IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 12, 2020, accepted November 30, 2020, date of publication December 23, 2020,
date of current version December 31, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3046912

Learning GPS Point Representations to Detect Anomalous Bus Trajectories

MICHAEL CRUZ AND LUCIANO BARBOSA
Centro de Informática (CIn), Universidade Federal de Pernambuco, Recife 50670-901, Brazil

Corresponding author: Michael Cruz (moc@cin.ufpe.br)

This work was supported by the INES 2.0, for his Ph.D. under Grant APQ 0388-1.03/14 e CNPq 465614/2014-0.

ABSTRACT Discovering anomalous bus trajectories can benefit transportation agencies to improve their services by helping them to deal with unexpected events such as detours or accidents. In this work, we propose a deep-learning strategy, which we name Spatial-Temporal Outlier Detector (STOD), that predicts the spatial/temporal anomaly degree of a bus trajectory by using learned representations of its GPS points. To calculate the score, STOD learns the regular behavior of bus trajectories by building a model that predicts the route id of buses. The degree of uncertainty on this prediction, measured by the entropy of the output class probability distribution, indicates the anomaly score of the trajectory. To perform the classification, STOD represents each point of a trajectory by the concatenation of two different representations. The first one (PAC embedding) is generated by the Point Activity Classifier (PAC) by leveraging temporal and spatial features on a stacked deep-learning model to predict the semantics of the point in terms of its bus activity (in route, bus stop, traffic signal, and other stops). The second representation (Geo embedding) captures the spatial relationship between a point and its geographical neighbors by applying a word embedding technique on the set of all trajectories. The experimental evaluation shows that our model is effective for filtering noisy trajectories since it outputs higher anomaly scores for both spatial and temporal anomalous trajectories than regular ones.

INDEX TERMS Anomaly score, representation learning, GPS bus trajectory.

I. INTRODUCTION

Public transportation is responsible for a reasonable part of ridership in big urban areas. To give some numbers: 35% of the workers in London commute by public transportation and 31% in São Paulo.[1] The quality of this service has hence a great impact on city people's lives.

Among the public transportation modes, transit buses are the most affected by unexpected factors (e.g., heavy traffic or accidents) since they share the roads with other vehicles and have a fixed route line. These factors can lead buses to present spatial or temporal behavior differently than usual. Finding such anomalies[2] might help transportation agencies improving its services, for example, by releasing more buses according to demand, redefining routes due to an accident or notifying bus drivers about detours.

Considering anomalies as instances that stand out as dissimilar to all others, different solutions have been proposed to detect them: statistical strategies based on distance [1] and density [2]; and machine learning approaches using supervised [3] and unsupervised [4] learning.

In this work, we aim to detect anomalous bus trajectories using supervised learning. A possible solution, in this direction, would be to train a binary classifier to predict a trajectory as anomalous or not. The main challenge in building such model for bus trajectories, though, is the lack of anomaly labeled data available. To deal with that, previous approaches [3], [5] have proposed to apply a classifier to model the regular behavior of bus trajectories by predicting their route line, and considering trajectories with classification probabilities lower than a certain threshold as outliers.

Similarly, we propose a multi-class classifier that learns the typical behavior of buses by predicting their line but, instead of performing a hard-boundary decision, our solution outputs an anomaly score of a bus trajectory based on the uncertainty of the classifier. Our assumption is that a trajectory classified with high uncertainty indicates a higher chance of it to

The associate editor coordinating the review of this manuscript and approving it for publication was Shajulin Benedict.

[1] https://www2.deloitte.com/xe/en/insights/focus/future-of-mobility/deloitte-urban-mobility-index-for-cities.html

[2] In this work, we use the terms outlier, anomaly, and noise mutually.

be spatial/temporal anomalous. Concretely, we calculate the anomaly score of a trajectory as the normalized entropy of the classifier's output class probability distribution: the higher the entropy, the higher the classifier uncertainty with respect to the route line of a given bus trajectory. In practice, our anomaly score can be used, for instance, by transportation agencies to rank bus trajectories in order to select the ones with the highest scores as candidate anomalous trajectories, serving thereby as filter to help analysts to inspect few trajectories, among many.

The trajectory classifier, which we call Spatial-Temporal Outlier Detector (STOD), is a deep learning model that heavily relies on representation learning [6]. Each GPS point in the input trajectories is represented by two learned vectors: (i) the PAC embedding, which embeds temporal and semantic patterns, and (ii) the Geo embedding, which captures geographical information.

The Point Activity Classifier (PAC), which is a stacked deep-learning model composed of recurrent and attention layers, learns a vector representation (PAC embedding) for each point in a given trajectory by classifying it into a set of activity points (in route, bus stop, traffic signal, and other stops) based on temporal and spatial features of the point and its neighboring points in the trajectory. This task is related to stay point prediction which allows detecting the semantic of stops in trajectories [7].

To create the Geo embedding, the points of the trajectories are mapped into a spatial grid where points in the same grid cell receives the same id. The Geo embeddings are then created by running a natural language word encoding [8] technique on the trajectories, in which each point is represented by its cell id. The Geo embedding, created by this process, captures the spatial relationship between a point and its geographical neighbors.

The trajectory points represented by the concatenation of their PAC and Geo embeddings feed a recurrent neural network, followed by a fully-connected network. The output of this network (STOD embedding) is passed to a softmax layer that outputs the class probability distribution of the input trajectory used to calculate the anomaly score.

The STOD embedding encodes relevant aspects regarding temporal and spatial behavior of the bus trajectory. As a result, STOD can also be used as a trajectory feature extractor to produce input features (STOD embedding) to any classification algorithm for bus route identification or to find similar trajectories [9].

We have performed an extensive experimental evaluation on datasets of two cities: Recife in Brazil and Dublin in Ireland. The results show that:

- Our anomaly score is in fact able to filter spatial and temporal anomalous trajectories since they present higher anomaly scores than regular ones in the experiments;
- STOD is effective in classifying trajectories in their respective route id and outperforms a previous approach [3] for the same task;
- The learned representation of the trajectories created by STOD produces high-quality results as input to different classifiers. In fact, the KNN classifier achieved the best results in the bus route classification, showing that the STOD embedding is also useful to find similar trajectories.
- PAC is effective to learn relevant information from the trajectory's points and classify stay points.

The source code of whole solution is available on Github.[3]

The remainder of the paper is organized as follows. In Section II, we present some background concepts and the problem statement. Section III delineates the proposed method and Section IV describes datasets, and setup experimentation. We compare our results with classical machine learning algorithms and previous researches in Section V. Section VI provides the state of the art on the anomaly detection in GPS trajectory. Finally, conclusions and futures works are drawn in Section VII.

II. PROBLEM DEFINITION

In this section, we provide some background concepts and state the problem that we deal with in this work.

Definition 1 (Bus Trajectory): We define a bus trajectory T_l as a sequence of consecutive GPS points collected from a bus trip, denoted as $T_l = \{p_1, p_2, \ldots, p_n\}$. Each point $p_i = \{lat_i, lng_i, tsp_i\}$ is composed of latitude (lat_i), longitude (lng_i) and timestamp (tsp_i). T_l is associated to a bus line l, and ordered by the points' timestamp, i.e., $tsp_i < tsp_{i+1}$.

