- Supplementary Material - GADformer: A Transparent Transformer Model for Group Anomaly Detection on Trajectories

Anonymous Authors
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

This document describes supplementary details of our GADFormer approach according to datasets, architectures, model training, hyperparameters and extended experiment results. In addition to that it contains further statements distinguishing our approach from related work. Datasets and code can be found at our repository for supplementary material[1].

I. GADFormer

A. Training

Algorithm 1 describes the training steps of our approach. The inputs of our algorithm are trajectory dataset $\mathcal{D}_\Lambda$ as group dataset $\mathcal{D}_\mathcal{G}$, the GADFormer model Ψ with initialized parameters and the loss objective function $\mathcal{L}_{BCE}$. Furthermore, we use the algorithm parameters dataset split ratios for train, validation and test, a model optimizer, the total number of epochs, batch size bs, learning rate η, weight decay and learning rate schedulers with patience parameter are in use. The output of the algorithm is model Ψ_{best} with parameters leading to the lowest validation loss, its GA scores $\hat{y}_{score_{[trn,vld,tst]}}$ and the related losses $\mathcal{L}_{[trn,vld,tst]}$. After variable initialization (lines 1-4) we repeat the training for the given number of epochs or until the model converges (line 5, 23, 24). In each training epoch (lines 5-25) we pass the group (trajectory) batches to the model, return its group abnormality probability $\hat{p}$ and use it to calculate the binary cross entropy loss $\mathcal{L}$ and the related group anomaly score $\hat{y}_{score}$. This is done for the training set (lines 6-11), the validation set (lines 12-16) and test set (lines 26-30) equally, just during training the gradients are calculated and model weight updates are done (line 9). At the end of each training epoch the best model with losses and GA scores is kept saved (lines 17-21) based on the validation loss $\mathcal{L}_{vld}$. Finally, as described for the output, the best model with related losses and group anomaly scores is returned (line 31).

B. Model Transparency

Despite the argument of [1], that "attention modules do not provide meaningful explanations", we could successfully

[1]https://anonymous.4open.science/r/gadf-994C

Algorithm 1 GADFormer Training Algorithm Pseudo Code

Input: groups $\mathcal{D}_\mathcal{G}$, model Ψ, loss objective $\mathcal{L}$
Parameter: ratios, optimizer, epochs, bs, lr η, wd, sched, patience
Output: model Ψ_{best}, GA scores $\hat{y}_{score_{[trn,vld,tst]}}$, losses $\mathcal{L}_{[trn,vld,tst]}$

1: $epoch = 0, \mathcal{L}_{best} = \infty$, earlystop=0, best=$\emptyset$.
2: opt = optimizer(Ψ, η, wd, sched).
3: split $\mathcal{D}_\mathcal{G}$ into $\mathcal{D}_{train}$, $\mathcal{D}_{valid}$, $\mathcal{D}_{test}$ by ratios.
4: $p = 0$.
5: **while** epoch $<$ epochs or earlystop $>$ patience **do**
6: **for all** $\mathcal{G}_m$ in $\mathcal{D}_{train}$ **do**
7: $\hat{p}_{trn} = \Psi(\mathcal{G}_m)$
8: $\mathcal{L}_{trn} = \mathcal{L}(p, \hat{p}_{trn})$
9: $w_\Psi = w_\Psi - \eta \frac{\partial \mathcal{L}_{trn}}{\partial \mathcal{G}_m}$
10: $\hat{y}_{score_{trn}} = \hat{p}_{trn}$
11: **end for**
12: **for all** $\mathcal{G}_m$ in $\mathcal{D}_{valid}$ **do**
13: $\hat{p}_{vld} = \Psi(\mathcal{G}_m)$
14: $\mathcal{L}_{vld} = \mathcal{L}(p, \hat{p}_{vld})$
15: $\hat{y}_{score_{vld}} = \hat{p}_{vld}$
16: **end for**
17: **if** $\mathcal{L}_{vld} < \mathcal{L}_{best}$ **then**
18: $\mathcal{L}_{best} = \mathcal{L}_{vld}$.
19: best = $(\Psi, \hat{y}_{score_{[trn,vld]}}, \mathcal{L}_{[trn,vld]})$
20: earlystop=0
21: **end if**
22: $\eta = sched(\eta, \mathcal{L}_{vld})$.
23: earlystop+=(0,1)$[\mathcal{L}_{vld} \geq \mathcal{L}_{best}]$
24: epoch+=1
25: **end while**
26: **for all** $\mathcal{G}_m$ in $\mathcal{D}_{tst}$ **do**
27: $\hat{p}_{tst} = \Psi_{best}(\mathcal{G}_m)$
28: $\mathcal{L}_{tst} = \mathcal{L}(p, \hat{p}_{tst})$
29: $\hat{y}_{score_{tst}} = \hat{p}_{tst}$
30: **end for**
31: **return** best $\cup(\hat{y}_{score_{tst}}, \mathcal{L}_{tst})$

