REAL-TIME DETECTION OF CROWDED BUSES VIA MOBILE PHONES

Alex Haig
Carnegie Mellon University
adhaig@andrew.cmu.edu

Shirley Anugrah Hayati
Carnegie Mellon University
shayati@cs.cmu.edu

Anthony Tomasic
Carnegie Mellon University
anthony.tomasic@gmail.com

Word Count: 4727 words + 6 figure(s) x 250 + 5 table(s) x 250 = 7477 words

Submission Date: November 15, 2018

ABSTRACT
Automated passenger counting (APC) technology is central to many aspects of the public transit experience. APC information informs public transit planners about utilization in a public transit system and operations about dynamic fluctuations in demand. Perhaps most importantly, APC information provides one metric to the rider experience – standing during a long ride because of a crowded vehicle is an unpleasant experience. Several technologies have been successfully used for APC including light beam sensing and video image analysis. However, these technologies are expensive and must be installed in buses. In this paper, we analyze a new source of data using statistical models: rider smartphone accelerometers. Smartphones are ubiquitous in society and accelerometers have been shown to accurately model user states such as walking and sitting. We extend these models to use accelerometers to detect if the rider is standing or sitting on a bus. Standing riders are a signal that the bus is crowded. This paper provides evidence that user smartphones are a valid source of participatory sensing and thus a new source of automated passenger counting data.

Keywords: Automatic Passenger Counter, Participatory Sensing, Machine Learning

INTRODUCTION

Accurate knowledge about the utilization of public transit vehicles by riders, such as bus fullness, is critical information for public transit planners. Automatic Passenger Counters (APC) are used to count riders in public transit systems. APC information is used by transit planners to detect transit bottlenecks. A vehicle with a passenger counter higher than the numbers of seats available indicates that capacity is approaching the limit. APC counting is also used in real time by operations teams to assess overcrowding of vehicles – a signal to dispatch additional vehicles on demand. Finally, APC data provides evidence that the rider experience is poor - high APC counts, relative to the capacity of the vehicle, indicate that riders are probably standing in a crowded vehicle during the trip.
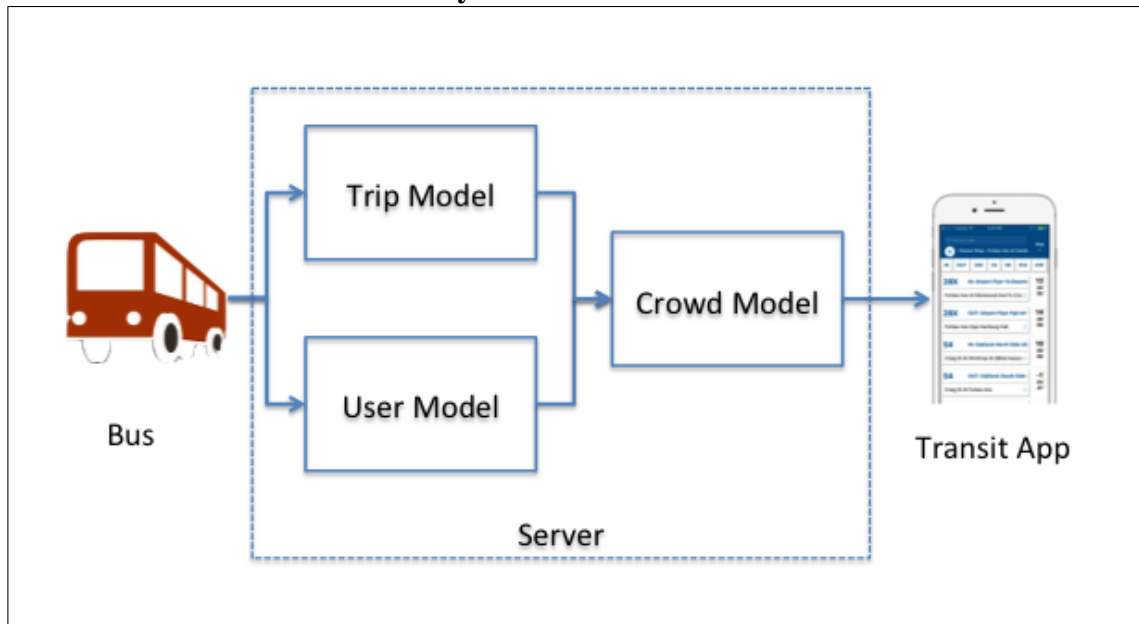
In this paper, we investigate a new technique for recording information about bus vehicle fullness. Participatory sensing, via a user's smartphone accerolmeter and GPS, can be used in transit systems to gather information about users and vehicles (*1*) without requiring action from the user – in this case whether the rider is sitting or standing. This information, aggregated across riders in a vehicle, provides a signal about current conditions. The fullness of a bus can be estimated by the number of riders standing or sitting on the bus. Since smartphones are now equipped with accelerometers, machine learning algorithms can predict the state of riders and thus infer the state of the vehicle with respect to fullness. This method of determining fullness is thus complementary to APC data.

