Cleaning Outdoor Activity Logs using Deep Learning

Davide Sbetti
Free University of Bozen–Bolzano
Email: davide.sbetti@gmail.com

Sergio Tessaris
Free University of Bozen–Bolzano
Email: tessaris@inf.unibz.it

***Abstract*—Nowadays, position recording personal tracking devices are ubiquitous and used by both athletes and outdoor enthusiasts to track and analyse their activities. These devices rely on Global Navigation Satellite Systems to obtain the position in real time. Although the nominal precisions of the different GNSS are high enough for analysis, there are several environmental factors that affect the precision of such devices. Most of the commercial services providing analysis of outdoor activities use techniques to "clean" the user-uploaded data (tracklogs). Most of these techniques require and exploit the huge amount of data that they collect and analyse, but the resulting logs still manifest outliers and recording errors. In this paper, we present a deep learning based technique to identify part of tracklogs that might be influenced by recording errors, in such a way that can be corrected using standard techniques. Our approach does not require geographical or crowdsourced data, and can be also used on low powered devices.**

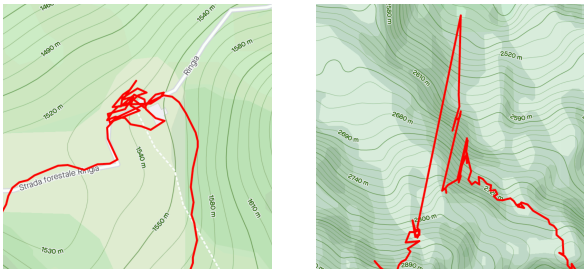
***Index Terms*—Data Cleaning, GPS Traces, Trajectory repairing, recurrent neural networks**

I. Introduction

Location tracking devices based on Global Navigation Satellite Systems (GNSS) are widely used by outdoor enthusiasts and athletes to track their activities for both analysis and social sharing purposes. Although the precision of GNSS-based geolocation is within a few metres under optimal conditions, there are several environmental and receiver-related factors that may introduce substantial errors [1].

Most commercial providers of services related to the analysis and sharing of outdoor activities employ techniques to correct imprecisions in the data provided by users (e.g. [2]). However, the results are not always satisfactory, even by exploiting the large amount of data collected by the big players in the field (see, e.g., Figure 1).

In our work, we focus on two types of error that are often present in recorded activity logs, which are demonstrated in Figure 1. Those are the errors that can be easily identified by users and appear in most outdoor activity recordings. Both these segments are extracted from the web interface of one of the biggest commercial service provider, after any correction that might have been applied to the raw data. The first segment shows a pause that has not been detected by the recording device (some devices feature pause detection algorithms, but often they do not provide a reliable outcome), while the second shows that some recorded points do not reflect the actual position of the receiver. Note that the nature of the two types of errors are different, since the behaviour of the first case is due to the intrinsic imprecision of the position pinpointing while the second is mostly due to environmental conditions, as rock formations. It might be argued that part of the tracking of the first case is due to inevitable small movements of the receiver; however, even in this case, it would be desirable to remove these segments from the recording.



(a) (b)

Fig. 1. Examples of recording errors

In our research, we explore the use of Deep Learning [3] (DL) techniques to improve the quality of recorded activities without resorting to big data techniques (e.g., heat maps) or background knowledge (e.g., network of roads and paths). The reason is twofold, from one side we would like to be able to apply our technique at the level of edge computing also on low powered devices; on the other hand, there are several outdoor activities that are not constrained by the network of paths or roads (e.g., ski touring or kayaking). We are also interested to develop techniques not relying on the behaviour of specific receivers or activities. In this way, it could be applied to data coming from different sources, or even when the information on the recording device or activity type has been stripped from the data.

The contributions of this work are as follows.

- Collection of an annotated dataset of outdoor activities for data repair.
- Evaluation of different DL architectures for GNSS data classification.
- Development and evaluation of a DL model for classifying GNSS receiver errors in activity logs.
- Evaluation and development of algorithms for repairing activity logs.

The next section will introduce the main concepts and related works, because of space restrictions we assume that

 ISBN: 978-1-61208-970-6

the reader is familiar with Deep Learning techniques and architectures (otherwise the reader is referred to the relevant bibliographic entries). Section III describes the development and evaluation of the classifier, while Section IV describes the repair techniques. A more detailed description of this work is available in the M.Sc. thesis [4].