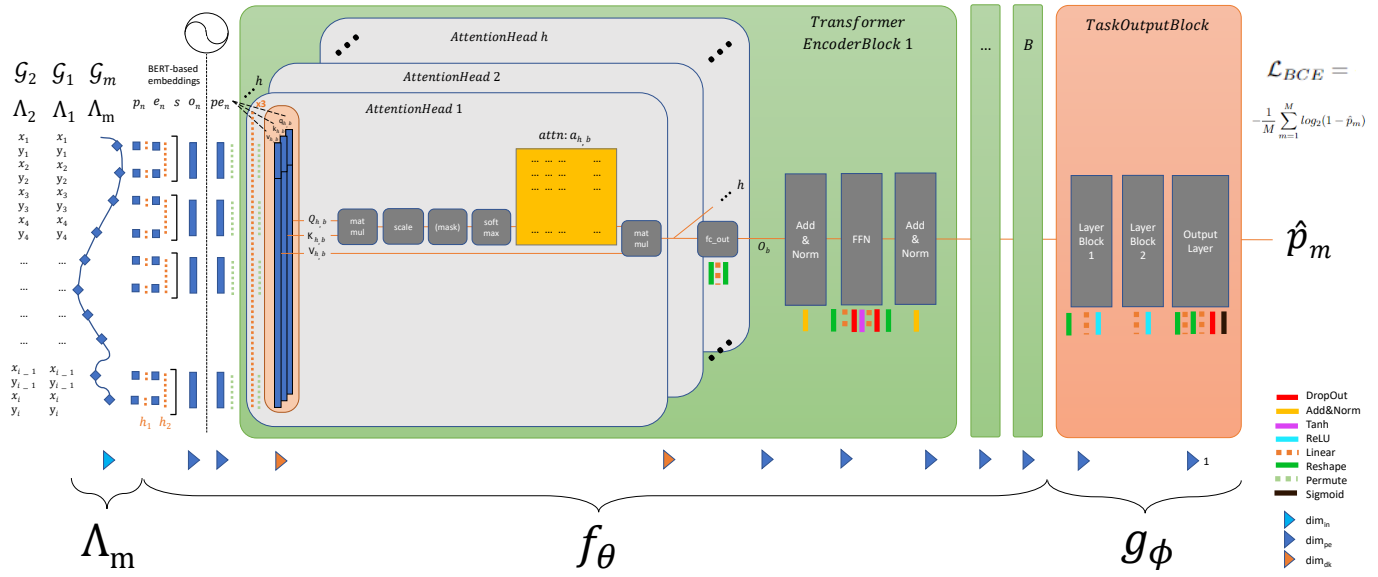

Fig. 1. GADFormer architecture overview.