Definition 2 (Spatial Anomaly Trajectory): A spatial anomaly trajectory T'_l contains points that spatially diverge from regular trajectories of the assigned bus line l of T'_l.

Definition 3 (Temporal Anomaly Trajectory): A temporal anomaly trajectory T''_l has a temporal behavior that deviates from regular trajectories of line l. We define two types of temporal anomalies. Trajectories with the Temporal Anomaly Type I have buses running slower than the regular behavior of trajectories of l (more congestion than usual). Trajectories with the Temporal Anomaly Type II, on the other hand, have vehicles moving faster than the regular behavior of trajectories of l (less congestion than usual).

Definition 4 (Problem Statement): Given a bus trajectory T_l, we aim to calculate the spatial-temporal anomaly score function $f : T_l \rightarrow \mathbb{R}$, such that $f(T_l) \in [0, 1]$.

III. METHODOLOGY

This work proposes a spatial-temporal outlier scoring method for bus trajectories data. Similar to previous approaches [3], [10], we calculate the score based on a classification task. More specifically, we build a multi-class classifier that learns spatio-temporal patterns of regular bus trajectories in k bus lines. The anomaly score of a trajectory T is based on the confidence degree of the classifier in predicting the class of T. We measure this confidence by calculating the entropy of the

[3]https://github.com/michaeloc/its_research

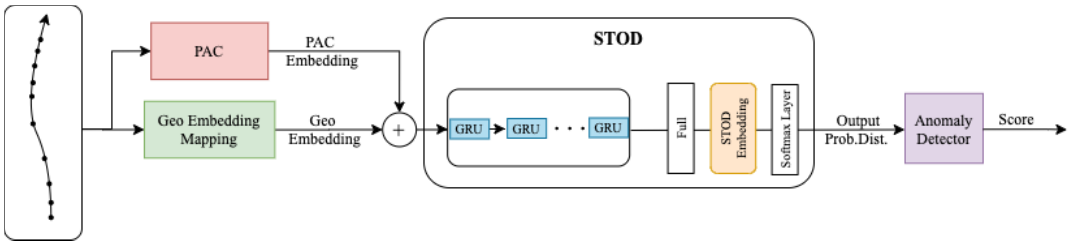


FIGURE 1. Overview of our anomaly score approach.

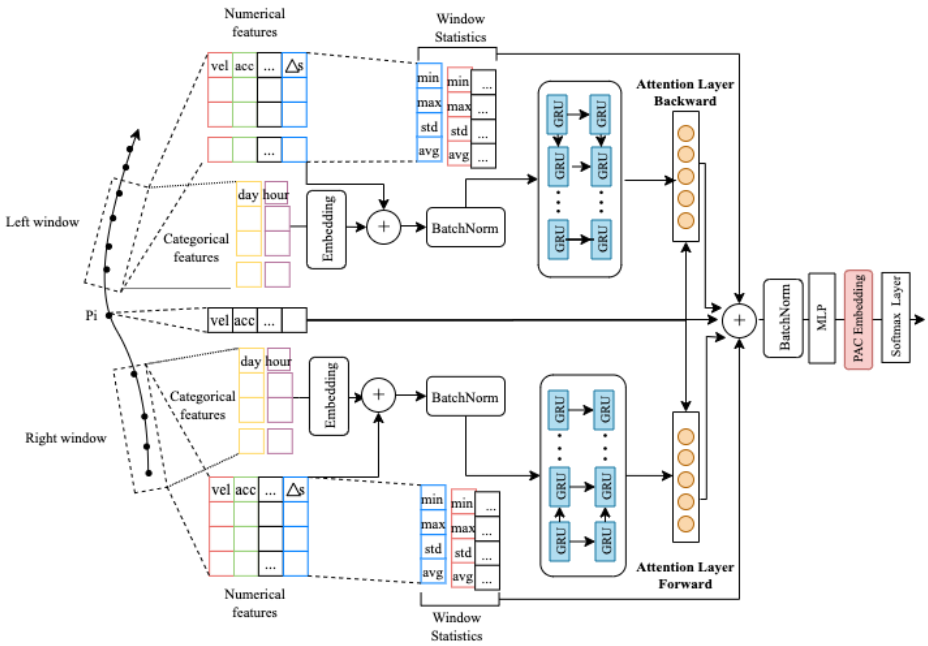


FIGURE 2. PAC model.

output probability distribution of T belonging to each one of k classes.

To perform the route classification, instead of representing each trajectory point p_i with its raw features (lat_i, lng_i, tsp_i), p_i is represented by a concatenation of vectors learned from two different tasks. In the first one, we use a neural network (Point Activity Classifier) as a feature extractor to learn a vector representation for p_i (PAC embedding) by predicting whether p_i is moving or is one of the 3 different types of stay points: bus stop, traffic light or other type of stay point. The second strategy uses a word embedding algorithm [11] to produce a vector representation (Geo embedding) for each point in a new space, which captures the spatial relationship between neighboring points. The points of T, represented by the concatenation of their PAC and Geo embedding, are fed into a deep learning model, called *STOD* (Spatial-Temporal Outlier Detection), to predict the probability distribution of T belonging to the k classes. Finally, this distribution is passed to the Anomaly Detector to calculate the anomaly score of T based on entropy, as mentioned before. An overview of the whole solution is presented in Figure 1. In the remaining of this section, we provide further details about the whole solution.

A. POINT ACTIVITY CLASSIFICATION

As mentioned before, the point activity classifier (PAC) learns representations of trajectory points that embed information of the point itself and its neighboring points using a supervised learning strategy. For that, we devised a deep learning model that predicts whether a given GPS point p_i in T is in one of the 4 distinct states: ''in route'', ''bus stop'', ''traffic light'' or ''other stop''. The class ''in route'' indicates that the bus is moving while ''other stop'' is any stop not labeled as ''bus stop'' or ''traffic signals'', which might occur, for example, due to an accident or heavy traffic.

Alongside the prediction, the model works as a feature extractor by producing a vector representation of p_i (PAC Embedding) as shown in Figure 2. The main goal of PAC is therefore to create a vector to represent any point in a trajectory. That is the reason we included the ''in route'' class as one of the PAC's output classes, even though it is straightforward to identify points in this class based on speed.

Point activity classification is very related to the task of stay point prediction that identifies the semantic of stops in trajectories [7], which helps to understand how vehicles are being utilized. For example, buses of a given route normally follow the same path which has the same number of bus stops

and traffic signals, but they can present different behaviors due to factors as period of the day, weather conditions and city dynamics. Discovering the meaning of each stop opens the opportunity to understand anomalous spatial-temporal behavior in trajectories. Stay point prediction can be considered as a sub-task of point activity classification, since 3 out of 4 classes predicted by PAC are related to stay points. As presented in Figure 2, to predict the class of a given point p_i, the PAC network receives as input p_i and the k consecutive points before $[p_{i-k} : p_{i-1}]$ and after it $[p_{i+1} : p_{i+k}]$, which we call left and right windows.

Since the bus behavior is heavily influenced by the day of week and hour of the day, they are used as features, extracted from the timestamp of the point. Similar to [12], instead of using their raw representation, PAC considers them as categorical values and applies Entity embeddings [13] for each feature set to map sparse one-hot encoded inputs of the categories to a dense and lower dimensionality.

