light sensors, which are continuous and always available, they stop sensing when the collection the number of features that the user set is done. While for wifi and bluetooth, they are not available in some scenes, such as for the elevator. In such cases, they stop after completing the initial broadcast. Furthermore, we can get the approximate number of bluetooth and wifi devices in the scene at that moment.

The collected features for audio, light, wifi and bluetooth are in different sizes. The features are derived from the data collected by FUNF probes. For audio, we use 12 mfccs [11] as the feature vector. Table I shows the features in detail.

TABLE I. FEATURES OF THE SENSOR DATA

Sensor	Feature	Type	Vector Size
Audio	MFCCs	Float	12
Light	Lux	Float	1
Wifi	# of Wifi device	Integer	1
Bluetooth	# of Bluetooth device	Integer	1

B. *Classification Modules and Algorithms*

The classification process includes two phases. At phase one, we trained individual sub-models using naive bayes classifier. Then we obtain the model profiles which include recognition rate and recognition precision on each scene class for the corresponding sub-model. At phase two, we build the ensemble learning model based on the sub-models and model profiles created at phase one. We then use the test dataset for testing the phase two ensemble model. Figure 4 shows the flowchart for this two-phase classification model.

In phase two, we exploited dynamic-weighted majority voting to build the ensemble classification model. Algorithm 1 is the algorithm for the ensemble classification model.

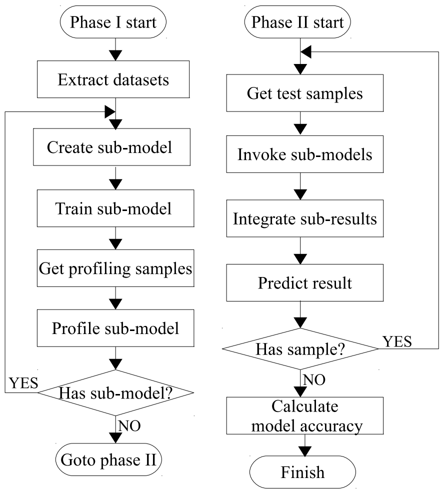


Fig. 4. Two-phase scene classification flowchart

Algorithm 1 Ensemble scene classification algorithm

```
1:  $learners \leftarrow [audio, light, wifi, bluetooth]$
2:  $reward\_ratio \leftarrow 1.1$
3:  $punish\_ratio \leftarrow 0.9$
    // Profile sub-model
4:  for all $i = 0$ to $len(learners)$ do
5:      $model \leftarrow \text{train}(learner[i])$
6:      $learner[i].weight \leftarrow model.recognition\_rate$
7:      $learner[i].precision \leftarrow model.precision$
8:  end for
    // Take sub-decisions from enrolled learners
9:  for all $i = 0$ to $len(learners)$ do
10:     $predict = learner[i].predict()$
11:     $vote\_vector \leftarrow \text{make\_vote}(learner[i], predict)$
12:     $result\_vector \leftarrow result\_vector + vote\_vector$
13: end for
14: $ensemble\_predict = \max(result\_vector)$
    // Update voting weights of learners
15: if $ensemble\_predict = target$ then
16:     for all $i = 0$ to $len(learners)$ do
17:         if $learner[i].predict = ensemble\_predict$ then
18:             $\text{update\_weight}(learner[i].weight * reward\_ratio)$
19:         end if
20:     end for
21: else
22:     for all $i = 0$ to $len(learners)$ do
23:         if $learner[i].predict = ensemble\_predict$ then
24:             $\text{update\_weight}(learner[i].weight * reward\_ratio)$
25:         end if
26:     end for
27: end if
```

The ensemble model starts by building and profiling sub-models for all enrolled learners (audio, light, wifi, bluetooth).

In the model profiling process, we initialize the voting weight of the sub-model to the recognition rate of the model. The recognition rate for each sub-model is calculated in equation (1) which we use the ratio of the number of correctly classified samples to the total number of samples.

$$Recognition\_rate = \frac{correct\_sample}{total\_sample} \tag{1}$$

And for each scene, we build a precision map which maps the predicted scene to the precision rate of this prediction. The precision of the scene indicates the probability that the predicted scene is correct for this scene in the sub-model. So we calculate this rate based on the true positive (TP) and false positive (FP) number for the predicted scene. Equation (2) shows how the precision map is calculated.

$$Precision(scene) = \frac{TP(scene)}{TP(scene) + FP(scene)} \tag{2}$$

Then we let each enrolled learner vote for the scene class. The value of the vote is calculated in equation (3). We consider both the predicted result (the precision factor) and the model (the weight factor) in making the voting decision.

$$Vote(scene) = precision(scene) \times weight \tag{3}$$

We also adjust the weight of the sub-models dynamically after each prediction in the ensemble model. If the predicted result is correct, we increase the weights of the models that make contributions in the voting process. On the contrary, we reduce the weights of the models that make the wrong decision if the predicted result is not the actual scene.