utilize it for model inspection in terms of plausible layer-wise correlations between ground truths and feature extraction capabilities as Fig. 2, Fig. 3 and related performance metrics show. Although it might be worth for further investigation, BAS does not aim like [1] or [2] for explainability in terms of how much each feature, in our case each group member instance, contributes to the predicted model output trying to answer why a model made that prediction. Instead, we address their findings of "heads only specialize to some extend and sometimes take into account a considerable amount of non-related tokens"[2] and "the existence of alternative heatmaps that yield equivalent predictions"[1], by utilizing the average attention across the group of each layers attention heads aiming for an aggregated attention per layer and with that for the attention-heads-based group anomaly score BAS following the assumption that in case of the aggregated attention of a group of layer heads is anomalous then also the model input, in our case the group member instances of a trajectory, is anomalous.

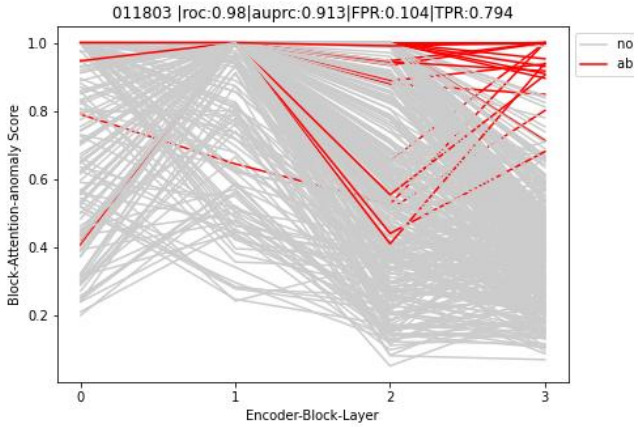

Fig. 2. BAS in case of good model performance.

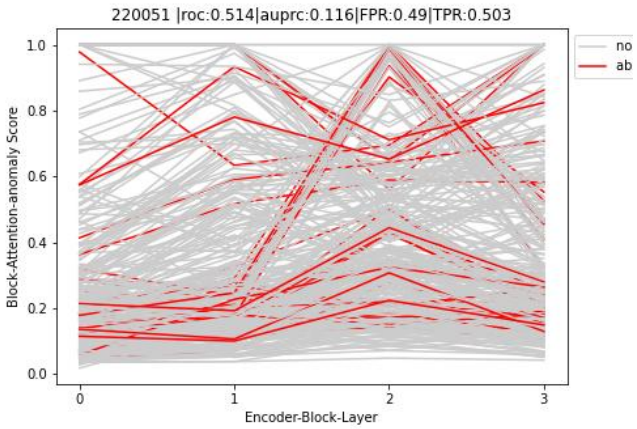

Fig. 3. BAS in case of bad model performance.

II. Experiments

In this section we evaluate the performance of our GADFormer approach on synthetic and real-world datasets and compare it against MTGAD and GRU, two technically most related works for individual trajectory anomaly detection.

A. Experimental Setup and Datasets

For our experiments we used an Ubuntu Focal 20.04.5 LTS server with AMD Ryzen 7 3700X 8-Core Processor with 16 CPUs, 64GB RAM and a 16GB NVIDIA RTX A4000 GPU. We conducted our experiments on four trajectory datasets, which are an own synthetic dataset[1] (cf. Fig. 8) with several noise and novelty data variants as well as three real-world datasets from Amazon[2] (cf. Fig. 9) describing walk or driving paths, from Deutsche Bahn describing cargo

[2]https://github.com/amazon-science/goal-gps-ordered-activity-labels

TABLE I
DATASET OVERVIEW.