PAC also uses the numerical features: latitude, longitude and timestamp of the point; acceleration and point-wise distance, which are calculated between consecutive points, since buses in stay points can have different behavior regarding these features. We utilize the Vincenty's formula [14] to compute the geographical distance between two points. Another feature is travel distance, which is the geographic distance between the initial point of the trip and the current point, calculated by the cumulative point-wise distance. It tries to capture events in specific locations of the trajectory. The final feature is the bearing rate [15], which is the absolute difference of the bearing of (p_{j-1}, p_j) minus the bearing of (p_j, p_{j+1}). The level of changing of direction of a bus in a point calculated by the bearing rate might indicate the type of stay point. For instance, buses usually change slightly their directions when they stop at a bus stop as opposed to at a traffic light. The bearing of two consecutive points p_{i-1} and p_i is calculated as:

$$y = \sin(lng_{p_i} - lng_{p_{i-1}}) * \cos(lat_{p_i}) \tag{1}$$

$$x = \cos(lat_{p_{i-1}}) * \sin(lat_{p_{i+1}}) \tag{2}$$

$$z = \sin(lat_{p_{i-1}}) * \cos(lat_{p_{i+1}}) * \cos(lng_{p_{i+1}} - lng_{p_{i-1}}) \tag{3}$$

$$bearing_{p_{i+1}} = \arctan(y, (x - z)) \tag{4}$$

where sin, cos, and arctan are the respectively trigonometric functions: sine, consine and arctangent. Latitude (lat) and longitude (lng) are passed in radians.

In each batch, the network applies batch normalization [16] on the values of these features to normalize them in order to give numerical stability to the model:

$$\bar{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{5}$$

$$z_i = \gamma * \bar{x} + \beta \tag{6}$$

where μ_B and σ^2 are respectively the batch mean and standard deviation, ϵ is a stability factor added to variance to avoid a division by zero, γ and β are learning parameters, and z_i is the normalized value of x_i.

After the batch normalization layer, we use two GRUs to capture the context before and after of p_i: the points in the left window $[p_{i-k} : p_{i-1}]$ are passed to a forward Gated Recurrent Unit (GRU), and the points in the right window $[p_{i+1} : p_{i+k}]$ to a backward GRU.

Next, the network applies the attention mechanism to learn which points before and after p_i are more relevant to classify it. More specifically, the hidden states of the forward GRU and the normalized features of p_i are fed into an attention layer, which weights the GRU hidden states according to p_i, creating an attention vector. Similarly, the hidden states of the backward GRU and p_i are used by an attention layer to produce an attention vector with respect to the right window. The attention vector v_{att} is calculated as follows:

$$e_j = f(p_i, v_j) \tag{7}$$

$$\alpha_j = \frac{exp(e_j)}{\sum_{k=1}^{h} e_k} \tag{8}$$

$$v_{att} = \sum_{j=1}^{h} \alpha_j v_j \tag{9}$$

where p_i is the input point, v_j is one of the GRU's hidden state, f is the hyperbolic tangent activation function in our implementation, and h is the number of output hidden states of the GRUs.

In addition to obtain the attention vectors from the left and right windows, the network also extracts statistics about points in both windows using a sliding window strategy, similar to [15]. Concretely, for each n consecutive points in $[p_{i-k} : p_{i-1}]$ and $[p_{i+1} : p_{i+k}]$, the model computes the mean, standard deviation, min, max, and median of the features: velocity, acceleration, distance, bearing and travel distance.

Lastly, the statistics from the left and right windows, their attention vectors and the features of p_i are passed to an MLP network with 3 fully-connected layers (a dropout layer is placed after the first full connected one). On top of the network, a softmax function predicts the probability of p_i belonging to each one of the 4 states (in route, bus stop, traffic light and other kind of stop). The output of the last hidden layer is the vector representation of p_i (PAC Embedding).

B. ANOMALY TRAJECTORY DETECTION

As we mentioned before, our solution predicts the anomaly score of bus trajectories based on a classification task. For that, our classifier predicts the probability of a trajectory T belonging to n possible route ids, and uses the classification's degree of confidence as the anomaly score. We calculate this confidence by measuring the entropy of the probability distribution of the n classes predicted by the classifier for T. We normalize the entropy to have values between 0 and 1: a value closer to 0 means the classifier has high confidence of predicting the class of T whereas values closer to 1 indicates the classifier is uncertain about the prediction. The anomaly

score of T is therefore the normalized entropy $E(T)$:

$$E(T) = \sum_{i=1}^{n} -P(T)_i \log_n P(T)_i \tag{10}$$

where $P(T)_i$ is the probability of the trajectory T being in class i, and n is the number of classes being predicted by the classifier. The normalization is performed by setting the log base equals to n, since the maximum entropy value is $log_n(n) = 1$.

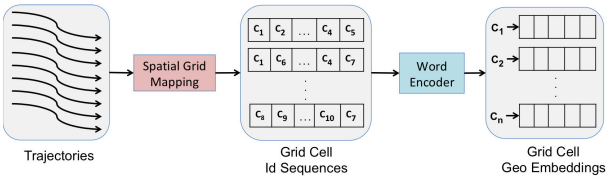


FIGURE 3. Geo embedding pipeline.

To perform the route id classification, the model represents each point of the trajectory using two vectors: PAC embedding, which is learned by the PAC model as explained in Section III-A; and Geo embedding that represents a point based on its geo location and its neighbors. To create Geo embedding, as shown in Figure 3, we first map the points of trajectories to a grid of hexagons using the H3 technique.[4] Each point is assigned to its corresponding cell (hexagon), represented by an id. A trajectory is then transformed into a sequence of cell ids in which its points lay on the grid. These grid cell id sequences are then used as input to the Word Encoder (Word2Vec [11] in our implementation) that outputs for each cell in the grid a dense vector (Geo embedding) that captures the spatial relationship between neighboring cells. At execution time, each point of a trajectory is mapped to its respective grid cell id to retrieve its geo embedding.

As depicted in Figure 1, the classification model, which we call Spatial-Temporal Outlier Detection (STOD), receives as input the points of the trajectory, in which each point is represented by the concatenation of its PAC and geo embeddings, feeding a forward GRU model. Next, the hidden states of this GRU are passed to an MLP with 1 fully connected layer.[5] On top of the network, a softmax layer predicts the probability distribution of T belonging to the k routes, which is used to calculate the anomaly score $E(T)$. Alongside the classification, STOD produces an embedding (STOD embedding) that is the input vector to the softmax layer and encodes relevant aspects regarding temporal and spatial behavior of buses as our experimental evaluation in Section V shows.

C. TRAINING

To train both models (PAC and STOD), we use the focal loss [17] function that deals with highly unbalanced data, which is the case of our input data as we show in Section V, leading to better performance than traditional multi-class loss functions. The focal loss is described as:

$$FL(p) = \sum_{t=1}^{n} \alpha_t (1 - p_t)^{\gamma_t} * log(p_t) \tag{11}$$

where n is the number of classes; p_t is the probability for the class t; α_t is a hyper-parameter that balances the importance of the classes; and the $\gamma_t \geq 0$ is the tunable focusing parameter for each class that is used to down-weight easy examples and pay more attention on hard classes, i.e., few instances in the training data.

