TRANSPORTES
www.revistatransportes.org.br
anpet
associação nacional de pesquisa e ensino em transportes
www.anpet.org.br

UAI-FI: using artificial intelligence for automatic passenger counting through Wi-Fi and GPS data

UAI-FI: Utilização de Inteligência Artificial para contagem automática de passageiros através de Wi-Fi e dados GPS

Marcos Paulino Roriz Junior[1], **Ronny Marcelo Aliaga Medrano**[2], **Cristiano Farias Almeida**[3]

[1]Federal University of Goiás, Goiás – Brazil, marcosroriz@ufg.br
[2]Federal University of Goiás, Goiás – Brazil, ronnymarcelo@ufg.br
[3]Federal University of Goiás, Goiás – Brazil, cristianofarias@ufg.br

Recebido:
13 de fevereiro de 2021
Aceito para publicação:
27 de maio de 2022
Publicado:
6 de agosto de 2022
Editor de área:
Helena Beatriz Bettella Cybis

Keywords:
Passenger counting.
Bus load.
Wi-Fi.
Machine learning.

Palavras-chave:
Contagem de passageiros.
Lotação.
Wi-Fi.
Aprendizado de máquina.

DOI:10.14295/transportes.v30i2.2555

ABSTRACT
An important piece of information for planning public transportation is the number of passengers using the system. Several initiatives have started to explore the Wi-Fi packets generated by passengers' smartphones as means to obtain this information. A sensing device located inside the bus can intercept and collect these packets. By applying filters, e.g., verifying if the signal strength is higher than a threshold, the sensor can infer passengers' presence/absence. However, such limits are set arbitrarily, leading to errors, for example, when close to bus stops. To address this issue, this article proposes a method (UAI-FI) based on an artificial intelligence technique (Support Vector Machine) to classify the origin of packets as inside or outside the bus. To validate UAI-FI, we applied and compared our approach to other methods in a bus line in Goiânia/Brazil. The results suggest that UAI-FI outperformed existing methods. Furthermore, it successfully classified the packet's origin, obtaining 83.3% and 88.5% of the total number of passengers boarding and alighting the line. Despite the overall similarity, we highlight that UAI-FI's counting curve presented a delay compared to the manual count indicating that the frequency that Wi-Fi packets are sent can cause the presence/absence of passengers to be perceived at different stops.

RESUMO
Saber a quantidade de passageiros que utilizam o sistema de transporte público é uma informação importante para planejá-lo. Várias iniciativas começaram a explorar pacotes Wi-Fi gerados pelos smartphones de passageiros como meio de obter essa informação. Esses pacotes podem ser interceptados por sensores dentro dos ônibus. Através da aplicação de filtros, por exemplo, que verificam se a intensidade do sinal é maior que um limiar, o sensor pode inferir a presença/ausência do passageiro. Entretanto, tais limites são definidos arbitrariamente, podendo causar erros, por exemplo, quando próximos de pontos de paradas. Para resolver esse problema, este artigo propõe um método (UAI-FI) baseado em uma técnica de inteligência artificial (Support Vector Machine) para classificar a origem dos pacotes como dentro ou fora do ônibus. Para validar o UAI-FI, foi feito um teste e comparação com outros trabalhos em uma linha de ônibus de Goiânia/Brasil. Os resultados sugerem que o método obteve um desempenho superior as outras abordagens. Ademais, foi capaz de classificar com sucesso a fonte dos pacotes, contabilizando 83,3% e 88,5% do total de embarques e desembarques da linha. Apesar da similaridade, ressalta-se que a contagem apresentou uma defasagem temporal com a feita manualmente, indicando que a frequência na transmissão de pacotes Wi-Fi pode fazer com que a presença/ausência de passageiros seja percebida em paradas diferentes.

1. INTRODUCTION

When planning public transportation systems, it is important to know the busloads, that is, the number of passengers traveling at a given period (Myrvoll *et al.*, 2017). This information can be used as an input to optimize the operation, *e.g.,* reallocating vehicles or the number of trips to decrease the system overload and waiting time (Mishalani *et al.*, 2016).

Many works have started to explore the data generated by passengers' smartphones aboard the vehicle to automate the process of counting passengers and obtaining the busloads (Ji *et al.*, 2017; Oransirikul *et al.*, 2014). The overall idea is to explore the trace of Wi-Fi packets automatically produced by such devices while they are inside the bus.

Each packet contains the smartphone identifier, enabling subsequent packets to trace back to the same passenger (IEEE, 1997). In addition, it also includes a wave data, called **R**eceived **S**ignal **S**trength **I**ndicator (RSSI), which indicates how close the transmitting smartphone is to the receiver end. Since these packets are transmitted periodically, they can be used to estimate the presence/absence of passengers inside the vehicle.

However, the parameters that filter the packets coming from smartphones inside the bus from outside noise are configured empirically, usually by setting arbitrary thresholds. This makes the application of such approaches problematic due to the complexity of estimating such limits and possible collateral effects (Oransirikul *et al.*, 2019; Paradeda *et al.*, 2019).

For example, suppose that the algorithm counts a passenger inside the bus if its devices emit a packet whose Wi-Fi strength is higher than a given threshold, which can be mapped indirectly to a distance range (*e.g.*, few meters). This situation can lead to erroneously counting other passengers, such as those waiting at a stop station, since the bus can pass close to them (Dunlap *et al.*, 2016).

Thereby, motivated by these issues, this paper investigates the feasibility of **U**sing **A**rtificial **I**ntelligence for Automatic Passenger Counting through Wi-**FI** signals and GPS data (UAI-FI). Hence, it proposes a method that uses a machine learning technique, Support Vector Machine, to discover the threshold limits to count and obtain the busloads. To evaluate UAI-FI, we built and deployed a prototype system in field tests to compare the passenger count produced to values obtained manually. We also implemented and compared existing works to our results.

The rest of this paper is structured as follows. First, Section 2 presents the fundamental concepts that underlie this work. Then, Section 3 presents and discusses the UAI-FI method, while Section 4 presents the experiment used to evaluate our approach. Finally, Section 5 states the conclusion and future work associated with the limitations of the proposed method.

2. FUNDAMENTAL CONCEPTS

2.1. A glimpse on Wi-Fi

The Wi-Fi technology has been standardized by the Institute of Electrical and Electronics Engineers (IEEE) to enable wireless communication between devices (IEEE, 1997). As such, the standard defines a network discovery process that periodically probes (*scans*) nearby devices by broadcasting a network packet called *probe request* (Freudiger, 2015).

All devices with a Wi-Fi chip turned *on* emit this packet, even those not connected to a wireless network. These packets are publicly broadcast on wavebands near the sender, allowing all nearby devices to receive them. Smartphones simultaneously send probe requests to multiple channels to amplify the discovery process. The broadcasted range can reach up to 100 meters in an open field and 30 meters in closed environments (IEEE, 1997).