II. Backgrounds and Related Work

Modern tracking devices use combinations of different GNSS to identify the current location; moreover, some devices integrate data from other sensors – e.g., barometer, gyroscope, compass – to compensate for possible reception errors. In this paper, we focus on the *Global Positioning System* (GPS), but similar considerations apply also to the other systems [5].

GPS uses 24 satellites on different orbits to enable the reception of at least four of them, regardless of the location of the receiver. In addition, there are control stations that precisely monitor the position of each satellite and share this information using a common registry. To calculate its position, the receiver listens to the electromagnetic waves of four different satellites. It then monitors the time taken by the signal to reach the device starting from the satellites and, since it knows the exact position of each satellite, is able to calculate its position with respect to them. Four different satellites are required to measure the three different coordinates (latitude, longitude, and altitude), while the fourth signal is used to synchronise the internal clock of the device [6].

According to a study conducted in 2015 [7], the positioning error of a smartphone using GPS is within $5m$ when there are no obstacles. However, the GPS signal can be influenced by the surrounding environment. In particular, the signal can be reflected by some surfaces resulting in what is known as multipath reception [1], depicted in Figure 2.

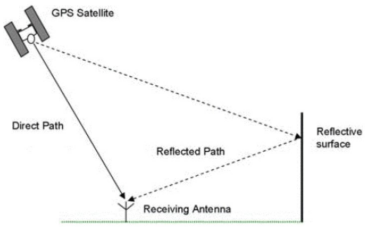


Fig. 2. Multipath effect. Reproduced from [1]

The reflected signal takes longer than the correct direct path, introducing a delay that could result in a potential error in the calculation of the position. Moreover, given that the satellites are constantly moving, the effect of multipath at a certain location changes over time, depending on the position of the satellites. Experiments have shown that the multipath effect can introduce an additional average of $8m$ error in the derived position [1]. This effect may occur in nature due to, for example, mountains, resulting in what is widely known as *canyon effect*. This effect is also occurring in urban contexts, where tall buildings substitute rock walls (urban canyons) [8]. Figure 3 shows the effects of the rocky environment on a recorded activity.

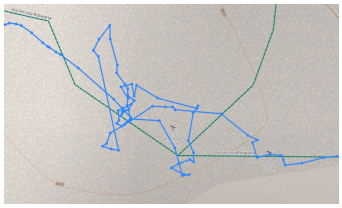


Fig. 3. Canyon effect on an activity log recorded during a via ferrata

In this paper, we use the term *tracklog* to denote a sequential record of geographic coordinates with associated timestamps (the *location points* or simply *points*) collected by a GNSS receiver. Timestamps are essential for most activity analysis tasks, and tracklogs are also referred as GNSS *traces* or *trajectories* in the literature. A *tracklog repair* is an editing of the sequence by removing or changing the geographic coordinates of some points. We assume the correctness of the timestamps; in particular, the relevant detail is the time interval between timestamps.

The task of cleaning GPS data, and as a more general case time series, has been widely studied in the literature. We identified six groups of related works based on the techniques they employ.

Smoothing-based techniques Moving average approaches or autoregressive algorithms are often used to smooth time series [9]. Kalman filters [9], which combine the observation and a prediction, based on the relation of the various components with external factors, were also applied, updating the external influence [10] or weighting the observations using the variance [11]. Smoothing techniques generally modify the entire time series, which may also lead to adjusting the correct points.

Data-driven techniques In [12], past observations are used to extract the main routes and areas in a region to correct the points falling into them. Predetermined trajectories, known a priori, were used to correct GPS readings [13]. Furthermore, in [14] the authors employed the detection of outlying sub-trajectories, considering the distance of their characteristics. Data driven techniques heavily rely on the regional availability of data, generally lacking generalisation.

Constraint-based techniques Ordered, or sequential, relations are also often employed; such as speed constraints to identify outliers [9]. However, their definition can be problematic for dynamic activities.

Statistics-based techniques Markov Models have been used to predict and then compare observations as a cleaning strategy [9]. Dynamic probabilistic models, such as STPM, were used to calculate the conditional probabilities of attributes, applying a threshold based cleaning [9]. Moreover, in [15], using available data, the corrected sequence is calculated as close as possible to the original using the largest likelihood in terms of speed change between consecutive points.