IV. DATA DESCRIPTION AND SETUP

A. EXPERIMENTAL SETUP

1) BUS TRAJECTORY DATASETS

We used datasets of two different cities in our evaluation:

- Recife[6] (Brazil): The Recife dataset comprises 82 bus routes and has GPS points collected every 30 seconds from 18 days between October and November 2017. Each data point contains longitude, latitude, timestamp, route id, vehicle id, instantaneous velocity, and travel distance. The dataset comprises 27,897 trajectories composed of 2,675,468 points from 238 buses. The trajectories cover an average distance of 10 kilometers and have an average of 70 GPS points.
- Dublin[7] (Ireland): The Dublin dataset contains 66 routes, 17,701 trajectories with a total of 1,699,022 points collected from 3 days on January 2013. The average size of a trajectory is 11 kilometers and each trajectory has 190 points on average. We used the following information of the GPS points: timestamp, line id, longitude, latitude, journey id, and vehicle id. We use line id along with journey id and vehicle id to form a composite key to uniquely identify trajectories. As opposed to the Recife dataset, the timestamp frequency on the Dublin dataset's GPS points does not have a regular pattern, the average time interval between consecutive points is 22 seconds.

To help building anomaly trajectories, used in our experiments in Section V, we collected GTFS (General Transit Feed Specification)[8] files for both cities. GTFS is a standard format used for agencies to publish transit data. It is composed of files that define, for instance, the location of bus stops, pre-assigned routes for bus lines etc. For this evaluation, we used the *shape.txt* file containing the latitude and longitude of each point in the pre-assigned routes for each bus line. Furthermore, for the point activity classification task, we obtain the labels traffic light and bus stop for Dublin and the bus stop

[4]UBER H3: https: //eng.uber.com/h3/

[5]Even though it is more common passing only the last hidden state, experimentally we did not find any significant difference in the performance of the model using this strategy.

[6]We obtained the Recife dataset from a collaboration with the local government.

[7]https://data.gov.ie/dataset/dublin-bus-gps-sample-data-from-dublin-city-council-insight-project

[8]https://developers.google.com/transit/gtfs

for Recife from *OpenStreetMap*,[9] and traffic light for Recife from the city's open data website.[10]

2) PRE-PROCESSING

We performed the following pre-processing tasks over the raw data to feed the models: cleaning, trajectory segmentation and feature extraction. The cleaning step removes points with missing attributes: points without latitude, longitude or timestamp. Next, we perform the trajectory segmentation by considering as a single trajectory consecutive points with time difference lower than 5 minutes. Finally, the feature extraction generates the set of features for each point mentioned in Section III.

3) POINT ACTIVITY APPROACHES

We assess the quality of the PAC embeddings by comparing traditional classification models using PAC embeddings as features versus PAC's original features. The PAC model was trained with the left and right windows equal to 16. For all experiments, we split the datasets in 64% for training, 16% for validation and 20% for test. We trained all approaches based on searching for the best hyperparameters. To choose the best values of the hyper-parameters for the PAC network, we used the hyper-parameter optimization framework Hyperas, which is a wrapper of hyperopt [18]. The hyper-parameters, the values that we used to search and the best values for each dataset based on the validation set is shown in Table 1.

TABLE 1. Values of hyper-parameters of PAC approach.

		Best Values	
Hyper-parameters	Values	Recife	Dublin
Dropout	$5*10^{-2}, 15*10^{-2}, 10^{-1}$	10^{-2}	10^{-2}
Dense1	32, 64, 128	32	32
Dense2	32, 64, 128	64	64
Learning rate	$7*10^{-4}, 10^{-2}, 10^{-1}$	10^{-2}	10^{-2}
Optimizers	adam, rmsprop, adagrad	adagrad	adagrad
Batch	32, 64, 128	64	64

4) CLASSIFIERS

We apply the following traditional classification models in this evaluation:

- Random Forest (random forest): it is an ensemble of decision trees classifiers that are trained with the bagging method [19]. The hyperparameters n-estimators and max-depth were optimized in a range of values $\{2 <= x <= 50\}$ and $\{1 <= x <= 32\}$.
- Gaussian Naive Bayes (gaussian nb): it is a classifier based on Bayes' theorem, and features independence [20]. No hyperparameter search was performed.
- Support Vector Machine (svm): svm is a non-probabilistic classifier characterized by finding hyperplanes (support vectors) that maximizes the margin between classes [21]. We used a linear kernel with the regularization hyperparameter C ranging between $\{1e^{-10} <= x <= 1e^{10}\}$.
- Light Gradient Boost (lgbm): lgbm is an improved Gradient Boosting which is based on an ensemble of decision trees and boosting method [22]. We varied the values of hyperparameters n-estimators [gbdt,dart,goss], number of leaves $\{10 <= x <= 150\}$, learning rate $\{1e^{-5} <= x <= 1\}$ and feature fraction $\{0 <= x <= 1\}$.
- K-Nearest Neighbors (knn): it is a lazy and non-parametric method that classifies examples based on a similarity measure [23]. We varied the number of neighbors in the range $\{1 <= x <= 30\}$ and the KNN's algorithm [auto, ball_tree, kd_tree, brute].

TABLE 2. Values of hyper-parameters of STOD.

		Best Values	
Hyper-parameters	Values	Recife	Dublin
GRU nodes	32, 64, 128	128	64
Full connected nodes	32, 500, 1000, 2000	1000	2000
Learning rates	$10^{-3}, 10^{-2}, 10^{-1}$	0.1	0.1
Optmizers	adam, rmsprop, adagrad	rmsprop	rmsprop
Batch size	32,64,128	64	128

In order to choose the values of the hyper-parameters, we used the hyperparameter optimization framework Optuna [24].

5) ROUTE CLASSIFICATION APPROACHES

We executed the following bus route classification strategies:

- Riobusdata [3] uses a Convolutional Neural Network (CNN) fed by raw bus trajectories where each point is composed of latitude, longitude, and timestamp.
- STOD(geo,pac) is our solution that uses as input the geo and PAC embeddings. To generate the geo embeddings, the Spatial Grid Mapping is implemented using UBER's H3 [25] and the Word Encoder is Word2Vec with the window size set to 5. We set the output size of attention layer in 128. We also executed a variation of our classifier that uses the point's timestamp instead of its PAC embedding STOD(geo,time).

For all approaches, the input trajectory has at most 100 points, and they were implemented using Keras.[11] We trained them for 100 epochs and selected the best models by early-stopping accuracy validation. For these experiments, we split the datasets in 64% for training, 16% for validation and 20% for test. We used Hyperas[12] to tune the some of the models' hyper-parameters. Table 2 and Table 3 show the tuning and best values used for STOD and Riobusdata respectively.

To measure the performance, we used weighted Precision (WP), weighted Recall (WR), and weighted F1 (WF1), which

[9] https://www.openstreetmap.org/

[10] http://dados.recife.pe.gov.br/dataset/localizacao-dos-semaforos/resource/ab6343e9-c3f2-4d62-9554-5778f9f33738

[11] https://keras.io/

[12] https://github.com/maxpumperla/hyperas

TABLE 3. Values of hyper-parameters of RioBusData.