dataset	setting	all	n	a	trajLen
synthetic[1]	unsup	3400	3083	317	72
synthetic[1]	semi	3400	3271	129	72
amazon[2]	unsup	805	760	45	72
amazon[2]	semi	776	760	16	72
dbcargo[3]	unsup	272	229	43	72
dbcargo[3]	semi	245	229	16	72
brightkite[4]	unsup	2241	2033	208	500
brightkite[4]	semi	2108	2033	75	500

ship container routes[3] (cf. Fig. 6) and from brightkite describing checkin routes [4] (cf. Fig. 7). The latter represents long trajectory sequences with a length 500 steps whereas amazon and dbcargo has a trajectory length of 72 steps. For real world datasets like amazon routes, brightkite checkin sequences or dbcargo ship container routes we have no domain expert verified ground truth labels available why we investigated in several methods delivering reasonable pseudo labels. One of them, using a z-score range of]-2.1;2.1[in order to label individual anomalous trajectories based on their PCA embedding turned out to be inappropriate since PCA does not preserve local neighborhoods for its embeddings. To address this issue we utilize UMAP[3] embeddings as basis for the required pseudo labels. These are obtained by applying OPTICS[4] clustering, whose noise labels meet not only our expectation according to target anomaly ratios (between 5 and 20%), also after manual investigation the samples showed pattern being actually anomalous compared to the majority of normal ones. Since all evaluated approaches have with these pseudo labels equal difficult conditions we consider them as appropriate for our tests. We split our data in a train-, valid- and testset with a ratio of 0.9 for normal data for each set (except for semi-supervised training when no abnormal data is in the training set). Each prepared dataset provides trajectories as trajectory steps, one step per row, with columns for Entity (TrajectoryID), Step (TrajectoryStepID), Coordinates (e.g. XCoord, YCoord) and Label (1 - anomalous, 0 - normal, all step records contain the same trajectory label). Further details can be seen in Table I.

For our ablation studies we first created a synthetic trajectory dataset with randomly chosen trajectory step coordinates in normal case and pattern-driven trajectory step coordinates in abnormal case. The default anomaly is represented by a random sequential amount of trajectory steps not changing one of its coordinates. These default anomalies get distorted randomly in case of noise ablation study up to a defined ratio (0. to 0.5). For the novelty ablation study we created anomalies with pattern different to the default anomaly pattern (e.g. sequential trajectory steps which shape a half-moon).

[3]https://data.deutschebahn.com/dataset/data-sensordaten-schenker-seefrachtcontainer.html

[4]https://snap.stanford.edu/data/loc-brightkite.html

TABLE II
HYPERPARAMETERS FOR TRAINING AND ARCHITECTURE.

Params	synthetic	amazon	dbcargo	brightkite
dim_{in}	72	72	72	500
dim_{feat}	2	2	2	2
dim_{h1}	16	16	16	16
seg_len	2	2	2	2
dim_{pe}	72	72	72	500
dim_{dk}	72	72	72	500
dim_{ffn}	2048	2048	2048	2048
heads	12	12	12	8
b	4	4	4	4
TLayers	3	3	3	3
bs	256	256	256	256
lr	1e-3	1e-3	1e-2	1e-4
wd	0	0	0	0
dropout	0	0	0	0
epochs	150	150	150	150

All hyperparameters are empirically selected by grid search based on validation loss convergence and additionally validated by model inspection with our proposed block attention anomaly score to find the ideal training parameters and model architecture avoiding overfitting or insufficient model complexity. We use a segment length of 2 for BERT segmentation, modeling two consecutive trajectory step coordinates as one segment. Additionally, we use progressive training, keeping task layers frozen until the validation loss converges for feature extraction layers.

For our experiments with a synthetic dataset and the real-world datasets amazon, dbcargo and brightkite, we chose the model- and hyperparameters according to Table II.

