Augmented temporal grid for Graph Neural Networks and Anomaly Detection in Spacecraft

Gamze Naz Kiprit[*,1,3], Andreas Koch[*,2,3](✉), Michael Petry[2,3] and Martin Werner[2]

1. *Technical University of Munich, Germany; TUM School of Computation, Information and Technology, Department of Electrical Engineering*
2. *Technical University of Munich, Germany; TUM School of Engineering and Design, Department of Aerospace and Geodesy, Professorship Big Geospatial Data Management*
3. *Airbus Defence and Space GmbH, Taufkirchen, Germany; Space Systems Engineering, Product Realisation Germany*

* *Contributed equally*

✉ andreas.c.koch@tum.de

Abstract

Graph Neural Networks for spatiotemporal problems incorporate a temporal module operating on a temporal grid of interconnected subsequent time steps. In this work, we explore adding connections to future time steps to the temporal grid in order to extract unique features for the task of anomaly detection in time series. We introduce strategies for determining future connections utilizing the autocorrelation function and different random sampling techniques. The effectiveness of resulting Graph Neural Networks is demonstrated on multiple anomaly detection benchmarks, including spacecraft telemetry datasets. A selection of Graph Neural Network models is further deployed on the AMD-Xilinx Versal AI Core SoC to measure the execution time and resource utilization when processing telemetry data.

Keywords

Anomaly detection · co-processor · edge AI · graph neural networks · time series analysis · telemetry data

1 Introduction

Numerous activities on Earth depend on satellites, such as radio signals, navigation and, in many countries, the internet access. These machines must execute critical tasks with exceptional reliability, even in the event of a failure. Fault Detection, Isolation and Recovery (FDIR) is a key strategy to fulfill

the purpose of reliability and aims to identify and isolate the faults as early as possible [1]. The most common approach in fault detection is to detect out-of limit (OOL) events. These events are identified by examining whether the design limits of any sensor, instrument, or subsystem are crossed. Instead of only applying fixed limits, behavioral system monitoring can be applied to analyze patterns in telemetry data and discover ”anomalies” which are to be reported to the operations team. This fault detection technique has the capability to detect faults earlier than OOL boundaries [2] and in case of high certainty by the monitoring system, an automatic transition to safe mode can be performed. Implementing behavioral system monitoring onboard of spacecraft presents the opportunity of analyzing data not being sent to the ground operations team due to downlink limitations. For this purpose, the deployment of behavioral system monitoring algorithms to different space-grade hardware needs to be studied. Algorithms applied within behavioral system monitoring belong to the research field of anomaly detection in time series. It encompasses many algorithms of different categories with an origin in fields such as data mining, statistics, and machine learning (ML) [3]. Deep learning algorithms are in particular attractive for onboard behavioral monitoring, as they can take advantage of specialized hardware already being developed for accelerating a multitude of deep learning algorithms. Examples for accelerators include the AMD-Xilinx Vitis AI Deep Learning Processor (DPU) and the Matlab Deep Learning Processor provided by the Deep Learning HDL toolbox.

In this work, we propose a novel algorithm for anomaly detection, which is based on a Graph Neural Network (GNN). It treats the time steps of a time series as graph nodes and adds connections to future time steps to extract a unique feature representation. To our knowledge, it is the first application of a GNN for anomaly detection, where graph nodes represent time steps of a time series and are not only connected to subsequent time steps as a ”temporal grid”. As graph nodes usually represent distinct channels of a multivariate time series with known channel relations that can inform graph edge weights, an important question in our approach is how to connect graph nodes representing time steps with no prior information on their relations. For this purpose, we present two strategies for randomly sampling graph edges, as well as a static connection scheme and a strategy for finding graph connections via the autocorrelation of a time series. The newly developed GNN is deployed on the AMD-Xilinx Versal AI Core SoC to show its potential for deployment on space-grade hardware.

Our contribution encompasses

- 4 strategies for augmenting the temporal grid in Graph Neural Network layers,
- an evaluation of Graph Convolutional Networks constructed with said strategies and
- the resource utilization and runtime of the proposed models when deployed to the AMD-Xilinx Versal AI Core SoC.