Devices reached by the probe request can compute a Received Signal Strength Indicator (RSSI), which measures the connectivity strength between them. This indicator, described in decibel-milliwatts (dBm), varies from 0 to $-\infty$, where the closer to zero, the closer the devices are. Thus, based on the receiving values, it is possible to indirectly estimate the distance range that the transmitting device is located. Alongside RSSI, each probe request also includes a value that globally identifies the smartphone that sent the packet through a digital fingerprint known as Media Access Control Address (MAC address). Due to privacy concerns, some companies have recently randomized the MAC address fingerprint sent in probe requests. However, several studies suggest that it is possible to trace back these packets to the same passenger device by comparing the payload data (Vanhoef *et al.*, 2016).

Since probe requests are sent periodically in public wavebands, a computer monitoring these frequencies can also receive them. These computers, called *sniffers*, can use these packets to infer the presence and absence of smartphones in their neighborhood.

To exemplify such concepts, consider the scenario illustrated in Figure 1, where a sniffer is located inside a bus that heads to a stop station. Here, passenger A is located inside the bus as well. When his/her device sends a probe request packet, it will be captured by the sniffer, which implies that A is in the vehicle. After A's alight, the sniffer will stop receiving packets from his/her device. When this happens, the sniffer infers that A left the bus. However, this approach needs to handle this workflow with care. Sniffers can capture data from noise sources, *e.g.*, from B's, which is waiting at the bus stop. Hence, we need a method to filter passengers' packets from noise to count them correctly.

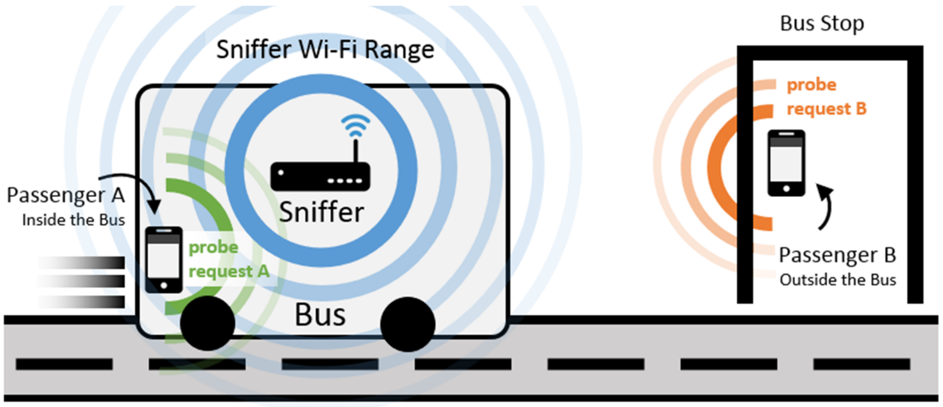


Figure 1. Example of using a sniffer to capture the presence of passengers

2.2. Current Approaches

Table 1 summarizes several strategies that explored Wi-Fi to obtain the busload. Such approaches usually define how to filter packets through arbitrary limits, *e.g.*, they check if the packet's RSSI is higher than a threshold to classify them as inside the bus. This can lead to wrongly counting passengers at bus stops or nearby drivers. For example, Oransirikul *et al.* (2016) reported a peak in false passengers when the vehicle arrives at a bus station. To further complicate, they present conflicting limits for the same features, demonstrating the difficulty in estimating them.

Oransirikul *et al.* (2019) and Nitti *et al.* (2020) included a frequency attribute, which measures the rate that packets are sent. However, defining this threshold is problematic because such frequency is not standardized (IEEE, 1997). Therefore, it is up to the smartphone vendor to define it. For instance, Freudiger (2015) reported that the average frequency of probe requests in iOS 8.1.3 is 330 seconds, while in Android 4.4.2 it is 72 seconds. Further, this rate is susceptible to how the smartphone is being used. For instance, the interval increases when the battery is low, while it decreases when the screen is turned on (Mikkelsen *et al.*, 2016). This variation can impact the algorithm's precision regarding the passenger's origin and destination. For example, the device of a given passenger may be perceived in subsequent stations where he/she boarded since his/her device can present a delay in transmitting its first probe request packet.

Table 1 – Comparison of Related Work

Authors	Features Used to Consider Incoming Packets From Passengers (≥)					
	RSSI	**Number of Packets**	**Frequency**	**Trav. Distance**	**Trav. Time**	**Trav. Speed**
Oransirikul *et al.* (2014)	−74 dBm	—	—	—	—	—
Mishalani *et al.* (2016)	−30 dBm	10	—	900 ft	—	—
Dunlap *et al.* (2016)	−79 dBm	3	—	300 ft	1 min	—
Mikkelsen *et al.* (2016)	−65 dBm	—	—	—	6 min	—
Oransirikul *et al.* (2016)	−72 dBm	3	—	—	1 min	—
Ji *et al.* (2017)	−60 dBm	10	—	400 m	—	—
Afshari *et al.* (2019)	−80 dBm	3	—	—	1 min	—
Ribeiro *et al.* (2019)	—	2	—	1000 m	10 min	—
Paradeda *et al.* (2019)	−79 dBm	2	—	—	—	—
Oransirikul *et al.* (2019)	−60 dBm	2	90 s	—	90 s	—
Nitti *et al.* (2020)	−65 dBm	5	4 min	—	—	—
Hidayat *et al.* (2020)	—	—	—	—	—	1 km/h
Chen *et al.* (2021)	ML	ML	—	—	—	—

Hidayat *et al.* (2020) propose an interesting approach to classify the packet's origin based on the smartphone's "traveling speed". If it is higher than a given threshold, the device packets can be associated with a passenger. They further refine this information by requiring that it occurs near bus stops ($\leq$ 10 m) and that their timestamps adhere to the bus line schedule. While interesting, we highlight that the approach can fail since there is no guarantee that probe requests will happen at stop stations.

Similar to our work, Chen *et al.* (2021) use machine learning to discover the thresholds. They explore the total number of packets, signal strength, and scanning time. This last feature contemplates the period that the sniffer actively perceives the smartphone in its neighborhood. The expectation is that noise sources would only appear for short periods in the sniffer range, while the passenger's device would be captured for longer periods. However, we note that this might not be necessary the case, as the transmission rate depends on how the smartphone is being used, *e.g.*, a device in the passenger's pockets sends few packets and appears for short periods (Freudiger, 2015). Finally, the authors test several techniques and highlight that Random Forest slightly outperforms (in 0.0077) other classification methods (Support Vector Machine and K-Nearest Neighbors). Nevertheless, it seems that the paper does not mention any calibration phase. Instead, they use default parameters, which can compromise the model performance.

Except for Chen *et al.* (2021), none of these works reported an experiment with buses to extract the features' limits. Instead, to understand the data, they fiddle with the thresholds empirically. Mishalani *et al.* (2016) noticed such limitations and suggested using machine learning to build a more robust method. As such, this work explores this path.

2.3. Machine Learning

The proposed method uses machine learning to discover the counting limits automatically instead of manually picking them. Hence, we need to provide a dataset of probe request packets and GPS data labeled as inside or outside the bus to discover such function (Flach, 2012).

The Support Vector Machine (SVM) stands out in binary classification among other techniques due to its high perfomance (Zhang *et al.*, 2017). The main idea of SVM is to discover the best hyperplane that can separate the data into different classes, *i.e.*, the one that has the largest margin between them (Cortes and Vapnik, 1995).

