UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

MICHAEL OLIVEIRA DA CRUZ

Applying Language Modeling to Detect Anomalies in Bus Trajectories

Recife
2022

MICHAEL OLIVEIRA DA CRUZ

Applying Language Modeling to Detect Anomalies in Bus Trajectories

Tese apresentada ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, como requisito para obtenção do título de Doutor em Ciência da Computação.

Área de Concentração: Inteligência Computacional
Orientador: Luciano de Andrade Barbosa

Recife
2022

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

C957a Cruz, Michael Oliveira da
Applying language modeling to detect anomalies in bus trajectories / Michael Oliveira da Cruz. – 2022.
99 f.: il., fig., tab.

Orientador: Luciano de Andrade Barbosa.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2022.

Inclui referências.

1. Inteligência artificial. 2. Modelos de linguagem. I. Barbosa, Luciano de Andrade (orientador). II. Título.

006.31 CDD (23. ed.) UFPE - CCEN 2022-124

Michael Oliveira da Cruz

“Applying Language Modeling to Detect Anomalies in Bus Trajectories”

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação. Área de Concentração: Banco de Dados

Aprovado em: 01/07/2022.

_____

Orientador: Prof. Dr. Luciano de Andrade Barbosa

BANCA EXAMINADORA

_____

Prof. Dr. Kiev Santos da Gama
Centro de Informática/UFPE

_____

Prof. Dr. Paulo Salgado Gomes de Mattos Neto
Centro de Informática/UFPE

_____

Prof. Dr. Cláudio de Souza Baptista
Centro de Engenharia Elétrica e Informática, Sistemas e Computação /UFCG

_____

Prof. Dr. Clodoveu Augusto Davis Junior
Departamento de Ciência da Computação /UFMG

_____

Prof. Dr. José Antônio Fernandes de Macedo
Departamento de Computação/UFC

To my family for all support and love.

ACKNOWLEDGEMENTS

First of all, I would like to thank my mother, Avanete Oliveira, who always supports me in all moments and reminds me that education is the most valuable treasure. Every day I learn more with her strength, love, and faith. Thank you for everything.

I sincerely thank my advisor, Prof. Luciano de Andrade Barbosa, for providing guidance and valuable insights into this research and for having patience in motivating and helping me to see beyond. Thank you for the opportunity to work with you.

My most enormous thanks to my family, primarily brothers and sisters, for supporting and advising me to continue pursuing my goals.

Thank you to Karla Mirella for supporting me and putting up with my stresses and complaints and for valuable advice and encouragement. Thank you for your patience.

Thank the Lab colleagues for all the conversations and insights and for making those years easier and more enjoyable.

And to many others who have been a part of this journey.

ABSTRACT

Discovering anomalous bus trajectories in urban traffic can help transportation agencies to improve their services by providing a better plan to deal with unexpected events such as weather phenomena, detours, or accidents. To help identifying the potential anomaly cause, we can detect anomalous trajectories and also pinpoint where the anomaly is located. However, a big challenge to performing this task is the lack of labeled anomalous trajectory data. The lack of labeled data hinders model evaluation, and the construction of inductive learning approaches. Additionally, previous approaches heavily rely on preprocessing tasks to segment trajectories before detecting the anomaly. This strategy is not only costly but also may lose important information because segments are analyzed individually without considering the relationship between segments. Lastly, only a few strategies in the literature propose online solutions, which restricts their real-time contribution. On this basis, this thesis aims to propose an online approach based on inductive learning to detect anomalous bus trajectories and pinpoint the anomalous segments. To do that, we initially observed that bus trajectories are pre-defined and well-formed, and they include route labels. Based on that, we supposed that a supervised approach could learn to classify those bus trajectories according to the routes and indirectly detect which ones are anomalous. Thus, we propose a multi-class classifier called Spatial-Temporal Outlier Detection as our first solution to detect anomalous trajectories. We use the uncertainty of the classification by applying the entropy function over the class distribution. Although extensive experiments have shown promising results, our first solution cannot pinpoint where the anomalous segments occur. To overcome that restriction, our intuition is that trajectories can be represented as a sequence of tokens similar to word sentences allowing us to apply a language model. Consequently, we propose using a deep generative encoder-decoder Transformer to learn the relationship between sequential trajectory points based on the self-attention mechanism. Our final solution does not require any manual feature extraction and can be easily adapted to other types of trajectories (e.g., car, people, and vessels). We have performed an extensive experimental evaluation that shows: (1) our approach is effective for both trajectory and sub-trajectory anomaly detection; and (2) it outperforms the baselines in some scenarios and statistically achieves comparable results in the others.

Keywords: anomaly score; anomalous segments; representation learning; language model; transformer; gps trajectory.

RESUMO

A descoberta de trajetórias anômalas de ônibus no tráfego urbano pode ajudar as agências de transporte a melhorar seus serviços, fornecendo um melhor planejamento para lidar com eventos inesperados, como fenômenos climáticos, desvios ou acidentes. Para ajudar a identificar a causa potencial da anomalia, podemos detectar trajetórias anômalas e também identificar onde a anomalia está localizada. No entanto, um grande desafio para realizar esta tarefa é a falta de *datasets* de trajetórias publicamente disponíveis. A falta desses dados rotulados para trajetórias anômalas também dificulta a avaliação de modelos e a construção de abordagens de aprendizagem indutivas. Além disso, as abordagens anteriores dependem fortemente de tarefas de pré-processamento para segmentar trajetórias antes de detectar a anomalia. Essa estratégia não é apenas cara, mas também pode perder informações importantes porque os segmentos são analisados individualmente sem considerar o relacionamento entre os segmentos. Por fim, poucas estratégias na literatura propõem soluções online, o que restringe sua contribuição em tempo real. Com base nisso, esta tese tem como objetivo propor uma abordagem online baseada em aprendizado indutivo para detectar trajetórias anômalas de ônibus e identificar os segmentos anômalos. Para isso, inicialmente observamos que as trajetórias de ônibus são pré-definidas e bem formadas, e incluem a informação de rotas. Com base nisso, supomos que uma abordagem supervisionada poderia aprender a classificar essas trajetórias de ônibus de acordo com as rotas e detectar indiretamente quais delas são anômalas. Assim, propomos um classificador multiclasse chamado *Spatial-Temporal Outlier Detection* como nossa primeira solução para detectar trajetórias anômalas. Usamos a incerteza da classificação aplicando a função de entropia sobre a distribuição de classes para gerar um *score* de anomalia. Embora experimentos extensivos tenham mostrado resultados promissores, nossa primeira solução não pode identificar onde ocorrem os segmentos anômalos. Para superar essa restrição, nossa intuição é que as trajetórias podem ser representadas como uma sequência de tokens semelhantes a frases de palavras que nos permitem aplicar um modelo de linguagem. Consequentemente, propomos usar uma abordagem baseada na arquitetura generativa de redes neuronais profundas chamada *encoder-decoder Transformer* para aprender a relação entre pontos de trajetória sequenciais com base no mecanismo de autoatenção. Nossa solução final não requer nenhuma extração manual de recursos e pode ser facilmente adaptada a outros tipos de trajetórias (por exemplo, carro, pessoas e embarcações). Realizamos uma extensa avaliação experimental que mostra: (1) nossa abordagem é eficaz para detecção de anomalias de trajetória e subtrajetória; e (2) supera os métodos usados para comparação em alguns cenários e alcança estatisticamente resultados comparáveis em outros cenários.

Palavras-chaves: score de anomalia; segmentos anômalos; representação de aprendizado; modelos de linguagem; *transformer*; trajetórias de gps.

LIST OF FIGURES

Figure 1 – Trajectory data mining context 17
Figure 2 – Segmentation approaches 26
Figure 3 – Outlier detection approaches 27
Figure 4 – Unrolled Recurrent Neural Network 28
Figure 5 – Encoder-Decoder architecture 29
Figure 6 – Self-Attention matrix correlation 30
Figure 7 – Overview of our anomaly score approach 45
Figure 8 – PAC model 46
Figure 9 – Geo embedding pipeline 49
Figure 10 – Visualization of PAC embeddings using T-SNE 55
Figure 11 – Example of spatial anomaly 59
Figure 12 – Results of spatial anomaly 59
Figure 13 – Example of temporal anomaly type I 60
Figure 14 – Results of temporal anomaly type I 61
Figure 15 – Example of temporal anomaly type II 61
Figure 16 – Results of temporal anomaly type II 62
Figure 17 – The trajectory anomaly detection solution proposed in this work 65
Figure 18 – Hamming distance 69
Figure 19 – Example of synthetic anomaly 70
Figure 20 – PR-AUC of the approaches for route 1 on the Dublin dataset and route 54 on the Recife dataset 74
Figure 21 – Detection on real-world trajectories 76
Figure 22 – MobApp Architecture 77
Figure 23 – WebApp Exploratory Data Analysis 78
Figure 24 – WebApp Anomaly Scores 79
Figure 25 – Precision-Recall curve of different anomaly detection strategies in a given route 80
Figure 26 – Detection on real-world trajectories 81
Figure 27 – MobApp Architeture Deployment Diagram 81
Figure 28 – Metadata end-point 82
Figure 29 – Trajectory end-point 82
Figure 30 – AE examples 88
Figure 31 – Latent space for 68 routes of bus trajectories for Recife Dataset 89
Figure 32 – VAE examples 89

Figure 33 – Anomalous trajectory prediction using DeepAnt approach. The orange trajectory is the prediction and the blue trajectory is the input anomalous trajectory 90

LIST OF TABLES

Table 1 – Stay point detection solutions 33
Table 2 – Trajectory Outlier detection solutions 42
Table 3 – Values of hyperparameters of PAC approach 51
Table 4 – Values of hyperparameters of STOD 53
Table 5 – Values of hyperparameters of RioBusData 53
Table 6 – Results of PAC classification using the original features and PAC embeddings 54
Table 7 – Results of PAC and CRF models for stay point classification 55
Table 8 – Average of results of our outlier model and RioBusData 56
Table 9 – Entropy Distribution of STOD on the Test Set 56
Table 10 – Results of STOD embeddings 58
Table 11 – Number of unique points before and after the Grid Mapping 70
Table 12 – Values of hyper-parameters of Transformer 71
Table 13 – Results of anomaly trajectory detection on the Dublin dataset 72
Table 14 – Results of anomaly trajectory detection on the Recife dataset 73
Table 15 – Results for the region anomaly detection models 74
Table 16 – Hypothesis test for F1 on the Dublin dataset 75
Table 17 – Hypothesis test for F1 on the Recife dataset 75

LIST OF ABBREVIATIONS AND ACRONYMS

AE	Autoencoder
ANNS	Artificial Neural Network
BCE	Binary Cross-Entropy
BPTT	Backpropagation Through Time
CNN	Convolutional Neural Network
CRF	Conditional Random Field
D-T	Displacement Time
D-S	Displacement Spatial
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DBTOD	Density-Based Outlier Trajectory Detection
ED	Edit Distance
GM-VSAE	Gaussian Mixture Variational Sequence AutoEncoder
GPS	Global Positioning System
GRU	Gated Recurrent Unit
GTFS	General Transit Feed Specification
HTTPS	Hypertext Transfer Protocol Secure
iBAT	Isolation-Based Anomalous Trajectory
iBOAT	Isolation-based Online Anomalous Trajectory Detection
IBTOD	Isolation-Based Method To Detect Outliers
iFOREST	Isolation Forest
JSON	JavaScript Object Notation
KNN	K-Nearest Neighbors
LM	Language Model
LOTAD	Long-Term Traffic Anomaly Detection
LRP	Layer-Wise Relevance Propagation
LRT	Likelihood Ratio Test
LSTM	Long Short-Term Memory
MEX	Minimal Examination
MSE	Mean Squared Error
MSN	Statical Method for Stop, Move and Noise

NLP	Natural Language Processing
OC-NN	One Class Neural Networks
OC-SVM	One Class SVM
OPTICS	Ordering Points To Identify the Clustering Structure
ORM	Object Relational Mapper
PAC	Point Activity Classification
POIs	Point of Interest
RNN	Recurrent Neural Network
S-D	Source and Destination
SMoT	Stops and Moves of Trajectories
STOD	Spatial-Temporal Outlier Detection
STR	Spatial-Temporal Relations
TF-IDF	Term Frequency-Inverse Document Frequency
TPRO	Time-dependent Popular Routes Based Trajectory Outlier Detection
TPRRO	Time-dependent Popular Routes Based Real-Time Trajectory Outlier Detection
TRDM	Trend-Residual Dual Modeling
TTG	Time-Dependent Transfer Graph
TTI	Time-Dependent Transfer Index
UT	Universal Transformer
UT-ATD	Universal Transformer for Anomalous Trajectory Detection
VAE	Variational Autoencoder

CONTENTS

1 INTRODUCTION 15
1.1 CONTEXTUALIZATION 15
1.2 MOTIVATION 17
1.3 OBJECTIVE 20
1.4 LIST OF PUBLICATIONS 21
1.5 STRUCTURE OVERVIEW OF THE PROPOSAL 21

2 BACKGROUND 23
2.1 TRAJECTORY DEFINITION 23
2.2 ANOMALY DETECTION 24
2.2.1 Anomaly detection on trajectory 25
2.3 DEEP LEARNING BACKGROUND 28
2.4 CONCLUSION 31

3 LITERATURE REVIEW 32
3.1 STAY-POINT DETECTION 32
3.2 ANOMALOUS BUS TRAJECTORIES 33
3.2.1 Rule-based Detection 34
3.2.2 Unsupervised Detection 37
3.2.3 Supervised Detection 39
3.2.4 Comparative Analysis 40
3.3 CONCLUSION 43

4 LEARNING GPS POINT REPRESENTATION TO DETECT ANOMALOUS BUS TRAJECTORIES 44
4.1 PROBLEM DEFINITION 44
4.2 METHODOLOGY 44
4.2.1 Point Activity Classification 45
4.2.2 Anomaly Trajectory Detection 48
4.2.3 Training 49
4.3 DATA DESCRIPTION AND SETUP 50
4.3.1 Experimental Setup 50
4.3.2 Evaluation Metrics 53
4.4 RESULT AND DISCUSSION 54
4.4.1 PAC Evaluation 54
4.4.2 Route ID Classifier Evaluation 55

4.4.3 Outlier Detection Assessment . . . 57
4.5 CONCLUSION . . . 62

5 APPLYING A TRANSFORMER LANGUAGE MODEL FOR ANOMALY DETECTION IN BUS TRAJECTORIES . . . 64
5.1 PROBLEM FORMULATION . . . 64
5.2 METHOD . . . 64
5.2.1 Grid Mapping . . . 65
5.2.2 Transformer Encoder . . . 66
5.2.3 Transformer Decoder . . . 67
5.2.4 Training . . . 68
5.2.5 Anomaly Detector . . . 68
5.3 DATA DESCRIPTION AND SETUP . . . 69
5.3.1 Experimental Setup . . . 69
5.4 RESULTS AND DISCUSSION . . . 72
5.4.1 Trajectory Anomaly Detection . . . 73
5.4.2 Region Anomaly Detection . . . 74
5.5 CONCLUSION . . . 76

6 MOBAPP: A DATA VISUALIZATION TOOL FOR TRAJECTORY ANALYSIS . . . 77
6.1 MOBAPP ARCHITECTURE . . . 78
6.1.1 WebApp . . . 78
6.1.2 REST API . . . 80
6.2 USE CASES . . . 83
6.3 CONCLUSIONS . . . 84

7 CONCLUSIONS AND FUTURE WORK . . . 85
7.1 THESIS CONTRIBUTIONS . . . 86
7.2 FAILED ATTEMPTS . . . 87
7.3 FUTURE WORK . . . 90

REFERENCES . . . 91

1 INTRODUCTION

1.1 CONTEXTUALIZATION

Anomaly detection is a multidisciplinary field. According to Chandola, Banerjee and Kumar (2009), the study of anomaly techniques starts in the Statistics community in the mid-19th century and extends to other research areas such as, cyber-security (YOUSEFI-AZAR et al., 2017), medical domain (IAKOVIDIS et al., 2018), military surveillance (BROTHERTON; JOHNSON, 2001), and intelligent transportation systems (LIU; PI; JIANG, 2013). Furthermore, the field presents solutions using largely consolidated techniques such as classification (BESSA et al., 2015), clustering (YING; XU; YIN, 2009), and semi-supervised (WU; PRASAD, 2017).

The literature introduces anomaly in different ways. For instance, (CHALAPATHY; CHAWLA, 2019) defines it as instances that stand out as dissimilar to all others. Another definition considers it as a pattern in data that is different from normal behavior (ZHENG, 2015). In the context of this work, we consider anomalous points the ones located further away from most of the data according to aspects such as spatial, temporal, spatio-temporal, and directional aspects. For example, spatial anomalies are related to points that spatially diverge from regular ones, such as bus deviation from regular routes [1] (CRUZ; BARBOSA, 2020). Temporal anomaly is related to the unusual changes in temporal continuity of the data, such as bus delays and flash crashes in financial markets (GUPTA et al., 2013).

Anomalies can reveal interesting insights since they can be induced for various reasons. For example, anomalous driving behavior from taxi GPS traces can be intentional to overcharge tourists by taking a longer path (LV et al., 2017; ZHANG et al., 2011; CHEN et al., 2013). Another example is credit/debit card fraud detection, which can reveal whether an incoming transaction fits well with the user's previous profile or behavior(KALID et al., 2020). Similarly, an outlier detection method can expose a hacking attack by discovering an unusual traffic pattern in a network(AHMED; MAHMOOD; HU, 2016).

Many aspects influence the behavior of public transportation vehicles in a city. In regular conditions, factors such as the day of the week, the hour of the day, and the current location of the buses in their pre-assigned route are good indicators of their behavior (KORMÁKSSON et al., 2014). On the other hand, due to unexpected situations (e.g., weather conditions, detours, accidents, celebrations, disasters, or other events), buses can present unusual spatial or temporal behavior. Finding such anomalies [2] may help transportation agencies improving its services, for example, by releasing more buses according to demand,

[1] A route can be defined as a set of road segments (LIU et al., 2017), whereby journeys in this route generate trajectories on different period of time

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

redefining routes due to accident or notifying bus drivers about detours.

Detecting bus anomalous trajectories is our main focus on this thesis. However, anomaly detection is not an isolated task because it depends on data quality, and Global Positioning System (GPS) trajectory datasets are traditionally noisy and voluminous which make this a challenging task. Additionally, algorithms that deal with trajectories have their performance hurt when the input trajectories are sparse (LI et al., 2013). Therefore, the literature suggests preprocessing data to improve data quality, as shown in Figure 1 that summarizes the trajectory data mining field and its tasks. The first activities are related to the preprocessing tasks such as noise filtering, segmentation, map-matching, compression, and stay-point detection (ZHENG, 2015). Among those tasks, we focus on segmentation (orage rectangle) and stay-point detection (blue rectangle) for some reasons. First, the segmentation task splits trajectories into segments to reduce computational complexity and enables mining sub-trajectories patterns. Second, the stay-point detection discovers points where objects or individuals stayed for a while in a trajectory. Those points are important because they can describe Point of Interest (POIs) (LIU et al., 2013), and they together can show, for example, individuals' daily moving activities, such as going from a workplace to a restaurant or from a hotel to an airport. In brief, the stay-point detection not only allows adding semantic meaning to raw trajectories but also reduces redundant points because one stay-point replaces a set of neighboring points in a raw trajectory.

After the preprocessing step, as shown in Figure 1, we focus our attention on the Trajectory Outlier Detection (dark blue rectangle), which is a core activity/task in the data mining field. There are some specific challenges in performing this task. The lack of labeled datasets limits the development of methods for learning such assumptions and thresholds by training with labels (i.e., supervised learning). In addition, the validation of outlier detection methods is complex due to the lack of labels. Fourth, little attention has been paid to finding anomalous regions in an online way (ZHANG et al., 2021).

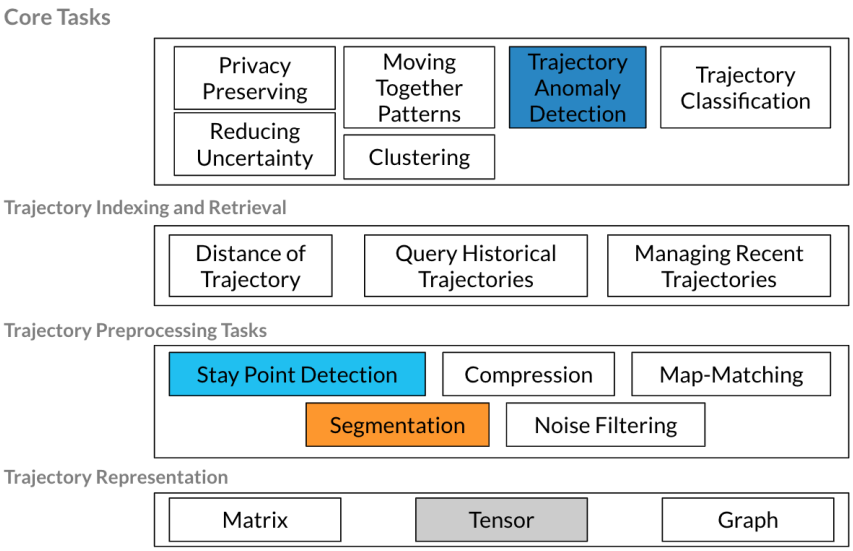
Previous works in this area consider methods that make assumptions or restrictions about the data. For example, statistical-based and distance-based techniques rely on thresholds that are hard to estimate because require profound knowledge about the domain (GE et al., 2010; PANG et al., 2011). We intend to explore those gaps in the trajectory data mining context, particularly in GPS trajectories.

Overall, Figure 1 presents other auxiliary and core tasks that are not our focus but are relevant to the field. For example, along with the previous preprocessing and trajectory outlier detection tasks, other relevant research topics include trajectory indexing and retrieval, trajectory representations (matrix, tensors, and graph), trajectory pattern mining, etc. In brief, as trajectory data mining is a large domain, our research focuses only on the preprocessing steps and trajectory outlier detection.

This thesis hence provides a research contribution in both scenarios: preprocessing and trajectory anomaly detection. For that, we propose a novel point activity representa-

tion for bus trajectories, a supervised and a semi-supervised approach for bus trajectory anomaly detection.

Figure 1 – Trajectory data mining context



Source: Adapted from ZHENG (2015)

1.2 MOTIVATION

With the advances in GPS devices and in smart city infrastructure, more and more mobility data has been generated (JI et al., 2020; ZHENG, 2015). Vehicles, smartwatches, traffic signals, and mobile phones, only to name a few, generate a massive volume of trajectory data. These data offer an unprecedented opportunity to discover rich information about traffic behavior, people mobility, and weather impact(ZHAO et al., 2020; BOURITSAS et al., 2019). Particularly in the traffic context, which is easily influenced by external factors (e.g., accidents, detours, events, and weather conditions), trajectory anomaly detection is crucial to understanding the traffic behavior and support better decision-making by transit authorities.

The goal of trajectory outlier detection is to determine which instances stand out as dissimilar to all others (CHALAPATHY; CHAWLA, 2019). Effective methods find anomalous instances in a data-driven fashion and open the opportunity to understanding the cause of abnormal trajectories. As an illustration, (PAN et al., 2013) use an approach to detect anomalous segments along with mining relevant terms on social media to describe the traffic anomaly. In the same direction, (WANG et al., 2018) propose a hierarchical cluster method to discover abnormal trajectories and analyze the cause of anomalies in the taxi context. In summary, detecting outlier trajectories along with the anomalous segments (region) can provide useful information for transportation agencies to improve their services and other real applications as follows:

Example 1: A car ridesharing company might store thousands of daily trajectories to suggest the best routes to its riders and drivers. Therefore, finding anomalous trajectories is an essential feature since it can be helpful to recommend alternative paths, free of anomalies such as traffic congestions, and alert riders or penalize drivers when the recommended paths by its app are not followed.
Example 2: Public transportation agencies face traffic problems such as congestion, accident, and poor weather conditions in large cities worldwide. At the same time, these agencies need to serve the population by making real-time decisions to deal with these issues, such as proposing alternative routes or releasing more buses. Given that, an online trajectory anomaly detection approach can identify problematic bus trips to allow transit authorities to intervene as quickly as possible.

In this work, we focus on detecting anomalies in bus trajectories. Buses follow a predefined route, and changes in traffic can affect it. For example, a fallen tree on the road can cause traffic congestion, and buses can rarely detour their paths. Furthermore, rainy weather conditions slow traffic flow. Conversely, taxis can drive away from traffic problems by getting alternative routes. In summary, a chain of events can modify traffic behavior, and bus routes do not usually change according to the driver's preferences (KONG et al., 2018).

In the literature, many solutions to detect anomalous trajectories have been proposed in unsupervised or supervised ways (KONG et al., 2018; WANG et al., 2018; CHEN et al., 2016). Unsupervised strategies are largely adopted, since they do not rely on data labeling to learn how to group similar points and separate the dissimilar ones. On the other hand, there are fewer supervised approaches to solve this problem because of the lack of available labeled data. Additionally, even with labeled data, supervised anomaly detectors have sub-optimal results because of the class imbalance problem (CHALAPATHY; CHAWLA, 2019). Even though it is a challenge to use a supervised approach, it can simplify the process of feature extraction and reduce some data assumptions. For instance, Bessa et al. (2015) and Raymond and Imamichi (2016) propose supervised approaches without using feature extraction or data distribution restrictions. In contrast, most methods that do not use a learning strategy depend on some constraints (e.g., distance threshold, probability distribution of data, etc.) (LEE; HAN; LI, 2008; HU et al., 2018; PANG et al., 2011).