Section 2 establishes previous work in the field of anomaly detection in time series and in particular the use of GNNs for this problem. The notation regarding GNNs is introduced in section 3, as well as the strategies regarding augmentations to the temporal grid of GNN layers. The approach towards benchmarking the newly developed methods, the corresponding evaluation results and the onboard resource utilization are presented in section 4 and the paper is concluded in section 5.

2 Related Work

Algorithms for anomaly detection in time series can be categorized into six distinct categories: forecasting-based, reconstruction-based, encoding, distance-based, distribution and isolation tree methods [3].

2.1 Anomaly Detection

Both **forecasting** and **reconstruction**-based methods process data of a window of sequential time steps and learn a model to predict: a) the subsequent time step value in case of forecasting methods; and b) the original window in case of reconstruction methods. The underlying models are optimized in a semi-supervised or unsupervised manner. Semi-supervised in this context refers to the task of learning to predict exemplary data which is assumed to be nominal. Since the model only learns to represent nominal data, the idea is that it will fail to predict data which is not represented in the nominal exemplars. This deviation is usually captured by a distance metric applied to the prediction and actual value and the result of this distance metric is referred to as the ”nonconformity score” in this work. In contrast, unsupervised models such as ARIMA [4] are built directly on the test set. The survey by Chandola *et al.* [5] classifies them as a regression-based statistical model, as they learn to emulate the underlying statistical process generating the time series. For reconstruction-based approaches, unsupervised methods project the input data window to a predefined latent space, whereby information is lost. As this extra precision is often needed to capture anomalies, the deviation of anomalies to the original data window is expected to be larger than for nominal data [3]. Examples of forecasting-based methods include DeepAnT [6], N-BEATS

[7], Telemanom [8], Random Forests [9] and Gradient Boosting Decision Trees [10] and examples for reconstruction-based methods are Convolutional Autoencoder [11], USAD [12], OmniAnomaly [13] and RobustPCA [14].

Encoding methods denote approaches that encode subsequences of a time series in low dimensional latent spaces, where the nonconformity scores are directly computed from the latent space representations [3]. An example for this are grammar-based approaches, where grammar induction is applied to discretized subsequences and rarely used substrings in the grammar rules are found to be anomalies [15]. **Distance**-based methods rely on applying a distance metric to samples or subsequences of a time series to then find anomalies by the distance to nearest neighbors [16], [17], the local density [18], or by the distance to cluster centroids [19]. Schmidl *et al.* [3] define the **distribution** category by methods that fit a distribution to the data and find anomalies at the distribution tails. Note that this is usually directly applied to the test data in an unsupervised manner, although there exist semi-supervised methods which learn distributions on nominal training data. Lastly, **isolation tree** methods form an ensemble of random trees that isolate samples of a time series. An isolation tree splits all samples at every node in order to isolate individual samples. The path length averaged over all isolation trees is then taken as the measure for nonconformity, where a shorter average path length indicates higher nonconformity [20]. This method was later extended to perform dataset splits with rotating boundaries [21].

2.2 Graph Neural Networks in Anomaly Detection

Jin *et al.* [22] present how Graph Neural Networks (GNNs) have been used for time series forecasting and anomaly detection. They argue that for multivariate time series, GNNs are able to capture both temporal and spatial dependencies among variables better than recurrent models. Besides forecasting and reconstruction GNN methods in anomaly detection, ”relational discrepancy” methods assume that the relationships between variables are significantly shifted for anomalous periods. Focusing on time series forecasting, GNN methods are often made up of a spatial module capturing inter-variable dependencies and a temporal module for inter-temporal dependencies. Considering the fact that spatial modules are designed to capture meaningful features for a problem with geographically distributed sources for information, such as traffic prediction, and comparing the problem setting of analyzing sensor measurements all located within one spacecraft, we focus on presenting methods regarding the GNN temporal module. The temporal module is often implemented via individual recurrent, convolutional or attentive layers within a larger GNN architecture. ”Diffusion Convolutional Recurrent Neural Network” (DCRNN) [23] captures