 ISBN: 978-1-61208-970-6

Anomaly detection techniques In [16], the authors partitioned the original trajectories using significant direction or speed changes, and then they apply a consistency model to the partitions in order to identify anomalies. Thresholds have also been employed, for example in [17], to the reachability speed between points to remove outliers, then applying the underlying geographic information to adjust their positions, or in [8] using the altitude and distance between consecutive points. Smoothing techniques, e.g., a Gaussian kernel, are applied to the points. However, identifying the right thresholds can be challenging for dynamic activities.

Machine learning techniques Clustering algorithms were used to merge directly reachable clusters of points [9], using DL models adopted to perform anomaly detection [9]. In [18], decision trees and the SVM were combined to classify the GPS points into different categories. In [19], a neural network is used to predict the destination of taxi trips among selected destinations, which are discovered applying a mean shift algorithm on the training trajectories. Furthermore, in [20] a regression model, trained on points collected by smartphones, is employed to identify a radius used to restrict candidates for the correct position within a circle centred on the point to be corrected.

III. Building and Training the Classifier

The first problem we faced is the lack of (annotated) datasets to develop and train a classification model. Therefore, we built a publicly accessible web application that allows anonymous upload and annotation of logs. The description of the application is outside the scope of this paper, but its source is available on [21].

The annotation app enables graphical annotation by selecting (ranges of) points and assigning one of the available classes: *pause*, *outlier*, *correct*. We decided to include the *correct* class in order to mitigate the fact that the majority of points are not annotated, so the dataset would be heavily unbalanced if we were considering all the not annotated points as correct. The *correct* class enables the annotator to specify points that they consider to be placed correctly. The annotator has the possibility of including additional metadata, such as the type of activity and the recording device, but this is not required.

Over 5 months, we collected 61 logs distributed among 7 types of activities, and most of them were hiking, walking, and ski mountaineering (they correspond to 80% of the logs). The number of distinct logs is important to ensure diversity, but the dataset should be understood in terms of location points and annotations. The total number of points is $232,238$ with $13,514$ (0.06%) annotations; of those, the majority – more than 77% – are marked as correct, followed by pauses (15%) and outliers (8%). The majority of the collected logs are located in the Trentino - South Tyrol Italian region, while more in general we could observe how all of them were in the northern Italy.

The fact that a point is identified as not correct does not depend on its location and timestamp, but on the variation w.r.t. preceding and following points. In fact, several techniques in the literature just rely on identifying wrong points on the basis of their variation w.r.t. neighbours (see, e.g., [16], [17]). Because of this, we normalise the dataset, transforming each point into a tuple representing the variation of location and time from the previous point (the *deltas*). Each delta represents the variation of the coordinates on the three spatial axes and the time elapsed between each pair of consecutive points (Figure 4).

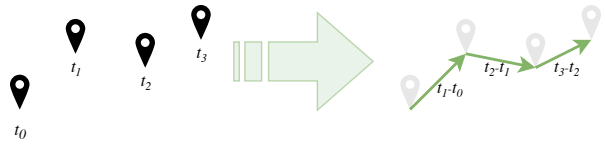


Fig. 4. Location points to deltas.

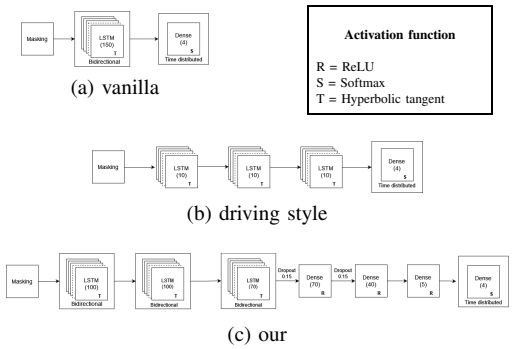
To simplify the computations of deltas, we convert the original location points encoded in longitude, latitude, and altitude into the Earth-Centred Earth-Fixed Coordinate System (ECEF) [22]. The latter being a Cartesian system, the deltas are just the difference between values of the corresponding axes, and their values do not depend on the actual geodetic position. Moreover, given the initial location and timestamp, the original sequence of geodetic coordinates can be reconstructed without data loss. In the rest of the paper, we will use the term points to refer to the deltas rather than the geographical coordinates.

