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

Andreas Lohrer* Darpan Malik† Claudius Zelenka‡ Peer Kröger§

This document describes supplementary details of our GADFormer approach according to datasets, architectures, model training, hyperparameters and evaluation. In addition to that it contains further statements distinguishing our approach from related work.

1 GADFormer

1.1 Training Anomalous trajectories are rare by definition and labeling by domain experts tends to be rather expensive. We address this challenge by two different learning settings which are: 1) Unsupervised learning, which requires no labels at all under the assumption that the ratio of anomalous trajectories is low and has no remarkable influence during model training. 2) Semi-Supervised Learning, which relies on verified normal samples only. As proposed in the section before, these learning settings allow us to set a fix auxiliary target probability $p_m = 0$, so that no ground truth for abnormal trajectories is needed for the GADFormer 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 Φ 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).

1.2 Model Transparency Despite the argument of [4], that "attention modules do not provide meaningful explanations", we could successfully 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 [4] 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"[4], 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.

2 Experiments

In this section we evaluate the performance of our GADFormer approach on synthetic and real-world datasets and compare it against GRU, one of the state of the art methods for individual trajectory anomaly detection.

*alo@informatik.uni-kiel.de, Kiel University
†stu225397@mail.uni-kiel.de, Kiel University
‡cze@informatik.uni-kiel.de, Kiel University
§pkr@informatik.uni-kiel.de, Kiel University

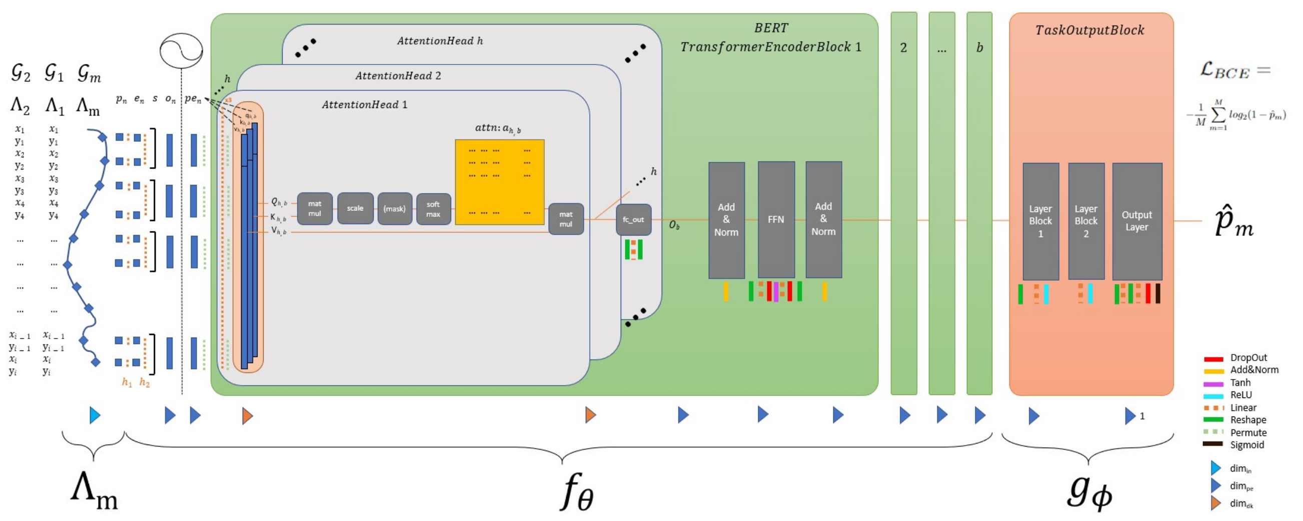

Figure 1: GADFormer architecture overview.

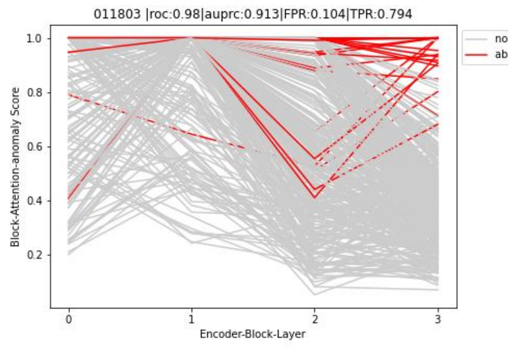

Figure 2: BAS in case of good model performance.

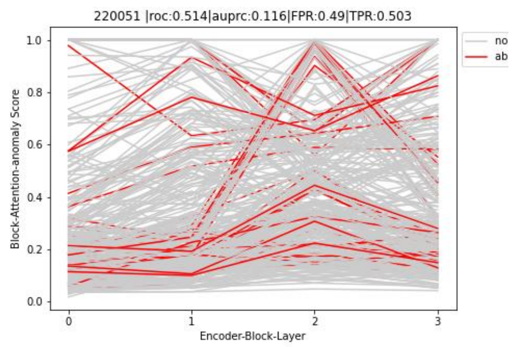

Figure 3: BAS in case of bad model performance.