While APC data may be a good measure of rider counts in aggregate, it can suffer from high error when considering individual vehicles (cf. Results Section) due to compounding errors. Thus, APC data is not always reliable. On the other hand, using phone sensors may not be able to produce an exact rider count, but aggregating across phone sensors provides an independent, direct, measurement of the state of riders. In addition, the error associated with each measurement is independent.

Third party transit information providers want information about the fullness of a bus prior to its arrival to provide to riders. Such information will be especially useful for riders with accessibility issues. However, these transit information providers typically do not have access to the APC data in real-time. Participatory sensing provides a direct method of providing this information, independent of the budgetary constraints of the transit agency (*2*).

PROPOSED ARCHITECTURE

The proposed architecture for the system consists of five components (Figure 1). The bus symbol represents riders utilizing a mobile smartphone that gathers accelerometer and telemetry data. This information is transmitted to a server in the cloud that contains three models. The Trip Model clusters users into the same bus trip using prior research methods ((*3*), (*4*)). The User Model, detailed below, models the state of the rider. The Crowd Model, detailed below, translates the aggregation of the results of the user model for each trip to an indication of the state of the bus (e.g., many seats, few seats, crowded, very crowded). This information is transmitted to users utilizing a transit information application. For simplicity, in this architecture, all models are run on a cloud server. However, the user model can also be on the client phones of users on the bus. This client architecture reduces network communication costs.

FIGURE 1 System Functional Architecture

METHODOLOGY

This study is organized into three linked sections. In the first section, we analyze APC data for the Port Authority of Allegheny County (PAAC) to estimate its accuracy. The next section describes the methodology, via direct field observations, for measuring the relationship between bus fullness and riders standing on the bus. In the third section, accelerometer data from smartphones is logged. The logs provide training data to model a rider's current state, specifically to detect if they were on a bus, and if so were they standing or sitting.

Automatic Passenger Counter Data Accuracy

To examine the APC data, we obtained two months worth of data for the PAAC bus system, from November of 2016 and March of 2017. PAAC buses use light beams mounted at the doors of buses to count riders entering and exiting buses. This data does not contain ground-truth measurements of rider counts. We estimate its accuracy by using an internal consistency argument: the total number of riders getting on a bus must equal the total number of riders getting off a bus for each bus's block. We use blocks because a rider can theoretically ride the entire chain of trips within a block continuously. However, a rider cannot get on the bus before the start of the block and must get off before or at the end of the block. The block error is thus the difference between the total number of counted embarking minus the total number of disembarking. This value should always be zero. Estimating the error in this case is just a simple matter of calculating the difference between the sum of the number of people recorded as getting on, and the sum of the number people recorded as getting off. Of course, this methodology does not account for errors that cancel each other out, such as person getting both on and off the bus without being recorded in either instance, but it still gives us a rough upper-bound on the accuracy overall.

Bus Fullness Observation

If someone is standing on the bus then there is a high probability that the bus is full or close to full. To test this hypothesis, we conducted field observations while riding buses. Every time a bus left a stop, we counted the number of riders sitting and the number of riders standing. In addition, we kept track of the bus number for each bus which uniquely identifies the bus model and thus the number of seats.

Overall we recorded 246 data points from March to April 2018 across 12 different buses on 5 routes over 168 stops.

This gathered data serves as our Crowd Model. This model is a simple function that translates the empirically gathered data into predefined categories.

User Activity State Modeling

Activity detection consists of multiple steps: gathering a supervised set of training data, iterative engineering of a machine learning model, and generation of the final evaluation statistics of the best model.

Data Collection

To determine the activity state of riders, we built a custom mobile app for Android and iOS that collected raw data from the GPS, accelerometer, and compass sensors in the phone. The app allows users to select one of the 15 activity conditions show in Table 1. Upon selection of a state, the app waits 5 seconds to allow the user to position the phone correctly, records data for 15 seconds, and then stops automatically. This cycle comprises one data session. The timing for the recording of data for each session results in clean data that is less affected by random real-world factors. Using this app and 5 different phones we recorded a total of 3209 sessions (approximately 13 hours) of real-world data from October 2017 to July 2018. Table 1 shows the specific breakdown of the data including how the data was split into training and test sets when training our machine learning models. The true distribution of users' activity states varies per user in the real-world, so we balanced our test set across the different states in order to get a fair comparison of the accuracy for each state. The entire dataset was split into two parts: a training set and a test set. During training, a validation set was split from the training set. This set was used to iteratively improve the machine learning models described below. The test set was used after the end of the iterative improvements to produce the final values reported in this paper.

Data Preprocessing

To clean the noise in the dataset, we filter the dataset by selecting sessions that have at least 200 points of sensor readings over a time span of at least 10 seconds. This cleaning was performed to account for recording errors caused by bugs in the app itself. Such errors would be significantly less common in a fully implemented production system. Table 1 shows our data after cleaning.