To exemplify how SVM works, consider a dataset with m n-dimensional data, where $\mathbf{x}^{(i)}$ is the i-th example ($i = [1, m]$). Each data is labeled to a class $y^{(i)} = \{-1, +1\}$. Consider Fig. 2 (a), here, there are nine ($m = 9$) samples with two dimensions ($n = 2$), x_1 and x_2, that are classified in a square ($+1$) and triangle class (-1). Note that any input data $\mathbf{x}^{(i)}$ located in the hyperplane satisfies the equation $\mathbf{w} \cdot \mathbf{x} + b = 0$, where $\mathbf{w}$ is a weight vector $(\mathrm{w}_1, \dots, \mathrm{w}_\mathrm{n})$, which is normal to the hyperplane, and b is a constant that indicates how far the plane is from the origin.

To find the hyperplane, SVM aims to discover a vector $\mathbf{w}$ such that:

$$\mathbf{w} \cdot \mathbf{x} + b \geq +1 \quad \forall \mathbf{x}^{(i)} \text{ that has class } y^{(i)} = +1, \text{ and}$$
$$\mathbf{w} \cdot \mathbf{x} + b \geq -1 \quad \forall \mathbf{x}^{(i)} \text{ that has class } y^{(i)} = -1$$

These two cases can be combined into one:

$$y^{(i)}\left(\mathbf{w} \cdot \mathbf{x}^{(i)} + b\right) \geq 1 \quad \forall \mathbf{x}^{(i)} \tag{1}$$

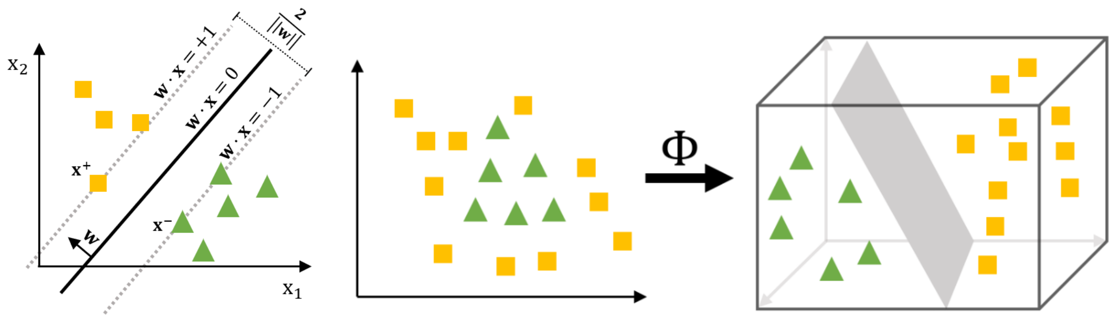


Figure 2. Basic concepts of Support Vector Machines (Cortes and Vapnik, 1995)

Vectors located on the margin are known as support vectors and they strictly satisfy Eq. 2. Hence, let $\mathbf{x}^+$ and $\mathbf{x}^-$ be two support vectors with classes $+1$ and -1, respectively. Then, one can calculate the margin size through the dot product:

$$\hat{\mathbf{w}} \cdot (\mathbf{x}^+ - \mathbf{x}^-) = \frac{(\mathbf{w}\cdot\mathbf{x}^+ - \mathbf{w}\cdot\mathbf{x}^-)}{\|\mathbf{w}\|} = \frac{(1-(-1))}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|} \tag{2}$$

SVM seeks to maximize this margin by minimizing $\|\mathbf{w}\|$. For mathematical convenience, it minimizes the quadratic norm $\|\mathbf{w}\|^2$. Based on that, the SVM problem can be stated as:

$$\begin{aligned} \min_{w,b} \quad & \|w\|^2 + C \textstyle\sum_{i=1}^{m} \xi^{(i)} \\ \text{s.t.} \quad & y^{(i)}\left(\mathbf{w} \cdot \mathbf{x}^{(i)} + b\right) \geq 1 - \xi^{(i)} \qquad \forall \mathbf{x}^{(i)} \\ \text{where} \quad & \xi^{(i)} = \max\left(0, 1 - y^{(i)}\left(\mathbf{w} \cdot \mathbf{x}^{(i)} + b\right)\right) \end{aligned} \tag{3}$$

Figure 2 (b) shows that not all data can be linearly separated. As such, SVM introduces a penalty variable to allow misclassifications. The slack variable $\xi^{(i)}$ represents the classification error for $\mathbf{x}^{(i)}$. Its value is zero if $\mathbf{x}^{(i)}$ is correctly classified, as $y^{(i)}(\mathbf{w} \cdot \mathbf{x}^{(i)} + b) \geq 1$. However, if the data is misclassified, it suffers a penalization proportional to its distance to the margin. The C parameter calibrates this impact. If it is small, it is preferable to maximize the margin and ignore some misclassifications. In contrast, if C is large, the error will have a higher impact on the objective function and, consequently, presents a smaller margin.

In addition, to deal with nonlinear problems, it is possible to apply a kernel function (ϕ) that maps the input data from $\mathbb{R}^n$ to $\mathbb{R}^k$, with $k \geq n$ (Cortes and Vapnik, 1995). For example, Figure 2 (b) illustrates the application of a kernel function $\phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ in a dataset that is not linearly separable in $\mathbb{R}^2$, but is in $\mathbb{R}^3$. SVM usually employ the following Radial Basis Function (RBF) kernel to project and compare data samples into a higher dimension:

$$\mathbf{RBF}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp(-\gamma \|\mathrm{x}^{(i)} - \mathrm{x}^{(j)}\|) \tag{4}$$

The γ parameter is the inverse degree of influence that a single piece of data has in the margin. If γ is large, then the radius of influence of a sample located far away from others is small. The advantage of using RBF is that it can compute the distance between projected data samples $\phi(\mathbf{x})$ in higher dimensions directly (Cortes and Vapnik, 1995).

3. METHODOLOGY

Motivated by the problem and concepts described, this section presents UAI-FI, a method based on three phases: building a sniffer, capturing packets and training an SVM classifier, and finally, execution and validation. Each phase is described in the following subsections.

3.1. Building a Sniffer

To capture the probe requests emitted by passengers' smartphones, we need to build a *sniffer* to monitor the packets sent in public wavebands. The sniffer should be placed inside the bus, as it can only intercept packets within its radius of influence, approximately 30 meters.

We recommend using embedded hardware platforms to build the device since they are small, cheap, and can be easily extended with other sensors (Richardson and Wallace, 2014). Thus, we built a sniffer using a Raspberry Pi 3B because it includes, by default, a Wi-Fi chip that can be used in monitor mode to capture packets. In addition, we coupled a GPS sensor (BU-353) to the platform to receive the bus's position data. The complete device, shown in Figure 3, costs approximately US$ 80.00 and has a dimension of 85,6 x 53,9 x 17 mm.