B. MainTulGAD

MainTulGAD (MTGAD) is an adapted approach of one of the most related works MainTUL [5] from technical design perspective. In order to compare our approach against theirs we had to modify MainTUL to be able to process sequential continuous coordinates instead of categorical checkin location and checkin timestamps as well as to predict a binary anomaly label instead of multi-class probabilities to predict the most probable user for a given checkin-sequence. For comparison of our modifications please see the following Fig. 4 and Fig. 5. In Fig. 5 we highlight our changes based on the original draft from Fig. 4 in red. Starting with the input, MainTUL originally uses categorical POI (Point Of Interest) sequences with hourly categorical checkin-timeranges. These input formats does not match semantically our continuous coordinates sequences, for which they had to be replaced. Instead of augmented long-term sequences we reuse the input trajectory coordinates and apply augmentation according to MainTUL by segmenting a new similar trajectory by randomly chosing segments of its k nearest neighbors based on the input trajectory's standardized UMAP embedding. These two input trajectories are then used twice as an embedding within one training epoch, once by the student encoder (RNN) and once by the teacher encoder (Transformer), whereas the output distributions are mapped together as close as possible utilizing KL-divergence. Since the task of MTGAD is to predict binary labels instead of

probabilities for most probable users for a trajectory we use binary cross entropy instead of categorical cross entropy as loss function. Similar as the original approach also MTGAD uses the predictions of the student encoder (RNN) for evaluation. Hyperparameters for MTGAD had been used as in the original paper except for the memory intensive dataset brightkite with a sequence length of 500. Hence, the hyperparameters are $poi\_nums = 72$ (500 for brightkite), $d\_model = 512$ (64 for brightkite), $num\_heads = 8$ (2 for brightkite), $num\_student\_layers = 2$, $num\_teacher\_layers = 4$, $temperature = 10$, $\lambda = 1$, $clipping\_value = 5$, $bs = 256$, $lr = 1e{-}4$ (1e-5 for U-Synthetic and dbcargo), $epochs = 100$, $wd = 0$, $dropout = 0$ and $k = 8$ (for self-supervised kNN-trajectory-augmentation).

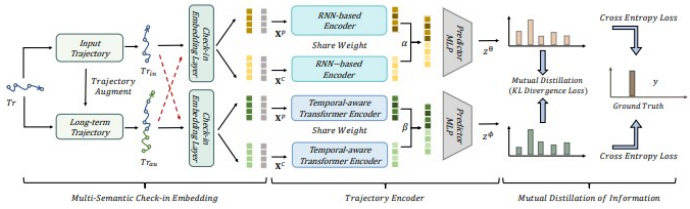

Fig. 4. Original MainTUL architecture from [5].

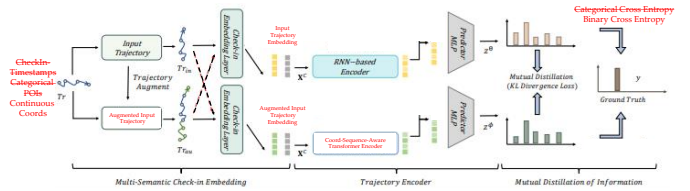

Fig. 5. Modified MainTulGAD (MTGAD) architecture adapted from [5].

C. GRU

In order to distinguish the capabilities of GRU as clearly as possible from these of GADFormer the inputs and loss targets had to be kept equal. For GRU approach only the Transformer layers are replaced by GRU layers. After grid search it turned out 2 GRU layers are the ideal choice instead of using 4 as for the GADF transformer layers. Moreover we use the sequence lengths as $hidden\_size$. The remaining hyperparameters are kept equal to that of GADFormer except $lr = 1e - 3$ for dbcargo.

D. Shared techniques

For all tested approaches we used RAdam as optmizer, early stopping and learning rate scheduling with patience thresholds of 20 and 10 respectively. Furthermore, we used seeds from 0 to 9.