Features

From the phone sensors, the app obtains seven values: acceleration in x-axis, acceleration in y-axis, acceleration in z-axis, latitude, longitude, speed and heading. Then, we follow Elhoushi et al. (5) for extracting statistical features from each of these raw sensor values – except for heading which reduced the performance of our machine learning models. Note that we do not implement all the features mentioned in Elhoushi et al. (5). The six values are translated into a set of features

TABLE 1 Dataset Statistics

	User State		Sessions	
State	Standing/Sitting	Phone Position	Train	Test
Bus	Standing	Pocket	198	20
	Standing	Hand	217	20
	Standing	Backpack	139	20
Bus	Sitting	Pocket	212	20
	Sitting	Hand	247	20
	Sitting	Backpack	210	20
Walking	✗	Pocket	197	20
	✗	Hand	194	20
	✗	Backpack	196	20
Stationary	Standing	Pocket	202	20
	Standing	Hand	187	20
	Standing	Backpack	165	20
Stationary	Sitting	Pocket	179	20
	Sitting	Hand	183	20
	Sitting	Backpack	183	20
Total			2909	300

that produce high accuracy (cf. Results Section) on the user states we are interested in. The total number of features used is 48: 8 summary statistics – as described below – for each of the 6 raw sensor values

- Mean: $\text{mean}(x) = \bar{x} = \dfrac{1}{N} \sum_{n=1}^{N} x_n$

- Mean absolute value: $\text{mean}(|x|) = |\bar{x}|$

- Median: Middle value separating the higher half of data from the lower half of data

- Variance: $\text{var}(x) = \sigma^2(x) = \dfrac{1}{N} \sum_{n=1}^{N} (x_n - \bar{x})^2$

- Standard Deviation: $\text{std}(x) = \sigma(x) = \sqrt{\text{var}(x)}$

- Average Absolute Difference: $\overline{|x - \bar{x}|}$

- Interquartile Range: Difference between 75th percentile and 25th percentile

- 75th Percentile: Value separating the higher 25% of data from the lower 75% of data

Model Construction
Using scikit-learn (*6*), we trained several classifiers to predict the activity state of a user from among the 15 classes defined in Table 1. The default parameters for the models were used unless otherwise specified. Three different classifiers were investigated: Random Forests, Support Vector

TABLE 2 APC Block Errors Sample Statistics

Dataset	Measurement	Result
11/16	Size (Blocks)	94,498
	Mean	-0.24
	Variance	34.61
	Standard Error of Mean	0.019
	Nonzero	43%
03/17	Size (Blocks)	77,797
	Mean	-0.09
	Variance	45.56
	Standard Error of Mean	0.024
	Nonzero	44%

Machines (SVM), and Multilayer Perceptrons (MLP).

Random forest is an ensemble classifier that randomly constructs many decision trees which vote for the most popular class (*7*). Individual trees were limited to a maximum depth of 20 nodes. In decision tree learning, each leaf represent a class; each interior node represent a criteria on a feature that selects one branch from a node to its children (either another node or a leaf). A decision tree is iteratively built top down using the Gini impurity criterion to split the feature space.

Support vector machines map non-linear input vectors into a higher dimensional feature space (Weston and Watkins (*8*)). For binary classification, a decision line is constructed to separate data points according to their labels and then the margin is maximized.