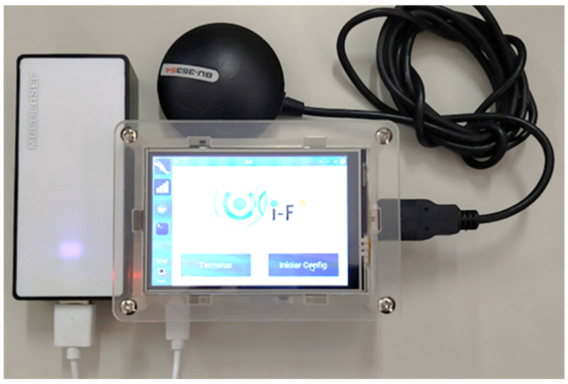


Figure 3. UAI-FI Sniffer: Raspberry PI 3B, with GPS sensor and Wi-Fi chip

Using the Wi-Fi and GPS sensors, we obtain the following features when intercepting a packet:

- MAC Address, the smartphone fingerprint, which is used to identify the passenger;
- RSSI, which indicates the Wi-Fi signal strength between the smartphone and the sniffer;
- Latitude and Longitude of the sniffer. It indirectly indicates the passenger's position in the bus line;
- Vehicle speed, indicating how fast the bus was when it received the packet;
- Sniffer timestamp, the instant where the packet was captured.

By using these variables, it is possible to derive the following features:

- Distance to the nearest bus stop, which aid in verifying if the received packet occurred close or far from a bus stop;
- Total number of packets received from the same device;
- Traveled distance and the travel time that the device stayed within the sniffer's presence.

As a result, each probe request forms an input vector $\mathbf{x} \in \mathbb{R}^6$ containing the following features: RSSI, bus speed, distance to the closest bus stop, number of packets sent, traveled distance, and traveled time.

3.2. Capturing Packets and Training the Model

The second phase of the method aims to capture the packets and train an SVM classifier. Therefore, first, it is necessary to obtain and label them manually. Each packet is associated with a class $y = \{-1, +1\}$, where -1 represents those that are outside the bus (noise), and $+1$ indicates the ones from passengers. This process should be done in a controlled manner. Specifically, packets should be captured using an empty bus with only known devices inside it. During the ride, all packets, except those produced by known devices, should be labeled as negative ($y = -1$) since they all come from noise sources. Similarly, data generated by known devices should be labeled as positive ($y = +1$).

After collecting the dataset, we train an SVM classifier to discriminate them. The dataset is split into two parts: training (70%) and testing (30%), as recommended by Flach (2012). The training dataset is used to find the support vectors that best separate the classes, while the test set is used to evaluate the accuracy of the classifier with unseen data.

During training, we try multiple SVMs with different combinations of C and γ hyperparameters. We vary them in the following range: $\gamma, C = [10^4, 10^3, 10^2, 10^1, 10^0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}]$, totalizing 81 different combinations. We also employ k-folding ($k = 10$), *i.e.*, the training set is repeatedly sampled in 10 different subsets of training and validation data to cross-validate the models. Out of the possible combinations of SVMs, we pick the one that presents the highest mean F_1 score considering the ten validation datasets.

The F_1 score represents the harmonic mean between precision and recall (Flach, 2012). We build a confusion matrix to compute its value, as it illustrates true positive (TP), true negative (TN), false positive (FP), and false negative (FN) contrast between the predicted classes and ground truth. Using these values, one can compute the metric as $F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$, where precision is expressed as $\frac{\text{TP}}{\text{TP+FP}}$ and recall as $\frac{\text{TP}}{\text{TP+FN}}$. Precision represents the percentage of packets UAI-FI assigned as inside the bus that is correct compared to the manual labeling. On the other hand, recall expresses the fraction of all passenger's packets that UAI-FI could retrieve and assign as inside the bus accordingly.

3.3. Execution and Validation

The last phase aims to incorporate the SVM classifier into the counting algorithm. In this step, researchers should conduct experiments on bus routes with ordinary passengers by capturing all packets generated throughout the trip. This dataset is used as input to count the number of passengers on the bus. Note that the objective here is different from the previous phase, as it aims to obtain the busload, whereas the last phase's goal was to find the function limits that classify the packets as inside or outside.

To count the bus passengers, UAI-FI follows the algorithm shown in Figure 4. First, it derives the input features from the captured data. Second, it predicts the packet location using the SVM classifier. If outside the bus, it proceeds to process the next packet. However, if it is inside, that passenger should be counted. Using the MAC address, the algorithm verifies if he/she has already been considered. If not, it increments the number of passengers and stores the packet for further queries. On the contrary, the algorithm updates its last seen timestamp if the passenger has already been counted. Finally, researchers must manually register the number of passengers who boarded and alighted the bus line. By doing so, it is possible to compare the results with the ones obtained by UAI-FI.

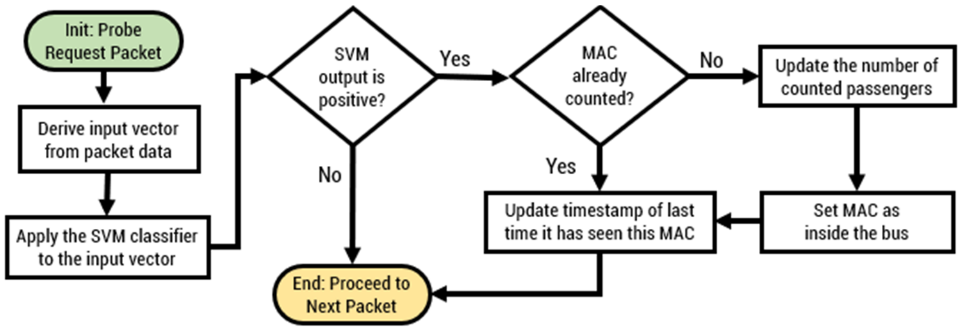

Init: Probe Request Packet
Derive input vector from packet data
Apply the SVM classifier to the input vector
SVM output is positive?
Yes
No
MAC already counted?
No
Yes
Update the number of counted passengers
Set MAC as inside the bus
Update timestamp of last time it has seen this MAC
End: Proceed to Next Packet

Figure 4. Algorithm for counting passengers

4. EXPERIMENTS AND RESULTS

To evaluate UAI-FI, we conducted experiments on a bus line of the Metropolitan Transit Network of the city of Goiânia, Goiás, Brazil. The analyzed bus line (305), illustrated in Fig. 5, has an extension of 12.41 km and interconnects two terminals. We choose to analyze this route as it is difficult to obtain its busload using other methods, such as those based on smartcards, since passengers do not swipe their cards when alighting in the terminal stations.

4.1. Captured Data

We captured the positive, negative, and ordinary packets on three days, totaling 10 trips. All experiments were conducted on working days from 09:00 to 12:00 on June 6 (Thursday), June 12 (Wednesday), and June 20 (Thursday) of 2019. During training rides, on empty buses, we carried 13 devices inside the vehicle.

The collected dataset contains a total of 20,470 packets. This dataset was further preprocessed to remove duplicates, as a smartphone can broadcast multiple probe requests simultaneously to different Wi-Fi channels (Freudiger, 2015). Hence, we summarize the packets received by a given MAC in the same instant by taking a mean of its features (*e.g.*, RSSI, bus speed). As a result, the preprocessed dataset contains 9,702 unique probe requests packets. Figure 6 illustrates a histogram of the features of the collected dataset. The histogram is in percentages, as the number of packets outside the bus significantly overcomes those insides.