E. Traditional methods

We extend our experiments also to technically non-related non-deep approaches[5] since they have still domain relevance. Therefore, we opted for traditional outlier detection methods

[5]https://scikit-learn.org/stable/modules/outlier_detection.html

TABLE III
RESULTS ON SYNTHETIC AND REAL WORLD DATASETS FOR (U)NSUPERVISED- AND S(E)MI-SUPERVISED SETTING.

	dataset	amazon		brightkite		dbcargo		synthetic	
		auroc	auprc	auroc	auprc	auroc	auprc	auroc	auprc
U	EE	0.755	0.135	0.458	0.088	0.477	0.154	0.469	0.088
	OCSVM	0.676	0.081	0.753	0.170	0.730	0.259	0.737	0.169
	LOF	0.717	0.092	0.859	0.264	0.618	0.200	0.551	0.155
	IF	*0.992*	0.778	0.689	0.199	0.716	0.250	0.802	0.422
	GRU	0.642	0.539	0.786	0.552	0.718	**0.664**	0.775	*0.431*
	MTGAD	0.956	*0.872*	*0.907*	*0.656*	*0.779*	*0.577*	*0.87*	0.371
	GADF	**0.997**	**0.955**	**0.948**	**0.672**	**0.797**	0.478	**0.982**	**0.887**
E	EE	0.635	0.182	0.447	0.087	0.490	0.157	0.722	0.155
	OCSVM	0.857	0.167	0.850	0.252	**0.838**	0.368	0.735	0.164
	LOF	0.906	0.233	**0.942**	0.465	0.571	0.279	0.878	*0.779*
	IF	*0.992*	*0.778*	0.861	0.330	0.635	0.206	*0.910*	0.363
	GRU	0.545	0.394	0.711	0.396	*0.701*	**0.64**	0.799	0.52
	MTGAD	0.445	0.325	0.887	*0.604*	0.678	*0.526*	0.889	0.549
	GADF	**0.998**	**0.976**	*0.933*	**0.612**	*0.801*	0.507	**0.997**	**0.982**

TABLE IV
RESULTS ON SYNTHETIC DATASET WITH NOISE ABLATIONS FOR (U)NSUPERVISED AND S(E)MI-SUPERVISED SETTING

	exp	noise .0		noise .2		noise .5	
		auroc	auprc	auroc	auprc	auroc	auprc
U	EE	0.503	0.093	0.402	0.085	*0.719*	0.160
	OCSVM	0.698	0.150	0.686	0.145	0.554	0.103
	LOF	0.499	0.092	0.512	0.104	0.526	0.124
	IF	0.859	*0.527*	0.768	*0.342*	0.718	*0.253*
	GRU	0.766	0.514	0.731	0.383	0.626	0.165
	MTGAD	*0.869*	0.376	*0.822*	0.256	0.717	0.149
	GADF	**0.97**	**0.892**	**0.949**	**0.831**	**0.863**	**0.537**
E	EE	0.707	0.148	0.698	0.145	0.737	0.162
	OCSVM	0.702	0.149	0.690	0.144	0.653	0.132
	LOF	0.878	*0.779*	0.811	*0.657*	0.554	0.191
	IF	0.832	0.291	0.846	0.304	*0.838*	0.304
	GRU	0.788	0.585	0.759	0.479	0.665	0.223
	MTGAD	*0.952*	0.766	*0.89*	0.547	0.792	*0.316*
	GADF	**0.989**	**0.95**	**0.98**	**0.919**	**0.944**	**0.803**

TABLE V
RESULTS ON SYNTHETIC DATASET WITH NOVELTY ABLATIONS FOR (U)NSUPERVISED AND S(E)MI-SUPERVISED SETTING.

	exp	novelty .0		novelty .01		novelty .05	
		auroc	auprc	auroc	auprc	auroc	auprc
U	EE	0.487	0.091	0.467	0.088	0.491	0.091
	OCSVM	0.698	0.150	0.759	0.177	0.738	0.168
	LOF	0.499	0.092	0.559	0.141	0.527	0.101
	IF	0.846	0.506	0.885	0.558	0.840	0.401
	GRU	0.766	*0.514*	0.832	0.585	0.818	0.496
	MTGAD	*0.882*	0.42	*0.935*	*0.588*	*0.923*	*0.504*
	GADF	**0.97**	**0.892**	**0.978**	**0.865**	**0.969**	**0.726**
E	EE	0.707	0.148	0.667	0.133	0.700	0.145
	OCSVM	0.702	0.149	0.734	0.162	0.717	0.156
	LOF	0.878	0.779	0.940	0.805	0.906	0.555
	IF	0.854	0.323	0.853	0.289	0.898	0.333
	GRU	0.788	0.585	0.849	0.652	0.841	0.574
	MTGAD	*0.964*	*0.802*	*0.977*	*0.867*	*0.97*	*0.797*
	GADF	**0.989**	**0.95**	**0.986**	**0.921**	**0.986**	**0.841**