Multilayer perceptron is a feed-forward neural network that consists of one or more hidden layers with nonlinear activation function (*9*). Our network has 1 hidden layer with 15 hidden units.
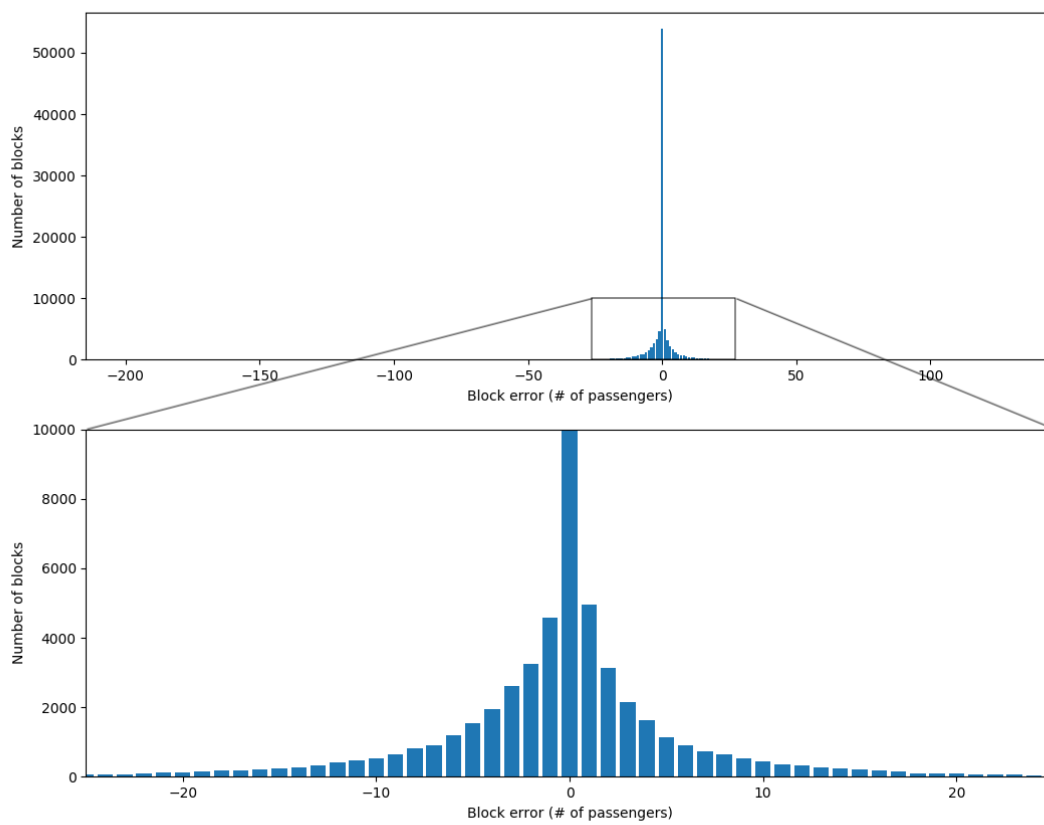
RESULTS
APC Data Accuracy
The APC data accuracy is measured by using internal consistency so the error represents over- or under-counting for a block. A histogram (Figure 2) of the data indicates that the vast majority of the data values are zero – see the top graph. The bottom graph, cropped and magnified, indicates that measurement error is balanced about the mean of zero. Standard statistics of the error (Table 2) confirms this analysis.

Overall we found that the mean error is near zero for both the 11/16 and 03/17 datasets, but the variance is quite high. The minimum error was -198 and the maximum was 188 for individual blocks. These statistics indicates that APC data is accurate for measuring the number of riders in aggregate, but with 43.6% of blocks containing errors, the reliability of APC counts needs improvement before it is used to measure the number of riders on buses in real-time.
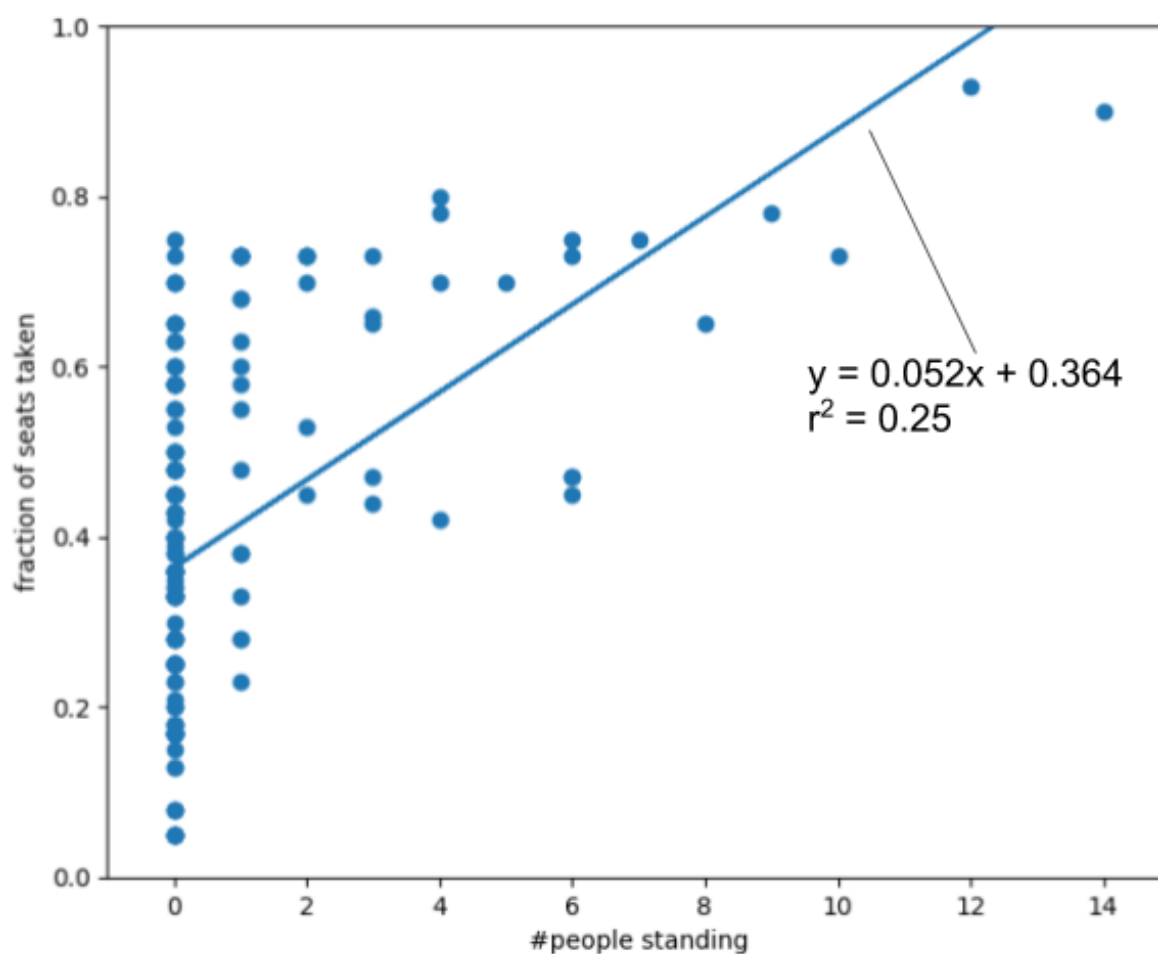
Bus Fullness, Standing and Sitting
To understand the behavior of riders on buses, we performed field observations on several routes and buses. Generally people preferred sitting on buses when seating was easily available. Anec-