		Best Values	
Hyper-parameters	Values	Recife	Dublin
Conv1	32, 64, 128	128	32
Conv2	32, 64, 128	64	128
Conv3	32, 64, 128	64	64
Conv4	64, 128, 256	128	128
Conv5	64, 128, 256	128	64
Conv6	64, 128, 256	128	256
Dense1	500, 1000, 2000	500	1000
Dense2	500, 1000, 2000	500	500
Learning rate	$10^{-3}, 10^{-2}, 10^{-1}$	10^{-2}	10^{-1}
Optimizers	rmsprop, adam, adagrad	adam	adam
Batch	32,64,128	128	128

are suitable to evaluate techniques that work on unbalanced data problems:

$$WF1 = \frac{\sum_{i=1}^{n} w_i * F1_i}{\sum_{i=1}^{n} w_i} \quad (12)$$

$$WR = \frac{\sum_{i=1}^{n} w_i * Recall_i}{\sum_{i=1}^{n} w_i} \quad (13)$$

$$WP = \frac{\sum_{i=1}^{n} w_i * Precision_i}{\sum_{i=1}^{n} w_i} \quad (14)$$

where w_i is the proportion of true instances of class i over all true instances and n is the number of classes.

6) ANOMALY DETECTION EVALUATION METRIC

We assess whether the confidence of the classifier measured by the entropy of the output probability distribution of the classes is a good indicator for trajectory anomaly detection. Since our datasets do not contain any trajectory labeled as outliers, we had to synthetically generate spatial and temporal anomaly trajectories. Based on the STOD's entropy results in Table 6, we assumed that the trajectories with the smallest chance of having anomalies are the ones that STOD correctly classified with high confidence, i.e., entropy between 0 and 0.1. On the Recife test set, there are 3,530 trajectories in this entropy range and 2,805 on the Dublin test set. We consider them as non-anomalous trajectories (NAT) and created from them anomalous ones (AT). For evaluation, we execute STOD(geo,pac) on both datasets and calculate our entropy-based anomaly score (see Equation 10) of all trajectories in NAT and AT. Then, we measure the relative change of the distribution of the anomaly scores of NAT regarding AT, which we call relative entropy, at different percentiles. More formally, the relative entropy is:

$$\frac{f_{ent}(AT) - f_{ent}(NAT)}{f_{ent}(NAT)} \quad (15)$$

where f_{ent} is the entropy value at a specific percentile. We used the 25th, 50th (median) or 75th percentiles in our evaluation. A positive value of relative entropy means that the anomaly score of the anomalous trajectories is higher than in the non-anomalous ones at a certain percentile, and a negative value means otherwise. We were not able to compare our anomaly detection with other baselines due to the difficulty of finding available code solutions or because the strategies in literature are very different from ours.

V. RESULT AND DISCUSSION

In this section, we evaluate on real bus trajectory datasets our proposed approaches: the Point Activity Classifier, the Route ID Classifier and the anomaly trajectory score method.

TABLE 4. Results of PAC classification using the original features and PAC embeddings.

		Recife			Dublin		
Model	Features	WP	WR	WF1	WP	WR	WF1
gaussian	original	0.619	0.586	0.586	0.565	0.627	0.580
nb	PAC emb	0.922	0.918	0.919	0.911	0.839	0.847
knn	original	0.733	0.755	0.731	0.602	0.698	0.623
	PAC emb	0.925	0.927	0.925	0.886	0.890	0.886
random	original	0.727	0.761	0.721	0.607	0.705	0.619
forest	PAC emb	0.927	0.927	0.927	0.888	0.892	0.888
svm	original	0.727	0.744	0.706	0.507	0.478	0.556
	PAC emb	0.927	0.927	0.926	0.891	0.893	0.888
lgbm	original	0.757	0.783	0.757	0.573	0.702	0.579
	PAC emb.	0.925	0.926	0.925	0.889	0.892	0.889
pac	original	0.924	0.924	0.923	0.890	0.894	0.889

A. PAC EVALUATION

We evaluate the quality of the embeddings created by the PAC model by comparing different classification algorithms trained using the PAC's original features and the embeddings created by our PAC model. The results in Table 4 show that the PAC embeddings improved the performance of all evaluated models. For instance, the WF1 value of Gaussian Naive Bayes (gaussian nb) increased from 0.586, using the original features, to 0.919 using the PAC embeddings on the Recife dataset, and from 0.58 to 0.847 on the Dublin dataset. The results also show that the performance of the softmax classifier (pac in Table 4) used in the PAC model is similar to the other classifiers using the PAC embeddings. The superior results of the classifiers using the PAC embedding confirm that the PAC network is indeed able to learn relevant information from the original features by embedding them in a single vector, working as an end-to-end encoder for this task.

In order to get a visual interpretation about the discriminative power of the PAC embeddings we present in Figures 4 (a) and Figures 4 (b) the 2D projection of learned PAC embeddings using T-SNE [26] on the two datasets. They illustrate desirable properties of good representations according to Bengio *et al.* [6]: natural clustering, since the points with similar labels are clustered in the space, maintaining a *spatial coherence* between similar points (e.g., stay points are closer to each other than to in route points).

B. ROUTE ID CLASSIFIER EVALUATION

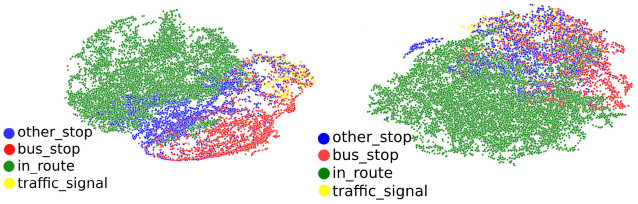
We assess in this section our proposed solution for the task of classifying bus trajectories in their assigned routes and compare it with a previous approach (RioBusData). The numbers in Table 5 show that STOD(geo,pac) and

TABLE 5. Average of results of our outlier model and RioBusData.

	Recife			Dublin		
Model	WP	WR	WF1	WP	WR	WF1
STOD(geo,pac)	**0.957**	**0.956**	**0.956**	0.964	0.964	0.964
STOD(geo,time)	0.953	0.952	0.952	**0.970**	**0.970**	**0.970**
RioBusData	0.930	0.927	0.927	0.952	0.949	0.949

TABLE 6. Entropy Distribution of STOD on the Test Set.

	Recife				Dublin			
Entropy	%Trajectories	WP	WR	WF1	%Trajectories	WP	WR	WF1
0.0 - 0.1	**95.2**	**0.97**	**0.97**	**0.97**	**96.4**	**0.98**	**0.98**	**0.98**
0.1 - 0.2	3.2	0.63	0.52	0.53	2.2	0.51	0.45	0.47
0.2 - 0.3	1	0.40	0.42	0.39	0.8	0.58	0.45	0.49
0.3 - 0.4	0.2	0.59	0.45	0.48	0.2	0.37	0.37	0.37
0.4 - 0.5	0.08	0.0	0.0	0.0	0.1	0.0	0.0	0.0
0.5 - 0.6	0.02	1.0	1.0	1.0	0.06	0.0	0.0	0.0



(a) Embeddings from Recife. (b) Embeddings from Dublin.

FIGURE 4. Visualization of PAC embeddings using T-SNE.