2.1 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 three trajectory datasets, which are an own synthetic dataset[1] with several noise and novelty data variants as well as two real-world datasets from amazon[2] describing walk or driving paths and brightkite [3]. The latter represents long trajectory sequences with a length 500 steps whereas amazon has a trajectory length of 72 steps. For real world datasets like amazon routes, brightkite checkin sequences or cargo container routes of Deutsche Bahn[4] 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[5] embeddings as basis for the required pseudo labels. These are obtained by applying OPTICS[1] 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

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

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

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

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

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($\Psi$, $\eta$, 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 $\cup(\hat{y}_{score_{tst}}, \mathcal{L}_{tst})$
```

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 1.

For our ablation studies and 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).

Table 1: Dataset Overview.

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

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.

The code for GADFormer is implemented in Python utilizing PyTorch and PyTorch Lightning. For model training, we use early stopping as well as gradient clipping to avoid exploding gradients. Furthermore, weight initialization is used in order to achieve better convergence. For our experiments with a synthetic dataset and the real-world datasets amazon, dbcargo and brightkite, we chose the training hyperparameters according to Table 2. The synthetically generated trajectory dataset consists of group member instances (trajectory steps), one per row, where one group member has different attributes such as id, sequence step, xcoord and ycoord. A GAD label is defined for one complete group (trajectory) stating whether a group (an individual trajectory) is anomalous (1) or normal (0). The used datasets can be found among the uploaded supplementary material[5].

2.2 MainTulGAD MainTulGAD (MTGAD) is an adapted approach of one of the most related works

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

Table 2: 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

MainTUL [3] 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 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 their described augmentation by segmenting a new similar trajectory by randomly chosing segments of its k nearest neighbors based on their standardized UMAP embedding. Both input trajectories are 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).

2.3 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 transformer layers. Moreover we use the sequence lengths as *hidden_size*. The remaining hyperparameter are kept equal to that of GADFormer except $lr = 1e-3$ for dbcargo.

2.4 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.

2.5 Evaluation For the evaluation of our model, we follow the goals of having a low miss rate (false negatives) as well as achieve as less false alerts (false positives) as possible. In addition to that, the quality of the model scores needs to be evaluated. Therefore and to be comparable to related approaches, we evaluate the model performance by AUROC and AUPRC.

- $TPR = \frac{TP}{P}; FPR = \frac{FP}{N}; FNR = 1 - TPR$
- Precision $= \frac{TP}{TP+FP}$
- Recall $= \frac{TP}{TP+FN}$

Areas Under the Curve (AUC) for thresholds $[t, 1]$:

- AUPRC = AP = Precision vs. Recall
- AUROC = TPR vs. FPR

References

[1] 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. DOI: 10.1145/304182.304187. URL: https://doi.org/10.1145/304182.304187.

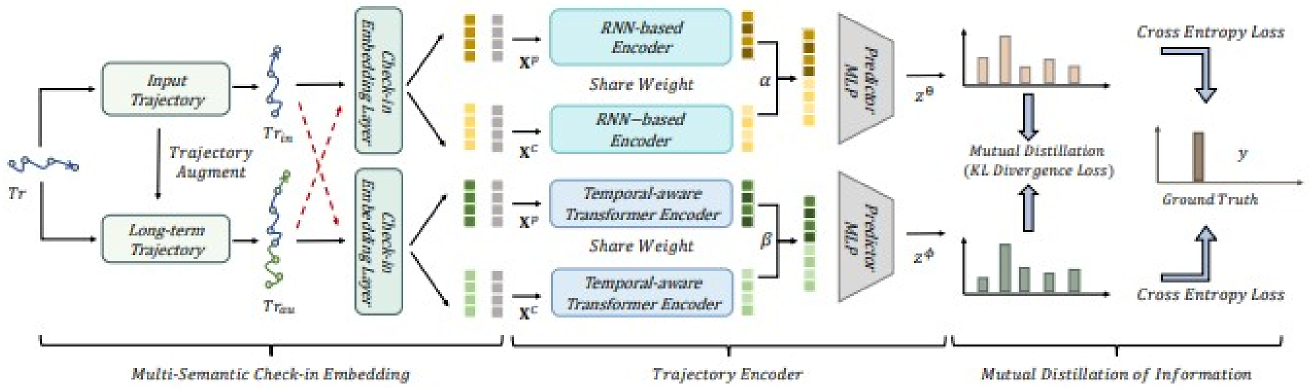

Figure 4: Original MainTUL architecture from [3].

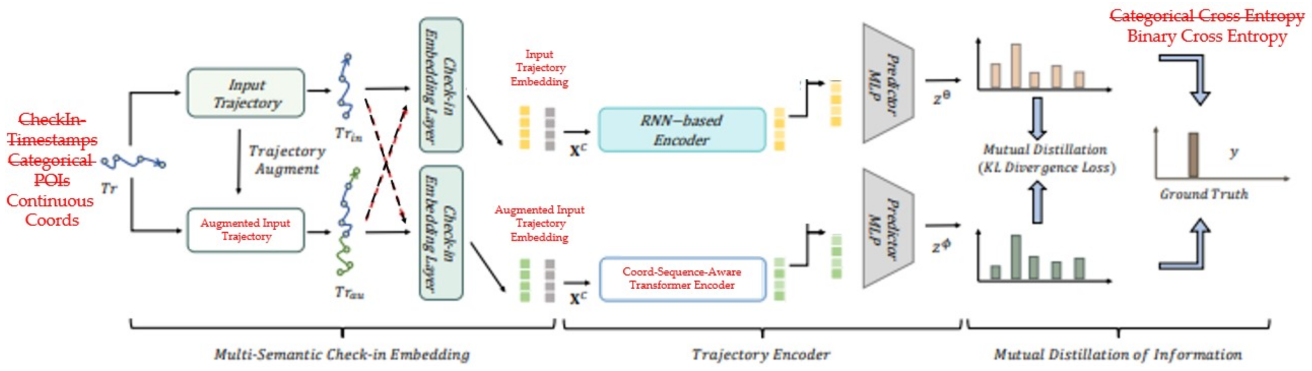

Figure 5: Modified MainTulGAD (MTGAD) architecture adapted from [3].

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

[3] 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.

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

[5] Leland McInnes and John Healy. “UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction”. In: *ArXiv* abs/1802.03426 (2018). URL: `https://api.semanticscholar.org/CorpusID:3641284`.