FIGURE 2 APC Block Errors for November, 2016 Dataset

dotal evidence provides some few exceptions to this rule. Riders stood when seating was difficult to reach, the rider was on the bus for a short period, or in special circumstances, such as a parent with a child in a stroller. Further research would investigate the behaviour of these riders through interviews to better account for them in our model. A regression analysis of the data (Figure 3) shows a clear relationship between the number of riders standing on the bus and the fullness of the bus. There is a clear upwards trend showing that the number of riders standing is indeed predictive of the fullness of the bus. Surprisingly, this behavior becomes apparent when the seats available on a bus reaches 50% or so. We had informally expected that the behavior would occur only when most of the seats were taken. An interesting future analysis would interview riders to discuss the conditions under which riders perceive a bus as crowded.

FIGURE 3 Riders Standing vs. Seats Taken

$$y = 0.052x + 0.364$$
$$r^2 = 0.25$$

Activity Detection

The results in this section are reported on the approximately 10% of the recorded data reserved for testing. Gathering 15 user states insured that the activity detection training set contains a robust set of different circumstances of mobile phone user for riders. Our iterative investigation of various machine learning models produced somewhat low performance (Table 3). The highest accuracy (79.33%) is achieved by using Random Forest.

The confusion matrix of the Random Forest Model (Figure 4) categorizes the errors that the model makes as counts off the diagonal. Table 4 is the key to the axes of the confusion matrix. All 20 test examples with the state of "walking hand" are predicted correctly. Another interesting finding is that 7 examples with the state "on the bus, sitting, hand" are predicted as "on the bus, standing, hand" although 18 examples with the state "on the bus, standing, hand" are predicted correctly. Another mistake that our model often makes is distinguishing the location of the phone (hand, pocket, or backpack). For instance, 4 examples of "on the bus, sitting, backpack" state are predicted as "on the bus, sitting, pocket." While the Random Forest model can distinguish the "on the bus" state from the "off the bus" state, it sometimes makes mistakes discriminating "standing" and "sitting". For instance, 4 examples of "on the bus, standing, backpack" are predicted as "on the bus, sitting, backpack".

TABLE 3 Performance of Model using All Classes

Algorithm	Accuracy %
Random Forest	79.33
SVM	56.17
MLP	72.65

TABLE 4 Key for confusion matrix classes

Symbol	Meaning
S	Stationary
W	Walking
B	On Bus
h	Hand
p	Pocket
b	Backpack/Bag
↑	Standing
↓	Sitting

Since the goal of our work is to detect if the rider is on the bus, and if so whether they are standing or sitting, we conducted two experiments with merged classes. The first experiment distinguishes the user's transportation mode by merging the 15 classes into three classes: "on bus", "walking", and "stationary". The confusion matrix for this experiment is shown in Figure 5. An analysis of the matrix shows very good performance with proportionately few errors.