As aforementioned, bus trajectories are predefined routes, and detecting anomalous trajectories can help to understand the outlier's causes. In the literature (CHEN et al., 2016; BESSA et al., 2015), most of the approaches to detect anomalies report their result using a score or classes (e.g., normal or anomalous). Typically, methods output a score when datasets have no labels, whereas binary approaches rely on labeled data. To the best of our knowledge, since labeled datasets are rare, supervised methods also use a score to report their results in the trajectory outlier detection task. For example, Bessa et al. (2015) and Raymond and Imamichi (2016) consider a hard score that is easy to

understand because they define a fixed threshold. However, it can hide result details since outliers are dynamic and evolve along time (GE et al., 2010). On the other hand, rule-based methods and unsupervised approaches require much effort in preprocessing tasks and rely on domain experts to make parameter assumptions and thresholds. Therefore, given a GPS bus trajectory dataset and the route labels, our first research question **RQ1** is: *how can we propose a supervised bus trajectory outlier detection method using classification confidence as an anomaly score?*

Answering the above question depends on other tasks. As we have mentioned, preprocessing trajectories is necessary to get useful information. Redundant points are a common issue on trajectory data since they represent similar locations, but with slightly different latitudes and longitudes. Also, sparseness or low-sample-rate is a data issue since it hinders analyzing trajectories. Therefore, preprocessing tasks are crucial in most works in the trajectory data mining field. For example, Zhang et al. (2011) convert raw trajectories to regions based on grid cells to decrease the complexity of dealing with redundant points. Likewise, Ouyang et al. (2018) also use a grid, but they replace latitude and longitude by cell id.

These aforementioned solutions regard trajectory points equally (i.e., points do not have a semantic meaning). In contrast, Zheng (2015) claims that trajectory's points are not equally important because some of them denote different locations (i.e., stay-points). Similarly, some works (YUAN et al., 2012; ZHANG et al., 2015) take advantage of representing trajectories using stay-points to decrease the redundancy of data as well as to facilitate the modeling problem by considering the semantics of those points instead of all points. In fact, representing trajectories using meaningful places is a way to characterize them. For example, buses as a public transportation service follow a predefined route with well-established stay-points (e.g., bus stops and traffic lights), then differentiating bus trajectories by stay-point representation seems simpler than dealing with raw data (i.e., latitude and longitude). Based on these observations, we believe that stay-points can facilitate the task of outlier detection, and we formulate our second research question **RQ2** as: *are there any advantages to representing trajectories by stay points for the outlier detection task?*

Identifying anomalous trajectories is useful for traffic understanding; detecting abnormal segments can help even more since they can limit a local where the outlier occurs. Lee, Han and Li (2008) tackle the anomaly segment problem by implementing a method to identify the anomalous segment partitioning trajectories before the detection while (LIU et al., 2011) builds a tree of anomaly links to discover the anomalous cause. From this, we formulate another research question **RQ3**: *how can we detect an anomalous trajectory segment without using anomalous labels?* This question is complementary to RQ1 since locating anomaly segments can be seen as a step further from trajectory outlier detection.

Although there are approaches to detect anomalous trajectories, practical solutions

also request online detection, mainly in the traffic context that demands real-time decisions. However, online approaches suffer two drawbacks. For example, approaches that only detect whether the whole trajectory is anomalous but do not pinpoint the anomalous segment or the ill-formed points(LIU et al., 2020). Trajectories can have thousands of points, and highlighting the erroneous points can improve their analysis. Another drawback is the necessity of searching mechanisms for recovering similar historical trajectories based on Source and Destination (S-D) pairs before detecting anomaly trajectories(CHEN et al., 2013). Those search mechanisms can hurt online detection performance in a massive volume of historical data. Conversely, our solution considers detecting whether the whole trajectory is anomalous and pinpointing the anomalous trajectory segment, both online and offline, without grouping trajectories using a search mechanism.

In the next section of this chapter we will present the objectives of this thesis.

1.3 OBJECTIVE

This thesis's main objective is to provide an end-to-end approach to: (i) detect anomalous trajectories in bus trajectory and (ii) pinpoint the abnormal points in these trajectories. In order to do that, we first investigate how to take advantage of supervised approaches to detect anomalous trajectories. For this purpose, we present Spatial-Temporal Outlier Detection (STOD), a method to detect bus trajectories outliers using flexible score values. STOD explores the preprocessing tasks to represent trajectories by meaningful points instead of raw data and uses multiclass classification to learn trajectories patterns. Furthermore, we propose detecting anomalies based on a flexible score measuring the classifier confidence. That is, STOD calculates the anomaly confidence score based on the entropy of the probability output of the classifier. The confidence score can be used to rank trajectories with respect their anomaly degree.

Secondly, we propose an end-to-end language model, commonly used in Natural Language Processing, for pinpointing the abnormal points in bus trajectories. The approach learns the pattern of well-formed trajectories and then identifies erroneous (ill-formed) trajectories, along with the trajectory points where the errors occur. In addition, our solution can be performed offline or online, i.e., as the buses move along their route, and it can also be adapted to other types of trajectories (e.g., car, people, and vessels), since our approach does not use any specific aspect from the bus domain (e.g., bus stops), and the input trajectory is the label.

Thus, in order to achieve the main objective of this thesis and answer our research questions, we had to investigate the following specific objectives:

1. Propose a Point Activity Classification (PAC) method to represent trajectory points into three types of stay point and moving point;

2. Define a Geo Embedding approach to capture spatial relationship between neighboring points of trajectories;

3. Propose a Spatial-Temporal Outlier Detection method composed by a multi-class classification method that learns to classify trajectories into their route lines and an anomaly detection module that outputs a soft anomaly score

1.4 LIST OF PUBLICATIONS

This section lists all the references published and submitted papers produced in the curse of the Ph.D. program. The list of papers is organized according to the status: published and submitted.

1. **Published papers**:

   - **Learning GPS Point Representations to Detect Anomalous Bus Trajectories.** Michael Cruz, Luciano Barbosa. IEEE Access, 2020.
   - **Análise do Impacto de Chuvas na Velocidade Média do Transporte Público Coletivo de Ônibus em Recife**. Alexandre Viana, Michael Cruz, Luciano Barbosa, Kiev Gama. In: WORKSHOP BRASILEIRO DE CIDADES INTELIGENTES (WBCI), 2018, Natal. I Workshop Brasileiro de Cidades Inteligentes, 2018

2. **Submitted papers**:

   - **Applying A Transformer Language Model for Anomaly Detection in Bus Trajectories**. Michael Cruz, Luciano Barbosa. Submitted to IEEE International Conference on Data Engineering 2023.
   - **MobApp: A Data Visualization Tool for Trajectory Analysis**. Michael Cruz, Fernando Neto, Luciano Barbosa. Submitted to 30th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems 2022.

1.5 STRUCTURE OVERVIEW OF THE PROPOSAL

The rest of this document is organized as follows:

1. **Chapter 2** will present the basic concepts of our proposal. The concepts will cover definitions of trajectory, point, spatial and temporal anomalies. Also, this chapter presents concepts of traffic anomaly, pre-processing techniques and outlier detection approaches.

2. **Chapter 3** presents an overview of related work. Here we present the works that lead this thesis on trajectory outlier detection and detecting anomalous trajectory segments. We also discuss methods to detect stay-points and summarize the works using comparative tables.

3. **Chapter 4** introduces the approach of learning GPS point representation to detect anomalous bus trajectories. Here, we propose a PAC to generate stay-point embeddings to feed a multi-class classifier called STOD. This chapter also proposes to detect anomalous bus trajectories indirectly by using the entropy function to measure the uncertainty of the classifier.

4. **Chapter 5** introduces the second research contribution of this thesis, a transformer language model for anomaly detection in bus trajectory. This model pinpoints the abnormal points and allows to perform this task offline or online. Lastly, this chapter discusses and presents an extensive experimental evaluation.

5. **Chapter 6** introduces the MobApp application as a data visualization tool for trajectory analysis. The chapter describes in detail the application architecture and presents a use case to illustrate how the MobApp can be used for each feature that the system provides.

6. **Chapter 7** closes this thesis by providing a summary of the contributions and presenting some failed attempts that were part of our research. Lastly, we suggest possible future research directions exploiting our results.

2 BACKGROUND

This chapter presents background concepts related to this thesis. Section 2.1 defines the concept of GPS trajectory. Section 2.2 discusses the key roles of detecting anomalous trajectories, while Section 2.3 presents deep learning architectures that are the basis of our solutions to answer our research questions. Finally, Section 2.4 concludes this chapter.

2.1 TRAJECTORY DEFINITION

Trajectories are traces that moving objects produce in a sequentially and chronological way (ZHENG, 2015). They allow researchers to understand more about traffic behavior due to the unprecedented advances in location-aware devices such as GPS, Wi-Fi, etc. Through those devices, more data is available allowing more knowledge about traffic problems. For example, detection of fraud in taxi drive behavior, the trajectory anomaly detection of animals, buses, and typhoons. Trajectories are continuous traces, but because of limitations on tracking devices, literature uses a sequence of points in discrete-time instances (MENG et al., 2019).

Literature has defined trajectories in a variety of ways. Here, we define a trajectory T_l as a sequence of consecutive GPS points collected from a trip, denoted as $T_l = \{p_1, p_2, ..., p_n\}$, where each l indicate the line id [1] i.e., each trajectory has a pre-defined itinerary (route). Each point $p_i = \{lat_i, lng_i, tsp_i\}$ is composed of latitude (lat_i), longitude (lng_i) and timestamp (tsp_i) ordered by the points' timestamp, i.e., $tsp_i < tsp_{i+1}$. Overall, point trajectories can have more or less attributes than latitude, longitude and timestamp. Conveniently, we assume those three features exist.

Since the scope of this thesis is to detect bus anomalous trajectories, and there are several types of anomalies in the literature, we only focus on the spatial and temporal anomalies in this research. Therefore, the first definition is that a spatially anomalous trajectory T_l' contains points that diverge from regular trajectories of the assigned bus line l of T_l'. On the other hand, we observe that a temporal anomaly trajectory T_l'' has a temporal behavior that deviates from regular trajectories of the line l (route id) in two ways. First, trajectories with the Temporal Anomaly Type I have buses running slower than the regular behavior of trajectories of l (more congestion than usual). Second, trajectories with the Temporal Anomaly Type II have vehicles moving faster than the regular behavior of trajectories of l (less congestion than usual).

[1] we use the terms route id and line id interchangeably

2.2 ANOMALY DETECTION

An anomaly is an instance that has an unexpected behavior compared to historical data. However, the notion of anomaly is not equal among the research areas because the perception about it depends on the data domain. For example, a few fluctuations on wireless devices may not be an anomaly, but the same level of deviations on ECG exams can be considered to be anomalous. Therefore, most of the works in the anomaly detection field are not orthogonal. Nonetheless, anomaly detection solutions have much in common because they require some decisions about the nature of data, types of anomalies, approaches to follow (e.g., machine learning, distance-based, etc), and how to represent an anomaly detection.

The nature of data can determine the applicability of anomaly detection because there are methods that do not work well with some types of attributes or with multivariate data. For example, some approaches (LEE; HAN; LI, 2008; LV et al., 2017) use Euclidean distance, which works better with numerical data instead of categorical. Also, the dimensionality is another concern since the performance of traditional machine learning methods deteriorates when they use high-dimensional data. Last, the relationship among data instances is another aspect to take into account before choosing which anomaly detection approach to use. Therefore, Chalapathy and Chawla (2019) classifies instances data as sequential and non-sequential. For example, text, time-series, spatio-temporal, music, protein sequence are sequential data because there are dependencies between previous and current instances. On the other hand, images are one example of non-sequential data. Overall, the data's nature and context domain can indicate which methods are more suitable to apply in the anomaly detection task.

Another requirement to consider before building an anomaly detection solution is the type of anomaly. Although the anomaly concept has a well-defined definitions in the literature, its semantics varies across domain areas. Chandola, Banerjee and Kumar (2009) classify outliers in some categories: (i) point anomaly, (ii) context anomaly, and (iii) grouped anomalies. The first considers problems in which a single instance can have anomalous behavior. For instance, suspicious network packets in network intrusion context and credit card transaction fraud. The second category of outlier fits the problem in this thesis. Here, data can have contextual outliers that rely on contextual and behavioral features of data. Contextual features determine the context of instances (e.g. latitude and longitude) while behavioral ones add information to the context of data, such as bus speed or the amount of rainfall. Last but not least, collective anomalies occur when two or more instances are an anomaly, but alone, they are not abnormal. Our proposal also accounts for grouped anomalies and we consider a single point as an anomaly as well.

Paying attention to the availability of labeled data also can steer which anomaly detection strategy to use. As discussed in previous sections, the lack of labels can limit the possibility of solutions. On this basis, Chalapathy and Chawla (2019) present an extensive set

of solutions based on supervised, semi-supervised, unsupervised, and hybrid approaches. On supervised anomaly detection, methods consider training models using normal and anomalous instances as a binary classification problem. In turn, semi-supervised detection methods train models considering only normal instances and test them using normal and abnormal examples. The unsupervised detection methods, on the other hand, do not rely on labels since they only depend on data properties. Also, they work on automatic labeling as well (PATTERSON; GIBSON, 2017). Lastly, hybrid techniques combine diverse approaches. For example, deep neural networks that are well-known as feature extractors can work along with traditional models such as One Class SVM (OC-SVM) or SVDD and One Class Neural Networks (OC-NN) (CHALAPATHY; MENON; CHAWLA, 2018; RUFF et al., 2018).

Another important aspect to consider is the way of outputting anomaly detection results. There are two traditional categories of outputs in literature: score, and binary. A score represents the level of an anomaly, in other words, instances can have more or fewer anomalous characteristics. This type of output is more flexible than a binary approach because outliers have different meanings and levels. On the other hand, binary results give a strict classification that can be more interpretative, but less informative. The decision of using one of those representations in most cases depends on the availability of labeled data.

Overall, anomaly detection has a rich plethora of definitions as well as a large application domain as detailed. However, in data mining trajectory domain, the detection of anomaly is not yet well-explored.

2.2.1 Anomaly detection on trajectory

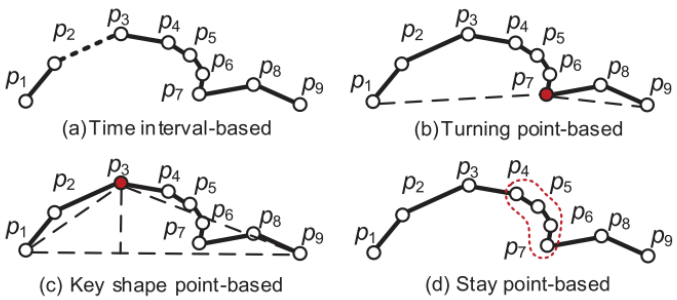
Trajectory anomaly detection has gained attention since more data about traffic objects has become available. Knorr, Ng and Tucakov (2000) were among of the first to propose an approach based on distance and multivariate data. Before (KNORR; NG; TUCAKOV, 2000), most of the solutions relied on univariate data and probability distribution assumptions (i.e., data has normal or X^2 distribution). They address anomalies only based on individual trajectories. That is, the approach looks at sub-trajectories or segments that are different from others considering the same historical trajectories. On the other hand, the trajectory data mining area also presents another concept named traffic anomalies for which the traffic flow behavior is important (e.g., traffic congestion caused by a car accident). Both previous concepts are different, but some works find traffic anomalies based on trajectory outliers (ZHENG, 2015). Overall, our focus is on trajectory outlier detection and, according to Meng et al. (2019) most of the works follow three steps to get a solution: preprocessing, transformation, and detection.

The preprocessing phase is well-known in the trajectory data mining domain. Among the techniques, the map-matching task, that is the process of aligning a sequence of ob-

served GPS positions into a road network (LOU et al., 2009), has some challenges because there are parallel roads, overpasses, and GPS device failure. According to Zheng (2015), there are four types of techniques based on additional information (i.e., road segments and maps). First, the geometric approaches that consider the shape of trajectories to match a point (GREENFELD, 2002). Second, probabilistic approaches calculate the transition probabilities between points, and they deal well with low-sampling rates. Third, topological algorithms that try to keep on the connectivity of the road network; they look for candidate road segments that fit well previous GPS sequences. Finally, advanced methods that combine previous approaches (YUAN et al., 2010).

A well-known pre-processing technique is segmentation. It is useful to decrease the complexity of analyzing the whole trajectory, reducing computational costs. For example, (LEE; HAN; LI, 2008; LV et al., 2017) discover anomalous segments by applying segmentation. Figure 2 summarizes the most common segmentation approaches in the literature. For example, Figure 2 (a) presents time interval-based approaches that split trajectories into parts according to the level of difference between two consecutive points. Similarly, Figure 2 (b) shows the turning point method that uses the level of heading direction between consecutive points to split trajectories. On the other hand, the key shape approach splits trajectories based on points that maintain a trajectory's shape as shown in Figure 2 (c). Lastly, looking at Figure 2 (d), the semantic meaning segmentation approach splits trajectories according to the stay-points. These methods are the most common in the literature but this is not an exhaustive list.

Figure 2 – Segmentation approaches



Source: ZHENG; XIE (2011)

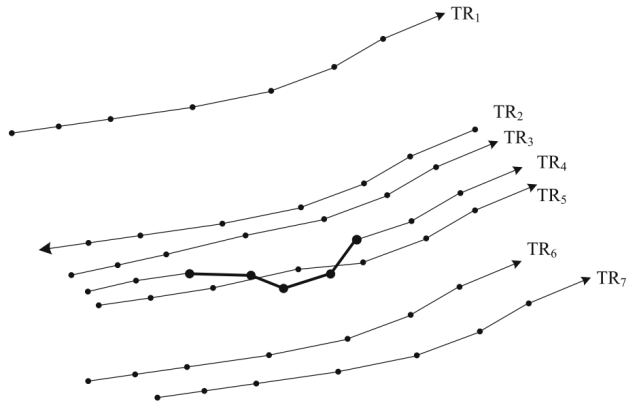
The second step to perform trajectory outlier detection is the transformation task. This activity aims to prepare data to feed a detection method since trajectories are complex data, and traditional algorithms, such as distance methods, cannot be applied directly (MENG et al., 2019). In the literature, grid discretization is a recurrent way to transform trajectories and facilitate the use of traditional methods. Some existing tools to create grids are: google-S2 [2] and the H3 [3] libraries. Both of them map trajectories into

[2] https://s2geometry.io/
[3] https://eng.uber.com/h3/

a collection of discrete cells (KULKARNI et al., 2018). As an illustration, (GE et al., 2010; PANG et al., 2011) split a city map into small regular grid-cells and map each trajectory point into them before detecting outliers. Lastly, it is worth mentioning that transformation is not a mandatory step. For example, (BESSA et al., 2015) uses raw trajectory data only applying a normalization.

Similar to other application domains in anomaly detection, literature also presents well-established methods to trajectory anomaly detection. (ZHENG, 2015; MENG et al., 2019) categorizes those methods as following: (i) distance-based, (ii) density-based, (iii) feature-based, (iv) sub-trajectory-based, and (v) activity-based. The first approach is based on distance functions (e.g., Hausdorff distance, Dynamic Time Warping, Longest Common Sub-Sequence, etc) to find close trajectories considering a distance threshold. For example, Figure 3 shows that TR1 is an outlier since it is far away from the other trajectories. The density methods looks at the density of trajectory neighbors instead of a global distance but also needs a threshold value. As an example, Figure 3 shows that TR6 and TR7 are anomalous trajectories compared to $TR1, ..., TR5$. The third type of methods look at the feature of trajectory. TR2 is an example of a feature outlier because it is going in the opposite direction of $TR3, ..., TR5$. The fourth approach does not look the trajectory as a whole; it splits them into sub-trajectories and then detects anomalous segments. For example, the bold segment of TR4 is an outlier. Last, activity methods that learn patterns about past activities of moving objects to discover divergent motion patterns representing anomalous activities. As an illustration, sudden changes in the moving trend of trajectories can be considered an anomaly (BASHARAT; GRITAI; SHAH, 2008).

Figure 3 – Outlier detection approaches



Source: MENG et al. (2019)

Finally, trajectory outlier detection also has real applications in some scenarios. For example, (LEE; HAN; LI, 2008) introduces an approach to find abnormal storms and typhoons trajectories. Also, (CHEN et al., 2011) proposes a detection outlier to discover taxi frauds, while (RAYMOND; IMAMICHI, 2016) suggests a bus trajectory outlier detection.

In brief, there are some works on trajectory outlier detection, but still, there are gaps, particularly in the traffic context.

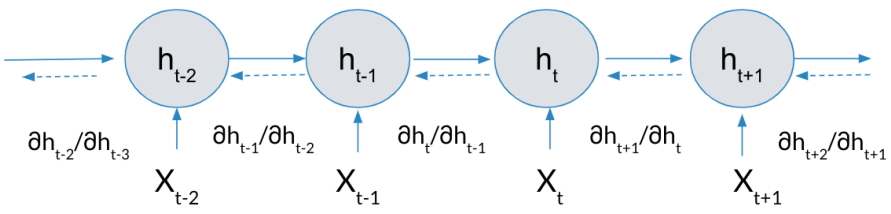
2.3 DEEP LEARNING BACKGROUND

Here, we explore a few essential deep learning concepts to understand our solutions. First, we briefly overview deep learning contributions in a few research areas, including in the anomaly detection field. Second, we present the main reasoning about rnn architecture and the Attention mechanism. Lastly, we introduce the Encoder-Decoder architecture and Self-Attention mechanism.

Deep Learning has been around for some time now. It is a subfield of machine learning inspired by Artificial Neural Network (ANNS) where successive layers can learn increasingly meaningful representations(CHOLLET, 2021). The deep learning approaches have been a breakthrough in some research areas such as Natural Language Processing (NLP) and Image Processing. Tasks such as translation, question answering (QA), and speech recognition, to name a few, have unprecedented results using deep learning approaches. In the same direction, deep learning methods have contributed to improving results in image processing tasks in such a way that overcomes humans. In recent years, deep learning has also been applied for anomaly detection in diverse tasks such as Illegal Traffic Flow detection (XIE et al., 2017), Retinal Damage detection (SCHLEGL et al., 2017), and Cyber-Network Intrusion detection (JAVAID et al., 2016).

In the trajectory anomaly detection field, deep learning-based anomaly detection algorithms have a few works in recent years. For example, the RioBusData method applies a Convolutional Neural Network (CNN) to learn a classification task and indirectly detects anomalous trajectories (BESSA et al., 2015). The GMVSAE approach uses a Variational AutoEncoder based on Long Short-Term Memory (LSTM) to detect anomalous trajectories (LIU et al., 2020). The Universal Transformer for Anomalous Trajectory Detection (UT-ATD) uses a universal Transformer to learn embeddings and feed them to a binary classifier that predicts if trajectories are anomalous or not (ZHANG et al., 2021).

Figure 4 – Unrolled Recurrent Neural Network



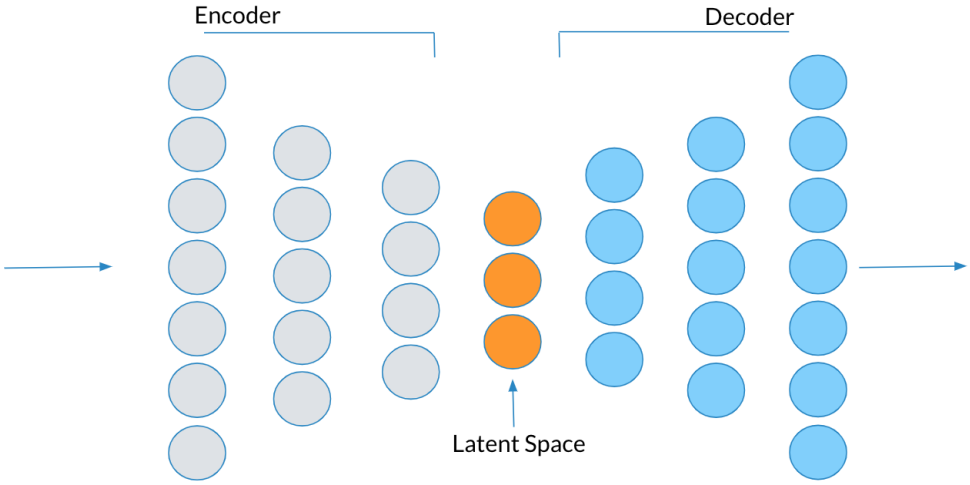
Source: Created by author (2022)

For this thesis, as trajectories are naturally sequential data, we first apply a well-known recurrent network called Gated Recurrent Unit (GRU) to answer our RQ1 and RQ2. The GRU is part of a family of Recurrent Neural Network (RNN), which is specialized for processing sequential data $x_1, ..., x_n$. Overall, the recurrent neural network predicts the future from a past sequence of events, where the network learns to capture historical information through the hidden state h to make predictions. For this, the network learns from the past using a mechanism called Backpropagation Through Time (BPTT) to calculate the network's loss and to infer the parameters from the followed equation:

$$h^{(t)} = f(h^{(t-1)}, x^{(t)}; \theta) \tag{2.1}$$

where x is the current sequence value, and θ is the network parameters. Figure 4 shows an example of BPTT in a time-unfolded computational graph representing a vanilla RNN. Observe that the loss are calculated by partial derivative in a recursive way such that $\frac{\partial h_{t+1}}{\partial h_t}$ relies on $\frac{\partial h_{t+2}}{\partial h_{t+1}}$. Although the BPTT is used to calculate the loss, the vanilla RNN has the problems called vanishing and exploding gradient(). Those problems occur when the partial derivatives, in the backpropagation process, generate small values and approach zero, or the values are larger, which causes the gradient descent to diverge. Given that, we explore the use of GRU that employ mechanisms for updating and resetting state h, which allows the network better control over the gradient.