like IF[6], LOF[7], OCSVM[8] and RobustCovariance (EE)[6] using default parameters and conducting gridsearch for neighbors k according to [9].

F. Result Discussion

Comparing the performances of GADFormer (GADF) with these from the main paper one observes that the superiority of GADF remains constant for noise and novelty ablation studies (cf. Table IV and Table V). Only in semi-supervised setting of Table III AUROC of GADF (0.933) is slightly worse than LOF (0.942) on brightkite and on dbcargo GADF (0.801) performs second best behind OCSVM (0.838).

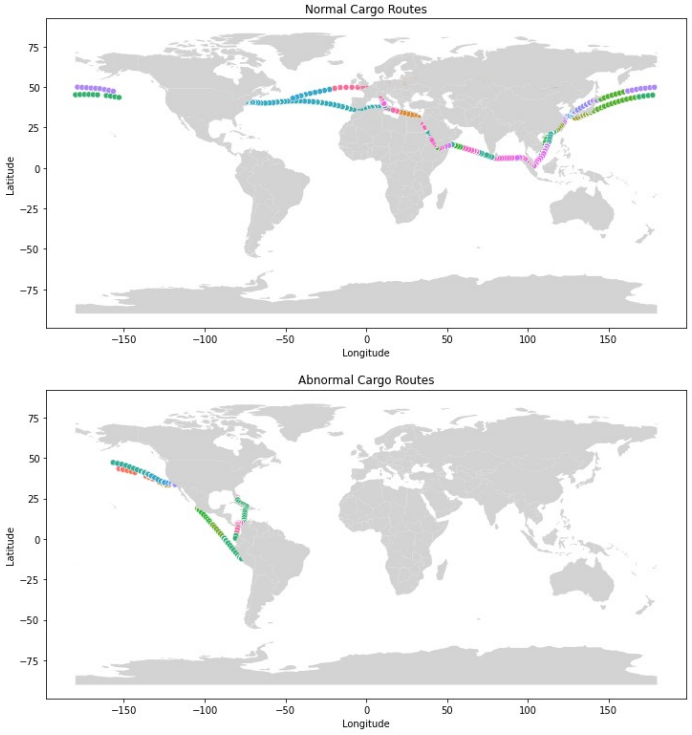

Fig. 6. Reconstructed cargo container routes of Deutsche Bahn.

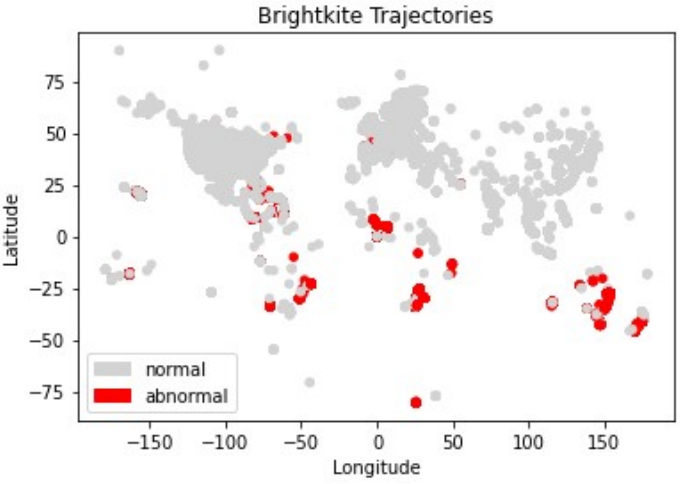

Fig. 7. Brightkite checkin routes.