As expected, there is an overlap in the RSSI of packets coming from devices located inside the bus from noise sources, see the interval from −80 dBm to −70 dBm as an example. For instance, a method that only considers probe requests whose signal strength is higher than −70 dBm as inside the bus would discard 21.6% of passengers' packets.

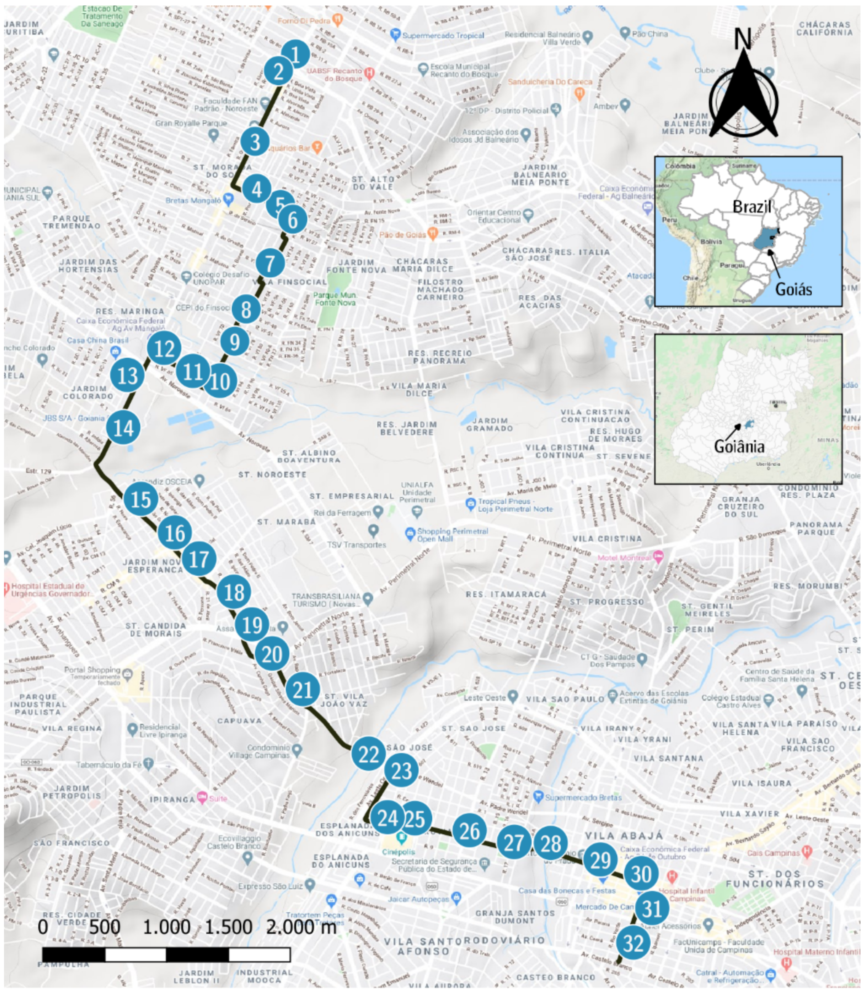


Figura 5. Route 305 in Goiânia, Goiás, Brazil. Adapted from RMTC (2018)

We expected the *bus speed* and *distance to the closest bus stop* features to be significantly larger for passengers, as their packets could be captured between stop stations. However, a closer look identified several noise sources in the same situation, potentially coming from nearby drivers and local stores. On the other hand, the traveled distance presented different values for the two cases. Over 91.4% of noise packets stay within the bus range for less than one km. The travel time feature also presented similar results, as over 80% of packets from outside sources stayed within the sniffer range for less than five minutes.

Finally, note how the number of packets received differs depending on the data origin. For example, the histogram shows that 84.3% of noise devices sent less than five probe requests, while it is 39% for those inside the bus. Combining these features, we aim to overcome their fuzziness and obtain the limit that separates them.

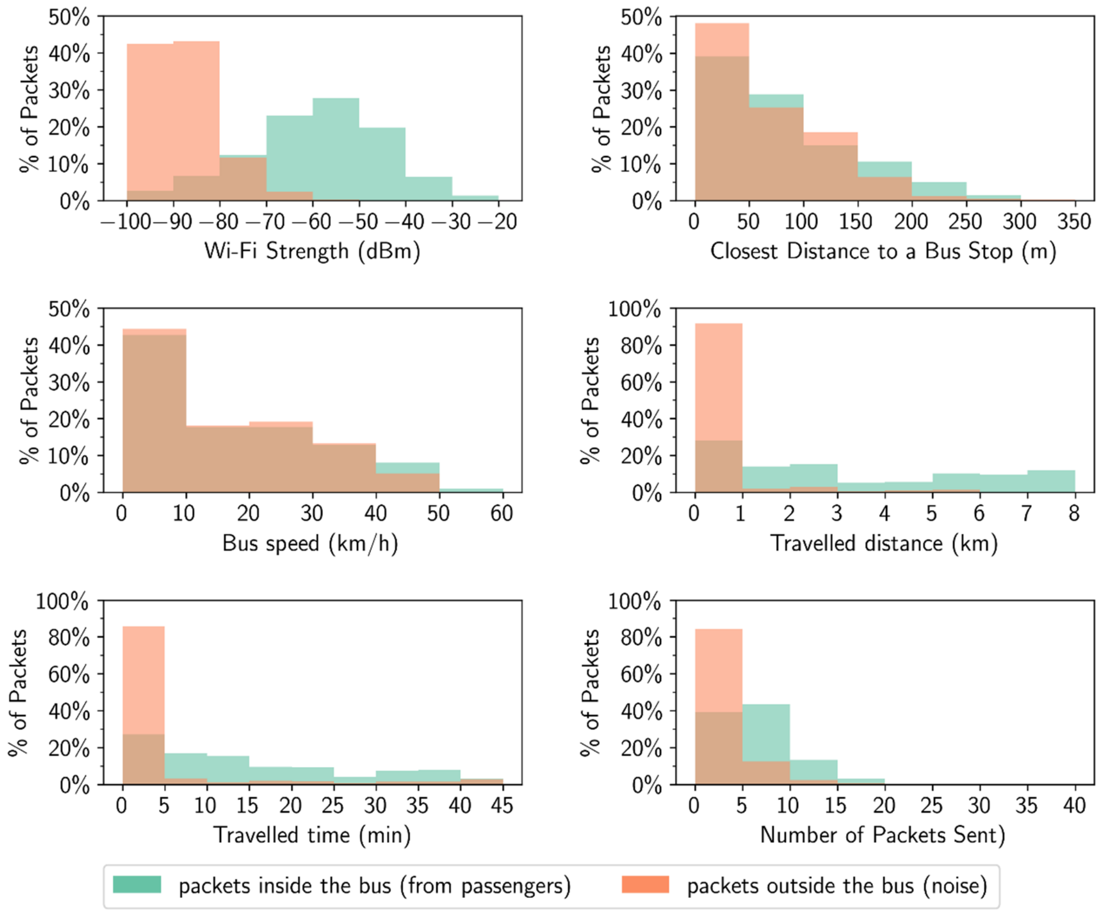


Figure 6. Histogram of collected packets' features

4.2. Training and testing

We trained 810 SVM classifiers as we employed 10-fold cross-validation and used 81 different combinations of C and γ with an RBF kernel. The model that yields the best result presented the parameters $C = 100$ and $\gamma = 1$. These values indicate that the best classifier is the one that significantly penalizes the errors committed (C). Further, a standard value for γ means that, although the misclassification penalty is significant, the influence of packets that are far away from others is reduced. Such value prevents the SVM from overfitting since outliers have a low influence on the remaining data.