Before committing to a specific deep learning architecture, we performed a set of preliminary experiments in order to identify the one that would be best suited to identify properties of GNSS logs. Our hypothesis was that Recurrent Neural Networks are well suited for this task, being widely adopted for *sequence labelling*, in particular focusing on *Long Short Term Memory* (LSTM) networks [23]. However, we decided to compare it with two other different architectures used to classify and repair GNSS data: Multi-Layer Perceptron [24], and Convolutional [25]. Since we were still collecting our data, we decided to evaluate these architectures on a simple activity recognition task (that is, identifying the type of sport activity from the recorded task) using 150 publicly available tracklogs of walking, hiking, and running (downloaded from *Wikiloc* and *Garmin Connect* sites with the consent of users). Our preliminary experiments confirmed that LSTM networks provided the best performance among the selected architectures.

In [26] different variations of the LSTM architecture have been evaluated and shown that "forget gate and the output activation function to be its most critical components", reinforcing our choice of adopting recurrent neural networks based on this cell type, considering also how some possible variations, tested in [26], did not show particular advantages in terms of accuracy. To select the right network topology and hyper-parameters we compared a common *vanilla LSTM* as baseline, with a three-layers LSTM architecture used in [27] to evaluate driving style, and our proposed architecture coupling three LSTM layers with dropout layers (see Figure 5). In our proposed architecture, the first (bidirectional) LSTM layers

 ISBN: 978-1-61208-970-6

outline the nonlinear temporal relations between the data; while the remaining layers gradually reduce the number of nodes before the target output to avoid overfitting.



(a) vanilla

(b) driving style

(c) our

Fig. 5. The evaluated LSTM architectures

For all selected LSTM architectures, the input layer is a sequence of contiguous points (4-tuples) of a given fixed size (the *window*), while the output layer is composed of a sequence (the same size as the input) with an array of units, each corresponding to one of the classification classes, activated using softmax. Each array of units provide the classification of the corresponding input point.

To classify the points of a tracklog, we employ a *sliding window* approach where sequences are created by moving a fixed size window over the original tracklog, by a given sliding step (Fig. 6).

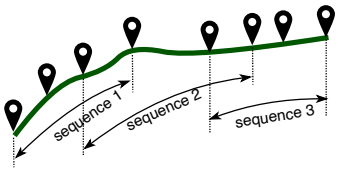


Fig. 6. Sliding window approach to segmenting (size 4, step 2)

When the classifier is used on a given tracklog, each point is classified multiple times (depending on the size of the window and sliding step), for selecting the class we decided to use a majority selection. This is used in the evaluation for training the network and in its deployment for cleaning new logs.

To select the best parameters we focused on the window size (3, 5, 10, 15), step size (1, 2, 3, 5), and number of training epochs (10, 20, 30, 40, 50). We performed a grid search on those combinations using a standard K-fold technique (3 folds). For the grid search, we used a subset of the final dataset (66% of the tracklogs), and the dataset was composed of all the sequences generated by the sliding-window approach over the (padded) annotated points. We ensured that training and testing folds did not share common points due to intersecting windows. Table I shows the best three combination across the different architectures. The *driving* architecture does not appear in the table because the best accuracy obtained among the different combinations of parameters was 0.83.

We performed a final evaluation of the selected architecture and parameters using the same k-fold procedure. The results

TABLE I
TOP THREE COMBINATIONS

Epochs	Window	Step	Accuracy	LSTM Architecture
30	15	2	0.864243	our
40	15	2	0.860122	vanilla
10	15	2	0.859688	vanilla

are summarised using the confusion matrices of the folds in Figure 7. Finally, we trained the network on the entire dataset, generating the model to be deployed to classify the points in tracklogs.

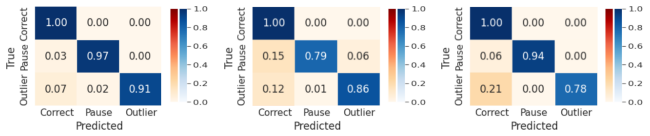


Fig. 7. Confusion matrices for model evaluation

Our goal is to be able to deploy the data repair on low-powered devices, so it is paramount to maximise the efficiency and minimise the resource consumption of the classifier. The standard *TensorFlow* library and models are not suitable for edge computing; however, *TensorFlow Lite* [28] is tailored for deployment on mobile, embedded, and IoT devices. Standard TensorFlow models cannot be used with the "lite" version of the library and need to be converted. Since the original model is not very large, we decided to convert it without any further optimisation; therefore, the properties of the "lite" model are the same as the original. This has been confirmed by comparing the results of the original and "lite" models over the entire training data set.