A second experiment discriminates the user's state on the bus and off the bus, and if the user is sitting or standing on the bus by merging classes into: "standing on the bus", "sitting on the bus", and "other". Again we get good performance as shown in Figure 6. Table 5 summarizes the performance of our models on both of these merged classes experiments. Both experiments have high accuracy.
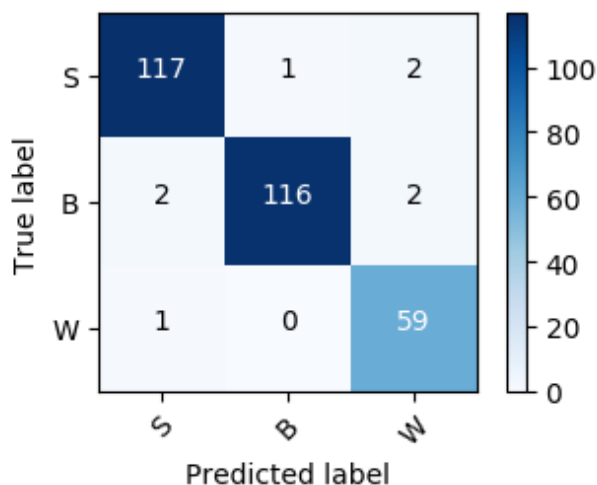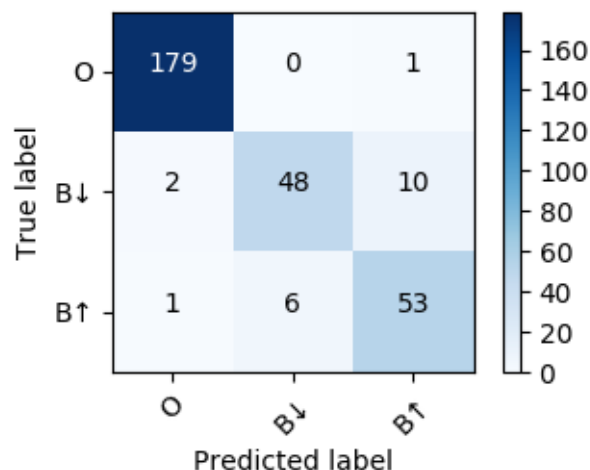
FIGURE 4 Random Forest Confusion Matrix for All Classes

These experiments show that even though the performance of the activity model on the full set of states is fairly low compared to previous work (5), the model still performs very well on detecting the user states we are most concerned with. The model very accurately determines whether riders are standing on a bus. Given that the model is operating on many riders on a bus (such as part of a smartphone operating system), aggregating the results of these models produces an estimate of the number of people standing on a bus. Given this value, the number of seats taken can be estimated by cross referencing the data in Figure 3.

TABLE 5 Performance of Model using Merged Classes

Classes	Accuracy %
Bus, Walking, Stationary	97.33
Bus Standing, On Bus Sitting, Other	93.33

FIGURE 5 Confusion matrix for Stationary, Walking, On Bus

FIGURE 6 Confusion matrix for On Bus Sitting, On Bus Standing, Other

DISCUSSION

The following discussion examines how our results relate to prior work. It is organized into sections on automated passenger counting, user activity sensing by mobile devices, mobile transit information systems, and participatory sensing.

Automated Passenger Counting

Although we did not use ground-truth data, our results on the errors found within APC data are consistent with those found in previous work that did use ground-truth data, Kimpel et al. (*10*). Using cameras to measure actual bus loads, they found that the difference in load between APC counts and ground-truth was 6.08%. Additionally, they used reverse regression to measure APC precision and found a bias of 8.81%. Both results were found to be statistically significant.

One other significant difference between the analysis in Kimpel et al. (*10*) and our own is that they performed significant cleaning on the data before measuring the APC accuracy by removing approximately 35% of their data. We did not perform such cleaning when examining the APC data because postprocessing the data in such a way would not be possible in a real-time

system.

Another method of counting riders (Chen et al. (*11*)) is to install cameras in buses and use computer vision techniques to detect riders getting on or off. Such a system can achieve an accuracy comparable to that found in APC systems, but it still requires bus agencies to install cameras in all of the buses, potentially a significant expense.

User Activity Sensing in Mobile Phones

Many prior works employ mobile phone sensors and supervised learning to determine user activity, either related to transportation mode or not. In detecting user activity, previous studies explored three main approaches: location-based (GPS), sensor-based (accelerometer), and hybrid of location-based and sensor-based approaches.

Ashbrook and Starner (*12*), Zheng et al. (*13*), Zheng et al. (*14*), and Stenneth et al. (*4*) use GPS to infer the motion or transportation mode of users. Sohn et al. (*15*) explored the usage of GSM sensors to detect daily physical activities such as walking, driving, or being stationary. Myrvoll et al. (*16*) counts the number of passengers on a bus through monitoring WiFi probe requests, a potential complementary source of passenger count data.

Microelectromechanical systems motion sensors (accelerometer, gyroscope, magnetometer, and barometer), without any location signals, can be used to detect indoor motion, such as walking, stationary, and going up/down stairs (Elhoushi et al. (*5*)). Nham et al. (*17*) collected acceleration data from mobile phones for predicting walking, biking, running, driving, riding a train, or riding a bus. Hemminki et al. (*18*) introduce a gravity estimation technique for accelerometers to produce more accurate horizontal acceleration measurements to discern information from user motion from noise like gravity. The user model in this paper extends this body of work to the specific case of classifying user activities on a bus.