V. Evaluation

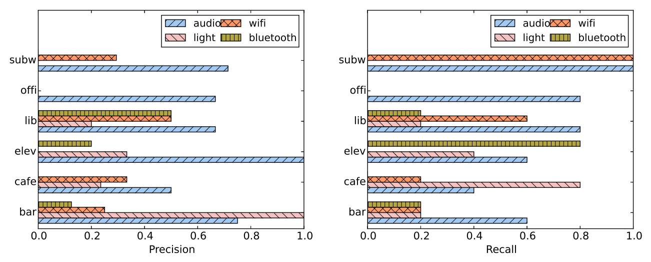
A. Evaluation Environment and Datasets

We tested our public recognition recognition system on a LG Nexus 4 with a Android 4.3.1 system and ubuntu server with sqlite 3.7.11 database, python 2.7.3 and django 1.6.6 installed. We collected training and testing data using the Android app we developed in six different scenes: the bar, Cafe, elevator (elev.), library (lib.), office (offi.) and subway station (subw.). For each selected scene, we collected several observations (obs.) in different spots and at different times of the day. And each observation include four samples (sam.) which are for audio, light, wifi, and bluetooth respectively. Currently, we have 1179 training samples and 1141 testing samples in our database. Table II shows the details about our training and testing datasets.

TABLE II. Training and testing datasets

Usage	Bar	Cafe	Elev.	Lib.	Offi.	Sub.	Obs.#
Train	63	78	71	63	28	59	362
Test	64	77	72	59	21	57	350

We randomly select 20 observations for each scene from the training dataset. We use these observation samples to train the sub-models of audio, light, wifi and bluetooth. Then we randomly select 10 observations for each scene from the testing dataset. We used half of these samples for profiling the sub-models which let us know the recognition rate of the sub-models and the recognition precision of each sub-model when predicting certain scene. With the remaining test samples, we fed them to the ensemble model to test our ensemble model.



(a) Precision bar chart (b) Recall bar chart

Fig. 5. Classification metrics on scenes

B. Performance of Sub-models

From the evaluation experiments, we find that different sensors have their strengths in recognizing certain scenes and are weak or disadvantaged to detect other certain scenes. To see which sensor models are strong and which sensor models are weak for certain scenes, we use two metrics: the precision and recall of each sensor model on the scenes. As introduced in equation (2), the precision is the fraction of samples that are truly positive from the samples predicted to be positive by the model. As shown in figure 5 (a), the bar chart clearly tells the precision difference of the sub-models on the scenes to be classified. The precision of the sub-models tells us the overall possibility that the recognition is correct. Therefore, we took this factor into consideration when constructing the ensemble model. The other metric is the recall. Recall measures the fraction of positive samples the model successfully classified on the scene. It also applies to each scene. Equation 4 shows how recall values of scenes are calculated. From the recall of the sub-models, we can know the recognition performance of the sub-model on a specific scene.

$$Recall(scene) = \frac{TP(scene)}{TP(scene) + FP(scene)} \tag{4}$$

To get to know the overall recognition performance and the robustness of each sub-model, we built the recognition confusion matrix for each sub-model as shown in figure 6. In the confusion matrix diagram, the darker the off-diagonal cells are, the more mistakes the model has made. From the confusion matrix, we found that the sub-model for audio features outperformed the others and is the most robust sub-model in recognizing the scenes.