Figure 5 – Encoder-Decoder architecture



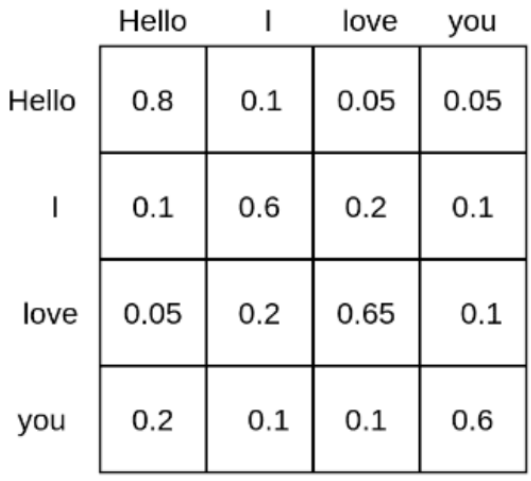
Source: Created by author (2022)

Although GRU and LSTM have remarkable results and have been firmly established in sequence modeling, their sequential nature restricts parallelization, which is critical for longer sequences (VASWANI et al., 2017). Also, those models represent all sequence information into a fixed-length vector (last hidden state), which can be a problem in coping with long sequences. To cope with those problems, most recurrence models adopt

the Attention mechanism (BAHDANAU; CHO; BENGIO, 2014) to help the recurrent models focus on different parts of the input sequence instead of static information compressed in a single vector. Overall, the Attention approach decides which parts of the sequence are more important than others, which is more helpful than a fixed representation. In recent years, Attention mechanisms have been used in conjunction with recurrent networks in almost all cases. For example, in neural machine translation, a well-known standard architecture called Encoder-Decoder can be built using attention and recurrent networks (ZHANG et al., 2020).

An Encoder-Decoder, the basis of our solution in Section 5 , is a machine learning model composed of two neural networks, as shown in Figure 5. The key idea behind that standard modeling paradigm is that the Encoder (first neural network) encodes a sequence to a new vector representation called latent space. Next, the second neural network, the Decoder, receives the latent space to map it to a new variable-length sequence. Although there are few ways to build Encode-Decoder, i.e., using CNN and RNN, we opt to use the Transformer approach instead of attention and RNN due to the constraint of sequential computation in recurrent models (VASWANI et al., 2017).

Figure 6 – Self-Attention matrix correlation



Source: ADALOGLOU (2021)

Lastly, as part of our solution Encoder-Decoder, we present a few concepts about the state-of-the-art deep learning model called Transformer. The Transformer is an architecture that uses only self-attention mechanisms to avoid the recurrence restriction. The core idea of Transformer is the self-attention mechanism, which is an approach to finding the correlation among each sequence element concerning all others. For example, Figure 6 shows an example of a correlation matrix calculated by a self-attention layer in a sequence of words. Note that the word **"love"** is more related to the words **"I"** and **"you"** than the word **"Hello"**. Based on this correlation matrix, a self-attention mechanism is

responsible for weighting each word, which helps the Encoder-Decoder to focus on parts of the sequence instead of looking for the entire sequence as occurs in traditional recurrence networks, for instance. Last but not least, it is worth mentioning that the translation task inspires our second solution in Section 5. However, our solution is slightly different because we use Encoder-Decoder to generate the same input, i.e., we apply an AutoEncoder, a particular type of Encoder-Decoder.

2.4 CONCLUSION

This chapter presented the theoretical basis for understanding this thesis, in which concepts and characteristics related to trajectory anomaly detection and data mining were described. For example, we defined trajectory and the concept of spatial and temporal anomalies. Also, we explained that anomaly detection solutions rely on a few requirements, such as the nature of data, type of anomalies, and anomaly detection outputs. Next, we presented a pipeline of tasks to get a trajectory anomaly detection solution and described the main techniques. Lastly, we described a few deep learning concepts that guide our solutions.

The following chapter presents the results of the most recent approaches that address the problem of trajectory outlier detection. Also, we present and discuss rule-based and machine-learning approaches.

3 LITERATURE REVIEW

In this chapter, we discuss some works that have similar goals with this thesis. In Section 3.1, we highlight some methods to detect stay point. Section 3.2 discusses in detail approaches to detect anomalous trajectories. Section 3.3 concludes this chapter.

3.1 STAY-POINT DETECTION

Among the preprocessing methods, stay-point detection is a method that proposes to identify trajectory points that indicate where moving objects or individuals stay for a while. It is important to detect stay points because they can inform snapshots of daily activities such as work in the office, restaurants, gas stations, etc. That is, stay points add semantic meaning to trajectories.

In the literature, Li et al. (2008) propose a stay-point detection method to find geographic regions. They use stay points to discover an individual's location history by measuring the similarity between users. The method uses distance and time to detect stay points based on rules, for example, if a user spends more than a threshold time within a distance threshold, the region is classified as a stay point. According to the authors, one of the reasons to use stay points is the computational cost since they use a clustering method to calculate similarity based on user locations.

Yuan et al. (2011) propose a recommendation system to support taxi. Before the recommendation task, the authors apply a stay-point method to find parking places (i.e., places where taxis frequently wait for passengers). The approach also relies on time and distance metrics. According to the authors, the algorithm keeps checking the distance between the current and earlier points. Thus, if the distance measure is smaller than a distance threshold, and the time interval between the points is superior to a time threshold, they are parking candidates. However, as the proposal considers individual trajectories, different parking places may be the same, then the authors apply the well-known density cluster approach named Ordering Points To Identify the Clustering Structure (OPTICS) algorithm (ANKERST et al., 1999) to discover which points represent the same parking places.

Nogueira et al. (2018) present the Statical Method for Stop, Move and Noise (MSN) method to detect stay points. For this purpose, the authors assume that stay points have slow speed and long duration. Differently from most works, the approach analyses trajectories individually as a three decomposed time-series based on duration, distance, and turning angle respectively.

Alvares et al. (2007), in turn, use stay-point detection to enrich raw trajectories with semantic information to facilitate queries. For example, the authors present an approach

named Stops and Moves of Trajectories (SMoT) that verifies which trajectory points intersect candidate stops (i.e., polygons previously chosen representing stops in a city). Then, considering the duration of those intersections, SMoT classifies the points as a stop. In a similar way, Moreno et al. (2014) propose the SMoT+, an extension of the SMoT method. The new approach allows discovering stops in nested regions according to different hierarchy levels instead of only considering disjoint spatial regions.

Hwang, Evans and Hanke (2017) propose to detect stay points to segment individual trajectories. The approach uses the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) (ESTER et al., 1996) algorithm adapted to temporal criteria to find Spatio-temporal clusters. However, as trajectory datasets have problems (e.g., low-sample-rate, devices issues), the authors claim that gaps can hide important stay points such as restaurants and malls. Therefore, they use the linear interpolation technique to treat the trajectories gaps before applying the extended DBSCAN. According to the authors, clusters alone with high density suggest a stay-point, however, without a duration threshold it is hard to find real stay points.

The most common criteria for stay point detection considers stop or moving points (HWANG; EVANS; HANKE, 2017). In this thesis, differently from the mentioned works that rely on empirically-based rules (e.g., distribution, distance threshold, duration, etc), we propose a sequential supervised method to detect stay points for some reasons. First, because a supervised method can discover the previous rules automatically, therefore, making a solution less dependent on domain experts. Second, GPS trajectory is a sequential data, then considering long-range dependencies can be more appropriate likewise occurs in NLP or in time-series field. In brief, Table 1 categorize the works highlighting the approach used as well as if there are thresholds and parameter dependencies to settup the approaches.

Table 1 – Stay point detection solutions

Method Used	Threshold dependency	Parameters dependency	References
Distance-based	True	False	(LI et al., 2008)
	True	True	(YUAN et al., 2011)
Intersection-based	True	False	(ALVARES et al., 2007)
	True	False	(MORENO et al., 2014)
Statistic-based	True	True	(NOGUEIRA et al., 2018)
Deep learning based (PAC)	False	False	(CRUZ; BARBOSA, 2020)

Source: Created by author (2022)

3.2 ANOMALOUS BUS TRAJECTORIES

Here, we split the literature approaches into some categories to give more granularity about the techniques used. The first category will describe works that apply distance-based and statistics-based methods that we name as rule-based methods. The second and third categories will show solutions that use unsupervised and supervised approaches.

3.2.1 Rule-based Detection

One of the first works applying distance-based methods to detect outliers is the partition-and-detect framework (LEE; HAN; LI, 2008). First, the authors partition trajectories, as a preprocessing, because segments can show unusual behavior and using summary information to compare the whole trajectories can smooth the differences in the parts of them. Next, they present the detection phase using a distance-based and density-based approach to find outliers. The process to find anomalies first considers the comparison among trajectories applying perpendicular, parallel, and angle distance. Second, as the closeness of trajectories can be affected by the density data instead of the sparseness ones, the authors suggest an adjustment coefficient based on computing the density of trajectory segments. In general, the framework presented is based on rules and depends on domain experts to determine if the results of the approach are meaningful and to set the distance thresholds.

Ge et al. (2010) develop the TOP-EYE method that focuses on generating an evolving outlier score. According to the authors, outliers evolve with time, then an evolving score to capture the outlying trajectories is necessary. That said, the TOP-EYE method proposed to detect directional and density outliers. The approach first maps trajectories into a grid and defines eight possible directions in each cell. Also, the authors calculate the visitation frequency that measures the frequency of a moving object in the grid cells. Then, through those values (i.e., direction frequency and pass frequency), the approach calculates the outlying score. However, to avoid the bias of historical outliers and finding a balance among past and current anomalies, the TOP-EYE approach suggests a cumulative decay function to give more attention to recent anomalies instead of past ones. The authors also claim that it is critical to keep the outlier measure updated in an accumulating way because outliers evolve. This approach is also distance-based, and it depends on a threshold to distinguish between abnormal and normal trajectories.

Zhang et al. (2011) introduce the Isolation-Based Anomalous Trajectory (iBAT) method to discover anomalous driving patterns from taxi GPS traces. The authors assume that outliers are few and different based on the isolation theory (LIU; TING; ZHOU, 2008). Based on that, to find the anomalies, the authors first extract taxi rides, split the city map into grid-cells, group all the taxi rides crossing the same source-destination cell-pair, and represent each trajectory as a sequence of cell symbols. Afterward, the iBAT that is an improvement of iForest (LIU; TING; ZHOU, 2008) isolates trajectories by randomly selecting grid-cells and removing trajectories that do not pass by the selected cells. Then trajectories with shorter paths (i.e., grid-cells) or with different paths are anomalies. However, differently from iForest, iBat focuses on test trajectories. That is, the isolation mechanism selects cells from the test trajectories instead of all trajectories, and then based on the number of cells used to isolate a trajectory the authors define an anomaly score.

In (PANG et al., 2011), the authors propose a Spatio-temporal method based on the Likelihood Ratio Test (LRT) statistic approach. The authors first split a city into a grid

and then estimate the count of the number of taxis in the grid cells given a time. The core idea is to identify a contiguous set of cells and time intervals which is statistically significantly different from expected behavior. Based on that, the approach uses the LRT method to separate the anomalous regions (i.e., regions here is a set of cells) from normal ones. To reach the aim, the authors claim that in regular conditions, the LRT follows a χ^2 critical value. Therefore, when the approach finds regions where the LRT score is above or below the χ^2 critical value, it considers the region as anomalous. It is worth mentioning that a Monte Carlo simulation should be used when the χ^2 is not applicable.

Pan et al. (2013) implement a method to detect traffic anomalies according to the routing behavior. The approach is based on topological variation in traffic flow between two points, and it is composed of three steps: offline mining, anomaly detection, and anomaly analysis. First, the offline task preprocesses data by creating an index to associate road segments to trajectories. Second, the authors implement an anomaly detection technique to find a set of sub-graphs (i.e., graphs of segments) of a road network where vehicles routing behavior is different from a pattern. Last, in the anomaly analysis, the authors use the location of anomalous sub-graphs and the time information to recover historical and current tweets. Based on that, they use the well-known Term Frequency-Inverse Document Frequency (TF-IDF) technique to filter the most common terms and rank them to analyze the possible anomalous causes.

Chen et al. (2013) introduce the Isolation-based Online Anomalous Trajectory Detection (iBOAT) method using the idea of isolating abnormal trajectories likewise (ZHANG et al., 2011). However, differently from iBAT, the iBOAT can discover the anomalous segment. To reach this aim, the authors first map trajectories into grid cells and create an index generation to simplify the retrieval of trajectory-position pairs for each possible grid cell. Next, they use an adaptative working window (i.e., a set of the latest incoming GPS points) to compare the ongoing trajectory with historical ones using similarity values based on a support function. iBOAT does real-time anomaly detection, then each new point receives a value of the support function. Overall, the approach uses a threshold to compare with the support function values to decide if a trajectory's new point is an anomaly or not. Yet, as the iBOAT also aims to rank trajectories according to the length of abnormal sub-trajectories, the authors implement a score function based on the sigmoid function.

Zhu et al. (2017) present the Time-dependent Popular Routes Based Real-Time Trajectory Outlier Detection (TPRRO) approach that extends the Time-dependent Popular Routes Based Trajectory Outlier Detection (TPRO) method proposed by (ZHU et al., 2015). According to the authors, TPRO can detect trajectory outliers, but not in real-time. TPRO method works detecting anomalies in historical data following some steps. First of all, the approach creates groups of trajectories with the same departure and destination. Second, it finds out the most popular routes through a graph structure called

Time-Dependent Transfer Graph (TTG). Finally, TPRO compares trajectories with popular routes (they represent the most used routes in the dataset) using a time-dependent edit distance method. As a result, if distances are greater than a threshold, TPRO finds out anomalous trajectories. The TPRRO method extends TPRO putting forward a Time-Dependent Transfer Index (TTI) to record which trajectories have passed through which location at which time. TTI index allows real-time outlier detection. Besides that, the novel approach further extends the TTG graph to work as a cache to boost the method. Note that TPRRO is an extension of TPRO that focuses on efficiency to answer in real-time.

Yu et al. (2017) introduce Minimal Examination (MEX) method to detect anomalous moving objects. The approach considers the idea of close neighbors to classify a trajectory as an outlier or not. That is, in a trajectory stream, if a test trajectory does not have enough close neighbors, the method classifies it as an anomaly. Before detecting outliers, the authors define three levels of anomaly detection: point-neighbors, trajectory-neighbors, and synchronized-neighbors. According to the authors, the levels of anomaly allow exploring higher granularity. For example, the point-neighbors measure how many neighbors are close looking at points while the trajectory-neighbor calculates how many trajectories are close within a time window. On the other hand, synchronized-neighbors is a time restriction over the previous two levels of anomaly detection. Overall, the anomaly task takes into account a time window and a distance measure to verify which points or trajectories are neighbors. For example, the approach computes an Euclidean distance between points of different trajectories in the same timebin to verify the neighbor points. They also use temporal windows to compare and calculate distances (i.e., a point $p_1^j \in T_1$, and a point $p_2^j \in T_2$ are within the same time window). Then, based on the levels of outliers, the time windows, and a distance measure, the MEX method builds data structures composed by a list of neighbors' timebins and a neighbor table. At last, as the authors keep those structures (i.e., timebins list and neighbor tables) for each trajectory, they can use query operations to get anomalous trajectories.

Hu et al. (2018) introduce the Isolation-Based Method To Detect Outliers (IBTOD) method based on the isolation concept (ZHANG et al., 2011). The authors define the approach following three steps: trajectory partitioning, feature extraction, and outlier detection. First, the partitioning splits trajectories into spatial and temporal levels using points with rapid spatial or velocity changes. Second, the feature extraction step also utilizes spatial and temporal features. The spatial attributes are three distances, as proposed by Lee et al. (LEE; HAN; LI, 2008) that produce the perpendicular distance, angular distance, and parallel distance. In turn, temporal features take into account the average velocity, the minimum, and the maximum velocity of sub-trajectories. Then, after feature extraction, the IBTOD adopts the isolation mechanism to detect outliers using the Isolation Forest (iFOREST) method to get abnormal sub-trajectories. However, as the

iForest only informs which sub-trajectories are anomalous, the authors propose a score function to give more flexibility in outlier detection. The approach also needs a threshold to distinguish between noises and outliers.

Zhao et al. (2020) propose an approach based on trajectory similarity measure and sparse subgraph to detect anomalous trajectory. According to the authors, using a similarity model that includes a semantic feature of position, time feature of trajectory, and velocity feature of object motion can reveal anomalous trajectory more effectively than using only one feature. To use that features efficiently, the authors propose three kernel functions for each mentioned feature and assign a weight for one of them. Also, they use a linear combination for feature fusion of the kernel results to generate a high-dimensional vector for trajectory. Then, with a new representation for each trajectory, the authors create a feature graph where the vertices are the trajectories, and the edges are the similarity between them (using fused features values). Thus, given a feature graph where the vertices are the trajectories and the edges the similarity between them (using fused features values), the approach discovers anomalous trajectories by searching the sparse subgraphs whose weight coefficient is less than a threshold. Overall, the approach presents promising results, but it only detects whether the whole trajectory is anomalous and does not clarify whether it works online.

3.2.2 Unsupervised Detection

Similar to (ZHANG et al., 2011), Liu, Ni and Krishnan (2013) propose a trajectory outlier detection on taxi's context. However, the authors use a cluster approach based on speed to calculate a fraud function. First, the method finds active areas where many taxis passed through and splits them into two clusters based on the average speed task. Each task has information about the average speed of an occupied (status) taxi trajectory. Hence, looking at the average speed, the authors divide taxi trajectories into two groups. One has taxis with speed above the average value, whereas the other includes taxis with speed below. Based on that, the anomaly detection relies on a suspicion function that describes the possibility of a taxi's average speed being an outlier in a specific active area because its speed is below the average. Also, the authors implement a fraud detection function that takes into account all suspicious functions from a complete trajectory to give a probability of fraudulent taxi behavior.

Lv et al. (2017) propose a method called Density-Based Outlier Trajectory Detection (DBTOD) to detect trajectory outliers. The authors approach the problem first building a road network (i.e., directed graph) to simplify the use of trajectories. Next, the work defines two functions: route distance function and Min Core Distance. The route function works similar to the Edit Distance, and it outputs a similarity value between two routes while the second function gives an outlier score. However, before applying the Min Core Distance, the authors use the DBSCAN method to discover clusters on a historical dataset.

After finding the groups, they select the core trajectories as representatives and use them to calculate a min core distance (i.e., outlier score) with a test trajectory. Hence, if the lowest similarity score is above a specific threshold outlier, the test trajectory will be an outlier. According to the authors, it is necessary to split routes into different groups (i.e., same origin and destination pairs) in advance to turn the result of clusters meaningful.

Wang et al. (2018) propose a method that combines the Edit Distance (ED) function with a hierarchy cluster technique as a solution to detect trajectory outliers. To achieve this objective, the authors first use an extended Edit Distance to calculate a similarity matrix among trajectories. Second, they implement an iterative process where they first consider each trajectory as an initial group, and when the distance between routes is minimal, a clustering method (i.e., hierarchical cluster) merges those previous groups. The process keeps on combining the trajectories until the number of groups reaches a threshold. Lastly, groups with occasional samples represent abnormal samples, while groups with more than one route are normal. Note that, according to the authors, the Edit Distance extension aims to avoid giving different weights to low-sampling-rate trajectories because long-trajectories have higher edit distance values than short-sequences. Also, the approach proposes four behavior patterns based on the trajectory's length and time to analyze causes of anomalies in the taxi context.

(KONG et al., 2018) propose Long-Term Traffic Anomaly Detection (LOTAD) to detect anomalous trajectory segments and abnormal regions in the bus context. Toward this end, the approach first extracts sub-trajectory named TS-Segment (Temporal-Spatial Segments) from two consecutive stop stations. Second, the authors use the TS-Segments from bus trajectory data to create a segment matrix where each element is a TS-segment composed of average speed and average stop time. Each bus line has a segment matrix that is treated as input data to find the anomalous segments. In this part, the LoTAD approach uses a Gaussian kernel function (KUMAR; MANGATHAYARU; NARSIMHA, 2016) to calculate segments' density and to get their relative distances (i.e., based on the features). On this basis, the authors use the well-known K-means to look for four groups: perfect segments that have higher density and three anomalies groups with the lowest density. Each anomaly group has a semantic meaning, for instance, points with higher stop time and lower average speed mean bigger demands on the stop stations and road congestion, while higher average speed and lower stop time mean the travel demand is high. After finding the groups, the authors remove the group with the highest density and explore only the anomalous segments. Last but not least, the approach proposes to discover abnormal regions. For this purpose, the authors split the trajectory's city into small areas according to the placement of bus stations. Next, looking at the regions, they generate a region's anomaly index based on anomalous segments to discover which areas have problematic traffic conditions.

3.2.3 Supervised Detection

Bessa et al. (2015) propose a multi-class Convolutional Neural Network to classify trajectories according to their route IDs. The main assumption is that a multi-class classifier can detect anomalous trajectories when a misclassification occurs or the classifier output probability is below a certain threshold. The classifier achieved a high accuracy value 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 datasets.

Raymond and Imamichi (2016) introduce a bus route classification approach that the authors claim that can be used to detect spatial and temporal outliers as (BESSA et al., 2015) but no specific method was presented. The classification is performed in two steps: First, it transforms the input GPS sequence into a sequence of road IDs based on OpenStreetMap [1] road network (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 K-Nearest Neighbors (KNN) model to perform bus route classification. Our approach detects candidate anomalous trajectories similar to (BESSA et al., 2015; RAYMOND; IMAMICHI, 2016) in the sense that the models are learning the spatial and temporal features of trajectories. However, our work differs from them in some aspects. We represent trajectories by two distinct types of information: activity information points embedding, and geo embedding. In addition, we propose a specific method for anomaly detection based on the classifier confidence and evaluate it on spatial and temporal anomalies.

Chen et al. (2016) propose the Trend-Residual Dual Modeling (TRDM) method. The approach detects abnormal points using a regressor method named Cubic Smooth Spline along with the ARMA algorithm. According to the authors, the Cubic approach models trajectory trends where trajectories are represented by (timestamps, latitude) and (timestamp, longitude). Looking at the trends, the TRDM uses the ARMA method to model the residuals since outliers stay into the residual data. Last, the TRDM implements an outlier score as output to compare with a critical value (i.e., threshold) to decide if a trajectory is anomalous or not.

In Gaussian Mixture Variational Sequence AutoEncoder (GM-VSAE), Liu et al. (2020) propose a deep learning encoder-decoder approach to detect whether the trajectory is anomalous or not. The approach uses an LSTM architecture to encode and decode trajectories. Overall, the approach uses the encoder to model trajectories by a gaussian distribution mixture. Then, the proposed method uses each distribution as the decoder's input information to calculate the probability of an ongoing trajectory belonging to one of the distributions. The trajectory is anomalous if the probability is below a certain threshold.

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

Zhang et al. (2021) introduce a supervised approach UT-ATD. The solution has three main components: preprocessing, Universal Transformer (UT), and MLP blocks. The first block pre-trains an embedding method. The second one uses a universal encoder transformer to learn the previous pre-trained embedding output and a mapped trajectories representation. Finally, to train the model and find the anomalies, the approach feeds the encoder output to the MLP block and applies the Binary Cross-Entropy (BCE) function to train the UT and MLP together. Although the results of UT-ATD are promising, the method is not an end-to-end solution because it has to pre-train the Word2Vec [2] approach to feed the UT component. Also, the approach needs labeled datasets that are not easy to find in our context.

Qian et al. (2021) propose an online Spatial-Temporal Relations (STR) method to detect abnormal taxi trajectory. The basic idea of the approach is to verify if the displacement (distance between $p1$ and p_i) increases continually along with the drive distance (distance between consecutive segments) and the drive time (delta time between consecutive segments). For that, the authors formulate two regression models to learn historical drive distance Displacement Spatial (D-S) and drive time Displacement Time (D-T). Once those models are learned, a test trajectory can be evaluated, and the STR method outputs an anomaly probability for each point. Next, an anomaly score is generated based on the sum of the anomaly probability for each point. Lastly, the trajectory is considered anomalous whether the anomaly score is greater than a threshold. Although the results of the experiments show greater precision by reducing false positives, likewise (CHEN et al., 2013) method, STR needs an inverted index to recover historical trajectories with the same S-D pairs. Also, the approach needs to train D-S and D-T every time a new trajectory arrives to be analyzed when a cache mechanism does not already contain fitted models.

3.2.4 Comparative Analysis

Table 2 summarizes the trajectory outlier detection solutions described in the previous sections. For each one of them, we highlight the type of the approach, its method, and the type of anomalies that it deals with. For example, the rule-based works described here mostly rely on distance functions, density, and statistics (LEE; HAN; LI, 2008; GE et al., 2010). On the other hand, unsupervised methods show that probably density-based solutions are more suitable than others (e.g., partitioning and hierarchical). Moreover, the anomaly type describes that spatial and temporal are the major concern in the works, and the localization characteristic reveals which solutions can discover anomalous segments since they are important to understanding anomalous causes. Those characteristics give us a notion of how robust the works are and how dependent they are from expert domains.

[2] https://radimrehurek.com/gensim/apiref.html

Lastly, we highlight which solutions work online since they are most suitable for real applications and highlight which works provide code implementation.

This thesis has the same interests as the above works. However, differently from the previous works, we propose to apply a language model for anomaly detection in bus trajectory. Overall, our proposal intends to detect only spatial bus trajectories anomalies and localize where the outlier occurs (i.e., anomalous sub-trajectories). Particularly, our method uses an approach commonly used in Natural Language Processing and allows the task to be performed offline and online. Lastly, it is worth mentioning that our solution does not require any manual feature extraction and can be adapted to other types of GPS trajectories.