STOD(geo,time) obtain high values and outperform RioBusData in all evaluation metrics. Looking at the WF1 measure on the Recife dataset, for instance, STOD(geo,pac) obtained the best WF1 result (0.956), slightly higher than RioBusData (0.927). On the Dublin dataset, on the other hand, STOD(geo,time) outperformed the other approaches with WF1 equals to 0.97 followed by STOD(geo,pac) with WF1 equals to 0.964.

We applied Mann Whitney statistical test [27] to verify whether there is a statistical difference between the pair of models STOD(geo,pac), STOD(geo, time), and STOD(geo,pac), RioBusData. We set the significance level to $\alpha = 5\%$ and our null hypothesis h_0 considers each of the pairs models generates WF1 results which have the same median and h_1 considers that the median of the results of the first model is greater than the second one. We ran each model 30 times for this evaluation. The results confirm that there is a statistical difference between STOD(geo,pac) and RioBusData on Recife (p-value $= 1.50e^{-11}$) and Dublin (p-value $= 4.63e^{-9}$), confirming that our route id classifier has superior performance than RioBusData. We can also note that STOD(geo, time) achieved close performance to STOD(geo, pac) on Recife (WF1 $= 0.956$, WF1 $= 0.952$) and Dublin (WF1 $= 0.970$, WF1 $= 0.964$). The statistical test shows evidence to reject the null hypothesis on Recife (p-value $= 2.04e^{-5}$), but not on Dublin (p-value $= 0.99$). A possible reason for the better performance of STOD(geo,time) in comparison to STOD(geo,pac) on the Dublin dataset is that it contains trajectories from only 3 days. As a result, the features of the PAC network that capture seasonality (day of the week and hour of the day) did not have much impact.

Since our anomaly detection score is based on the entropy of the output class distribution of STOD, we verified the performance of STOD(geo,pac) on different ranges of entropy on the trajectories of the test set. The results in Table 6 show a clear degradation in the performance of STOD as the entropy of the classification probability output increases. The WF1 metric dropped from 0.97 to 0.53 from the entropy interval [0.0 - 0.1] to [0.1 - 0.2] on the Recife dataset and from 0.98 to 0.47 on the Dublin dataset. Furthermore, the percentage of trajectories classified in these entropy intervals decreased from 95.2% to 3.2% on the Recife dataset and from 96.4% to 2.2% on the Dublin dataset. These results confirm the high quality of our proposed classifier (STOD) since it classified with high confidence and correctly a very high percentage of the trajectories on both datasets. In addition, there is a clear correlation between the entropy value and the performance of the classifier: the higher the entropy (or the lower the confidence of the classifier), the lower its performance. This might also indicate that trajectories with high entropy values are anomalous, which is the main assumption behind our trajectory anomaly detector.

STOD as a Trajectory Encoder: We also evaluated the STOD(geo,pac) model as a trajectory encoder that receives as input a raw trajectory and uses the intermediate mappings and non-linearities performed in the network to produce a dense vector representation of the trajectory (STOD embedding). This vector is the input of the final layer of the STOD network (see Figure 1). In this experiment, in particular, this trajectory encoder transforms a trajectory with 100 points and 50 features per point into an embedding of 1000 dimensions on Recife dataset, and 2000 on Dublin dataset. To further reduce the dimensionality of the vectors and ease the training

TABLE 7. Results of STOD embeddings.

	Recife			Dublin		
Model	WP	WR	WF1	WP	WR	WF1
gaussian_nb	0.943	0.934	0.936	0.958	0.953	0.954
knn	**0.955**	**0.953**	**0.954**	**0.968**	**0.968**	**0.968**
random forest	0.948	0.947	0.946	0.963	0.965	0.964
svm	0.948	0.946	0.946	0.967	0.968	0.967
lgbm	0.941	0.939	0.939	0.963	0.964	0.963

process, we apply PCA (Principal Component Analysis) [28], decreasing the STOD vectors to 32 dimensions. Table 7 presents the results of classifiers for the route id classification task using this dimensionality reduction strategy. The KNN approach outperformed all the evaluated classifiers on both datasets: WF1=0.954 (Recife) and WF1=0.968 (Dublin). This result indicates that our trajectory encoder in fact can capture the complexities associated with the features of the trajectory allowing such a simple model as KNN, which relies heavily on the representation of the instances to perform the classification, to achieve superior performance than more sophisticated algorithms such as Random Forest, SVM and LightGBM. This result might indicate as well that our sentence encoder can also be used in the task of trajectory similarity [9], which we let as a future work.

C. OUTLIER DETECTION ASSESSMENT

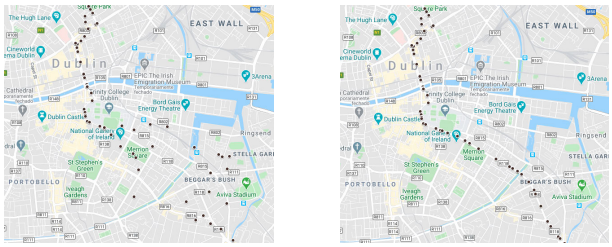
We also evaluate our approach for detecting spatio-temporal anomaly trajectory.

1) SPATIAL ANOMALY

We consider spatial anomaly trajectories the ones whose points are (totally or partially) spatially away from their respective pre-assigned bus route line. We synthetically generate them by randomly picking k consecutive points in the trajectory, and adding noise to their latitude and longitude. To calculate the noise, we use the *geo-py* [13] library, in which we pass the current point k_i, a *distance* in kilometers which represents the distance between k_i and its respective noisy one, and a bearing value, which is the angle between the trajectory from k_i and the location where the noisy point is placed. We vary the distances to simulate points getting away from the original trajectory and returning to it. For example, for $k = 5$ we define 5 distances [0.1, 0.2, 0.3, 0.2, 0.1], then the 5 noisy points are placed at those distances from the original points. We varied the number of points k in 10 (noise_10), 20 (noise_20) and 30 (noise_30). Figure 5 (a) shows an example of a synthetic trajectory and Figure 5 (b) the original one. The noisy points are at the bottom of Figure 5 (a) which shows points getting away, and returning to the original shape of trajectory.

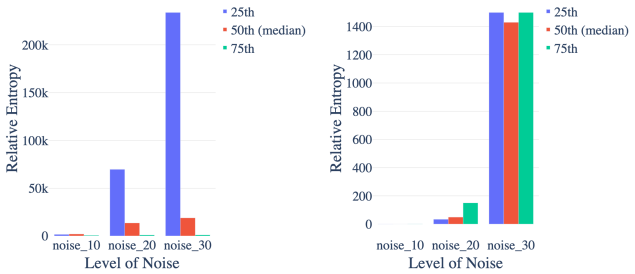
Figure 6 (a) and Figure 6 (b) show the relative entropy of the STOD classifier on the synthetic spatial anomaly on the Recife and Dublin datasets respectively. On both datasets, the

[13]https://pypi.org/project/geo-py/



(a) Anomalous trajectory. (b) Raw trajectory.

FIGURE 5. Example of spatial anomaly.



(a) Recife. (b) Dublin.