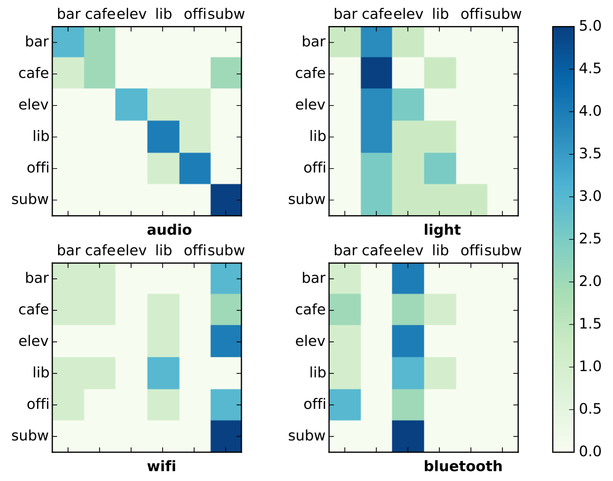


Fig. 6. Confusion matrices of sub-models

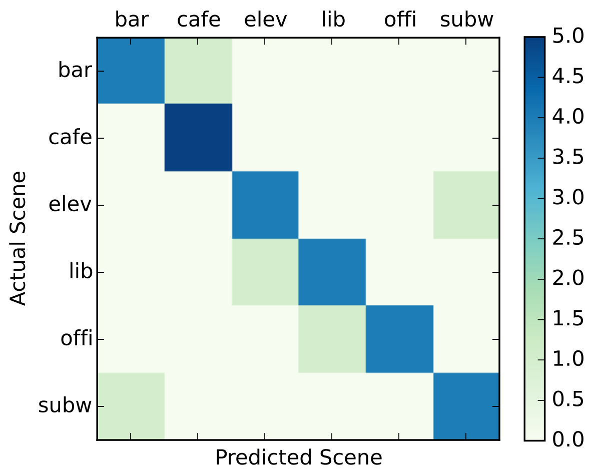


Fig. 7. Confusion matrix of ensemble model

C. *Classification Performance*

Our ultimate goal is to build an accurate and robust model to recognize public scenes. To this end, we obtained insights from sub-models of sensors and built our ensemble model based on dynamically-weighted majority voting. We considered both the recognition precision of individual sub-model on the scenes and the overall recognition rate per sub-model. So the ensemble model is at least as accurate as the best sub-model while is better in robustness.

TABLE III. RECOGNITION RATES

Classifiers	Recognition rate (%)
Audio	70
Light	26.67
Wifi	33.33
Bluetooth	20
Ensemble	83.33

Table III shows the recognition rate of the ensemble model together with the sub-models of audio, light, wifi and bluetooth. The accuracy improvement shows that our ensemble model is better than using individual sub-models. And the confusion matrix diagram of the ensemble model that is shown in figure 7 also confirms that the model is more robust and rarely makes error.

VI. CONCLUSION

We proposed a public scene recognition system based on the sensors embedded in smartphones. Compared with recognition systems using only individual sensor data, the ensemble module performs better in accuracy and robustness. We designed and implemented the recognition system and evaluated it using the data collected in six different scenes. For each scene, we collected samples in different places and different hours of the day. We investigated the performance of each sub-module with specific scenes and use that knowledge in building our ensemble model based on dynamic-weighted majority voting. Since the classification models are not limited to certain sensor features, they can apply to other smartphone sensors.

ACKNOWLEDGMENT

This work was supported in part by the US NSF under grants CNS-1022552, CNS-1065444, as well as the US Army Research Office under grant WF911NF-14-1-0518.

REFERENCES

[1] L.-J. Li, R. Socher, and L. Fei-Fei, “Towards total scene understanding: Classification, annotation and segmentation in an automatic framework,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[2] J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba, “Sun database: Large-scale scene recognition from abbey to zoo,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[3] J.-H. Lim, Y. Li, Y. You, and J.-P. Chevallet, “Scene recognition with camera phones for tourist information access,” in *International Conference on Multimedia and Expo*, 2007.

[4] V. Peltonen, J. Tuomi, A. Klapuri, J. Huopaniemi, and T. Sorsa, “Computational auditory scene recognition,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2002.

[5] S. Ravindran and D. Anderson, “Audio classification and scene recognition and for hearing aids,” in *IEEE International Symposium on Circuits and Systems*, 2005.

[6] J. Chen, A. H. Kam, J. Zhang, N. Liu, and L. Shue, “Bathroom activity monitoring based on sound,” in *Pervasive Computing*. Springer, 2005.

[7] J.-C. Wang, H.-P. Lee, J.-F. Wang, and C.-B. Lin, “Robust environmental sound recognition for home automation,” *IEEE Transactions on Automation Science and Engineering*, 2008.

[8] A. Mesaros, T. Heittola, A. Eronen, and T. Virtanen, “Acoustic event detection in real life recordings,” in *18th European Signal Processing Conference*, 2010.

[9] M. Rossi, S. Feese, O. Amft, N. Braune, S. Martis, and G. Troster, “Ambientsense: A real-time ambient sound recognition system for smartphones,” in *International Conference on Pervasive Computing and Communications Workshops*, 2013.

[10] “Funf open sensing framework,” http://www.funf.org/.

[11] “Mel-frequency cepstrum,” http://en.wikipedia.org/wiki/Mel-frequency_cepstrum.