As a result, UAI-FI presented a mean precision of 99.6% and a recall of 99.7% considering the 10 validation dataset used in training, yielding a mean F_1 score of 99.7%. We obtained similar results when we applied the model in the isolated test dataset, a 99.9% precision and a recall of 98.5%, resulting in a F_1 score of 99.2%. This means that over 99% of the manually labeled packets, from known devices and outside sources, are correctly classified. Such results suggest that UAI-FI can discriminate the packets' origin successfully.

To further compare UAI-FI, we implemented three different algorithms described in Subsection 2.2. The results, shown in Table 2, indicate the confusion matrices for the methods on the test dataset, *i.e.*, the data separated from training. Further, Figure 7 shows the number of passengers detected by each algorithm in a given trip of the test dataset.

The first comparison relates to the machine learning approach reported by Chen *et al.* (2021), which uses RSSI based-values, such as mean and standard deviation, alongside the total number of packets and scanning time received in a time window. The approach aggregates the receiving packets in temporal batches of 20 seconds. After that, it applies a Random Forest classifier to the collected data. To compare their approach, we reimplemented their algorithm and trained a Random Forest using the parameters they recommend, 11 trees and with depth equal to 7, in the same training dataset we used for UAI-FI. Table 2 (b) reports their result in the test dataset.

Table 2 – Comparison of UAI-FI with Related Works

		Manual Count	
		Inside	Outside
Prediction	Inside	873	1
	Outside	13	2024

(a) UAI-FI

		Manual Count	
		Inside	Outside
Prediction	Inside	675	19
	Outside	211	2006

(b) Chen *et al.* (2021)

		Manual Count	
		Inside	Outside
Prediction	Inside	763	44
	Outside	123	1981

(c) Afshari *et al.* (2019)

		Manual Count	
		Inside	Outside
Prediction	Inside	492	0
	Outside	394	2025

(d) Ji *et al.* (2017)

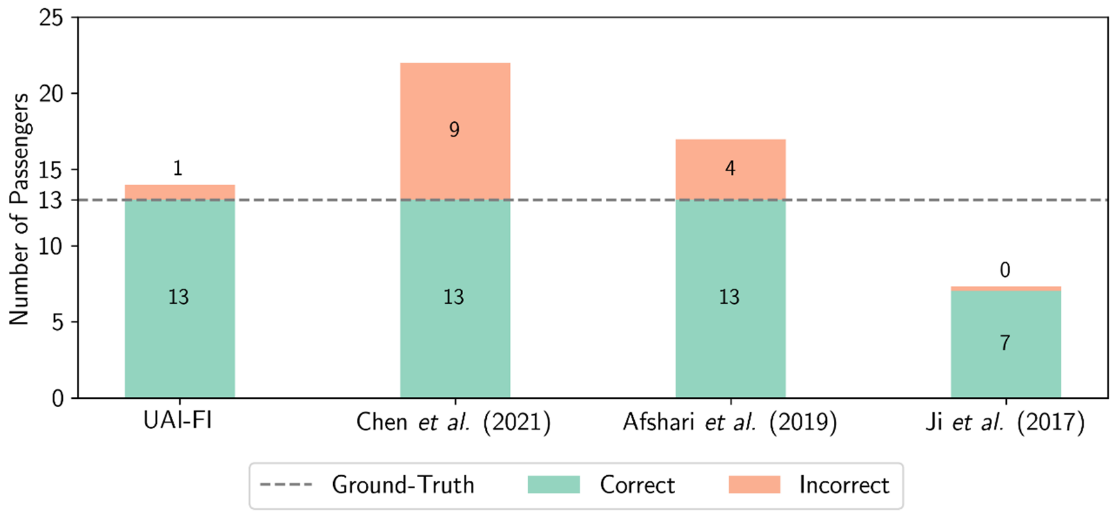


Figure 7. Number of passengers detected by each algorithm in a given trip of the test dataset

When classifying the packet's origin, the approach presented a precision of 96.7% and a recall of 74.4%, resulting in a F_1 score of 85.7%. However, a closer look shows that the algorithm counted several passengers from noise data, see Figure 7. A possible reason for this issue is the lack of distance or temporal features. A noise packet presenting a high RSSI signal strength can be classified as inside the bus since the approach analyzes each batch individually.

To overcome the duration problem, the filtering algorithm described in Afshari *et al.* (2019) counts a passenger only if his/her smartphone stays within the sniffer range for more than 30 seconds and sends at least 2 packets with an RSSI higher than −80 dBm. The approach presented a precision of 94.5% and a recall of 86.1% concerning the packets associated with inside the bus, resulting in an F_1 score of 90.1%. Yet, the algorithm also counted noise data as passengers, as shown in Figure 7. This is because smartphones only need to send few packets and stay within the sniffer ranger for a short period to be considered inside the bus.

On the other hand, Ji *et al.* (2017) presented a conservative algorithm requiring smartphones to 'travel' with the bus for at least 400 meters and send more than 10 packets with a mean RSSI value higher than −60 dBm to be considered as inside the vehicle. Consequently, the method has a high precision (100%). However, it fails to detect several passengers (a recall of 55.5%), having an accuracy of 71.4%. We understand that such recall is due to many smartphones presenting a low frequency in transmitting probe requests.

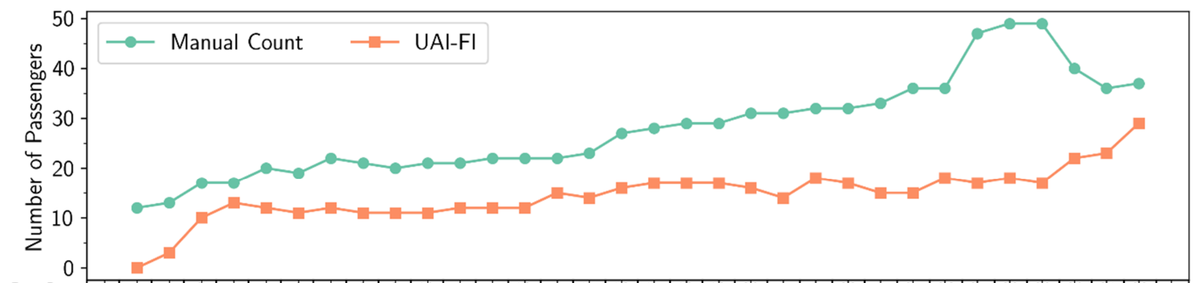
As can be seen, it is complicated to tinker with the parameters to count the passengers that are inside the vehicle. Soft limits lead to false positives, while hard thresholds fail to detect passengers rightfully. Further, the machine learning method proposed by Chen *et al.* (2021) also presented difficulty in counting the passengers. A possible reason is that it uses solely RSSI-based features and analyzes packets in individual batches. By including distance and temporal features in the method, in addition to a calibration phase and counting strategy, UAI-FI was able to learn the parameters and provide an F_1 score of 99.2%, correctly detecting all passengers in the test with only one false positive.