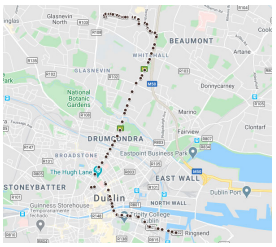
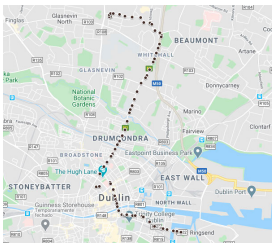
FIGURE 6. Results of spatial anomaly.

results show that: (1) the entropy of the anomaly trajectories are higher than the non-anomalous ones (positive relative entropy); and (2) the higher the noise level, the higher the relative entropy. For example, on the Recife dataset, the median relative entropy for noise_10 is 1,959, for noise_20 is 13,550, and noise_30 is 18,868. These numbers confirm that the entropy of the probability distribution of the predicted classes provided by STOD is an effective approach to score trajectories in order to identify spatial anomaly.

2) TEMPORAL ANOMALY

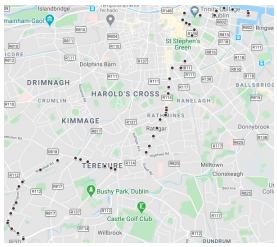
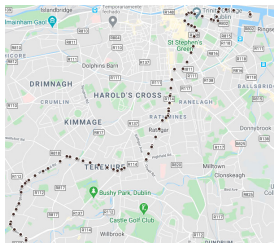
We also evaluate our approach for detecting temporal anomaly trajectories on the two types described in Section II: Temporal Anomaly Type I (more congestion than usual) and Temporal Anomaly Type II (less congestion than usual). We simulate the Temporal Anomaly Type I as follows. First, given a non-anomalous trajectory T and the its assigned route line l in the GTFS shape file, we randomly pick k consecutive points from l, find the closest point to each one of them in T using the Vincenty's distance, and then add these k consecutive points to T. Figure 7 (a) depicts a noisy trajectory where points were added at the beginning of the original trajectory (top of the figure), simulating a congestion, and Figure 7 (b) the original bus trajectory without noise.

Figure 8 (a) and Figure 8 (b) show the relative entropy for Temporal Anomaly Type I. On the Recife dataset, the results follow a similar trend of spatial anomaly: the entropy values are higher for the noisy trajectories than the regular ones, and there is a relation between the level of noise and the entropy value. For instance, the median of the relative entropy for

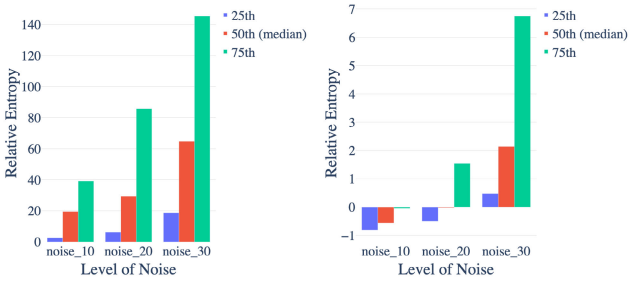
(a) Anomalous trajectory. (b) Raw trajectory.

FIGURE 7. Example of temporal anomaly type I.

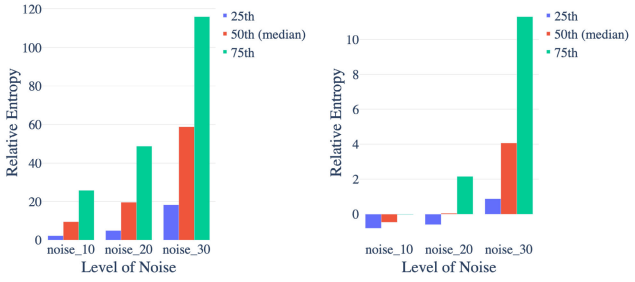
(a) Anomalous trajectory. (b) Raw trajectory.

FIGURE 9. Example of temporal anomaly type II.



(a) Recife (b) Dublin.

FIGURE 8. Results of temporal anomaly type I.



(a) Recife (b) Dublin.

FIGURE 10. Results of temporal anomaly type II.

noise_10 is 19.50, 20.41 for noise_20 and 64.75 for noise_30. On the Dublin dataset, also the increasing of noise led to higher values of relative entropy, but as opposed to the Recife dataset, the relative entropy was not positive for all scenarios. In the noisiest scenario, noise_30, though the relative entropy was positive for 25th, 50th and 75th percentiles. These results confirm that our anomaly score strategy can also be used to detect Temporal Anomaly Type I.

For generating the Temporal Anomaly Type II, we remove 10, 20, and 30 points of non-anomalous trajectories to simulate less traffic than usual. To avoid having unrealistic scenarios, for example, bus speeds above 200 km/h, we only remove a GPS point k_i if the speed between points k_{i-1} and k_{i+1} is less than the trajectory maximum speed and two standard deviations from the mean. Figure 9 (a) presents an example of Temporal Anomaly Type II, and the original trajectory in Figure 9 (b). As one can see there are sparse points mostly at the bottom of Figure 9 (a) simulating less congestion.

Figure 10 (a) and Figure 10 (b) present the relative entropy of Temporal Anomaly Type II for both cities. The results are similar to the other anomaly types: on the Recife dataset, the relative entropy is positive for the 25th, 50th and 75th percentiles; and on the Dublin dataset, the positive relative entropy occurs on the 50th and 75h percentiles. Looking at the median at noise_30, for instance, the relative entropy for Recife is almost 60 times higher than the non-anomalous trajectories and for Dublin, 4 times higher. The anomaly score calculated by the entropy of the probability class distribution of the trajectories provided by STOD is in fact effective to identify trajectories with Temporal Anomaly Type II.

VI. LITERATURE REVIEW

Lee *et al.* [29] propose a partition-and-detect framework for trajectory outlier detection. The partitioning step focuses on splitting trajectories into a set of line segments and feed them to the detector task. To find the anomalies, the authors consider the number of close trajectories based on the distances (i.e., perpendicular, parallel, and angle) from neighboring trajectories. Then, if a fraction of segments from trajectories are not close to a given segment, it is considered an outlier. Overall, trajectories composed from a set of anomalous segments are anomalies according to a threshold.

Kong *et al.* [4] introduce an unsupervised approach named LoTAD (Long-term Traffic Anomaly Detection) to detect anomalous trajectory segments and abnormal regions in the bus context. Toward this end, LoTAD first obtains temporal-spatial segments (TS-Segments) from two consecutive stop stations to extract two features: average speed and average stop time. Next, based on segments composed of the previous two features, it builds a matrix for each bus line and calculates an anomaly index by comparing the density of a point to all other points using a gaussian kernel function [30]. With the anomaly index results, LoTAD apply the K-means algorithm to find anomalous points. Also, the LoTAD maps the TS-segments in small regions (i.e., the approach splits a city into regions) and cumulatively calculates regions' anomaly index based on previous segments' anomaly index.

Bessa *et al.* [3] propose a supervised-based strategy to detect bus anomalous trajectories. For that, they introduce a multi-class Convolutional Neural Network that classifies trajectories according to their route IDs. A trajectory is

considered anomalous when it is misclassified or the classifier's output probability is below a certain threshold. Their classifier achieved a high accuracy for the route id classification but no results were presented to evaluate the outlier detection method. As we show in our experimental evaluation, STOD outperforms their approach for the bus route classification task on both evaluation datasets. Another approach that uses supervised learning is proposed by Raymond and Imamich [5]. They introduce a bus route classifier to detect trajectory outliers. The classification is performed in two steps: First, it transforms the input GPS sequence into a sequence of road IDs based using map-matching. Next, a bag-of-roads method, similar to bag-of-words, generates vectors of the buses and vectors of the predefined routes, which are used by a KNN model to perform bus route classification. Song *et al.* [31] introduce a binary classifier to detect spatial anomalies in taxi trajectories. For that, they first map trajectories into a regular grid map. Next, the mapped trajectories are feed into an RNN that provides input to an MLP with a sigmoid function on top of it to perform the binary classification.