Several works focus on the transportation mode of a user. Fang et al. (*19*) classified transportation mode using the accelerometer, magnetometer, and gyroscope sensors from smartphones and ran three machine learning algorithms (decision trees, K-nearest neighbor, and support vector machine). Yu et al. (*20*) applied four strategies which are big data, small footprint, data substitution, and multi-tier design in minimizing power consumption and memory requirement to detect transportation mode. Reddy et al. (*21*) used both accelerometer and GPS to identify transportation mode (stationary, walking, running, biking, in motorized transport). Jahangiri and Rakha (*22*) tried to detect the transportation mode (including driving a car, riding a bicycle, riding a bus, walking, and running) of a user via mobile phones using K-nearest neighbor, support vector machines (SVMs), and tree-based models such as decision tree, bagging, and random forest. Their best result was achieved by using random forest and SVM, similar to our results.

Mobile Transit Information Systems

This paper also extends a body of previous work using mobile phones to collect transit data from users and then using this data to improve the users experience on transit systems. For example the Tiramisu Transit system (*1, 3*) used crowd sourcing to generate real-time arrival information for the PAAC bus system before real-time information was provided by PAAC itself. The system presented in this study would be a novel extension by using user's phones to detect real time bus fullness, and then improving the user's transit experience by presenting fullness information. Zimmerman et al. (*3*) provides methodology for clustering users into the associated bus trips that they are on. The system simply asks the user directly as a form of crowdsourcing. Incentives to

encourage users to report this information are studied in Tomasic et al. (*23*). Stenneth et al. (*4*) uses automated methods to gather the same information.

Participatory Sensing
Our work is also related to participatory sensing for applications related to public transportation tracking. Zhou et al. (*24*) presented a crowdsourced bus arrival time estimation system by utilizing low-energy sensing resources, such as cell tower signals, movement statuses, and audio recordings. Thiagarajan et al. (*25*) also proposed crowdsourced transit (bus and train) tracking. They introduced power-efficient activity classification to detect if riders are in the vehicle or not, low-memory route-matching algorithm, and a method to track underground trains.

Pi et al. (*26*) studied the how users perceived bus fullness by investigating the relationship between APC data and crowdsourced bus fullness ratings from users of Tiramisu. They built a classification model to infer riders' fullness perception from APC data and other factors. Their model could also be used by transit agency to help predict bus fullness. Our work is distinguished by the fact that we indirectly but independently estimate bus fullness using phone sensors whereas they use APC data.

CONCLUSION
Real time bus fullness (the crowd on a bus) information is important to both transit users and planners. In this paper we showed that the reliability of APC data needs to be improved before presenting this information in real time. We then conjectured that the same information could be derived by determining how many people were standing in a bus. Field observations confirmed a direct relationship between people standing and the number of seats taken on a bus. To make this information available to riders in real time, the paper demonstrated that smartphone sensors can provide relevant real time data. Several machine learning models were applied to the logs of users' activities, both on and off buses, and were shown to be highly accurate in detecting when a user was standing or sitting on a bus. Thus, adding these models to mobile phone operating systems or applications, and transmitting the results of these models in real-time, would provide an accurate foundation for this feature in transit information systems.

In future work, the results of this work can easily be extended to many different public transit systems, such as trains or subways. Additionally, with more data we could improve the results of our activity model and better learn the relationship between bus fullness and the number of people standing.

AUTHOR CONTRIBUTION STATEMENT
The authors confirm contribution to the paper as follows: study conception and design: Alex Haig, Shirley Anugrah Hayati, Anthony Tomasic; data collection: Alex Haig, Shirley Anugrah Hayati, Anthony Tomasic; analysis and interpretation of results: Alex Haig, Shirley Anugrah Hayati, Anthony Tomasic; draft manuscript preparation: Alex Haig, Shirley Anugrah Hayati, Anthony

REFERENCES

[1] Tomasic, A., J. Zimmerman, C. Garrod, Y. Huang, T. Nip, and A. Steinfeld, The performance of a crowdsourced transportation information system. In *TRB 94th Annual Meeting Compendium of Papers*, Transportation Research Board of the National Academies, 2015.

[2] Steinfeld, A., J. Zimmerman, A. Tomasic, D. Yoo, and R. Aziz, Mobile transit information from universal design and crowdsourcing. *Transportation Research Record: Journal of the Transportation Research Board*, , No. 2217, 2011, pp. 95–102.

[3] Zimmerman, J., A. Tomasic, C. Garrod, D. Yoo, C. Hiruncharoenvate, R. Aziz, N. R. Thiruvengadam, Y. Huang, and A. Steinfeld, Field trial of tiramisu: crowd-sourcing bus arrival times to spur co-design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2011, pp. 1677–1686.

[4] Stenneth, L., O. Wolfson, P. S. Yu, and B. Xu, Transportation Mode Detection Using Mobile Phones and GIS Information. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ACM, New York, NY, USA, 2011, GIS '11, pp. 54–63.

[5] Elhoushi, M., J. Georgy, A. Noureldin, and M. J. Korenberg, Motion mode recognition for indoor pedestrian navigation using portable devices. *IEEE Transactions on Instrumentation and Measurement*, Vol. 65, No. 1, 2016, pp. 208–221.