The code used for our experiments is available in [29].

IV. REPAIRING TRACKLOGS

The process we envisage for cleaning the activity logs is composed by two stages: in the first, location points are classified using the trained model; after, one of the various techniques presented in the literature can be applied to the points identified as non-correct.

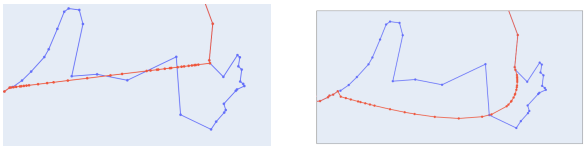
As introduced in Section III the classification of points in a tracklog is performed by first converting them into a sequence *deltas* and then classifying them by majority vote using a sliding window to obtain multiple classifications for each point.

After the classification process, the correction focuses on the points assigned to either *pause* or *outlier* classes. The first kind of points are easy to deal with, since they represent a situation in which the device should have been stationary, so they should be all referring to the same position. Therefore, a sequence of "pause" points can be removed or replaced with a single point by averaging their data. In fact, there might be actual small movements, but they are irrelevant, or even misleading, for activity analysis purposes.

To correct "outliers" we considered different techniques applied in the relevant literature. Most techniques that do

not exploit crowd-sourced data are based on the idea of "smoothing" segments in which points diverge w.r.t. a set of predefined parameters. The smoothing can be performed using linear or splines [30] interpolation. *Kalman filters* [31] have also been used to improve the quality of GNSS data [9]. This family of transformations join together the observed value and the predicted one to obtain the final observation.

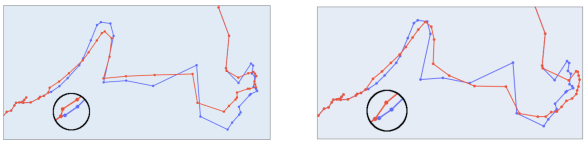
We compared linear and spline interpolation (Figure 8) with Kalman filters (Fig 9) and we observed that the latter provided the best results. Note that the quality of the result is based on the visual inspection of the corrected tracklog; we reckon that more objective measures should be investigated, but to the best of our knowledge, there is no consensus in the research community on how to evaluate the quality of a tracklog.



(a) linear (b) spline

Fig. 8. Example of interpolation (in red the changes)

There are different ways to apply the filter; we excluded the common approach of applying it to all data points because our classifier enables us to focus on only the points that are considered "outliers". We apply filtering only to the segments that include "outliers". We also noticed that the application of the filter along the direction of the timestamps introduces an unnatural joining of the corrected segment with the rest of the tracklog due to an "inertial" effect of the filter (it tends to maintain the current direction of the movement). To mitigate this effect, we applied the filter in both forward and backward directions, averaging their changes to the points (Figure 9).



(a) unidirectional (b) bidirectional

Fig. 9. Example of Kalman filter (in red the changes)

The code we used for our repair experiments is available in [32].

V. Conclusions

In this paper, we described a Deep Learning approach to identify parts of the recorded activity logs that could be affected by GNSS receiver errors. We identified two types of error, namely those deriving from the so-called "canyon effect" (outliers) and those deriving from a stationary receiver (pauses). Moreover, we show how different techniques can be applied to modify the identified errors, and suggest the ones that provide good quality results. Note that the modularity of our approach enables the use of different techniques and/or preferences (e.g., pause segments might be left as they are).

Our empirical evaluation shows that an LSTM architecture is well suited to identify points affected by receiver errors, and the classifier can also be used on low-powered (edge) devices. Our work is also showing that the identification of errors can also be performed without a prior knowledge about the type of device and activity. The training process does not require a large amount of computational resources, so individual users could adapt the model to their kind of specific activities or devices.

Moreover, we developed a web application to enable the collection and annotation of tracklogs, and used it to create a dataset for our experiments. Clearly, the quality of the dataset is paramount to ensure the accuracy of the predictions. In the future we plan to extend the dataset, in particular w.r.t. the geographical area.