Table 2 – Trajectory Outlier detection solutions

Approach	Methods	Anomaly type	Localization	Online	Code	References
Rule-based	distance, density	spatial	Yes	No	No	(LEE; HAN; LI, 2008)
	statistic-based (frequency)	directional, density	No	Yes	No	(GE et al., 2010)
	statistics-based	spatial, temporal	Yes	No	No	(PANG et al., 2011)
	similarity-based	speed	No	No	No	(LIU; NI; KRISHNAN, 2013)
	distance-based	spatial, temporal	Yes	Yes	No	(PAN et al., 2013)
	frequency-based	spatial	Yes	Yes	No	(CHEN et al., 2013)
	distance-based	spatial, temporal	No	Yes	No	(ZHU et al., 2017)
	distance-based	spatial	Yes	Yes	No	(YU et al., 2017)
	distance-based	spatial, temporal	Yes	No	No	(HU et al., 2018)
	similarity-based	spatial, temporal	No	No	No	(ZHAO et al., 2020)
Unsupervised	frequency(lazy-learning cells)	spatial	No	No	No	(ZHANG et al., 2011)
	density-based	spatial	No	No	No	(LV et al., 2017)
	distance-based	spatial	No	No	No	(WANG et al., 2018)
	distance-based	spatial	Yes	No	No	(KONG et al., 2018)
Supervised	CNN	spatial, temporal	No	No	No	(BESSA et al., 2015)
	KNN	spatial, temporal	No	No	No	(RAYMOND; IMAMICHI, 2016)
	ARMA	spatial	Yes	No	No	(CHEN et al., 2016)
	Encoder-Decoder LSTM	spatial	No	Yes	Yes	(LIU et al., 2020)
	BIGRU+Attention (STOD)	spatial, temporal	No	No	Yes	(CRUZ; BARBOSA, 2020)
	Regression	spatial, temporal	Yes	Yes	True	(QIAN et al., 2021)
	Transformer Encoder	spatial	No	No	No	(ZHANG et al., 2021)

Source: Created by author (2022)

3.3 CONCLUSION

In this chapter, we have reviewed several works that are relevant and have common purposes with the objectives of this thesis.

We first described and discussed stay point detection methods. According to the literature, most works are rule-based, such as distance and time interval between points. Also, some works apply density cluster methods to find such stay points. In addition, we saw that stay-points are used for some reasons, such as enriching raw trajectories with semantic information, segmenting trajectories, and reducing computational costs.

The second half of this chapter presented and discussed state-of-art trajectory anomaly detection approaches. We split the works based on the strategy/approach adopted, such as rule-based, unsupervised, semi-supervised, and supervised, to facilitate the understanding. Note that most of the works are rule-based, but more recently, methods based on learning have shown to be competitive. Furthermore, it is worth mentioning that the lack of labeled datasets also restricts supervised approaches. Table 2 summarizes the methods according to a few features that might be important to other researchers. For example, the feature method describes which type of strategy the solution implements more specifically. Also, the anomaly type and localization describe which type of anomalies the solutions can detect and if they can localize the anomalous segments/regions in trajectories. Lastly, we highlight the solutions that work online and which ones make the implementation code available.

The following chapters will present our solutions to detect bus anomalies trajectories. Our first solution is a supervised approach that focuses on spatial-temporal anomalies. However, it only works in a batch way and can not pinpoint the abnormal segments. On the other hand, the Chapter 5 will present our online model language approach that focuses on detecting spatial anomalies and can localize anomalous segments. Last but not least, the Chapter 6 will present our demo tool for visualizing and analyzing trajectories.

4 LEARNING GPS POINT REPRESENTATION TO DETECT ANOMALOUS BUS TRAJECTORIES

In this section, we aim to answer the two first research questions RQ1 and RQ2: *how can we propose a bus supervised trajectory outlier detection using classification confidence as an anomaly score?* and *Are there any advantages to representing trajectories by stay points on the outlier detection task?* For the first goal, we propose a multi-class classifier that learns the typical behavior of buses but, instead of performing a hard detection decision as previous approaches (BESSA et al., 2015; RAYMOND; IMAMICHI, 2016), our solution calculates an anomaly score based on the uncertainty of the classifier. More specifically, our classifier, which we call Spatial-Temporal Outlier Detector (STOD), predicts the route line of a given bus trajectory. Our main assumption is that when the classifier is uncertain about the bus route of a given trajectory, there is a high chance that this trajectory is anomalous. To reach our second goal, we propose the Point Activity Classifier (PAC) which is a stacked deep-learning model that 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. Overall, we feed the PAC embedding into the STOD method. In the remainder of this chapter, we describe in details these two strategies.

4.1 PROBLEM DEFINITION

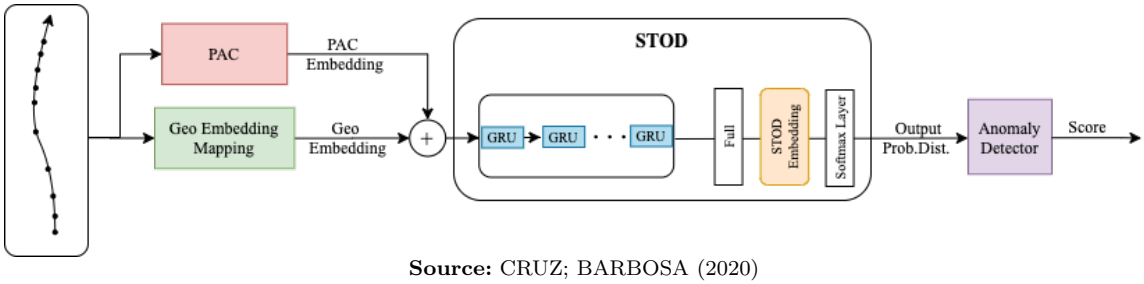
Definition 1. ***[Problem Statement]****. Given a bus trajectory T_l, we aim to calculate the spatial-temporal anomaly score function $f : T_l \rightarrow \mathbb{R}$, such that $f(T_l) \in [0, 1]$.*

4.2 METHODOLOGY

This work proposes a spatial-temporal outlier scoring method for bus trajectories data. Similar to previous approaches (BESSA et al., 2015; BONTEMPS et al., 2016; LI et al., 2007), 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 (CHANDOLA; BANERJEE; KUMAR, 2009). 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 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 Clas-

Figure 7 – Overview of our anomaly score approach



Source: CRUZ; BARBOSA (2020)

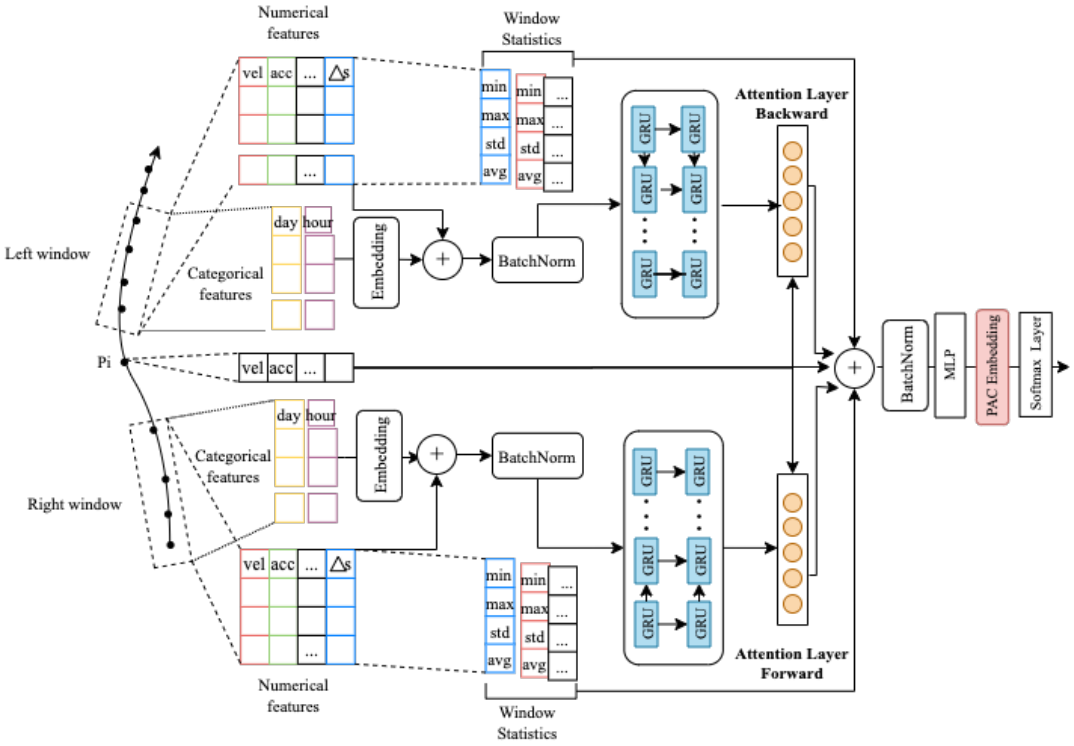
sifier) as a feature extractor to learn a vector representation for p_i (PAC embedding) by predicting whether p_i is moving or is one of the 3 different types of stay points: bus stop, traffic light or other type of stay point. The second strategy uses a word embedding algorithm (MIKOLOV et al., 2013) 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 7. In the remainder of this section, we provide further details about the whole solution.

4.2.1 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 another kind of stop. Alongside the prediction, the model works as a feature extractor by producing a vector representation of p_i (PAC Embedding) as shown in Figure 8. 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 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 (MOHANADAS, 2018), 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 8, to predict the class of a

Figure 8 – PAC model



Source: CRUZ; BARBOSA (2020)

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, those data are used as features, extracted from the timestamp of the point. Similar to (WANG et al., 2018), instead of using their raw representation, PAC considers them as categorical values and applies Entity embeddings (GUO; BERKHAHN, 2016) 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 Vincenty's formula (VINCENTY, 1975) to compute the geographical distance between two points. Another feature, travel distance, is the geographic distance between the initial point of the trip and the current point. It tries to capture events in specific locations of the trajectory. The final feature is the bearing rate (DABIRI; HEASLIP, 2018), 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' and p'' is calculated as:

$$y = \sin(lng_{p''} - lng_{p'}).\cos(lat_{p''}) \tag{4.1}$$

$$x = \cos(lat_{p'}).\sin(lat_{p''}) \tag{4.2}$$

$$z = \sin(lat_{p'}).\cos(lat_{p''}).\cos(lng_{p''} - lng_{p'}) \tag{4.3}$$

$$bearing_{p'} = \arctan(y, (x - z)) \tag{4.4}$$

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

In each batch, the network applies batch normalization (IOFFE; SZEGEDY, 2015) on the values of these features to normalize them in order to give numerical stability to the model:

$$\bar{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{4.5}$$

$$z_i = \gamma.\bar{x} + \beta \tag{4.6}$$

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 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:

$$e_j = f(p_i, v_j) \tag{4.7}$$

$$\alpha_j = \frac{exp(e_j)}{\sum_{k=1}^{h} e_k} \tag{4.8}$$

$$v_{att} = \sum_{j=1}^{h} \alpha_j v_j \tag{4.9}$$

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 (DABIRI; HEASLIP, 2018). 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).

4.2.2 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 $E(T)$:

$$E(T) = \sum_{i=1}^{n} -P(T)_i \log_n P(T)_i \tag{4.10}$$

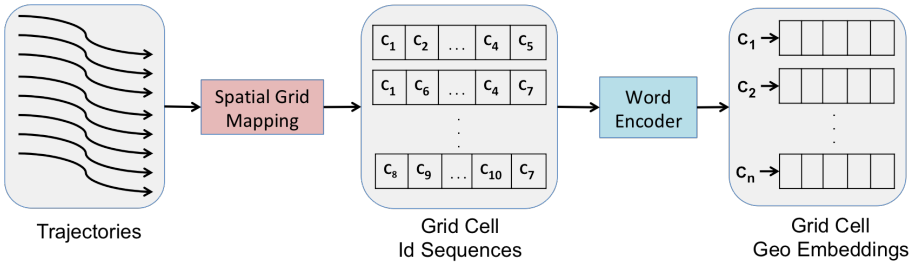
where $P(T)_i$ is the probability of the trajectory T being in class i, and n is the number of classes being predicted by the classifier. The normalization is performed by setting the log base equals to n, since the maximum entropy value is $log_n(n) = 1$.

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 4.2.1; and Geo embedding that represents a point based on its geo location and its neighbors. Similar to (GAO et al., 2017), to create Geo embedding, as shown in Figure 9, first the Spatial Grid Mapping maps the points of trajectories to a grid of hexagons using the H3 technique[1].

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

[1] <https: //eng.uber.com/h3/>

Figure 9 – Geo embedding pipeline



Source: CRUZ; BARBOSA (2020)

grid. These grid cell id sequences are then used as input to the Word Encoder (Word2Vec (MIKOLOV et al., 2013) 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 7, 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[2]. 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 classfication, 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 4.4 shows.

4.2.3 Training

To train both models (PAC and STOD), we use the focal loss (LIN et al., 2017) function that deals with highly unbalanced data, which is the case of our input data as we show in Section 4.4, leading to better performance than traditional multi-class loss functions. The focal loss is described as:

$$FL(p) = \sum_{t=1}^{n} \alpha_t (1 - p_t)^{\gamma_t} . log(p_t) \tag{4.11}$$

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.

[2] 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.

4.3 DATA DESCRIPTION AND SETUP

4.3.1 Experimental Setup

Bus Trajectory Datasets. We used datasets of two different cities in our evaluation:

- Recife[3] (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[4] (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 (our route id), longitude, latitude, vehicle journey id or travel id, and vehicle id. 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 4.4, we collected General Transit Feed Specification (GTFS)[5] 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 for Recife from *OpenStreetMap*[6], and traffic light for Recife from the city's open data website[7]. The label "in route" is defined by points with speed greater than 5Km/h, and the label "other stop" is set to any stop not labeled as bus stop or traffic signals.

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

[3] We obtained the Recife dataset from a collaboration with the local government.
[4] https://data.gov.ie/dataset/dublin-bus-gps-sample-data-from-dublin-city-council-insight-project
[5] https://developers.google.com/transit/gtfs
[6] https://www.openstreetmap.org/
[7] http://dados.recife.pe.gov.br/dataset/localizacao-dos-semaforos/resource/ab6343e9-c3f2-4d62-9554-5778f9f33738

Table 3 – Values of hyperparameters 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

Source: CRUZ; BARBOSA (2020)

consecutive points with time difference lower than 5 minutes. Finally, the feature extraction generates the set of features for each point mentioned in Section 4.2.

Point Activity Approaches. We assess the quality of the PAC embeddings by comparing traditional classification models using as features PAC embeddings versus PAC's original features. The PAC model was trained with the left and right windows equal to 16. We also evaluate PAC on the stay point prediction task by training and evaluating the PAC model with only the 3 stay-point classes (bus stop, traffic light and another kind of stop). We compare this approach with Conditional Random Field (CRF) (LAFFERTY; MCCALLUM; PEREIRA, 2001), a widely used probabilistic sequence-based model. In this experiment, we used the CRF implementation CRFSuite[8]. For all experiments, we split the datasets in 64% for training, 16% for validation and 20% for testing. We trained all approaches based on searching for the best hyperparameters. For CRF, we selected the best values for the L1 and L2 regularization coefficients among 4 values generated by a exponential continuous random variable for each coefficient. 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 (BERGSTRA; YAMINS; COX, 2013). The hyper-parameters, the values that we used to search and the best values for each dataset based on the validation set are shown in Table 3. We executed the search for both tasks but the same values were selected.

Classifiers. We apply the following traditional classification models in this evaluation:

- Random Forest (random forest): it is an ensemble of decision tree classifiers that are trained with the bagging method (HO, 1995). The hyperparameters n-estimators and max-depth were optimized in a range of values [2-50] and [1-32].
- Gaussian Naive Bayes (gaussian nb): it is a classifier based on Bayes' theorem, and features independence (RUSSELL; NORVIG, 2002). No hyperparameter search was performed.

[8] https://sklearn-crfsuite.readthedocs.io/

- Support Vector Machine (svm): svm is a non-probabilistic classifier characterized by finding hyperplanes (support vectors) that maximize the margin between classes (CORTES; VAPNIK, 1995). We used a linear kernel with the regularization hyperparameter C ranging between $[1e^{-10} - 1e^{10}]$.

- Light Gradient Boost (lgbm): lgbm is an improved Gradient Boosting which is based on an ensemble of decision trees and a boosting method (KE et al., 2017). We varied the values of hyperparameters n-estimators [gbdt,dart,goss], number of leaves [10-150], learning rate $[1e^{-5} - 1]$ and feature fraction [0-1].

- K-Nearest Neighbors (knn): it is a lazy and non-parametric method that classifies examples based on a similarity measure (DUDA; HART; STORK, 2012). We varied the number of neighbors in the range [1-30] and the KNN's algorithm [auto, ball_tree, kd_tree, brute].

In order to choose the values of the hyper-parameters, we used the hyperparameter optimization framework Optuna (AKIBA et al., 2019).

Route Classification Approaches. We executed the following bus route classification strategies:

- Riobusdata (BESSA et al., 2015) 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 (BRODSKY, 2018) 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 100 points on average, and they were implemented using Keras[9]. We trained them for 100 epochs and selected the best models by early-stopping accuracy validation to avoid overfitting. For these experiments, we split the datasets in 64% for training, 16% for validation and 20% for test. We used Hyperas[10] to tune the some of the models' hyper-parameters. Table 4 and Table 5 show the tuning and best values used for STOD and Riobusdata respectively.

[9] https://keras.io/
[10] https://github.com/maxpumperla/hyperas

Table 4 – Values of hyperparameters 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

Source: CRUZ; BARBOSA (2020)

Table 5 – Values of hyperparameters 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

Source: CRUZ; BARBOSA (2020)

4.3.2 Evaluation Metrics

We used weighted Precision (WP), weighted Recall (WR), and weighted F1 (WF1) for this evaluation due to the unbalanced nature of our datasets:

$$WF1 = \frac{\sum_{i=1}^{n} w_i.F1_i}{\sum_{i=1}^{n} w_i} \tag{4.12}$$

$$WR = \frac{\sum_{i=1}^{n} w_i.Recall_i}{\sum_{i=1}^{n} w_i} \tag{4.13}$$

$$WP = \frac{\sum_{i=1}^{n} w_i.Precision_i}{\sum_{i=1}^{n} w_i} \tag{4.14}$$

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

Table 6 – 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

Source: CRUZ; BARBOSA (2020)

4.4 RESULT AND DISCUSSION

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

4.4.1 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 6 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 sofmax classifier (pac in Table 6) 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 10 (a) and Figures 10 (b) the 2D projection of learned PAC embeddings using T-SNE (MAATEN; HINTON, 2008) on the two datasets. They illustrate desirable properties of good representations according to Bengio et al. (BENGIO; COURVILLE; VINCENT, 2013): 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).

PAC as a Stay Point Classifier. We also evaluated the PAC model as a stay point classifier by training it only considering the stay point classes: bus stop, traffic light and another kind of stop. We compare its results with a CRF model. Due to performance

Table 7 – Results of PAC and CRF models for stay point classification

		Recife			Dublin		
Model	Label	WP	WR	WF1	WP	WR	WF1
pac	bus_stop	**0.608**	0.547	0.576	**0.770**	**0.832**	**0.799**
	other_stop	0.649	**0.713**	**0.680**	**0.805**	**0.779**	**0.792**
	traffic_signal	**0.349**	**0.294**	**0.319**	**0.545**	**0.439**	**0.486**
	Average	0.535	0.518	0.525	0.762	0.766	0.763
crf	bus_stop	0.583	**0.615**	**0.599**	0.580	0.732	0.648
	other_stop	**0.666**	0.647	0.656	0.485	0.414	0.447
	traffic_signal	0.050	0.010	0.016	0.256	0.086	0.129
	Average	0.433	0.424	0.424	0.505	0.538	0.511

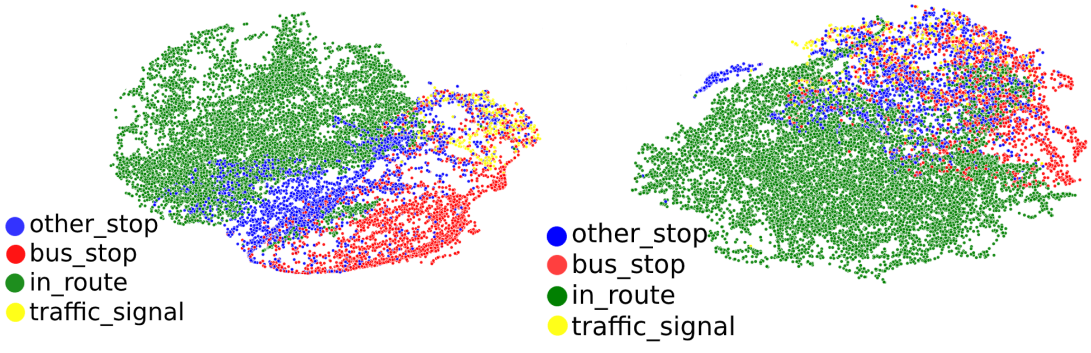
Source: CRUZ; BARBOSA (2020)

limitations in training CRF, we limited the training data used by both approaches in 5,000 points that represent 104 trajectories. Table 7 shows that the average values of WP, WR and WF1 for the PAC model are higher than the CRF for both datasets. On WF1, for instance, PAC achieved 0.525 and CRF 0.424 for Recife and 0.763 versus 0.511 for Dublin. Looking at individual labels, PAC outperforms CRF on *traffic_signal* for both datasets for all three metrics. For *bus_stop*, PAC achieved the best result on the Dublin dataset: WF1 value of 0.799 for PAC and 0.648 for CRF, whereas in the Recife dataset CRF obtained a slightly better WF1 than PAC: 0.599 and 0.576 respectively. These results show that our PAC model is also effective for the stay point classification task.

4.4.2 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 8 show that STOD(geo,pac) and 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),

Figure 10 – Visualization of PAC embeddings using T-SNE



(a) Embeddings from Recife.

(b) Embeddings from Dublin.

Source: CRUZ; BARBOSA (2020)

Table 8 – 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

Source: CRUZ; BARBOSA (2020)

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 (MANN; WHITNEY, 1947) 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.

Table 9 – 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

Source: CRUZ; BARBOSA (2020)

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 9 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 7). 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 process, we apply PCA (Principal Component Analysis) (JOLLIFFE; CADIMA, 2016), decreasing the STOD vectors to 32 dimensions. Table 10 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 a simple model such 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 (ZHANG et al., 2019), which we leave as a future work.

4.4.3 Outlier Detection Assessment

We also evaluate our approach for detecting spatio-temporal anomalous trajectories. Concretely, 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 trajectories labeled as outliers, we had

Table 10 – 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

Source: CRUZ; BARBOSA (2020)

to synthetically generate spatially and temporally anomalous trajectories. Based on the STOD's entropy results in Table 9, 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 4.10) 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:

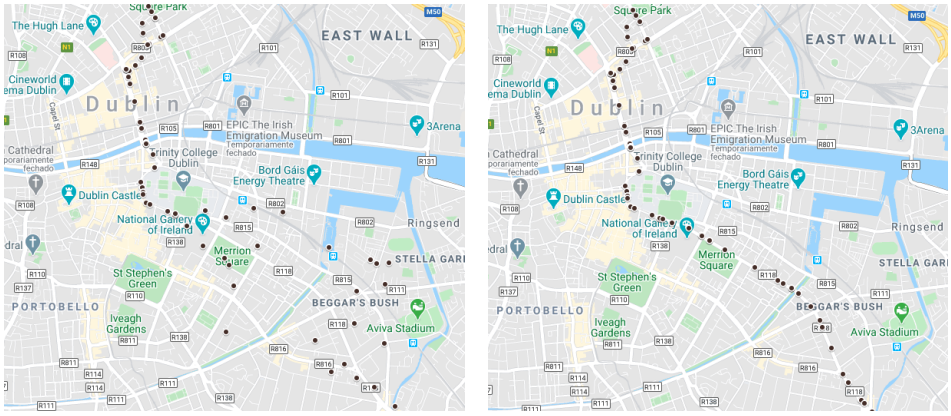
$$\frac{f_{ent}(AT) - f_{ent}(NAT)}{f_{ent}(NAT)} \tag{4.15}$$

where f_{ent} is the entropy value at a specific percentile. We used the 25th, 50th (median) and 75th percentiles in our evaluation. A positive value of relative entropy means that the anomaly score is higher in the anomalous trajectories than in the non-anomalous ones at a certain percentile, and a negative value means otherwise.

Spatial Anomaly. We consider as spatially anomalous the trajectories 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* [11] 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 21 (a)

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

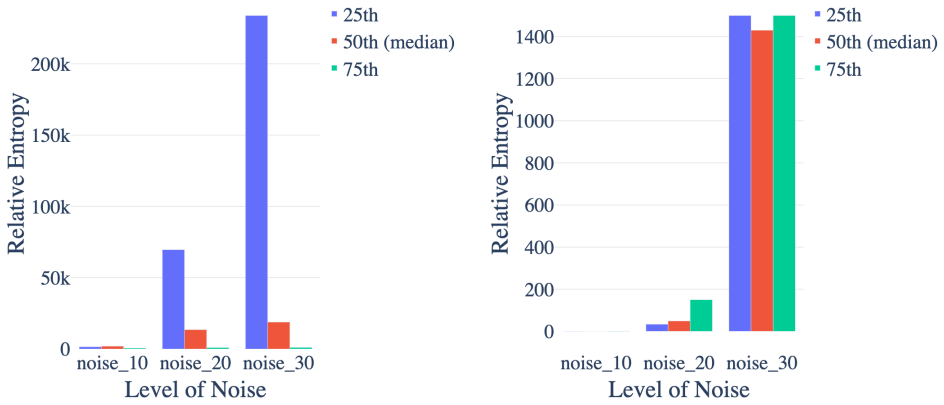
Figure 11 – Example of spatial anomaly



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

Source: CRUZ; BARBOSA (2020)

Figure 12 – Results of spatial anomaly



(a) Recife. (b) Dublin.