[6] *Scikit Python Machine Learning Library*, 2018, http://scikit-learn.org/stable/.

[7] Breiman, L., Random forests. *Machine learning*, Vol. 45, No. 1, 2001, pp. 5–32.

[8] Weston, J. and C. Watkins, *Multi-class Support Vector Machines*, 1998.

[9] Freund, Y. and R. E. Schapire, Large Margin Classification Using the Perceptron Algorithm. *Mach. Learn.*, Vol. 37, No. 3, 1999, pp. 277–296.

[10] Kimpel, T., J. Strathman, D. Griffin, S. Callas, and R. Gerhart, Automatic passenger counter evaluation: Implications for national transit database reporting. *Transportation Research Record: Journal of the Transportation Research Board*, , No. 1835, 2003, pp. 93–100.

[11] Chen, C.-H., Y.-C. Chang, T.-Y. Chen, and D.-J. Wang, People counting system for getting in/out of a bus based on video processing. In *Intelligent Systems Design and Applications, 2008. ISDA'08. Eighth International Conference on*, IEEE, 2008, Vol. 3, pp. 565–569.

[12] Ashbrook, D. and T. Starner, Using GPS to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous Computing*, Vol. 7, No. 5, 2003, pp. 275–286.

[13] Zheng, Y., Q. Li, Y. Chen, X. Xie, and W.-Y. Ma, Understanding mobility based on GPS data. In *Proceedings of the 10th international conference on Ubiquitous computing*, ACM, 2008, pp. 312–321.

[14] Zheng, Y., Y. Chen, Q. Li, X. Xie, and W.-Y. Ma, Understanding Transportation Modes Based on GPS Data for Web Applications. *ACM Trans. Web*, Vol. 4, No. 1, 2010, pp. 1:1–1:36.

[15] Sohn, T., A. Varshavsky, A. LaMarca, M. Y. Chen, T. Choudhury, I. Smith, S. Consolvo, J. Hightower, W. G. Griswold, and E. de Lara, Mobility Detection Using Everyday GSM Traces. In *UbiComp 2006: Ubiquitous Computing* (P. Dourish and A. Friday, eds.), Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 212–224.

[16] Myrvoll, T. A., J. E. Håkegård, T. Matsui, and F. Septier, Counting public transport passenger using WiFi signatures of mobile devices. In *Intelligent Transportation Systems (ITSC), 2017 IEEE 20th International Conference on*, IEEE, 2017, pp. 1–6.

[17] Nham, B., K. Siangliulue, and S. Yeung, Predicting mode of transport from iphone accelerometer data. In *Machine Learning Final Projects*, Stanford, CA, USA, 2008.

[18] Hemminki, S., P. Nurmi, and S. Tarkoma, Accelerometer-based Transportation Mode Detection on Smartphones. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, ACM, New York, NY, USA, 2013, SenSys '13, pp. 13:1–13:14.

[19] Fang, S.-H., H.-H. Liao, Y.-X. Fei, K.-H. Chen, J.-W. Huang, Y.-D. Lu, and Y. Tsao, Transportation modes classification using sensors on smartphones. *Sensors*, Vol. 16, No. 8, 2016, p. 1324.

[20] Yu, M.-C., T. Yu, S.-C. Wang, C.-J. Lin, and E. Y. Chang, Big Data Small Footprint: The Design of a Low-power Classifier for Detecting Transportation Modes. *Proc. VLDB Endow.*, Vol. 7, No. 13, 2014, pp. 1429–1440.

[21] Reddy, S., M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava, Using Mobile Phones to Determine Transportation Modes. *ACM Trans. Sen. Netw.*, Vol. 6, No. 2, 2010, pp. 13:1–13:27.

[22] Jahangiri, A. and H. A. Rakha, Applying Machine Learning Techniques to Transportation Mode Recognition Using Mobile Phone Sensor Data. *IEEE Transactions on Intelligent Transportation Systems*, Vol. 16, No. 5, 2015, pp. 2406–2417.

[23] Tomasic, A., J. Zimmerman, A. Steinfeld, and Y. Huang, Motivating contribution in a participatory sensing system via quid-pro-quo. In *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*, ACM, 2014, pp. 979–988.

[24] Zhou, P., Y. Zheng, and M. Li, How long to wait?: predicting bus arrival time with mobile phone based participatory sensing. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, ACM, 2012, pp. 379–392.

[25] Thiagarajan, A., J. Biagioni, T. Gerlich, and J. Eriksson, Cooperative Transit Tracking Using Smart-phones. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, ACM, New York, NY, USA, 2010, SenSys '10, pp. 85–98.

[26] Pi, X., Z. S. Qian, A. Steinfeld, and Y. Huang, Understanding Human Perception of Bus Fullness: An Empirical Study of Crowdsourced Fullness Ratings and Automatic Passenger Counter Data. In *Transportation Research Board 97th Annual Meeting*, TRB, 2018.