Another problem we identified is the evaluation of the quality of tracklog repairs. In this work, we adopted a subjective approach based on the appearance of the resulting tracklog, but we think that identifying a proper quality measure is an open and relevant problem which does not seem to be addressed in any of the works we reviewed. An approach could be to collect a "golden standard" dataset in a controlled environment; but it is difficult to cover the variety of outdoor activities and the environments in which they occur.

References

[1] T. Kos, I. Markezic, and J. Pokrajcic, "Effects of multipath reception on GPS positioning performance," in *Proceedings ELMAR-2010*, Sep. 2010, pp. 399–402, iSSN: 1334-2630.

[2] "Elevation on Strava FAQs," Mar 2022, [Online; accessed 30. Mar. 2022]. [Online]. Available: https://support.strava.com/hc/en-us/articles/115001294564-Elevation-on-Strava-FAQs

[3] S. Skansi, *Introduction to Deep Learning*. Cham, Switzerland: Springer, 2018.

[4] D. Sbetti, "Cleaning GPS trajectories using machine learning techniques," Master's thesis, Free University of Bozen-Bolzano – Computer Science Department, Bolzano, Italy, 2022. [Online]. Available: https://doi.org/10.5281/zenodo.6369847

[5] X. Li, M. Ge, X. Dai, X. Ren, M. Fritsche, J. Wickert, and H. Schuh, "Accuracy and reliability of multi-GNSS real-time precise positioning: GPS, GLONASS, BeiDou, and Galileo," *Journal of Geodesy*, vol. 89, no. 6, pp. 607–635, Jun. 2015. [Online]. Available: https://doi.org/10.1007/s00190-015-0802-8

[6] N. Crato, "How GPS Works," in *Figuring It Out: Entertaining Encounters with Everyday Math*, N. Crato, Ed. Berlin, Heidelberg: Springer, 2010, pp. 49–52. [Online]. Available: https://doi.org/10.1007/978-3-642-04833-3_12

[7] F. van Diggelen and P. Enge, "The World's first GPS MOOC and Worldwide Laboratory using Smartphones," in *Proceedings of the 28th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2015)*, Sep. 2015, pp. 361–369, iSSN: 2331-5954. [Online]. Available: http://www.ion.org/publications/abstract.cfm?jp=p&articleID=13079

[8] N. Schuessler and K. W. Axhausen, "Processing Raw Data from Global Positioning Systems without Additional Information," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2105, no. 1, pp. 28–36, Jan. 2009. [Online]. Available: http://journals.sagepub.com/doi/10.3141/2105-04

[9] X. Wang and C. Wang, "Time Series Data Cleaning: A Survey," *IEEE Access*, vol. 8, pp. 1866–1881, 2020.

[10] A. H. Mohamed and K. P. Schwarz, "Adaptive Kalman Filtering for INS/GPS," *Journal of Geodesy*, vol. 73, no. 4, pp. 193–203, May 1999. [Online]. Available: http://link.springer.com/10.1007/s001900050236

[11] S. Shokri, N. Rahemi, and M. R. Mosavi, "Improving GPS positioning accuracy using weighted Kalman Filter and variance estimation methods," *CEAS Aeronautical Journal*, vol. 11, no. 2, pp. 515–527, Jun. 2020. [Online]. Available: http://link.springer.com/10.1007/s13272-019-00433-x

[12] L. Li, X. Chen, Q. Liu, and Z. Bao, "A Data-Driven Approach for GPS Trajectory Data Cleaning," in *Database Systems for Advanced Applications*, Y. Nah, B. Cui, S.-W. Lee, J. X. Yu, Y.-S. Moon, and S. E. Whang, Eds. Cham: Springer International Publishing, 2020, vol. 12112, pp. 3–19, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/978-3-030-59410-7_1

[13] R. Milton and A. Steed, "Correcting GPS Readings from a Tracked Mobile Sensor," in *Location- and Context-Awareness*, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, T. Strang, and C. Linnhoff-Popien, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, vol. 3479, pp. 83–94, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/11426646_8

[14] J.-G. Lee, J. Han, and X. Li, "Trajectory Outlier Detection: A Partition-and-Detect Framework," in *2008 IEEE 24th International Conference on Data Engineering*. Cancun, Mexico: IEEE, apr 2008, pp. 140–149. [Online]. Available: http://ieeexplore.ieee.org/document/4497422/