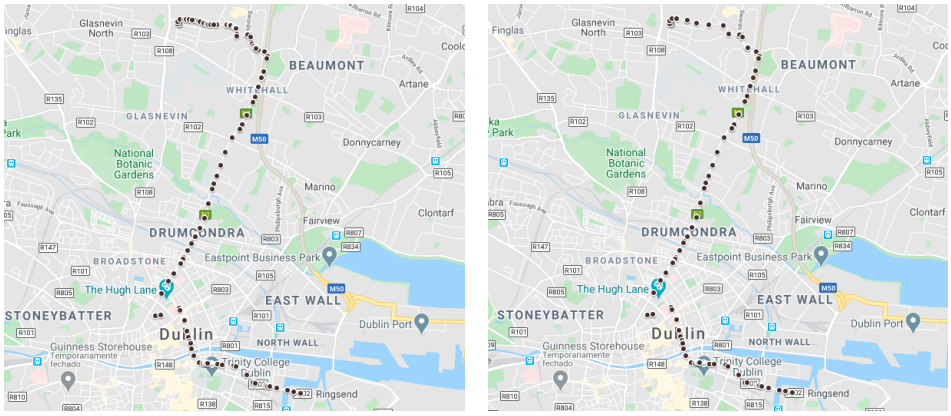
Source: CRUZ; BARBOSA (2020)

shows an example of a synthetic trajectory and Figure 21 (b) the original one. The noisy points are at the bottom of Figure 21 (a) which shows points moving away, and returning to the original shape of trajectory.

Figure 12 (a) and Figure 12 (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 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.

Temporal Anomaly. We also evaluate our approach for detecting temporal anomaly

Figure 13 – Example of temporal anomaly type I



(a) Anomalous trajectory.

(b) Raw trajectory.

Source: CRUZ; BARBOSA (2020)

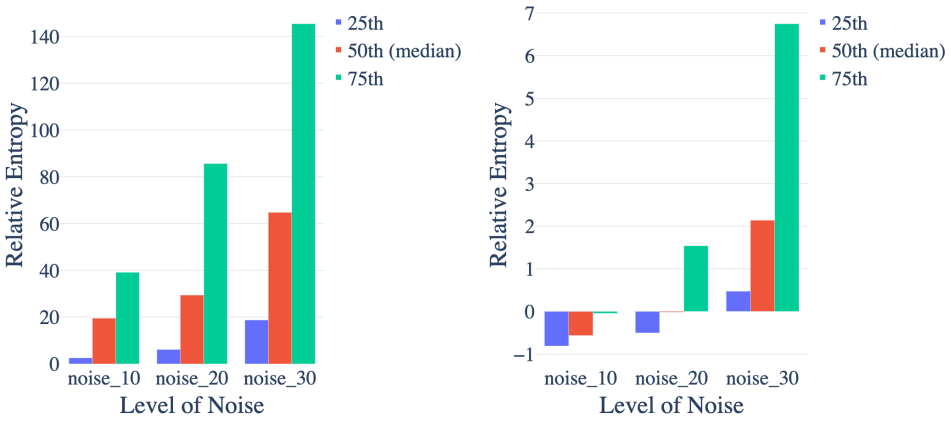
trajectories on the two types described in Section 2.1: Temporal Anomaly Type I (more congestion than usual) and Temporal Anomaly Type II (less congestion than usual). We simulate the Temporal Anomaly Type I as follows. First, given a non-anomalous trajectory T and the its assigned route line l in the 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 13 (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 13 (b) the original bus trajectory without noise.

Figure 14 (a) and Figure 14 (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 noise_10 is 19.50, 20.41 for noise_20 and 64.75 for noise_30. On the Dublin dataset, also the increase 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 15 (a) presents an example of Temporal Anomaly Type II, and the original trajectory in Figure 15 (b). As one can see there are sparse points mostly at the bottom of Figure 15 (a) simulating less congestion.

Figure 16 (a) and Figure 16 (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

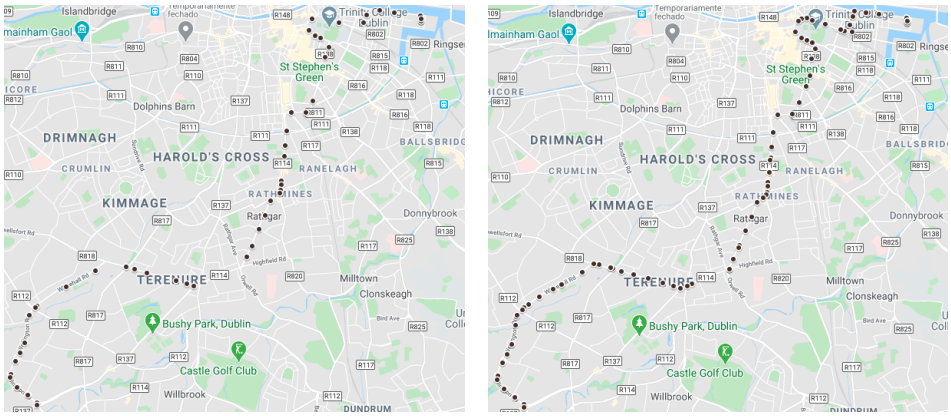
Figure 14 – Results of temporal anomaly type I



(a) Recife (b) Dublin.

Source: CRUZ; BARBOSA (2020)

Figure 15 – Example of temporal anomaly type II

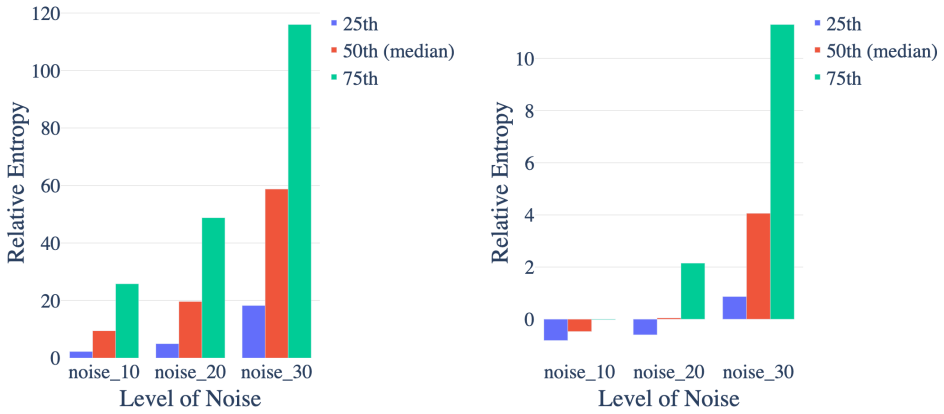


(a) Anomalous trajectory. (b) Raw trajectory.

Source: CRUZ; BARBOSA (2020)

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.

Figure 16 – Results of temporal anomaly type II



(a) Recife (b) Dublin.

Source: CRUZ; BARBOSA (2020)

4.5 CONCLUSION

In this chapter, we proposed a solution for detecting anomalous spatial-temporal bus trajectories. The solution relies on a deep learning multi-class classification model (STOD), where the anomaly score is based on the classifier's confidence, which is calculated by a normalized entropy. 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, in turn, 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 also performs well for this task. Finally, the evaluation of the anomaly score showed that, generally, the distribution of anomaly score of trajectories with spatial or temporal anomalies is higher than in non-anomalous trajectories. According to the results, we realized that PAC embeddings do not contribute to improving the STOD method, but the STOD has exciting results. So then, we finally answer our RQ1 (STOD) and RQ2 (PAC).

Although the STOD has competitive results, it can not perform the anomaly detection task online, which limits its use to work as a filter in batch processing. Also, the method does not highlight which are anomalous points/regions. Both restrictions are essential to help domain experts (transit authorities) to make real-time decisions. Then, in the next Section 5, we propose a method that not only detects if the whole trajectory is anomalous but also pinpoints which are the anomalous points. However, the approach only detects

spatial anomalies since we did not have enough time to implement the temporal feature.

5 APPLYING A TRANSFORMER LANGUAGE MODEL FOR ANOMALY DETECTION IN BUS TRAJECTORIES

In this section, we aim to answer the third request question RQ3: *Using an end-to-end solution, how can we detect spatially anomalous trajectory segments without anomalous labels?*. For this, we propose an approach that uses language modeling, commonly used in Natural Language Processing tasks, to: (1) detect anomalous trajectories in bus trajectories, and (2) pinpoint the abnormal points in these trajectories (sub-trajectory anomaly detection). Our solution allows these tasks to be performed either offline or online, i.e., as the buses are moving along their route. In addition, it can be adapted to other types of trajectories (e.g., car, people, and vessels), since our model does not use any specific aspect from the bus domain (e.g., bus stops).

The key idea of our model is to learn the language of well-formed trajectories, and then identify erroneous (ill-formed) trajectory and also the trajectories' points where the errors occur. For that, given an input trajectory T mapped by our solution to a sequence of tokens, our language model (LM) generates $\hat{T}$, the most likely (common) sequence of points for T, which represents T supposedly without anomalies. Our assumption is therefore that, since anomalous points are typically few and different from the others (CHEN et al., 2013), there is a small chance that abnormal points are present in $\hat{T}$. Based on that, our solution produces an anomaly score for T and pinpoints anomalous regions in T by comparing T with $\hat{T}$.

The rest of this chapter is organized as follows. In Section 5.1, we present the problem formulation. Section 5.2 delineates our model, and Section 5.3 describes the datasets and the setup experimentation. We compare our results with the state-of-the-art algorithms and previous research in Section 5.4. Finally, the conclusion and future work are drawn in Section 5.5.

5.1 PROBLEM FORMULATION

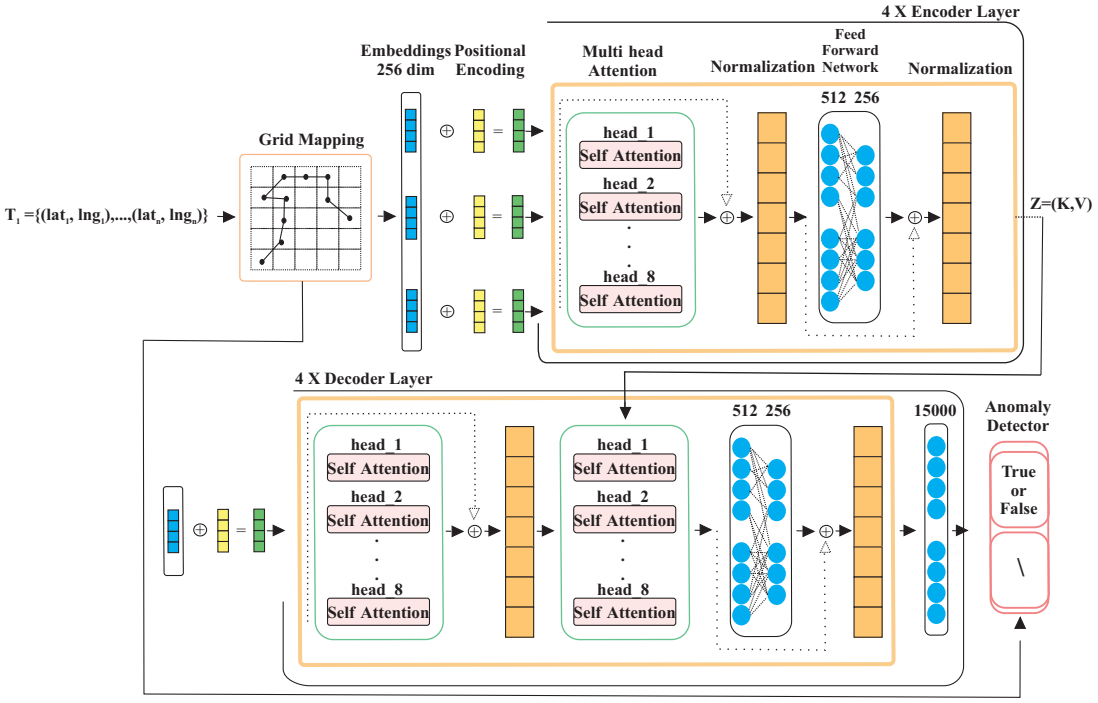
In this section, we provide some background concepts and state the problem that we deal with in this work.

Definition 2. ***[Problem Statement]****. Given a bus trajectory T, calculate the spatial anomaly score of T and detect the anomalous points of T.*

5.2 METHOD

In this section, we introduce our approach to discover abnormal trajectories and localize anomaly regions in trajectories. As Figure 17 shows, our solution is composed of three main components: Grid Mapping, Transformer Language Model and Anomaly Detector.

Figure 17 – The trajectory anomaly detection solution proposed in this work



Source: Created by author (2022)

They work as follows. Given the input bus trajectory, the Grid Mapping discretizes it by mapping each of its points to a geographical grid cell, represented by a token, generating a sequence of grid cell tokens. This sequence is passed to a language model, a deep generative encoder-decoder Transformer, that produces a series of predicted grid cell tokens. Finally, the Anomaly Detector compares the original token sequence with the predicted one to calculate the trajectory's anomaly score and to identify anomalous points, if they exist. In the remaining of this section, we provide further details about each one of these components.

5.2.1 Grid Mapping

The first step of our approach, Grid Mapping, maps the trajectories, which are a multivariate time series, into a univariate sequence of tokens. For this purpose, we use the H3 (Hexagonal Hierarchical Geospatial Indexing System)[1] library to create a grid system based on hexagonal cells and a hierarchical index. More specifically, given a raw trajectory $T = (p_1, p_2, .., p_n)$, where p_i is each trajectory point, represented by its latitude and longitude $(lat, long)$, and a grid cell G, we use the geoToH3 function in the H3 library to map points of trajectories into grid cell locations c_i, generating $T' = (c_i, c_2, ..., c_n)$. As a result, every point that falls into the same cell has the same identifier (token). Therefore, this mapping reduces the complexity of dealing with a multi-dimensional domain to a uni-dimensional one, which language models can properly process.

[1] https://eng.uber.com/h3/

5.2.2 Transformer Encoder

The first component of our language model is the Transformer Encoder that maps the sequence of trajectory tokens T' into a set of vectors that feeds the Transformer Decoder. For that, it encodes the tokens in T' based on the other tokens (points) in T' by applying the self-attention strategy.

More concretely, as Figure 17 depicts, the encoder first generates a sequence of embeddings from the trajectory tokens. An embedding is a vector representation of a token in an n-dimension space[2]. Next, a positional encoding adds position information to the embeddings. Similar to (VASWANI et al., 2017), our position encoder is calculated as:

$$PE(pos, 2_i) = sin(pos/1000^{2_i/d_{model}}) \tag{5.1}$$

$$PE(pos, 2_{i+1}) = cos(pos/1000^{2_i/d_{model}}) \tag{5.2}$$

where *sin* and *cos* are the trigonometric functions sine and cosine respectively, *pos* is the position of the point in the trajectory, and d_{model} is the dimension in the embedding vector. To create the final embedding representation for each input token, the model performs an element-wise addition of the token embedding with the positional encoding vector.

These embeddings are then passed to the Transformer block with four identical encoder layers, which experimentally worked well. The model uses the so-called multi-head self-attention to allow the network to attend different input sequence positions and learn which points in the sequence are relevant to the current one. Multiple heads create multiple representation subspaces to learn a set of queries Q and keys K of dimension d_k, and values V dimension d_v weight matrices. Each head computes the attention weights with respect to a given token embedding j ($Embedding_j$) as follows:

$$Q_{i,j} = W_i^Q.Embedding_j \tag{5.3}$$

$$K_{i,j} = W_i^K.Embedding_j \tag{5.4}$$

$$V_{i,j} = W_i^V.Embedding_j \tag{5.5}$$

$$Head = softmax(\frac{Q_i K_i}{\sqrt{d_k}})V_i \tag{5.6}$$

with parameters matrices $W^Q \in \mathbb{R}^{d_{model} X d_k}$, $W^K \in \mathbb{R}^{d_{model} X d_k}$, and $W^V \in \mathbb{R}^{d_{model} X d_v}$.

The multi-head attention combines the individual heads as follows:

$$Multi\_Head = concat(Head_1, Head_2, ..., Head_n)W^O \tag{5.7}$$

[2] We use embeddings of 256 dimensions in our solution as default.

where $W^O \in \mathbb{R}^{hd_v X d_v}$ is a weight matrix learned during training.

On top of the multi-head attention, there are two skip connections, two normalization layers interspersed with fully connected feed-forward networks. The residual connection helps the encoder to combine features from different layers, merging different levels of representations (HUANG et al., 2017). The normalization layers standardize the residual connection and the feed-forward outputs, giving numerical stability to the model. They are calculated as follows:

$$\bar{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{5.8}$$

$$z_i = \gamma.\bar{x} + \beta \tag{5.9}$$

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. Note that x_i is the concatenation between $Multi\_head$ vector and positional Embeddings (skip connection) and the normalization vector along with feed-forward output as shown in Figure 17.

Lastly, the feed-forward network has two layers on the top of the encoder. Their goal is to project the normalization of the multi-head attention to another dimension space and add non-linearities between them.

The encoder's final output is the matrix $Z = (K, V)$, where K are key vectors and V value vectors of the tokens in the input sentence. This matrix is used by the decoder to focus on appropriate tokens in the input sentence to generate the predicted sequence.

5.2.3 Transformer Decoder

The Transformer Decoder is the second component of our language model. Its goal is to produce a grid cell token sequence from the input sentence encoded by the Transformer Encoder. For that, it uses the auto-regressive method, i.e., it predicts each token in the sequence based on the previous ones produced by the model.

Similar to the encoder, the decoder is also composed of Transformer blocks. The decoder self-attention works, however, in a slightly different way. While the self-attention in the encoder considers all tokens from the trajectory to generate the attention weights, the decoder only considers tokens preceding the current one to predict the next. For that, Transformers mask future positions by using the look-ahead mask approach (VASWANI et al., 2017).

In addition, the first decoder multi-head attention layer learns a query matrix Q_{dec} from the previously predicted tokens. For that, first the decoder receives the output from the previous layer (embeddings). Then, similar to the encoder, the decoder augments it with a positional embedding layer and feeds it to multi-head attention to generate Q_{dec}.

After, the query vector and the residual connection feed a normalization layer similar to the decoder. Next, a second multi-head attention layer receives the learned decoder Q_{dec} and the matrix $Z = (K, V)$ from the encoder output to guide the query/search process. This second multi-head layer allows the decoder to focus on which trajectory points from the encoder are relevant to predict the next token/point. After the second multi-head layer generates the encoder-decoder attention vector, the decoder passes it to a feed-forward layer, followed by another normalization to add non-linearities and stability to the values.

Finally, the last layer implements a feed-forward neural network that projects the decoder vectors in a large dimension to represent the logit vector[3]. Each logit represents a token/cell score, which is turned into a probability by the softmax function. In our decoding strategy, the model outputs the highest probability token for each position, generating the predicted sequence $\hat{T}$.

5.2.4 Training

To train our model, we use the sparse categorical cross-entropy loss [4] since the labels are integers. The loss is described as:

$$L(y, \hat{y}) = -\sum_{j=0}^{M}\sum_{i=0}^{N}(y_{ij}.log(\hat{y}_{ij})) \tag{5.10}$$

where y_{ij} is the target, and $\hat{y}_{ij}$ represents the prediction. To train the model, we use the Adam optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.9$, and $\epsilon = 1e^{-9}$) with a flexible learning rate that increases at the beginning of training and decreases slowly in the remaining training steps conform (VASWANI et al., 2017). We also apply residual dropout with a rate of 0.1 for each layer in the encoder and decoder.

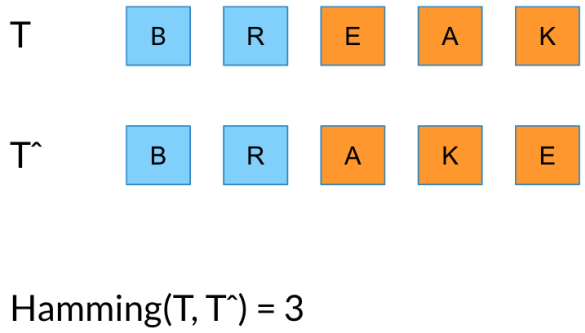
It is worth mentioning that our approach learns to generate the input trajectory, then input and targets are the same for training. In addition, during training, we use the teacher-forcing, i.e., we pass the true output to each successive step in the decoder. Finally, in the inference step, we provide the input to the encoder and a starting token to the decoder that outputs prediction one token at a time.

5.2.5 Anomaly Detector

Given the encoder's input token sequence and the decoder's output, as aforementioned, our solution produces two outputs for a given bus trajectory: its anomaly score and the regions where the anomaly occurs in the trajectory. Our main assumption is that our trained language model predicts the correct sequence, and any token in the input sequence (trajectory) that diverges from the predicted ones are considered anomaly.

[3] The logit vector dimension is the total number of grid cell tokens (vocabulary).
[4] https://www.tensorflow.org/api_docs/python/tf/keras/losses/SparseCategoricalCrossentropy

Figure 18 – Hamming distance



Source: Created by author (2022)

Thus, to calculate the trajectory's anomaly score, the detector compares the sentence predicted by the decoder with the encoder's input sentence by aligning them and computing their Hamming distance (LUU et al., 2020), as follows:

$$score = 1 - \frac{Hamming(T, \hat{T})}{n}, \tag{5.11}$$

where $\hat{T}$ is the decoder's predicted sequence, T is the language model input sequence, and n represents their size. Figure18 shows how Hamming distance works using words.

We consider the anomalous regions in the input sequence T the trajectory points represented by the unmatched tokens between T and $\hat{T}$.

5.3 DATA DESCRIPTION AND SETUP

5.3.1 Experimental Setup

In this section, we provide details about the setup of our experimental evaluation.

Datasets. We conducted our experiments in two real-world bus trajectory datasets. The first dataset is from Recife, Brazil. It comprises 19,290 trajectories (100 points on average) from 82 bus lines generated by 238 buses from October 2017 to November 2017. Each bus reports points at intervals of 30 seconds, and each point contains longitude, latitude, timestamp, route id, vehicle id, instantaneous velocity, and travel distance (from the beginning of the trip). The second dataset is from Dublin[5], Ireland. It contains 60,084 trajectories (206 points on average) and 68 bus lines. Each trajectory point is reported between 20 and 50 seconds. In total, 12,497.472 points were collected during Jan 01 2013 to Jan 04 2013. Each point contains the attributes: latitude, longitude, timestamp, line id, journey id, and vehicle id.

Pre-processing. As mentioned in Section 5.2.1, we map the trajectories into a geographical grid. Table 11 presents the statistics of trajectories before and after the grid-mapping transformation using the H3 parameter resolution 10 [6] (16 is the maximum resolution),

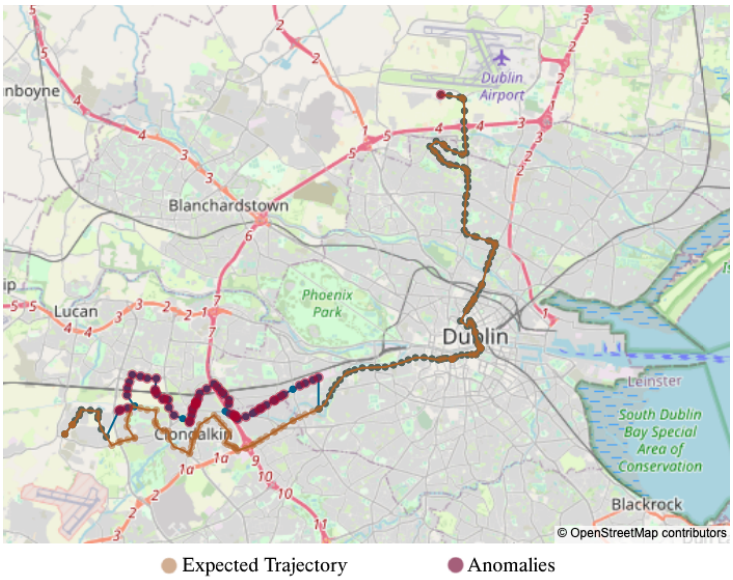
[5] https://data.gov.ie/dataset/dublin-bus-gps-sample-data-from-dublin-city-council-insight-project
[6] https://h3geo.org/docs/core-library/restable/

Table 11 – Number of unique points before and after the Grid Mapping

	Recife	Dublin
	Unique Points	Unique Points
Before Mapping	1,929,000	12,497,472
After Mapping	2,416	6,401

Source: Created by author (2022)

Figure 19 – Example of synthetic anomaly



Source: Created by author (2022)

which was chosen by experimentation[7]. As can be seen, this transformation provides a great reduction of the dimensionality on both datasets.

Ground Truth. Since there is no label available in our datasets, one can try manually label anomalies as (ZHANG et al., 2011; WANG et al., 2020) or generate artificial ones as (ZHENG et al., 2017; LIU; NI; KRISHNAN, 2013; LI et al., 2018). We chose to generate synthetic anomalies, as manual labels are time-consuming, by adding some perturbation in the real trajectories. We do so by randomly choosing the first point in a real trajectory t and shifting it along with the next n points in t sequentially. Figure 19 presents an example of a synthetic anomalous trajectory created by this method. In the experiments, we use two parameters to create different types of anomaly trajectories from real ones: d (the distance in kilometers from the real points) and p (the percentage of shifted points). For example, using $d = 0.5$ and $p = 0.1$, 10% of trajectory points is moved 500 meters from the real point. We generate anomalous trajectories for our experiments considering the values of $p = [0.1, 0.2, 03]$ and $d = 1.0$. Note that we generate synthetic anomalies different from those shown in Chapter 4, where we applied unrealistic synthetic spatial anomalies since the anomalous points are spread without a well-defined route shape.