4.3. Validation with Ordinary Passengers

We also tested the method on bus trips with ordinary passengers. During this test, we manually counted that 72 passengers boarded and 35 exited the bus throughout the route's 32 stations, see Figure 8. In parallel, UAI-FI counted a total of 60 passengers boarding and 31 exiting the bus, *i.e.*, by incorporating the SVM model, which has an accuracy of over 99% for classifying the packet's origin, the method was able to imply for 83.3% and 88.5% of the total number of passengers that boarded and exited the vehicle.

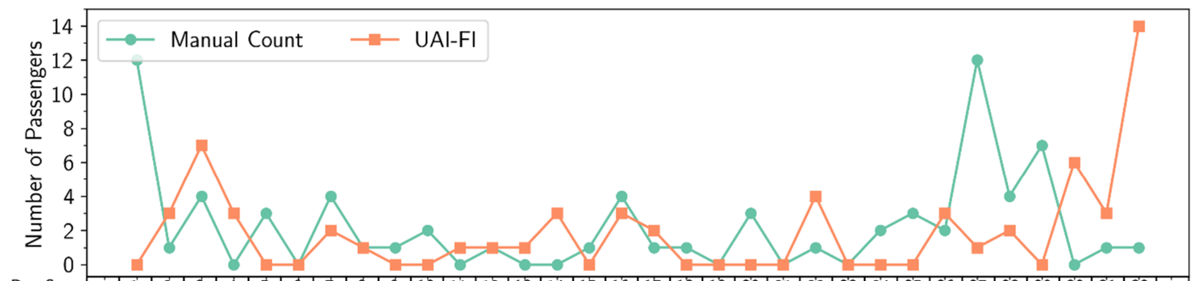
As expected, the total load in UAI-FI is inferior to the manual approach, as not all passengers have a smartphone or leave Wi-Fi *on*. Despite this, the results suggest that the method detects a significant part of the passengers. Specifically, note that UAI-FI counting curve shape is similar to the one obtained manually, but presents a delay of 2 to 5 bus stops. One explanation for this phenomenon is that some bus stops are reasonably close. Hence, the time to travel to the next stop can be lower than the frequency used by the passenger's device to send packets. For example, we observed that the time required for the vehicle to travel from bus stop 2 to 3 was 31 seconds. If the passenger's smartphone takes longer than this to send a packet, he/she will only be counted in the following stations. As reported by Freudiger (2015), some devices can take several minutes to transmit a packet, explaining the delay in the counting curve.

The delay phenomenon can be grasped when viewing the boarding and alighting curves at each bus stop, see Figures 8 (b) and (c). Note how UAI-FI results are slightly out of phase with the ground truth. For example, the manual curve counts 12 passengers at the first bus stop, the terminal station. However, due to the proximity of consequent bus stops, this was only reflected in UAI-FI at the fourth bus stop.



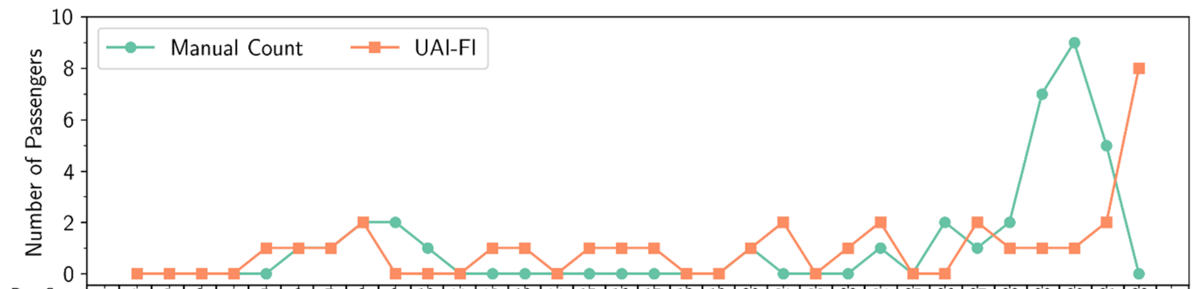
Bus Stop	–	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	–
Manual	–	12	13	17	17	20	19	22	21	20	21	21	22	22	22	23	27	28	29	29	31	31	32	32	33	36	36	47	49	49	40	36	37	–
Uai-Fi	–	0	3	10	13	12	11	12	11	11	11	12	12	12	15	14	16	17	17	17	16	14	18	17	15	15	18	17	18	17	22	23	29	–

(a) Total load inside the bus



Bus Stop	–	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	–
Manual	–	12	1	4	0	3	0	4	1	1	2	0	1	0	0	1	4	1	1	0	3	0	1	0	2	3	2	12	4	7	0	1	1	–
Uai-Fi	–	0	3	7	3	0	0	2	1	0	0	1	1	1	3	0	3	2	0	0	0	0	4	0	0	0	3	1	2	0	6	3	14	–

(b) Boarding curve



Bus Stop	–	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	–
Manual	–	0	0	0	0	0	1	1	2	2	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	2	1	2	7	9	5	0	–
Uai-Fi	–	0	0	0	0	1	1	1	2	0	0	0	1	1	0	1	1	1	0	0	1	2	0	1	2	0	0	2	1	1	1	2	8	–

(c) Alighting curve

Figure 8. Number of passengers that are boarding or alighting at each bus stop

A similar situation occurs when considering the load of passengers exiting. Note that in some cases, UAI-FI associates the passenger's exit with a bus stop before the one he left. One of the reasons for this is that that given station is the closest one to the last packet that UAI-FI received from that passenger.

Finally, it is worth noting the similarity between UAI-FI and ground truth curves throughout the bus line, especially its evolution. The data suggests that they follow a similar trend within a given delay.

5. CONCLUSION

This work investigated the viability of using Wi-Fi packets emitted by passengers' smartphones inside the bus, alongside the vehicle GPS data, as means to obtain the busload. The main reason for doing so is that unrestricted use of the Wi-Fi technology can eventually lead to counting false passengers, specifically when handling data from noise sources, such as nearby drivers. To address this problem, we proposed UAI-FI, a method that uses machine learning to learn how to discriminate the packet's origin. The method relies on Support Vector Machines, a technique that aims to find a hyperplane which can separate the data, in our case, the receiving packets.

We applied the method on a bus line in the city of Goiânia, Goiás, Brazil. During the controlled experiments, we manually labeled packets as inside and outside the bus. Using the training dataset, UAI-FI was able to obtain an F_1 score of 99.7% for the packet's origin, *i.e.*, the method was able to distinct packets from passengers from outside sources in over 99% of the cases. When considering the test dataset, the method also presented robust results, with an F_1 score of 99.2%. Furthermore, our results indicate that UAI-FI was able to outperform related works.