[15] A. Zhang, S. Song, and J. Wang, "Sequential Data Cleaning: A Statistical Approach," in *Proceedings of the 2016 International Conference on Management of Data*. San Francisco California USA: ACM, jun 2016, pp. 909–924. [Online]. Available: https://dl.acm.org/doi/10.1145/2882903.2915233

[16] X. Yang, L. Tang, X. Zhang, and Q. Li, "A Data Cleaning Method for Big Trace Data Using Movement Consistency," *Sensors*, vol. 18, no. 3, p. 824, Mar. 2018. [Online]. Available: http://www.mdpi.com/1424-8220/18/3/824

[17] A. Hendawi, S. S. Sabbineni, J. Shen, Y. Song, P. Cao, Z. Zhang, J. Krumm, and M. Ali, "An Interactive Map-based System for Visually Exploring and Cleaning GPS Traces," in *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. Chicago IL USA: ACM, Nov. 2019, pp. 572–575. [Online]. Available: https://dl.acm.org/doi/10.1145/3347146.3359105

[18] J. Zhang and Y. Sun, "An Automatic Data Cleaning Method for GPS Trajectory Data on Didi Chuxing GAIA Open Dataset Using Machine Learning Algorithms," in *2019 6th International Conference on Systems and Informatics (ICSAI)*. Shanghai, China: IEEE, Nov. 2019, pp. 1522–1526. [Online]. Available: https://ieeexplore.ieee.org/document/9010403/

[19] A. De Brébisson, E. Simon, A. Auvolat, P. Vincent, and Y. Bengio, "Artificial neural networks applied to taxi destination prediction," in *Proceedings of the 2015th International Conference on ECML PKDD Discovery Challenge - Volume 1526*, ser. ECMLPKDDDC'15. Aachen, DEU: CEUR-WS.org, Sep. 2015, pp. 40–51.

[20] P. Zhao, A. Zhang, C. Zhang, J. Li, Q. Zhao, and W. Rao, "ATR: Automatic Trajectory Repairing With Movement Tendencies," *IEEE Access*, vol. 8, pp. 4122–4132, 2020, conference Name: IEEE Access.

[21] D. Sbetti and S. Tessaris, "Track annotation app," Feb. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.5977242

[22] I. Skog and P. Handel, "In-car positioning and navigation technologies - a survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 1, pp. 4–21, 2009.

[23] A. Graves, *Supervised sequence labelling with recurrent neural networks*, ser. Studies in computational intelligence. Berlin New York: Springer, 2012, no. v. 385.

[24] S. R. Bashir and V. Misic, "Detecting Fake Points of Interest from Location Data," in *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, Dec 2021, pp. 5347–5356.

[25] S. Dabiri, N. Marković, K. Heaslip, and C. K. Reddy, "A deep convolutional neural network based approach for vehicle classification using large-scale GPS trajectory data," *Transportation Research Part C: Emerging Technologies*, vol. 116, p. 102644, Jul 2020.

[26] K. Greff, R. K. Srivastava, J. Koutnk, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A Search Space Odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017, conference Name: IEEE Transactions on Neural Networks and Learning Systems.

[27] S. Wrtz and U. Ghner, "Driving Style Analysis Using Recurrent Neural Networks with LSTM Cells," *Journal of Advances in Information Technology*, vol. 11, p. 1, Jan. 2020.

[28] "TensorFlow Lite | ML for Mobile and Edge Devices," Dec 2021, [Online; accessed 4. Feb. 2022]. [Online]. Available: https://www.tensorflow.org/lite

[29] D. Sbetti and S. Tessaris, "Gps tracks: transform and model," Feb. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.5978571

[30] S. Dyer and J. Dyer, "Cubic-spline interpolation. 1," *IEEE Instrumentation Measurement Magazine*, vol. 4, no. 1, pp. 44–46, Mar. 2001, conference Name: IEEE Instrumentation Measurement Magazine.

[31] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 03 1960. [Online]. Available: https://doi.org/10.1115/1.3662552

[32] D. Sbetti and S. Tessaris, "Gpsclean," Feb. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.5978000

 ISBN: 978-1-61208-970-6