[7] The resolution allows the library to increase or decrease the size of grid cells. Thus, the higher resolution, the smaller the cell areas.

Table 12 – Values of hyper-parameters of Transformer

Hyper-parameters	Values
Num Heads	8
Embedding	250
Num Layers	4
Beta_1	0.9
Beta_2	0.98
Epsilon	1e-9
Optimizer	Adam
Dropout	0.1

Source: Created by author (2022)

Baselines. We evaluate the following anomaly detection methods in our experiments:

- **RioBusData** (BESSA et al., 2015). This is a supervised method to detect bus anomalous trajectories by classifying them in bus routes. It uses a Convolutional Neural Network (CNN) fed by raw bus trajectories. On the top of CNN, a softmax function outputs a vector of probability where each value is the probability of class membership for each route/label. A trajectory is considered abnormal if its highest class probability is below a threshold.

- **STOD** (CRUZ; BARBOSA, 2020). This is also a supervised method that detects bus anomalous trajectories. It learns to classify bus trajectories in their routes using a deep-learning network. The model outputs the routes' class distribution of a given trajectory. From this distribution, it calculates the uncertainty of the classifier using entropy as a measure of anomaly degree. The higher the classifier uncertainty, the higher the entropy. A trajectory is then considered anomalous if the entropy of the classifier's probability distribution output for it is higher than a threshold.

- **GM-VSAE** (LIU et al., 2020). This approach uses an encoder-decoder strategy to detect anomalous trajectories. To perform that, firstly, the encoder infers a disentangled latent space to discover the distribution of each trajectory based on this space. This distribution is then fed to the decoder that generates a trajectory. The method calculates a score comparing the generated trajectory with the input trajectory. A high score ≈ 1.0 means that the input trajectory has a high probability of being an anomaly.

- **iBOAT** (CHEN et al., 2013). This method is based on the isolation mechanism (LIU; TING; ZHOU, 2012) and an adaptive windows approach. It performs the detection of both trajectory and sub-trajectory anomaly detection. Overall, the iBOAT calculates the frequency of points mapped into a grid cell to isolate "few and different" points. Based on that, trajectories that visit cells with low frequency get small scores, meaning that those points are highly likely to be an anomaly. Conversely

Table 13 – Results of anomaly trajectory detection on the Dublin dataset

	p=0.1			p=0.2			p=0.3		
	F1	Rec	Prec	F1	Rec	Prec	F1	Rec	Prec
STOD	0.636	0.595	0.683	0.696	0.635	0.772	0.724	0.638	0.838
RioBusData	0.651	0.738	0.605	0.660	0.749	0.612	0.673	0.753	0.631
GMVSAE	0.682	0.624	0.758	0.708	0.628	0.827	0.713	0.611	0.876
iBOAT	0.668	0.566	0.833	0.673	0.557	0.871	0.684	0.563	0.889
Transformer	**0.840**	**0.776**	**0.930**	**0.853**	**0.768**	**0.972**	**0.854**	**0.763**	**0.979**

Source: Created by author (2022)

trajectories with high-frequent visited cells have high scores of non-anomaly. Similar to the previous methods, iBOAT also needs a threshold to detect anomalies.

- **Transformer**. This is our proposed approach. We train our model on both datasets with the hyper-parameter values shown in Table 12. The source code of our method is available on Github[8].

It is worth pointing out that all those methods identify anomalous trajectories, but only our approach (Transformer) and iBOAT detect anomalous sub-trajectories. We randomly selected 8,200 trajectories from the Recife dataset and 6,800 from Dublin to evaluate the approaches. Based on a data analysis, we defined the maximum number of points 100 for the Recife dataset and 208 for Dublin. To train and test the models, we use 80% and 20% of the trajectories, respectively. We implemented the neural-based approaches on Tensorflow 2.1.0 [9].

Evaluation Metrics. We use F1-measure, Precision, Recall, and PR-AUC as evaluation metrics, since they are usually applied to evaluate outlier detection methods(BELHADI et al., 2020; LIU et al., 2020). To verify whether the F1-measure values of our model are statistically different than the baselines, we execute the Wilcoxon statistical test (SIEGEL, 1956). The test verifies whether two paired samples (F1-measure values of our solution vs a baseline) come from the same distribution. Given that, we set the significance level $\alpha = 5\%$. In our context, the null hypothesis h_0 considers that the median difference between the F1 values of a pair of models is zero. This statistical test is performed on the instances in the test set.

5.4 RESULTS AND DISCUSSION

In this section, we first present the evaluation of the trajectory outlier identification and, subsequently, the region anomaly detection.

[8] https://github.com/michaeloc/its_research
[9] https://www.tensorflow.org/

Table 14 – Results of anomaly trajectory detection on the Recife dataset

	p=0.1			p=0.2			p=0.3		
	F1	Rec	Prec	F1	Rec	Prec	F1	Rec	Prec
STOD	0.589	0.552	0.634	0.630	0.576	0.703	0.660	0.590	0.760
RioBusData	0.533	0.549	0.524	0.537	0.551	0.529	0.545	0.555	0.544
GMVSAE	0.654	**0.590**	0.748	**0.673**	**0.594**	0.797	**0.682**	**0.597**	0.819
iBOAT	**0.668**	0.533	0.915	0.668	0.528	0.928	0.673	0.534	0.932
Transformer	0.665	0.520	**0.948**	0.669	0.518	**0.967**	0.670	0.518	**0.970**

Source: Created by author (2022)

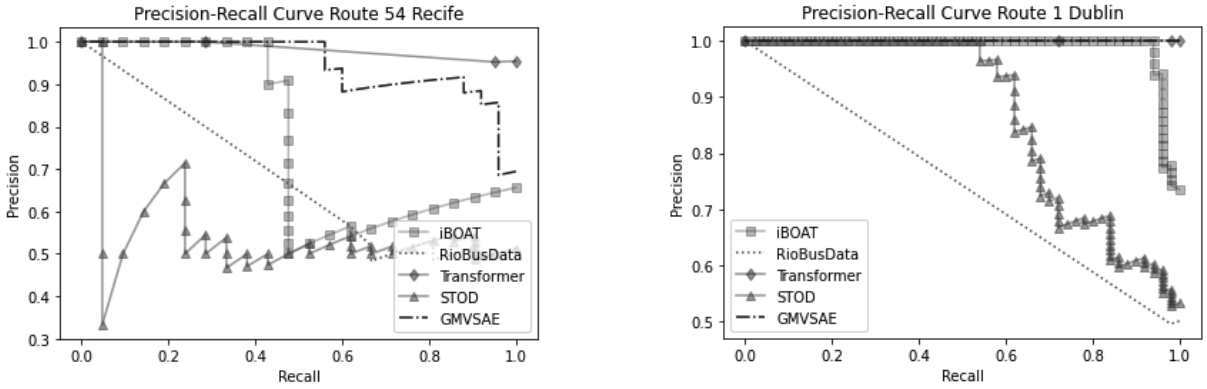
5.4.1 Trajectory Anomaly Detection

Table 13 presents the results for the trajectory anomaly detection task on the Dublin dataset. Transformer outperforms the baselines in all scenarios and metrics. For example, considering the best results on F1, our approach is at least 17% better than all baselines. To confirm this, Table 16 depicts the p-values for the results of each baseline in comparison to Transformer on the Dublin dataset. All the p-values are smaller than the significance level (0.05), which supports that our Transformer network has in fact a superior performance than the baselines on this dataset.

Regarding the results on the Recife dataset, presented in Table 14, our method obtains better F1-measure values than STOD and RioBusData, and comparable ones with GMVSAE and iBOAT. In fact, the p-values of the hypothesis tests on this dataset for F1-measure, Table 17, confirm this: the p-values of GMVSAE and iBOAT versus Transformer in all scenarios are higher than the significance level of 0.05, meaning there is no statistical difference in terms of F1-measure between our approach and them. With respect to RioBusData and STOD, however, the p-values are lower than 0.05 in two out of the three anomaly cases. Looking at precision values, Transformer achieved the best overall results, but lower recall than GMVSAE and iBOAT. In practice, having better precision can be an advantage, since an anomaly detection model can be thought of a filter that identifies possibly few anomalous trajectories in a large set of trajectories. The more precise this filter is, the few false negative anomalies need to be inspected.

Overall, the methods built to directly detect anomalies (Transformer, GMVSAE and iBOAT) outperformed in almost all scenarios the ones that try to do this in an indirect way (STOD and RioBusData), i.e., learning to classify routes instead of anomalies. For example, for $p = 0.1$, RioBusData obtained the lowest F1 (0.651) on Dublin and Recife (0.533). However, on the Dublin dataset, STOD shows better F1 results than iBOAT for $p = 0.2$ (0.69 vs 0.673 respectively), and for $p = 0.3$ obtained the F1 second best value (0.724) only behind Transformer (0.854). We also observed that STOD is more sensible than our method over the percentage outlier variation p. For example, for $p = 0.1$, and $p = 0.3$, the F1-measure is 0.58 and 0.66 on Recife (difference of 0.8), respectively. In contrast, our method is more stable regarding the outlier level. For instance, for $p = 0.1$ and $p = 0.3$, the F1-measure is respectively 0.84 and 0.85 on Dublin and 0.66 and 0.67

Figure 20 – PR-AUC of the approaches for route 1 on the Dublin dataset and route 54 on the Recife dataset



Source: Created by author (2022)

on Recife, i.e., there is not much difference.

Table 15 – Results for the region anomaly detection models

	p=0.1				p=0.2				p=0.3			
Recife												
	F1	Prec	Rec	p-value F1	F1	Prec	Rec	p-value F1	F1	Prec	Rec	p-value F1
iBOAT	0.911	0.845	0.990	**7.70e-10**	0.905	0.841	0.980	**2.13e-8**	0.901	0.839	0.973	**1.91e-7**
Transformer	0.989	0.988	0.992		0.983	0.975	0.992		0.975	0.961	0.991	
Dublin												
iBOAT	0.978	0.969	0.988	**0.12**	0.970	0.966	0.976	**0.65**	0.963	0.964	0.963	**0.10**
Transformer	0.986	0.979	0.995		0.975	0.957	0.995		0.962	0.932	0.994	

Source: Created by author (2022)

To provide a detailed analysis of the approaches on individual routes, Figure 20 shows the PR-AUC curves of all methods on both datasets in two different routes, one from each dataset with $p = 0.3$. In route 54 from Recife, our approach has the highest area under curve ≈ 0.98, outperforming both the unsupervised methods (GMVSAE ≈ 0.94 and iBOAT ≈ 0.77) and the supervised ones (STOD ≈ 0.54 and RioBusData ≈ 0.67). Looking at route 1 from Dublin, we observe that the encoder-decoder methods have almost the perfect curve AUC ≈ 0.99, i.e., the models can adequately distinguish anomalous trajectories from non-anomalous ones. Conversely, the RioBusData has the worst PR-AUC curve ≈ 0.70. Finally, we can see that the precision of iBOAT degrades with a recall close to 1.

5.4.2 Region Anomaly Detection

Table 15 shows the results between Transformer and iBOAT on region anomaly detection. We observe that Transformer outperforms iBOAT on the Recife dataset in all scenarios. This occurs mainly because our model achieved the high values of precision

Table 16 – Hypothesis test for F1 on the Dublin dataset

	Transformer		
	p=0.1	p=0.2	p=0.3
	p-value	p-value	p-value
STOD	4.05e-12	1.77e-12	1.09e-12
RioBusData	1.48e-12	1.04e-12	8.73e-13
GMVSAE	4.472e-12	4.05e-12	3.26e-12
iBOAT	1.22e-12	7.98e-13	1.13e-12

Source: Created by author (2022)

Table 17 – Hypothesis test for F1 on the Recife dataset

	Transformer		
	p=0.1	p=0.2	p=0.3
	p-value	p-value	p-value
STOD	0.38	2.11e-4	4.26e-9
RioBusData	1.13e-11	5.61e-13	8.12e-14
GMVSAE	0.21	0.68	0.46
iBOAT	0.73	0.80	0.73

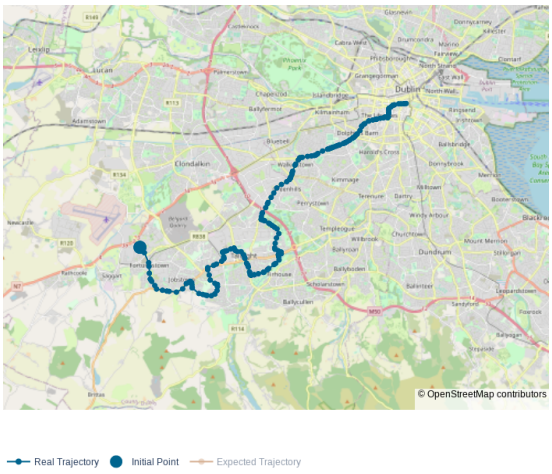
Source: Created by author (2022)

$(0.988, 0.975, 0.961)$. The methods, however, are similar regarding recall for $p = 0.1$, for instance, Transformer's recall is 0.992, and iBoat 0.990. On the Dublin dataset, the methods are qualitatively similar. Note that the most difference between the models occurs for $p = 0.1$: Transformer's F1 is 0.986 and iBOAT's F1 is 0.978.

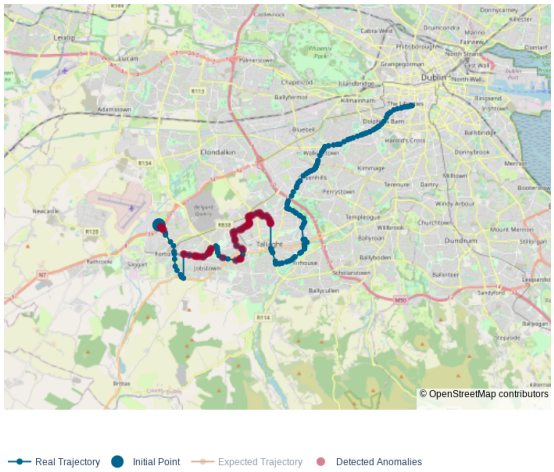
We applied the Wilcoxon test to verify whether there is a statistical difference between the F1-measure values of the methods on this task. On the Recife dataset the models are statistically different with p-value lower than our significance level of 0.05 for all scenarios. On the Dublin dataset, however, there is no statistical evidence to reject the h_0, since the p-values are higher than 0.05 and, therefore, both models are statistically equivalent in terms of F-1 measure.

To present concrete examples of the detection of region anomalies on real trajectories by our model, Figure 21a shows the expected trajectories of Dublin route 1, and Figure 21b depicts the anomalous regions identified (represented by the red dots) by Transformer. One can clearly see by these plots that our approach identifies very precisely the anomalous regions in this trajectory.

Figure 21 – Detection on real-world trajectories



(a) Expected trajectory.



(b) Anomalous regions (represented by the red dots) detected by our model.

Source: Created by author (2022)

5.5 CONCLUSION

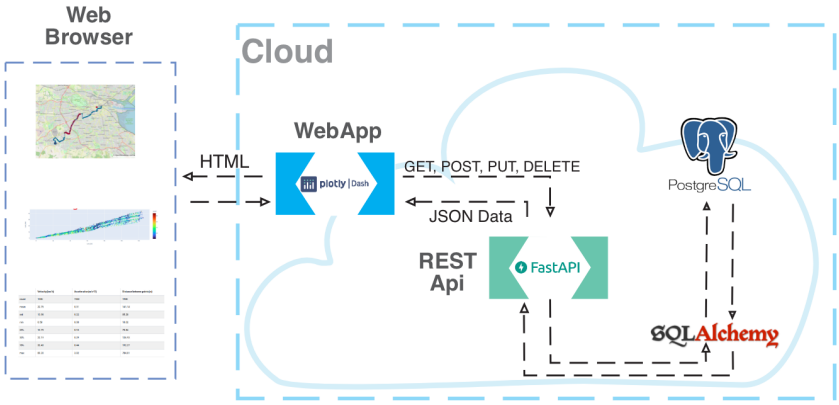
In this chapter, we propose a solution that applies a encoder-decoder transformer language model in bus trajectory data to solve two problems: trajectory and sub-trajectory anomaly detection. For that, our solution transforms a trajectory into a discrete token sequence by mapping its points to tokens that represent geographical grid cells. This sequence is then passed to the Transformer language model that outputs a predicted sequence, supposedly without anomalies. Also, our solution calculates the trajectory's anomaly score by applying the hamming distance between the two sequences, and identifies the anomalous regions by looking at the unmatched tokens between them. Experiments in two real-world bus trajectory datasets demonstrate that our approach is effective for anomalous trajectory detection and anomalous region detection tasks.

In the next Chapter 6 we present a demo application called MobApp. It is a data-driven open-source web application developed to help domain analysts and practitioners/scientists to analyze real-world trajectories. The application is not a thesis contribution with a respective request question. However, we developed it because of two main motivations. First, we must have implemented a few baseline methods since we did not find literature code implementation, which limited our solution comparison. Second, we provided a simulation of our methods in a real application, i.e., in production. Therefore, the MobApp have born to allow other researchers to add their anomaly detection results and datasets to help the research community compare their methods to others and save implementation time.

6 MOBAPP: A DATA VISUALIZATION TOOL FOR TRAJECTORY ANALYSIS

In this section, we present MobApp[1] (NETO, 2021) as part of this thesis. Here, we focus on describing the application features fed by datasets and experimentation results generated by previous Chapters 4, 5. MobApp supports three different visual analyses. The first one allows analysts to visualize trajectories on a map in a specific period along with plots showing information about the trajectory points (e.g., speed and acceleration). In the second analysis, the user can observe anomalous trajectories and also the region where the anomaly occurs, detected by an anomaly detection algorithm. The last one allows users to visually compare on the map the output of different anomaly detection models on the same trajectory. Different users might be interested in such functions. For instance, data scientists might use MobApp to inspect trajectories, looking for issues in the data or the result of anomaly detection models, which is helpful for model diagnostics(BUJA et al., 2009). Likewise, transit authorities can use MobApp to automatically identify anomalous bus trajectories, indicating traffic congestion or driver misconduct.

Figure 22 – MobApp Architecture



Source: Created by author (2022)

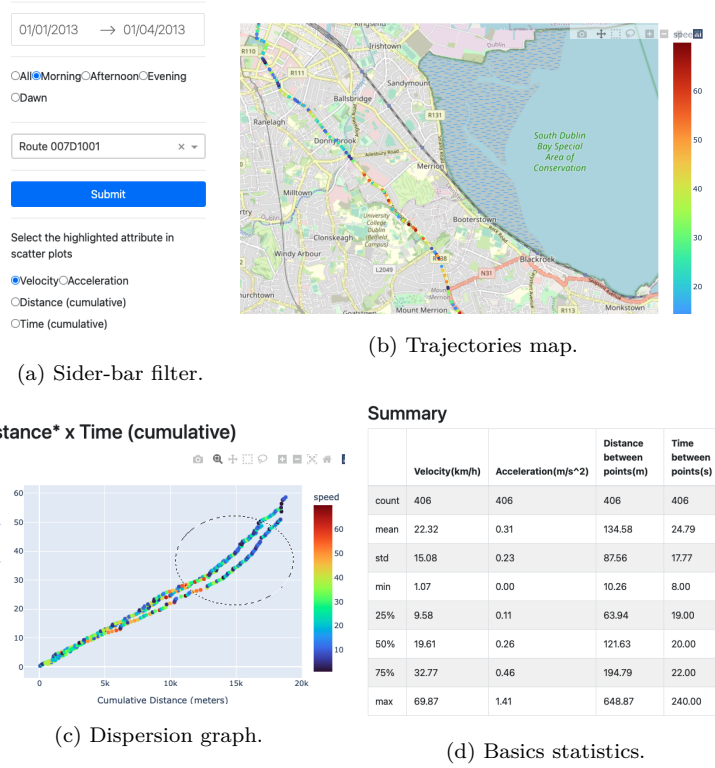
The MobApp application uses a dataset from the city of Dublin[2] with 1,699.022 points collected from Jan 01, 2013, to Jan 07, 2013. In this Chapter, we show the following use cases:

- An exploratory data analysis based upon speed and days of the week;
- Insights from the results of the anomaly detection visualization;
- A visual comparison of different anomaly detection models.

[1] https://github.com/Mobapp-Dashboard
[2] https://data.gov.ie/dataset/dublin-bus-gps-sample-data-from-dublin-city-council-insight-project

The remainder of this Chapter is organized as follows. Section 6.1 presents the MobApp architecture: WebApp and API. In Section 6.2, we provide three user cases that demonstrate how MobApp can be used. Finally, the conclusion and future work are drawn in Section 6.3.

Figure 23 – WebApp Exploratory Data Analysis



(a) Sider-bar filter.

(b) Trajectories map.

(c) Dispersion graph.

	Velocity(km/h)	Acceleration(m/s^2)	Distance between points(m)	Time between points(s)
count	406	406	406	406
mean	22.32	0.31	134.58	24.79
std	15.08	0.23	87.56	17.77
min	1.07	0.00	10.26	8.00
25%	9.58	0.11	63.94	19.00
50%	19.61	0.26	121.63	20.00
75%	32.77	0.46	194.79	22.00
max	69.87	1.41	648.87	240.00

(d) Basics statistics.

Source: Created by author (2022)

6.1 MOBAPP ARCHITECTURE

Figure 22 shows a high-level architecture of MobApp. The application comprises two modules: (i) WebApp and (ii) REST API. The first module, the WebApp, is the system's user interface (UI), which interacts with the REST API to provide end-user functionalities. The second module, the API, receives client requests and returns the selected analysis results.

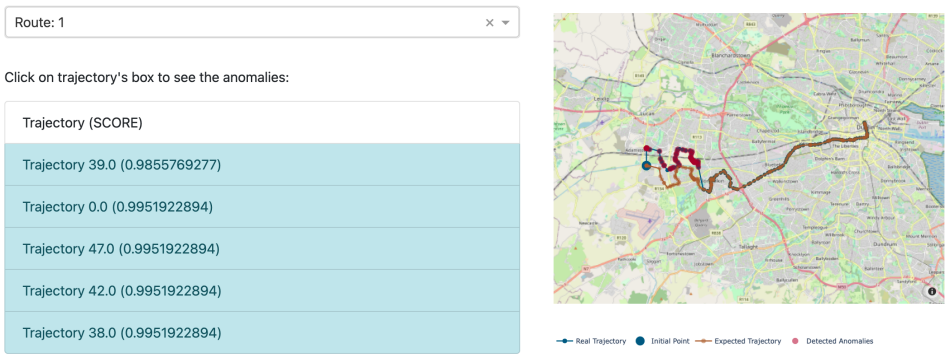
6.1.1 WebApp

Our WebApp module is the application's front-end. It was developed using the library Dash[3], which is a Web-based Python UI framework for data science apps. The MobApp presents three screens: (i) Exploratory Data Analysis (EDA), (ii) Anomaly Trajectory Filter, and (iii) Model Evaluation.

[3] https://plotly.com/dash/

Exploratory Data Analysis (EDA). The EDA screen shows general analyses of the trajectories. Using this functionality, users can: (1) apply filters to select trajectories in the sidebar to select trajectories based on period of time (Figure 23a); (2) visualize on a map the chosen trajectories (Figure 23b), a cumulative scatter plot (Figure 23c), and a summarization table (Figure 23d) with statistics about speed, acceleration, distance and time between consecutive points.

Figure 24 – WebApp Anomaly Scores



(a) Top-5 anomaly trajectories

(b) Anomaly region

Source: Created by author (2022)

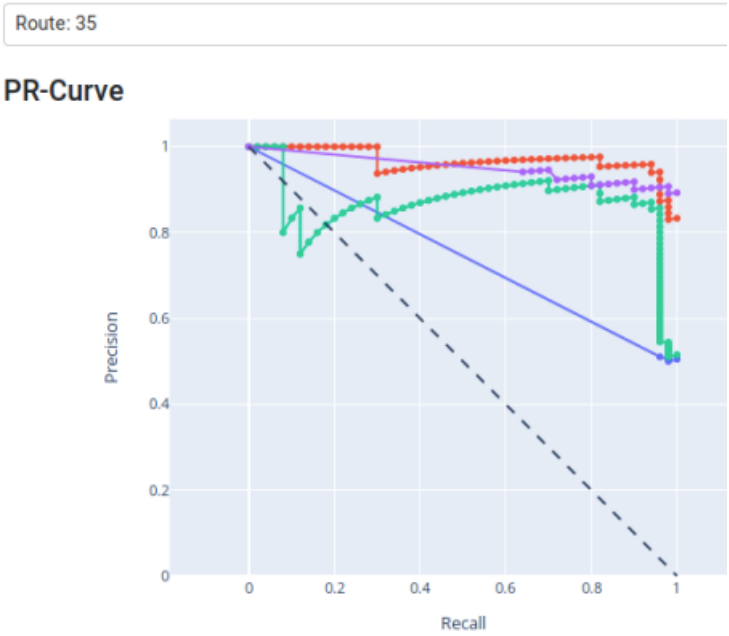
Anomaly Trajectory Filter. This screen allows users to visualize anomalous trajectories for a given route [4]. As shown in Figure 24, it returns the top-k anomalous trajectories (id trajectory and score) according to a pre-defined anomaly detection model (Figure 24a). Note that the application allows users to choose a route and click on each returned trajectory to analyze its behavior. For that, MobApp shows the anomaly region (red points in Figure 24b) on the map, also detected by an anomaly detection model, which helps users to visually localize where the anomaly likely occurred. In addition, the tool also shows the expected trajectory of the bus (yellow dots in Figure 24b), if this information is provided.