VII. CONCLUSION

In this paper, we proposed a solution for detecting anomalous spatial-temporal bus trajectories. The solution relies on a deep learning multi-class classification model (STOD). The anomaly score is based on the confidence of the classifier in classifying a trajectory, and is measured by the normalized entropy of the output probability distribution of the classes. This classifier receives as inputs trajectories in which each point is a concatenation of the PAC embedding, which captures behavior and time information, and Geo embedding, which catches spatial relationships among points. The PAC embedding is created by a deep-learning network that predicts the type of activity of GPS points. The Geo embedding is generated by mapping the GPS points to grid cells and applying a word encoder algorithm in the trajectories, in which each point is represented by its grid cell id. The experimental results indicate that PAC produces high-quality embedding vectors and is effective for the task of stay point classification. Furthermore, STOD outperforms a baseline for bus route id classification and its generated trajectory encoder used in different classifiers has good performance for this task as well. Finally, our evaluation of the anomaly score showed that in general the score distribution of trajectories with spatial or temporal anomalies is higher in anomalous trajectories than non-anomalous ones. As future work, we plan to detect the exact points of the trajectories that have anomalies and evaluate our trajectory encoder strategy in other tasks such as trajectory similarity.

ACKNOWLEDGMENT

The authors would like to thank Grande Recife Consórcio de Transporte Metropolitano for providing one of the data sets used in this article.

REFERENCES

[1] E. M. Knox and R. T. Ng, "Algorithms for mining distancebased outliers in large datasets," in *Proc. Int. Conf. Very Large Data Bases*. Citeseer, Aug. 1998, pp. 392–403.
[2] S. Papadimitriou, H. Kitagawa, P. B. Gibbons, and C. Faloutsos, "LOCI: Fast outlier detection using the local correlation integral," in *Proc. 19th Int. Conf. Data Eng.*, 2003, pp. 315–326.
[3] A. Bessa, F. Silva, R. Nogueira, E. Bertini, and J. Freire, "Riobusdata: Outlier detection in bus routes of Rio de Janeiro," in *Proc. Symp. Vis. Data Sci. (VDS)*, 2015, pp. 1–6.
[4] X. Kong, X. Song, F. Xia, H. Guo, J. Wang, and A. Tolba, "LoTAD: Long-term traffic anomaly detection based on crowdsourced bus trajectory data," *World Wide Web*, vol. 21, no. 3, pp. 825–847, May 2018.
[5] R. Raymond and T. Imamichi, "Bus trajectory identification by map-matching," in *Proc. 23rd Int. Conf. Pattern Recognit. (ICPR)*, Dec. 2016, pp. 1618–1623.
[6] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.
[7] R. Mohanadas, "Discerning truck stop semantics through latent space clustering," Ph.D. dissertation, 2018. [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-240598
[8] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, Feb. 2003.
[9] Y. Zhang, A. Liu, G. Liu, Z. Li, and Q. Li, "Deep representation learning of activity trajectory similarity computation," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jul. 2019, pp. 312–319.
[10] L. Bontemps, J. McDermott, and N. A. Le-Khac, "Collective anomaly detection based on long short-term memory recurrent neural networks," in *Proc. Int. Conf. Future Data Secur. Eng.* Cham, Switzerland: Springer, 2016, pp. 141–152.
[11] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*. [Online]. Available: http://arxiv.org/abs/1301.3781
[12] D. Wang, J. Zhang, W. Cao, J. Li, and Y. Zheng, "When will you arrive? Estimating travel time based on deep neural networks," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 1–8.
[13] C. Guo and F. Berkhahn, "Entity embeddings of categorical variables," 2016, *arXiv:1604.06737*. [Online]. Available: http://arxiv.org/abs/1604.06737
[14] T. Vincenty, "Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations," *Surv. Rev.*, vol. 23, no. 176, pp. 88–93, Apr. 1975.
[15] S. Dabiri and K. Heaslip, "Inferring transportation modes from GPS trajectories using a convolutional neural network," *Transp. Res. C, Emerg. Technol.*, vol. 86, pp. 360–371, Jan. 2018.
[16] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015, *arXiv:1502.03167*. [Online]. Available: http://arxiv.org/abs/1502.03167
[17] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2980–2988.
[18] J. Bergstra, D. Yamins, and D. D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 115–123.
[19] T. Kam Ho, "Random decision forests," in *Proc. 3rd Int. Conf. Document Anal. Recognit.*, vol. 1, 1995, pp. 278–282.
[20] S. Russell and P. Norvig, "Artificial intelligence: A modern approach," Tech. Rep., 2002.
[21] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.
[22] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3146–3154.
[23] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. Hoboken, NJ, USA: Wiley, 2012.
[24] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 2623–2631.
[25] I. Brodsky. (2018). *H3: Uber's Hexagonal Hierarchical Spatial Index*. Uber Engineering. [Online]. Available: https://eng.uber.com/h3/[22 June 2019]

[26] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008. [Online]. Available: http://www.jmlr.org/papers/v9/vandermaaten08a.html
[27] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *Ann. Math. Statist.*, vol. 18, no. 1, pp. 50–60, Mar. 1947.
[28] I. T. Jolliffe and J. Cadima, "Principal component analysis: A review and recent developments," *Phil. Trans. Roy. Soc. A: Math., Phys. Eng. Sci.*, vol. 374, no. 2065, Apr. 2016, Art. no. 20150202.
[29] J.-G. Lee, J. Han, and X. Li, "Trajectory outlier detection: A partition-and-detect framework," in *Proc. IEEE 24th Int. Conf. Data Eng.*, Apr. 2008, pp. 140–149.
[30] G. R. Kumar, N. Mangathayaru, and G. Narsimha, "An approach for intrusion detection using novel Gaussian based kernel function," *J. UCS*, vol. 22, no. 4, pp. 589–604, 2016.
[31] L. Song, R. Wang, D. Xiao, X. Han, Y. Cai, and C. Shi, "Anomalous trajectory detection using recurrent neural network," in *Proc. Int. Conf. Adv. Data Mining Appl.* Cham, Switzerland: Springer, 2018, pp. 263–277.

MICHAEL CRUZ received the M.Sc. degree in computer science from Univerdidade Federal Sergipe. He is currently pursuing the Ph.D. degree with the Centro de Informática, Universidade Federal de Pernambuco. His research interests include machine learning, data mining, and data analytics.

LUCIANO BARBOSA is currently an Associate Professor with the Computer Science Department, Universidade Federal de Pernambuco. In addition to his experience in academia, he also held different positions in industry research labs as a Visiting Scholar Researcher at Google AI (formerly Google Research); a Research Scientist at IBM Research and AT&T Research Labs; and a Ph.D. Summer Intern at Yahoo! Research (Europe and USA). His research interests include web mining, text mining, natural language processing, information retrieval, and data analytics.