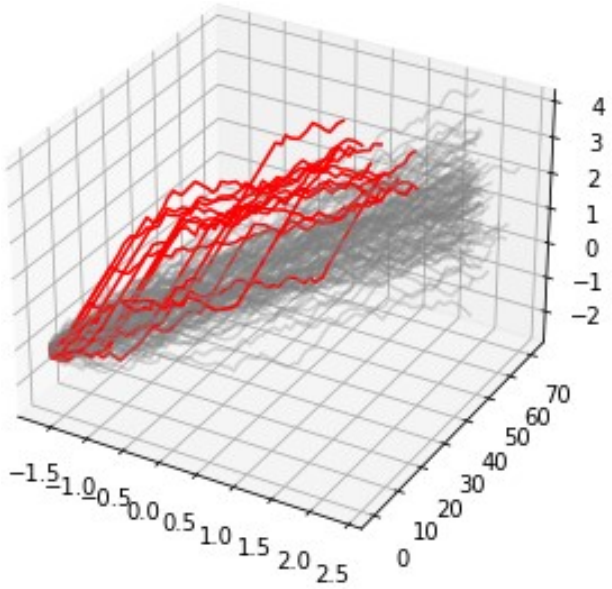

Fig. 8. Synthetic trajectory data.

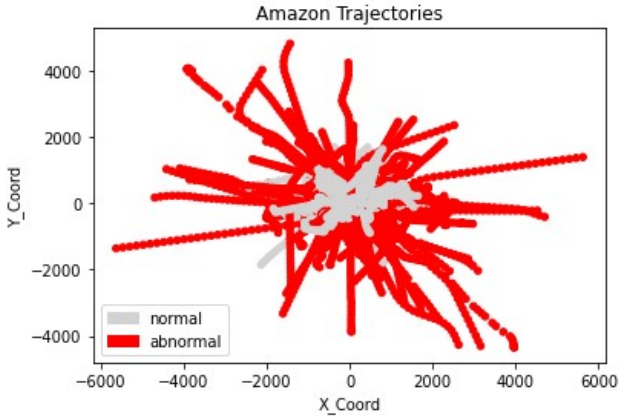

Fig. 9. Amazon trajectories.

REFERENCES

[1] Sarthak Jain and Byron C. Wallace. *Attention is not Explanation*. 2019. arXiv: 1902.10186 [cs.CL].

[2] Joris Baan et al. "Do Transformer Attention Heads Provide Transparency in Abstractive Summarization?" In: *ArXiv* abs/1907.00570 (2019).

[3] Leland McInnes and John Healy. "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction". In: *ArXiv* abs/1802.03426 (2018).

[4] Mihael Ankerst et al. "OPTICS: Ordering Points to Identify the Clustering Structure". In: *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*. SIGMOD '99. Philadelphia, Pennsylvania, USA: Association for Computing Machinery, 1999, pp. 49–60. ISBN: 1581130848.

[6]https://scikit-learn.org/stable/modules/generated/sklearn.covariance.EllipticEnvelope.html

[5] Wei Chen et al. “Mutual Distillation Learning Network for Trajectory-User Linking”. In: *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*. International Joint Conferences on Artificial Intelligence Organization, July 2022, pp. 1973–1979.

[6] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. “Isolation Forest”. In: *2008 Eighth IEEE International Conference on Data Mining* (2008), pp. 413–422.

[7] Markus M. Breunig et al. “LOF: identifying density-based local outliers”. In: *ACM SIGMOD Conference*. 2000.

[8] Koby Crammer and Gal Chechik. “A Needle in a Haystack: Local One-Class Optimization”. In: *Proceedings of the Twenty-First International Conference on Machine Learning*. ICML ’04. 2004, p. 26.

[9] Guilherme Oliveira Campos et al. “On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study”. In: *Data Mining and Knowledge Discovery* 30 (2016), pp. 891–927.