Model Evaluation. This screen provides visual comparison between anomaly detection models. It is divided into two parts: route and trajectory. The route screen allows users to compare the performance of anomaly detection models on trajectories in the same route. For this purpose, the application provides a drop-down component for users to select routes and visualize precision and recall curves, a well-known plot for evaluating unbalanced classification problems as outlier detection. Figure 25 shows this screen and the four available state-of-art algorithms used in the MobApp application: GMVSAE (LIU et al., 2020), Transformer [5], RioBusData (BESSA et al., 2015) and iBOAT (CHEN et al., 2013). GMVSAE is an autoencoder method to detect spatial anomaly trajectories by

[4] A *route* can be defined as a set of road segments (LIU et al., 2017), whereby journeys in this route generate trajectories on different periods of time.

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

Figure 25 – Precision-Recall curve of different anomaly detection strategies in a given route



Source: Created by author (2022)

disentangling latent space to discover each trajectory's distribution and recreating the input trajectory. In the same direction, the Transformer is an encoder-decoder language model that applies the self-attention mechanism to encoder trajectories and decoder them. Similar to GMVSAE, the Transformer learns to recreate the input trajectories without anomalies. The RioBusData approach, in turn, is a supervised method that learns to classify trajectories in their routes and detects anomalous trajectories by comparing the highest class probability value and a given threshold. On the other hand, the iBOAT is a method based on the isolation mechanism(LIU; TING; ZHOU, 2012) and an adaptive window approach. Shortly, it calculates the frequency of points in a grid cell to isolate "few and different" trajectories.

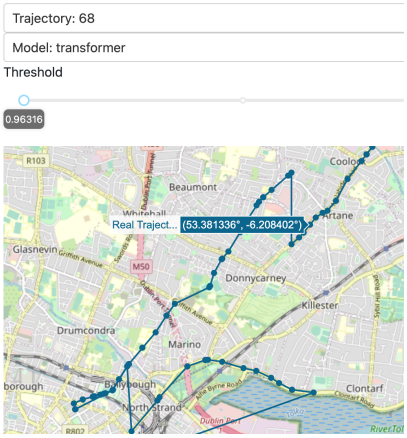
The other screen in this module allows users to inspect a particular trajectory based on an anomaly detection algorithm and a given anomaly score threshold. Figure 26 depicts a map with a selected trajectory and a summary table composed of the anomaly score of this trajectory according to the model, a defined threshold, the precision and recall values of the route which the trajectory belongs to, and an indication whether the trajectory is anomalous or not, regarding the chosen threshold.

6.1.2 REST API

MobApp's backend has a REST API that provides endpoints for the analysis. It is implemented using the Python FastAPI framework [6]. The API's data is stored in a Postgres [7]

[6] https://fastapi.tiangolo.com/
[7] https://www.postgresql.org/

Figure 26 – Detection on real-world trajectories



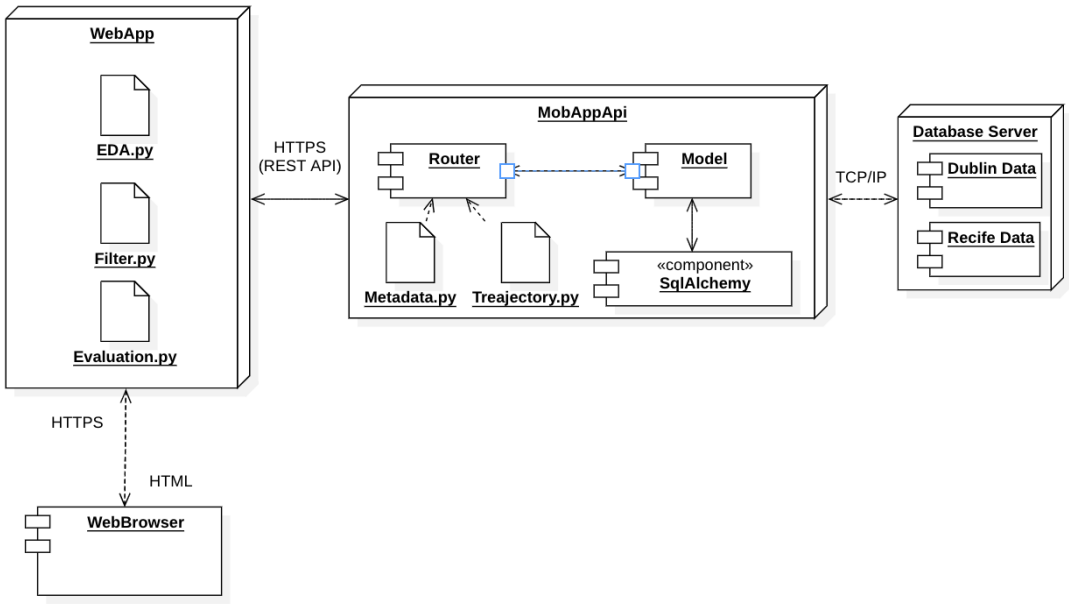
(a) Trajectory and Model selection.

Route	35
Trajectory	68
Model	transformer
Anomaly Direction	Below threshold
Threshold	0.9631578900000001
Precision / Recall (on route)	0.89285714 / 1.0
Score	0.8051947951
Result	Anomaly DETECTED

(b) Summary of model results.

Source: Created by author (2022)

Figure 27 – MobApp Architeture Deployment Diagram



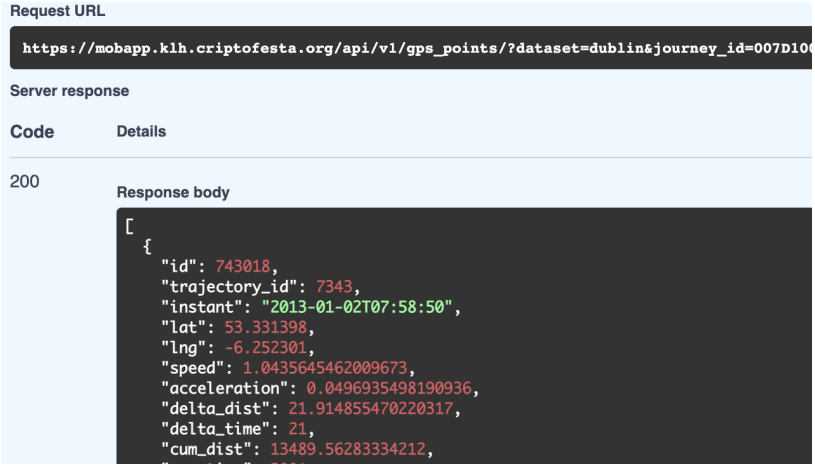
Source: Created by author (2022)

SGBD due to its support for spatial and geographic objects, i.e., geo-based queries and spatial index. Furthermore, the API makes the endpoints available in JavaScript Object Notation (JSON) format, allowing clients to consume the provided trajectory analyses. Specifically, as shown in Figure 27, the MobApp API implements the endpoints using a router component to answer the WebApp requests. For this, the router imports a few classes (Metadata.py and Trajectory.py) that can handle the Hypertext Transfer Protocol Secure (HTTPS) requests. Since the router uses those classes, it calls the model component

Figure 28 – Metadata end-point

Source: Created by author (2022)

Figure 29 – Trajectory end-point



Source: Created by author (2022)

to query the request information using the SQLAlchemy [8] component. The SQLAlchemy is a Object Relational Mapper (ORM) responsible for communicating with the Database Server by TPC/IP protocol and returning the model queries. Therefore, when the router has the retrieved information, it makes available a JSON to the WebApp, which handles the information retrieved by one of the classes (EDA.py, Filter.py, or Evaluation.py).

Currently, the API presents four endpoints. The first one is the Trajectory Metadata that returns trajectories' ids, given a start and end date as shown in Figure 28. This endpoint is essential to choose proper trajectories when users select a range of data on the Exploratory Data Analysis screen, as shown in Figure 23a. The second, the GPS endpoint, returns trajectories for a given route[9], date interval, and selected features on the side-bar (e.g., period of the day and speed), as shown in Figure 23. The returned information of this endpoint is used on the EDA screen, where users can observe basic statistics from

[8] https://www.sqlalchemy.org/

[9] A *route* can be defined as a set of road segments (LIU et al., 2017), whereby journeys in this route generate trajectories on different periods of time.

trajectories behavior. To illustrate, Figure 29 shows a request and the payload returning a list of trajectories and their attributes, respectively, such as timestamp, speed, acceleration, delta distance, delta time (time difference between two consecutive points), and cumulative distance. The third endpoint, the anomaly detection one, returns the top-k anomalous trajectories of a given route, according to a anomaly detection model, along with their respective scores and trajectories ids. It allows users to filter anomalous trajectories in descending order of scores. This endpoint is used by the WebApp in the Anomaly Trajectory Filter. Lastly, the model evaluation endpoint requests a route and the name of a anomaly detection method to return evaluation metrics namely recall, precision.

6.2 USE CASES

Here, we will be able to see how MobApp can be used from three use cases, one for each analysis the system provides.

Exploratory Analysis. Visualizing general statistics of the data is the first step to understanding its characteristics and distribution. Using MobApp, users can explore trajectory data to discover, for instance, which period of the day vehicles run slower than usual, the impact of known events (e.g., heavy rain) in the traffic flow, etc. To illustrate a real scenario, a user using the EDA screen selects the data period between 2013-01-01 and 2013-04-01 on the sider-bar, as shown in Figure 23a. Next, she chooses the morning option as a period of the day and the route 007D1001 along with the velocity filter. Based on these selections, the MobApp presents on the map the speed distribution of the points along the selected trajectories, as shown in Figure 23b. In addition, the application shows a dispersion graph that might indicate a traffic jam between 10k and 16k of trajectories' cumulative distance since their points have low speed, as shown in Figure 23c. Overall, the EDA functionalities can help MobApp users understand traffic behavior and provide information for better decision-making.

Bus Anomaly Trajectories. Identifying anomalous vehicle trajectories is essential for public transportation agencies to monitor and understand traffic behavior in order to implement better policies in a city. For example, consider an analyst from a department of public transportation who needs to analyze thousands of daily trajectories (trips) to identify anomaly trajectories to support decisions such as releasing or retaining buses regarding detected anomalies. Unfortunately, it is unfeasible to manually inspect a massive volume of trajectories, since they are multivariate temporal series with hundreds of points per trip. Considering this context, the analyst can use MobApp to rank anomalies trajectories by their anomaly level provided by a pre-built model. To do that, the user can use the Anomaly Detection screen and select which route she wants (by the selection input component) to filter the top-5 anomalous trajectories and analyze each one by the map on the right side. Figure 24a shows an example in which a user/analyst selected route 1, and the MobApp returned the most anomalous trajectories on the map, along with

the trajectory regions where the anomaly occurs (red points). Additionally, Figure 24b depicts the expected trajectory (yellow points) for that particular bus to help the analyst to get some insight by comparing with the actual trajectory.

Model Comparison. Comparison between anomaly detection models is essential in quantitative research. In this direction, a data scientist looking to compare trajectory anomaly detection models can use the MobApp application. For that, users can access the Model Evaluation screen and compare models based on a selected route through the Precision-Recall curve. In addition, on the same screen on the bottom, users can analyze a particular trajectory from the selected route using the models individually. For example, an analyst browses the Evaluation screen and selects route 35. MobApp application, in turn, returns the Precision-Recall curve of the approaches for this route, as shown in Figure 25. Observing these particular results, the analyst identifies that Transformer and GMVSAE methods have similar performance and outperform the other methods. On the other hand, RioBusData has the worst recall and precision values for trajectories in the chosen route. Second, at the bottom of the same screen, the user selects trajectory 68 along with the Transformer method to analyze results individual trajectories, Figure 26a and 26b show the results of this interaction.

6.3 CONCLUSIONS

In this chapter, we developed a visualization tool for three analyses: Exploratory Data Analysis, Anomaly Trajectory Filter, and Model Evaluation. The MobApp is a demo application that allows researchers and practitioners to analyze trajectories and compare literature model results. Overall, the tool is a possibility to allow users to save implementation time to compare their trajectory anomaly detection methods with others.

7 CONCLUSIONS AND FUTURE WORK

Anomaly detection task consists of detecting instances that stand out as dissimilar to all others. Finding anomaly patterns impacts several research areas, such as cyber-security, medical domain, and military surveillance. In Intelligent Transportation Systems, particularly in trajectory data mining, many works in literature propose solutions to detect anomalous trajectories. In this context, approaches in the literature can be split based on the operation mode and based on the type of output. For example, methods can operate offline as a batch method or work online, receiving input data almost real-time and outputting the results simultaneously. Also, some methods detect the whole trajectory as anomalous only and exist those that also pinpoint the anomalous region. This research thesis contributes on both fronts: we propose a supervised outlier detection using classification confidence as an anomaly score and an online end-to-end transformer language model for detecting anomalous trajectories and pinpointing the abnormal points.

In Chapter 4, we presented STOD as a supervised anomaly detection approach that generates an anomaly score based on classification confidence. Most of the methods in the literature are typically rule-based or unsupervised approaches, and they require much effort in preprocessing tasks and rely on domain experts to make parameter assumptions. Supervised approaches also exist, but they output a rigid anomaly score. Then, we propose the STOD, a multi-class supervised method to generate a flexible anomaly score to indicate if the whole trajectory is an anomaly or not. Our approach is particular to bus trajectories data since it learns to classify trajectories among route ids and uses the classification probability distribution to measure a level of confidence. Therefore, based on that confidence level, the whole trajectory can be classified as anomalous. Also, in this chapter, we tried to validate our assumption that the segmentation process based on bus stay points could improve anomaly detection methods. For that, we propose the PAC method to generate embeddings from bus stop activity labels. The embedding presented a visible group separability, but experiment results have shown that they did not contribute to the STOD results. Thus, in this chapter, our solutions answered the following research questions: RQ1 *how can we propose a supervised bus trajectory outlier detection method using classification confidence as an anomaly score?*, and RQ2 *Are there any advantages to representing trajectories by stay points on the outlier detection task?*. Although the STOD presented promising results and contributed to this thesis, it is very restricted to bus trajectories or pre-defined trajectories.

Chapter 5, in turn, presented a language model for anomaly detection in bus trajectories. We applied a well-known encoder-decoder Transformer method to learn well-formed trajectories, detect the ill-formed trajectories, and pinpoint which points are anomalous. Also, our solution can perform the anomaly detection task either online or offline. The

Language Model (LM) approach was validated on the bus trajectories, but it does not rely on trajectories route ids. Furthermore, our encoder-decoder method can be adapted to other types of trajectories since only raw trajectories are necessary to train the model. Note that feeding a trajectory T to our LM model, it generates $\hat{T}$ as the most likely (common) sequence of points for T, which represents T supposedly without anomalies. Lastly, our method generates an anomaly score and finds the abnormal points using the hamming distance between T and $\hat{T}$. Extensive experimentation show that our method outperformed a few state-of-art methods in some scenarios and had competitive results in others. Given that, we answered the following research question RQ3: *Using an end-to-end solution, how we can detect an anomalous trajectory segment without anomalous labels?*

Lastly, our Chapter 6 presented the MobApp: a data visualization tool for trajectory analysis. This tool was built with two goals: to facilitate comparison among trajectory outlier detection methods from literature and to show a real scenario to use those methods. First, providing implemented literature methods and public datasets to help other researchers to save time and focus on their solutions. Second, the tool has shown a real application scenario where outlier methods can be used to assist, for instance, public transport agencies' authorities.

The remaining sections of this chapter are organized as follows. Section 7.1 summarizes the main contribution of this thesis. Section 7.2, in turn, describes our failed attempts before reaching a solution for this thesis. Lastly, Section 7.3 discusses possible directions for future research.

7.1 THESIS CONTRIBUTIONS

In the following, we consider the main contribution of this thesis to be:

- A novel supervised approach called STOD to detect anomalous bus trajectories using classification confidence to generate an anomaly score. This classifier shows to be competitive with other baselines methods.

- A supervised approach called PAC to predict the type of activity GPS points in bus trajectory (bus stop, traffic light, and another kind of stop). The experimental results indicate that PAC produces high-quality embedding vectors, and it is effective for the task of stay point classification. However, experiments indicated that the PAC embeddings did not improve the STOD results.

- A language model encoder-decoder approach to detect anomalous bus trajectories and pinpoint the respective anomalous region. The approach can be applied offline and online and does not rely on hand-crafted features. The approach presented competitive results in the Recife dataset, and it surpassed the baselines in Dublin

dataset. Lastly, the results also indicate that the problem of trajectory anomaly detection can be well-modeled as a model language problem.

7.2 FAILED ATTEMPTS

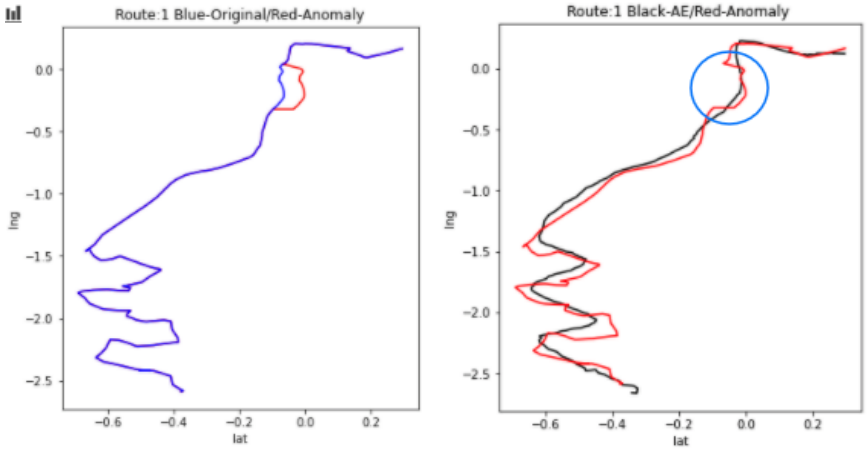
Throughout this thesis's development, we attempted several neural network approaches to build an efficient spatial anomaly detection model, and most of them were met with failure. Thus, this section describes these failures to help researchers/practitioners facing similar that work in this topic.

The first failed attempt to highlight was using explainability techniques such as the guided-backpropagation or Layer-Wise Relevance Propagation (LRP) technique (HOLZINGER et al., 2022; GU; YANG; TRESP, 2018) to detect anomalous segments in trajectories since those techniques have great results in detecting objects in images data. Then, we expected those techniques to be an efficient way to discover which part of an anomalous trajectory decreases the STOD confidence in classifying the anomalous trajectory among the route labels. Unfortunately, although the technique works pretty well in the image localization task, we failed to apply those technique to detect anomalous trajectory segments. We did not discover the semantic meaning of the results after applying LRP or Guided-backpropagation in the STOD.

We also attempted approaches based on unsupervised Autoencoder (AE) to detect anomaly trajectories and pinpoint the anomalous segments. Our assumption to test AE architecture was based on its capacity to denoising images (XIE; XU; CHEN, 2012). The key idea behind that architecture is that the input and output are the same. The network receives a trajectory and compresses it into a lower-dimension code, then reconstructs the output from this representation. Then, we initially considered testing AE to reduce trajectory anomalies. For that, we fed raw trajectories to a convolutional AE (Conv1D) that outputs the reconstructed trajectory and the reconstruction error that is calculated using the Mean Squared Error (MSE) between the input trajectory and the reconstructed trajectory. The first results were promising since the reconstruction error in the training and validation seemed to converge. However, although the AE approach reconstructed the trajectories very well, we realized that the approach also reconstructed the anomalous points. For example, Figure 30a shows two trajectories, blue and red, where the blue is the original, and the red trajectory is the original added by a small synthetic anomalous segment. Observe that, Figure 30b shows a black trajectory which is the reconstructed trajectory, while the red trajectory is the input. Here, note that the black trajectory is a smoothed version of the red trajectory, and, unfortunately, the anomalous segment is also reconstructed.

Another attempt was to use a Variational Autoencoder (VAE) architecture to improve the AE results. Although VAE is an AE in its essence, VAE allows regularizing the latent space to a well-known normal distribution. The regularization might solve the

Figure 30 – AE examples



(a) Original image

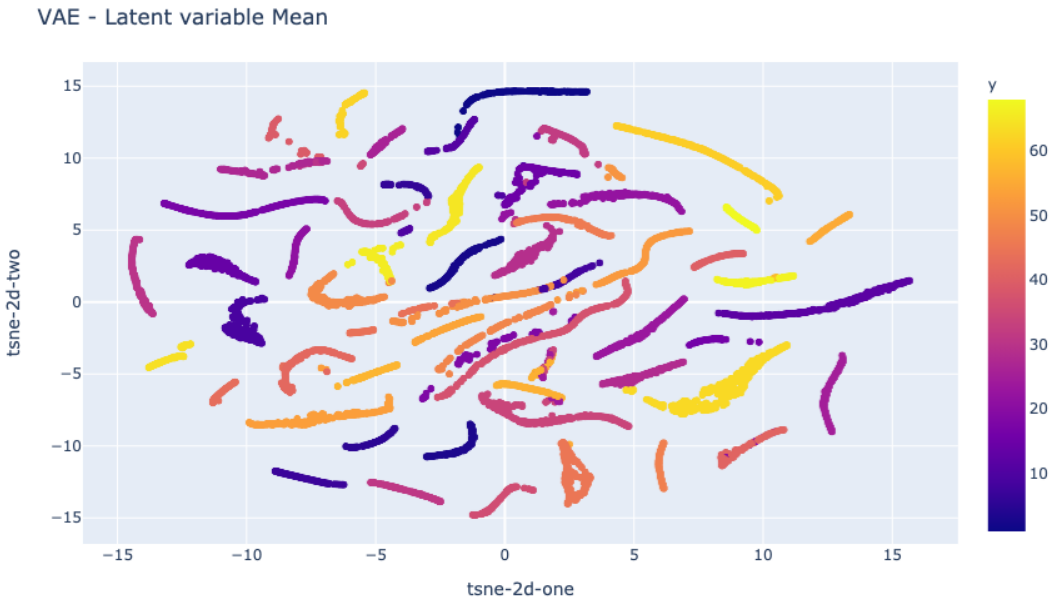
(b) Reconstruction

Source: Created by author (2022)

possible overfit noted in the AE results. In addition, it organizes the latent space retaining relevant features to improve the generative process (decoder). Overall, the variational encoder codifies the input trajectories as a normal distribution $N(u, \sigma)$, and the decoder learns to reconstruct the input from a normal distribution. Unlike AE, VAE adds a regularization in the loss to organize the latent space using the Kulback-Leibler divergence $||x - x^*|| + KL[N(u_x, \sigma_x), N(0, 1)]$. Note that the slight difference to the AE method is the regularization term. As shown in Figure 31, the modification (regularization term) allows the Variational Autoencoder approach grouping trajectories according to their routes (shapes), even without the route information. We realized that the VAE approach reconstructed the trajectories and discovered similar trajectories information through the latent space, which is useful for other tasks. For example, in taxi trajectories that do not have route information, VAE can work to group similar trajectories. However, this solution also failed to reconstruct the anomalies trajectories similarly to the AE approach, as shown in Figure 32b. Even with the regularization term (Kulback-Leibler divergence), the network continues to reconstruct the anomalous segments. In other words, we note that the VAE approach almost learns an identity function causing overfitting. Note that the reconstructed red trajectory follows the anomalous segment on the orange trajectory. Although Figure 32b shows a visible distance between the trajectories, we failed to use that reconstructed error to localize the anomalous segment. The main reason was that the regularization might decrease the overfit or the identity function effect in several segments in trajectory, as shown in Figure 32b, but, particularly, in the anomalous segment, the overfit continues.

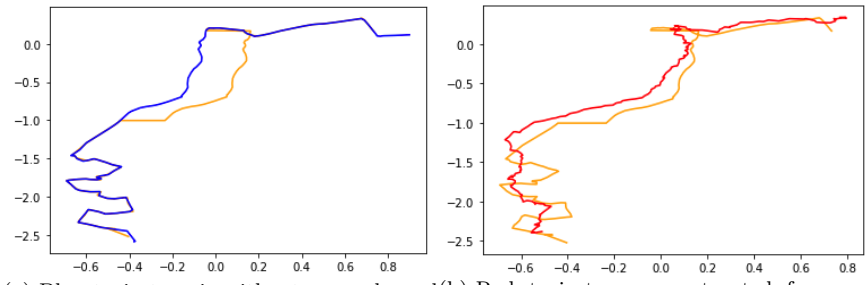
Following the Auto-Encoder architecture, we also attempted other methods such as Adversarial Auto-Encoder using CNN (Convolutional Neural Network) and LSTM (Long-Short Term Memory) and Conditional Adversarial Auto-Encoder. Unfortunately, they did

Figure 31 – Latent space for 68 routes of bus trajectories for Recife Dataset



Source: Created by author (2022)

Figure 32 – VAE examples



(a) Blue trajectory is without anomaly and Orange has synthetic anomaly.

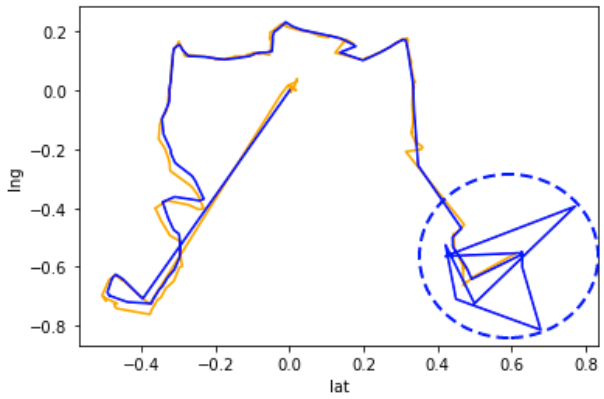
(b) Red trajectory reconstructed from orange anomaly trajectory.