We also validated UAI-Fi with ordinary passengers in the same bus line. During the test, we manually observed that 72 passengers boarded the vehicle, while 35 exited the bus throughout the route. UAI-FI detected 60 passengers boarding and 31 exiting the vehicle in the same period. This result suggests that the method was able to *imply* respectively 83.3% and 88.5% of the total number of passengers that boarded and exited the bus considering the manual count.

Despite an overall similarity with ground truth, the results indicate a delay in UAI-FI's counting curve and, consequently, in the busload obtained. A closer look reveals that the primary suspect for this problem is the frequency that packets are received. Since Wi-Fi does not standardize the transmission rate that devices should emit probe requests, it can happen that a passenger board a vehicle at a bus stop and his/her smartphone only send in packet further down the bus line, *e.g.*, in subsequent stations. Since UAI-FI only detects the passenger when it receives the packets, this can delay the counting curve. For example, the results indicate that the busload at the first station was only detected in the fourth station by UAI-FI.

This problem can make it challenging to capture short trips, which can impact the overall busload. Furthermore, it complicates the extraction of origin-destination matrixes. In future works, we intend to investigate if clustering the first and last location of the same MAC addresses throughout a large timespan can yield a more precise origin and destination pair.

ACKNOWLEDGEMENTS

The work has been partially funded by the Brazilian National Council for Scientific and Technological Development (grant number 436660/2018-0).

REFERENCES

Afshari, H. H.; S. Jalali; A. H. Ghods and B. Raahemi (2019) An Intelligent Traffic Management System Based on the Wi-Fi and Bluetooth Sensing and Data Clustering. In K. Arai, R. Bhatia e S. Kapoor (eds) *Proceedings of the Future Technologies Conference*. Cham: Springer. DOI:10.1007/978-3-030-02686-8_24

Chen, T. Z.; Y. Y. Chen and J. H. Lai (2021) Estimating bus cross-sectional flow based on machine learning algorithm combined with wi-fi probe technology. *Sensors*, v. 21, n. 3. DOI:10.3390/s21030844

Cortes, C. and V. Vapnik. (1995) Support-vector networks. *Machine Learning*, v. 20, n. 3, p. 273–297. DOI:10.1007/BF00994018

Dunlap, M.; Z. Li; K. Henrickson and Y. Wang (2016) Estimation of Origin and Destination Information from Bluetooth and Wi-Fi Sensing for Transit. *Transportation Research Record: Journal of the Transportation Research Board*, v. *2595*, n. 1, p. 11–17. DOI:10.3141/2595-02

Flach, P. (2012) *Machine Learning: The Art and Science of Algorithms That Make Sense of Data*. New York: Cambridge University Press.

Freudiger, J. (2015) How Talkative is your Mobile Device? An Experimental Study of Wi-Fi Probe Requests. In *Proc. of the 8th ACM Conf. on Security & Privacy in Wireless and Mobile Networks*. New York: ACM. DOI:10.1145/2766498.2766517

Hidayat, A.; S. Terabe and H. Yaginuma. (2020) Estimating bus passenger volume based on a Wi-Fi scanner survey. *Transportation Research Interdisciplinary Perspectives*, v. 6. DOI:10.1016/j.trip.2020.100142

IEEE (1997) Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification. New York: Institute of Electrical and Electronics Engineers.

Ji, Y.; J. Zhao, Z. Zhang and Y. Du. (2017) Estimating Bus Loads and OD Flows Using Location-Stamped Farebox and Wi-Fi Signal Data. *Journal of Advanced Transportation*, v. 2017, p. 1–10. DOI:10.1155/2017/6374858

Mikkelsen, L.; R. Buchakchiev; T. Madsen and H. P. Schwefel (2016) Public transport occupancy estimation using WLAN probing. In *Proceedings of 8th International Workshop on Resilient Networks Design and Modeling*. New York: IEEE, p. 302–308. DOI:10.1109/RNDM.2016.7608302

Mishalani, R. G.; M. R. McCord and T. Reinhold (2016) Use of Mobile Device Wireless Signals to Determine Transit Route-Level Passenger Origin–Destination Flows: Methodology and Empirical Evaluation. *Transportation Research Record: Journal of the Transportation Research Board*, v. 2544, n. 1, p. 123–130. DOI:10.3141/2544-14

Myrvoll, T. A.; J. E. Håkegård; T. Matsui and F. Septier (2017) Counting public transport passenger using WiFi signatures of mobile devices. In *20th International Conference on Intelligent Transportation Systems*. New York: IEEE, p. 1–6. DOI:10.1109/ITSC.2017.8317687

Nitti, M.; F. Pinna; L. Pintor; V. Pilloni and B. Barabino (2020) iABACUS: A Wi-Fi-Based Automatic Bus Passenger Counting System. *Energies*, v. 13, n. 6. DOI:10.3390/en13061446

Oransirikul, T.; R. Nishide; I. Piumarta and H. Takada (2014) Measuring Bus Passenger Load by Monitoring Wi-Fi Transmissions from Mobile Devices. *Procedia Technology*, v. 18, p. 120–125. DOI:10.1016/j.protcy.2014.11.023

Oransirikul, T.; I. Piumarta and H. Takada (2019) Classifying passenger and non-passenger signals in public transportation by analysing mobile device Wi-Fi activity. *Journal of Information Processing*, v. 27, p. 25–32. DOI:10.2197/ipsjjip.27.25

Paradeda, D. B.; W. Kraus Junior and R. C. Carlson (2019) Bus passenger counts using Wi-Fi signals: some cautionary findings. *TRANSPORTES*, v. 27, n. 3, p. 115–130. DOI:10.14295/transportes.v27i3.2039

Ribeiro, J.; A. Zúquete and S. Sargento (2019) Survey of Passengers' Origin-Destination in Public Transportation Networks Using Wi-Fi. In *Communications in Computer and Information Science*. Cham: Springer, p. 367–388. DOI:10.1007/978-3-030-26633-2_18

Richardson, M and S. Wallace (2014) *Getting Started with Raspberry Pi: Electronic Projects with Python, Scratch, and Linux* (2nd ed). Santa Rosa: Maker Media.

RMTC. (2018) "RMTC Goiânia". Rede Metropolitana de Transportes Coletivos. Available at <http://rmtcgoiania.com.br> (accessed on: 13/02/2021).

Vanhoef, M.; C. Matte; M. Cunche; L. S. Cardoso and F. Piessens (2016) Why MAC address randomization is not enough: An analysis of Wi-Fi network discovery mechanisms. In *Proc. of the 11th ACM on Asia Conf. on Computer and Communications Security*. New York: ACM., p. 413–424. DOI:10.1145/2897845.2897883

Zhang, C.; C. Liu; X. Zhang and G. Almpanidis (2017) An up-to-date comparison of state-of-the-art classification algorithms. *Expert Systems with Applications*, v. 82, p. 128–150. DOI:10.1016/j.eswa.2017.04.003