Source: Created by author (2022)

not show good results.

Finally, we considered modeling our problem as an anomaly detection task in time series. The idea was to use a deep learning approach as a regression approach to predict the next element p^*_{t+1} for a given time series $p_0, p_1, ..., p_t$. Given that, we attempted the DeepAnt approach (MUNIR et al., 2018), which is a convolutional regression method to detect anomalous points in time series. The method generates a score for each point using the euclidean distance between the predicted point and the real point $|p - p*|$. Then, based on a threshold, anomaly points are detected. According to our experiments, as shown in Figure 33, DeepAnt failed to detect anomalous trajectory because it reproduces the anomalous points similar to AE approaches.

Figure 33 – Anomalous trajectory prediction using DeepAnt approach. The orange trajectory is the prediction and the blue trajectory is the input anomalous trajectory



Source: Created by author (2022)

7.3 FUTURE WORK

As possible improvements and extensions of this thesis, we suggest the following:

- Extending our LM model for temporal anomaly detection. Resulting in Spatial-Temporal Anomaly detection.
- Analyzing the encoder-decoder solution in other datasets, such as taxi and human trajectories datasets.
- Pre-training the encoder-decoder in multiple trajectory datasets to verify whether it can learn general trajectory patterns (deep representation). Once our approach learns those patterns, we want to exploit other tasks such as trajectory similarity and trajectory classification using transfer learning.
- Connecting the selected trajectories across screens as an end-to-end analysis in the MobApp Web App. For example, one can select a specific route on the EDA screen and use their trajectories on the other screens to get a complete analysis.
- Developing an end-point to store other datasets following a default pattern to be generic enough to feed the anomalies detection approaches.
- Developing another end-point to receive other approach results and evolving the user interface (UI).

REFERENCES

ADALOGLOU, N. Transformers in computer vision. *https://theaisummer.com/*, 2021.

AHMED, M.; MAHMOOD, A. N.; HU, J. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, Elsevier, v. 60, p. 19–31, 2016.

AKIBA, T.; SANO, S.; YANASE, T.; OHTA, T.; KOYAMA, M. Optuna: A next-generation hyperparameter optimization framework. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. [S.l.: s.n.], 2019. p. 2623–2631.

ALVARES, L. O.; BOGORNY, V.; KUIJPERS, B.; MACEDO, J. A. F. de; MOELANS, B.; VAISMAN, A. A model for enriching trajectories with semantic geographical information. In: *Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems*. [S.l.: s.n.], 2007. p. 1–8.

ANKERST, M.; BREUNIG, M. M.; KRIEGEL, H.-P.; SANDER, J. Optics: ordering points to identify the clustering structure. *ACM Sigmod record*, ACM New York, NY, USA, v. 28, n. 2, p. 49–60, 1999.

BAHDANAU, D.; CHO, K.; BENGIO, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

BASHARAT, A.; GRITAI, A.; SHAH, M. Learning object motion patterns for anomaly detection and improved object detection. In: IEEE. *2008 IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.], 2008. p. 1–8.

BELHADI, A.; DJENOURI, Y.; LIN, J. C.-W.; CANO, A. Trajectory outlier detection: Algorithms, taxonomies, evaluation, and open challenges. *ACM Transactions on Management Information Systems (TMIS)*, ACM New York, NY, USA, v. 11, n. 3, p. 1–29, 2020.

BENGIO, Y.; COURVILLE, A.; VINCENT, P. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 35, n. 8, p. 1798–1828, 8 2013. ISSN 0162-8828.

BERGSTRA, J.; YAMINS, D.; COX, D. D. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. Jmlr, 2013.

BESSA, A.; SILVA, F.; NOGUEIRA, R.; BERTINI, E.; FREIRE, J. Riobusdata: Outlier detection in bus routes of rio de janeiro. In: *Proceedings of Symposium on Visualization in Data Science (VDS)*. [S.l.: s.n.], 2015.

BONTEMPS, L.; MCDERMOTT, J.; LE-KHAC, N.-A. et al. Collective anomaly detection based on long short-term memory recurrent neural networks. In: SPRINGER. *International Conference on Future Data and Security Engineering*. [S.l.], 2016. p. 141–152.

BOURITSAS, G.; DAVEAS, S.; DANELAKIS, A.; THOMOPOULOS, S. C. Automated real-time anomaly detection in human trajectories using sequence to sequence networks. In: IEEE. *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. [S.l.], 2019. p. 1–8.

BRODSKY, I. *H3: Uber's Hexagonal Hierarchical Spatial Index*. 2018.

BROTHERTON, T.; JOHNSON, T. Anomaly detection for advanced military aircraft using neural networks. In: IEEE. *2001 IEEE Aerospace Conference Proceedings (Cat. No. 01TH8542)*. [S.l.], 2001. v. 6, p. 3113–3123.

BUJA, A.; COOK, D.; HOFMANN, H.; LAWRENCE, M.; LEE, E.-K.; SWAYNE, D. F.; WICKHAM, H. Statistical inference for exploratory data analysis and model diagnostics. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, The Royal Society Publishing, v. 367, n. 1906, p. 4361–4383, 2009.

CHALAPATHY, R.; CHAWLA, S. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*, 2019.

CHALAPATHY, R.; MENON, A. K.; CHAWLA, S. Anomaly detection using one-class neural networks. *arXiv preprint arXiv:1802.06360*, 2018.

CHANDOLA, V.; BANERJEE, A.; KUMAR, V. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, ACM New York, NY, USA, v. 41, n. 3, p. 1–58, 2009.

CHEN, C.; ZHANG, D.; CASTRO, P. S.; LI, N.; SUN, L.; LI, S. Real-time detection of anomalous taxi trajectories from gps traces. In: SPRINGER. *International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*. [S.l.], 2011. p. 63–74.

CHEN, C.; ZHANG, D.; CASTRO, P. S.; LI, N.; SUN, L.; LI, S.; WANG, Z. iboat: Isolation-based online anomalous trajectory detection. *IEEE Transactions on Intelligent Transportation Systems*, IEEE, v. 14, n. 2, p. 806–818, 2013.

CHEN, X.; CUI, T.; FU, J.; PENG, J.; SHAN, J. Trend-residual dual modeling for detection of outliers in low-cost gps trajectories. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 16, n. 12, p. 2036, 2016.

CHOLLET, F. *Deep learning with Python*. [S.l.]: Simon and Schuster, 2021.

CORTES, C.; VAPNIK, V. Support-vector networks. *Machine learning*, Springer, v. 20, n. 3, p. 273–297, 1995.

CRUZ, M.; BARBOSA, L. Learning gps point representations to detect anomalous bus trajectories. *IEEE Access*, IEEE, v. 8, p. 229006–229017, 2020.

DABIRI, S.; HEASLIP, K. Inferring transportation modes from gps trajectories using a convolutional neural network. *Transportation research part C: emerging technologies*, Elsevier, v. 86, p. 360–371, 2018.

DUDA, R. O.; HART, P. E.; STORK, D. G. *Pattern classification*. [S.l.]: John Wiley & Sons, 2012.

ESTER, M.; KRIEGEL, H.-P.; SANDER, J.; XU, X. et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: *Kdd*. [S.l.: s.n.], 1996. v. 96, n. 34, p. 226–231.

GAO, Q.; ZHOU, F.; ZHANG, K.; TRAJCEVSKI, G.; LUO, X.; ZHANG, F. Identifying human mobility via trajectory embeddings. In: *IJCAI*. [S.l.: s.n.], 2017. v. 17, p. 1689–1695.

GE, Y.; XIONG, H.; ZHOU, Z.-h.; OZDEMIR, H.; YU, J.; LEE, K. C. Top-eye: Top-k evolving trajectory outlier detection. In: *Proceedings of the 19th ACM international conference on Information and knowledge management*. [S.l.: s.n.], 2010. p. 1733–1736.

GREENFELD, J. S. Matching gps observations to locations on a digital map. In: WASHINGTON, DC. *81th annual meeting of the transportation research board*. [S.l.], 2002. v. 1, n. 3, p. 164–173.

GU, J.; YANG, Y.; TRESP, V. Understanding individual decisions of cnns via contrastive backpropagation. In: SPRINGER. *Asian Conference on Computer Vision*. [S.l.], 2018. p. 119–134.

GUO, C.; BERKHAHN, F. Entity embeddings of categorical variables. *arXiv preprint arXiv:1604.06737*, 2016.

GUPTA, M.; GAO, J.; AGGARWAL, C. C.; HAN, J. Outlier detection for temporal data: A survey. *IEEE Transactions on Knowledge and data Engineering*, IEEE, v. 26, n. 9, p. 2250–2267, 2013.

HO, T. K. Random decision forests. In: IEEE. *Proceedings of 3rd international conference on document analysis and recognition*. [S.l.], 1995. v. 1, p. 278–282.

HOLZINGER, A.; SARANTI, A.; MOLNAR, C.; BIECEK, P.; SAMEK, W. Explainable ai methods-a brief overview. In: SPRINGER. *International Workshop on Extending Explainable AI Beyond Deep Models and Classifiers*. [S.l.], 2022. p. 13–38.

HU, K.; DUAN, P.; HU, B.; DUAN, Q. Ibtod: An isolation-based method to detect outlying sub-trajectories on multi-factors. In: IEEE. *2018 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*. [S.l.], 2018. p. 1–1918.

HUANG, G.; LIU, Z.; MAATEN, L. V. D.; WEINBERGER, K. Q. Densely connected convolutional networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2017. p. 4700–4708.

HWANG, S.; EVANS, C.; HANKE, T. Detecting stop episodes from gps trajectories with gaps. In: *Seeing Cities Through Big Data*. [S.l.]: Springer, 2017. p. 427–439.

IAKOVIDIS, D. K.; GEORGAKOPOULOS, S. V.; VASILAKAKIS, M.; KOULAOUZIDIS, A.; PLAGIANAKOS, V. P. Detecting and locating gastrointestinal anomalies using deep learning and iterative cluster unification. *IEEE transactions on medical imaging*, IEEE, v. 37, n. 10, p. 2196–2210, 2018.

IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

JAVAID, A.; NIYAZ, Q.; SUN, W.; ALAM, M. A deep learning approach for network intrusion detection system. *Eai Endorsed Transactions on Security and Safety*, v. 3, n. 9, p. e2, 2016.

JI, Y.; WANG, L.; WU, W.; SHAO, H.; FENG, Y. A method for lstm-based trajectory modeling and abnormal trajectory detection. *IEEE Access*, IEEE, v. 8, p. 104063–104073, 2020.

JOLLIFFE, I. T.; CADIMA, J. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, The Royal Society Publishing, v. 374, n. 2065, p. 20150202, 2016.

KALID, S. N.; NG, K.-H.; TONG, G.-K.; KHOR, K.-C. A multiple classifiers system for anomaly detection in credit card data with unbalanced and overlapped classes. *IEEE Access*, IEEE, v. 8, p. 28210–28221, 2020.

KE, G.; MENG, Q.; FINLEY, T.; WANG, T.; CHEN, W.; MA, W.; YE, Q.; LIU, T.-Y. Lightgbm: A highly efficient gradient boosting decision tree. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2017. p. 3146–3154.

KNORR, E. M.; NG, R. T.; TUCAKOV, V. Distance-based outliers: algorithms and applications. *The VLDB Journal*, Springer, v. 8, n. 3-4, p. 237–253, 2000.

KONG, X.; SONG, X.; XIA, F.; GUO, H.; WANG, J.; TOLBA, A. Lotad: Long-term traffic anomaly detection based on crowdsourced bus trajectory data. *World Wide Web*, Springer, v. 21, n. 3, p. 825–847, 2018.

KORMÁKSSON, M.; BARBOSA, L.; VIEIRA, M. R.; ZADROZNY, B. Bus travel time predictions using additive models. In: IEEE. *2014 IEEE International Conference on Data Mining*. [S.l.], 2014. p. 875–880.

KULKARNI, V.; TAGASOVSKA, N.; VATTER, T.; GARBINATO, B. Generative models for simulating mobility trajectories. *arXiv preprint arXiv:1811.12801*, 2018.

KUMAR, G. R.; MANGATHAYARU, N.; NARSIMHA, G. An approach for intrusion detection using novel gaussian based kernel function. *J. UCS*, v. 22, n. 4, p. 589–604, 2016.

LAFFERTY, J.; MCCALLUM, A.; PEREIRA, F. C. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

LEE, J.-G.; HAN, J.; LI, X. Trajectory outlier detection: A partition-and-detect framework. In: IEEE. *2008 IEEE 24th International Conference on Data Engineering*. [S.l.], 2008. p. 140–149.

LI, Q.; ZHENG, Y.; XIE, X.; CHEN, Y.; LIU, W.; MA, W.-Y. Mining user similarity based on location history. In: ACM. *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*. [S.l.], 2008. p. 34.

LI, X.; ZHAO, K.; CONG, G.; JENSEN, C. S.; WEI, W. Deep representation learning for trajectory similarity computation. In: IEEE. *2018 IEEE 34th international conference on data engineering (ICDE)*. [S.l.], 2018. p. 617–628.

LI, Y.; FANG, B.; GUO, L.; CHEN, Y. Network anomaly detection based on tcm-knn algorithm. In: *Proceedings of the 2nd ACM symposium on Information, computer and communications security*. [S.l.: s.n.], 2007. p. 13–19.

LI, Y.; HUANG, Q.; KERBER, M.; ZHANG, L.; GUIBAS, L. Large-scale joint map matching of gps traces. In: *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. [S.l.: s.n.], 2013. p. 214–223.

LIN, T.-Y.; GOYAL, P.; GIRSHICK, R.; HE, K.; DOLLÁR, P. Focal loss for dense object detection. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2017. p. 2980–2988.

LIU, B.; FU, Y.; YAO, Z.; XIONG, H. Learning geographical preferences for point-of-interest recommendation. In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.: s.n.], 2013. p. 1043–1051.

LIU, F. T.; TING, K. M.; ZHOU, Z.-H. Isolation forest. In: IEEE. *2008 Eighth IEEE International Conference on Data Mining*. [S.l.], 2008. p. 413–422.

LIU, F. T.; TING, K. M.; ZHOU, Z.-H. Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, Acm New York, NY, USA, v. 6, n. 1, p. 1–39, 2012.

LIU, K.; GAO, S.; QIU, P.; LIU, X.; YAN, B.; LU, F. Road2vec: Measuring traffic interactions in urban road system from massive travel routes. *ISPRS International Journal of Geo-Information*, Multidisciplinary Digital Publishing Institute, v. 6, n. 11, p. 321, 2017.

LIU, S.; NI, L. M.; KRISHNAN, R. Fraud detection from taxis' driving behaviors. *IEEE Transactions on Vehicular Technology*, IEEE, v. 63, n. 1, p. 464–472, 2013.

LIU, W.; ZHENG, Y.; CHAWLA, S.; YUAN, J.; XING, X. Discovering spatio-temporal causal interactions in traffic data streams. In: *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.: s.n.], 2011. p. 1010–1018.

LIU, Y.; ZHAO, K.; CONG, G.; BAO, Z. Online anomalous trajectory detection with deep generative sequence modeling. In: IEEE. *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. [S.l.], 2020. p. 949–960.

LIU, Z.; PI, D.; JIANG, J. Density-based trajectory outlier detection algorithm. *Journal of Systems Engineering and Electronics*, BIAI, v. 24, n. 2, p. 335–340, 2013.

LOU, Y.; ZHANG, C.; ZHENG, Y.; XIE, X.; WANG, W.; HUANG, Y. Map-matching for low-sampling-rate gps trajectories. In: *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*. [S.l.: s.n.], 2009. p. 352–361.

LUU, V.-T.; FORESTIER, G.; WEBER, J.; BOURGEOIS, P.; DJELIL, F.; MULLER, P.-A. A review of alignment based similarity measures for web usage mining. *Artificial Intelligence Review*, Springer, v. 53, n. 3, p. 1529–1551, 2020.

LV, Z.; XU, J.; ZHAO, P.; LIU, G.; ZHAO, L.; ZHOU, X. Outlier trajectory detection: A trajectory analytics based approach. In: SPRINGER. *International Conference on Database Systems for Advanced Applications*. [S.l.], 2017. p. 231–246.

MAATEN, L. van der; HINTON, G. Visualizing data using t-SNE. *Journal of Machine Learning Research*, v. 9, p. 2579–2605, 2008. Available at: <http://www.jmlr.org/papers/v9/vandermaaten08a.html>.

MANN, H. B.; WHITNEY, D. R. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, JSTOR, p. 50–60, 1947.

MENG, F.; YUAN, G.; LV, S.; WANG, Z.; XIA, S. An overview on trajectory outlier detection. *Artificial Intelligence Review*, Springer, v. 52, n. 4, p. 2437–2456, 2019.

MIKOLOV, T.; CHEN, K.; CORRADO, G.; DEAN, J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

MOHANADAS, R. *Discerning truck stop semantics through latent space clustering*. 2018.

MORENO, F. J.; PINEDA, A. F.; FILETO, R.; BOGORNY, V. Smot+: Extending the smot algorithm for discovering stops in nested sites. *Computing and Informatics*, v. 33, n. 2, p. 327–342, 2014.

MUNIR, M.; SIDDIQUI, S. A.; DENGEL, A.; AHMED, S. Deepant: A deep learning approach for unsupervised anomaly detection in time series. *Ieee Access*, IEEE, v. 7, p. 1991–2005, 2018.

NETO, F. d. B. W. Mobapp-dashboard para visualização de trajetórias de ônibus. 2021.

NOGUEIRA, T. P.; CELES, C. S.; MARTIN, H.; LOUREIRO, A. A.; ANDRADE, R. M. A statistical method for detecting move, stop, and noise: A case study with bus trajectories. *Journal of Information and Data Management*, v. 9, n. 3, p. 214–214, 2018.

OUYANG, K.; SHOKRI, R.; ROSENBLUM, D. S.; YANG, W. A non-parametric generative model for human trajectories. In: *IJCAI*. [S.l.: s.n.], 2018. p. 3812–3817.

PAN, B.; ZHENG, Y.; WILKIE, D.; SHAHABI, C. Crowd sensing of traffic anomalies based on human mobility and social media. In: *Proceedings of the 21st ACM SIGSPATIAL international conference on advances in geographic information systems*. [S.l.: s.n.], 2013. p. 344–353.

PANG, L. X.; CHAWLA, S.; LIU, W.; ZHENG, Y. On mining anomalous patterns in road traffic streams. In: SPRINGER. *International conference on advanced data mining and applications*. [S.l.], 2011. p. 237–251.

PATTERSON, J.; GIBSON, A. *Deep learning: A practitioner's approach*. [S.l.]: " O'Reilly Media, Inc.", 2017.

QIAN, S.; CHENG, B.; CAO, J.; XUE, G.; ZHU, Y.; YU, J.; LI, M.; ZHANG, T. Detecting taxi trajectory anomaly based on spatio-temporal relations. *IEEE Transactions on Intelligent Transportation Systems*, IEEE, 2021.

RAYMOND, R.; IMAMICHI, T. Bus trajectory identification by map-matching. In: IEEE. *2016 23rd International Conference on Pattern Recognition (ICPR)*. [S.l.], 2016. p. 1618–1623.

RUFF, L.; VANDERMEULEN, R.; GOERNITZ, N.; DEECKE, L.; SIDDIQUI, S. A.; BINDER, A.; MÜLLER, E.; KLOFT, M. Deep one-class classification. In: *International conference on machine learning*. [S.l.: s.n.], 2018. p. 4393–4402.

RUSSELL, S.; NORVIG, P. Artificial intelligence: a modern approach. 2002.

SCHLEGL, T.; SEEBÖCK, P.; WALDSTEIN, S. M.; SCHMIDT-ERFURTH, U.; LANGS, G. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In: SPRINGER. *International conference on information processing in medical imaging*. [S.l.], 2017. p. 146–157.

SIEGEL, S. Nonparametric statistics for the behavioral sciences. McGraw-hill, 1956.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, Ł.; POLOSUKHIN, I. Attention is all you need. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2017. p. 5998–6008.

VINCENTY, T. Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. *Survey review*, Taylor & Francis, v. 23, n. 176, p. 88–93, 1975.

WANG, D.; ZHANG, J.; CAO, W.; LI, J.; ZHENG, Y. When will you arrive? estimating travel time based on deep neural networks. In: *Thirty-Second AAAI Conference on Artificial Intelligence*. [S.l.: s.n.], 2018.

WANG, J.; YUAN, Y.; NI, T.; MA, Y.; LIU, M.; XU, G.; SHEN, W. Anomalous trajectory detection and classification based on difference and intersection set distance. *IEEE Transactions on Vehicular Technology*, IEEE, v. 69, n. 3, p. 2487–2500, 2020.

WANG, Y.; QIN, K.; CHEN, Y.; ZHAO, P. Detecting anomalous trajectories and behavior patterns using hierarchical clustering from taxi gps data. *ISPRS International Journal of Geo-Information*, Multidisciplinary Digital Publishing Institute, v. 7, n. 1, p. 25, 2018.

WU, H.; PRASAD, S. Semi-supervised deep learning using pseudo labels for hyperspectral image classification. *IEEE Transactions on Image Processing*, IEEE, v. 27, n. 3, p. 1259–1270, 2017.

XIE, J.; XU, L.; CHEN, E. Image denoising and inpainting with deep neural networks. *Advances in neural information processing systems*, v. 25, 2012.

XIE, X.; WANG, C.; CHEN, S.; SHI, G.; ZHAO, Z. Real-time illegal parking detection system based on deep learning. In: *Proceedings of the 2017 International Conference on Deep Learning Technologies*. [S.l.: s.n.], 2017. p. 23–27.

YING, X.; XU, Z.; YIN, W. G. Cluster-based congestion outlier detection method on trajectory data. In: IEEE. *2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery*. [S.l.], 2009. v. 5, p. 243–247.

YOUSEFI-AZAR, M.; VARADHARAJAN, V.; HAMEY, L.; TUPAKULA, U. Autoencoder-based feature learning for cyber security applications. In: IEEE. *2017 International joint conference on neural networks (IJCNN)*. [S.l.], 2017. p. 3854–3861.

YU, Y.; CAO, L.; RUNDENSTEINER, E. A.; WANG, Q. Outlier detection over massive-scale trajectory streams. *ACM Transactions on Database Systems (TODS)*, ACM New York, NY, USA, v. 42, n. 2, p. 1–33, 2017.

YUAN, J.; ZHENG, Y.; ZHANG, C.; XIE, X.; SUN, G.-Z. An interactive-voting based map matching algorithm. In: IEEE. *2010 Eleventh international conference on mobile data management*. [S.l.], 2010. p. 43–52.

YUAN, J.; ZHENG, Y.; ZHANG, L.; XIE, X.; SUN, G. Where to find my next passenger. In: ACM. *Proceedings of the 13th international conference on Ubiquitous computing*. [S.l.], 2011. p. 109–118.

YUAN, N. J.; ZHENG, Y.; ZHANG, L.; XIE, X. T-finder: A recommender system for finding passengers and vacant taxis. *IEEE Transactions on knowledge and data engineering*, IEEE, v. 25, n. 10, p. 2390–2403, 2012.

ZHANG, A.; LIPTON, Z. C.; LI, M.; SMOLA, A. J. *Dive into Deep Learning*. [S.l.: s.n.], 2020. <https://d2l.ai>.

ZHANG, D.; LI, N.; ZHOU, Z.-H.; CHEN, C.; SUN, L.; LI, S. ibat: detecting anomalous taxi trajectories from gps traces. In: ACM. *Proceedings of the 13th international conference on Ubiquitous computing*. [S.l.], 2011. p. 99–108.

ZHANG, F.; YUAN, N. J.; WILKIE, D.; ZHENG, Y.; XIE, X. Sensing the pulse of urban refueling behavior: A perspective from taxi mobility. *ACM Transactions on Intelligent Systems and Technology (TIST)*, ACM, v. 6, n. 3, p. 37, 2015.

ZHANG, Y.; LIU, A.; LIU, G.; LI, Z.; LI, Q. Deep representation learning of activity trajectory similarity computation. In: IEEE. *2019 IEEE International Conference on Web Services (ICWS)*. [S.l.], 2019. p. 312–319.

ZHANG, Y.; NING, N.; ZHOU, P.; WU, B. Ut-atd: Universal transformer for anomalous trajectory detection by embedding trajectory information. *Proceedings of the 27th International Conference on Distributed Multimedia Systems*, 2021.

ZHAO, X.; RAO, Y.; CAI, J.; MA, W. Abnormal trajectory detection based on a sparse subgraph. *IEEE Access*, IEEE, v. 8, p. 29987–30000, 2020.

ZHENG, G.; BRANTLEY, S. L.; LAUVAUX, T.; LI, Z. Contextual spatial outlier detection with metric learning. In: *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. [S.l.: s.n.], 2017. p. 2161–2170.

ZHENG, Y. Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology (TIST)*, ACM, v. 6, n. 3, p. 29, 2015.

ZHENG, Y.; XIE, X. Learning travel recommendations from user-generated gps traces. *ACM Transactions on Intelligent Systems and Technology (TIST)*, ACM, v. 2, n. 1, p. 2, 2011.

ZHU, J.; JIANG, W.; LIU, A.; LIU, G.; ZHAO, L. Time-dependent popular routes based trajectory outlier detection. In: SPRINGER. *International Conference on Web Information Systems Engineering*. [S.l.], 2015. p. 16–30.

ZHU, J.; JIANG, W.; LIU, A.; LIU, G.; ZHAO, L. Effective and efficient trajectory outlier detection based on time-dependent popular route. *World Wide Web*, Springer, v. 20, n. 1, p. 111–134, 2017.