temporal dependencies via a Gated Recurrent Unit (GRU) adapted to operate on a graph via diffusion convolution, whereas the models from the framework "Spatio-Temporal Graph Convolutional Networks" (STGCN) [24] extract temporal features via a gated graph convolution operating on a temporal grid of subsequent time steps. "Dynamic Spatiotemporal Graph Convolutional Neural Network" (DSTGCN) [25] uses a bi-directional LSTM applied to each node individually to capture temporal dependencies. While "Hierarchical Graph Convolution Network" (HGCN) [26] first applies a gated convolution in temporal domain, it further processes the spatiotemporal representations by a temporal attention mechanism. The "Graph Multi-Attention Network" (GMAN) [27] assigns each time step an embedding vector and accumulates spatiotemporal embeddings for each time step and node to then apply an attention mechanism across time steps to find non-linear correlations. An attention mechanism is also applied by "Attention Temporal Convolution Network" (ATCN) [28] to extract features from the temporal domain. This is equivalent to a fully connected temporal graph that applies the attention operation to all time steps within a time window. The topological relationship between sensors is captured by graph convolutional layers in "Graph Learning with Transformer for Anomaly Detection" (GTA) [29] and the temporal relationships are learned with a transformer model.

3 Methodology

While previous approaches use recurrent, convolutional or attentive layers operating on a graph which is a temporal grid of subsequent time steps, we choose to employ a GNN layer on a graph where time steps are not only connected to their subsequent time steps, but can also be connected to arbitrary future time steps. This graph design allows the GNN layer to extract a different set of features when compared to a temporal grid of subsequent time steps. The difference of features in turn is helpful for either a forecasting or reconstruction anomaly detection approach, as these approaches employ a model that learns how to represent nominal data within a feature space. In section 3.2, we further present 4 strategies for determining which time steps to connect within the graph. Let $S \coloneqq \{s_1, ..., s_t | s_i \in \mathbb{R}^M\}$ be a stream with consecutive M-dimensional stream vectors s_i. In relation to the streaming anomaly detection framework of Calikus *et al.* [30], the data representation at time t will consist of the past W stream vectors, yielding $x_t \coloneqq [s_{t-W+1}, ..., s_t] \in \mathbb{R}^{W \times M}$.

3.1 Graph Neural Networks

Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ be a graph consisting of a node set $\mathcal{V}$ and an edge set $\mathcal{E}$. The adjacency matrix $A \in \mathbb{R}^{N \times N}$ defines the connections of nodes by edges, where $N = |\mathcal{V}|$ is the total number of nodes. All node attributes are collected in the node attribute matrix $X \in \mathbb{R}^{N \times C}$, with C being the number of features per node. The goal of a Graph Neural Network (GNN) is to learn effective node representations $H^k \in \mathbb{R}^{N \times F}$, where H^k is the node representation matrix after the k-th GNN layer and F is the dimension of node representations. Wu *et al.* [31] define a framework for Graph Neural Networks, where the node attributes are given as $H^0 = X$ and a GNN layer with index k consists of two functions:

$$a_v^k = \textbf{AGGREGATE}^k\{H_u^{k-1} : u \in N(v)\},$$
$$H_v^k = \textbf{COMBINE}^k\{H_v^{k-1}, a_v^k\},$$

where $N(v)$ is the set of neighbors of node v. For the specific case of a graph convolutional neural networks (GCN) [32], these two rules are implemented in the overall updating rule as

$$a_v^k = \sum_{j \in N(i)} \frac{A_{ij}}{\sqrt{\tilde{D}_{ii}\tilde{D}_{ij}}} H_j^{k-1} W^k,$$
$$H_i^k = \sigma\left(a_v^k + \frac{1}{\tilde{D}_i} H_i^{k-1} W^k\right),$$

with $\tilde{A} = A + \mathbb{I}$ as the adjacency matrix with self-connections, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ as the corresponding degree matrix and σ as an activation function.

3.2 Graph Connection Strategies

For a data representation of W stream vectors being encoded as the node attributes, the goal of the following approaches is to determine an adjacency matrix A, which is restricted to be a temporal grid with optional connections to future time steps. Since $X = x_t$, the number of stream vectors is the same as the number of nodes, $W = N$, and the stream vector dimension is the same as the number of features of the first GNN layer, $M = C$. As the connections to future time steps impact the extracted features and thus the overall anomaly detection capability, the connections can be seen as hyperparameters for the overall Graph Neural Network and a brute force search would yield the best adjacency matrix for any given dataset. Apart from random search being presented in section 3.2.3, computationally less expensive alternatives are introduced below.

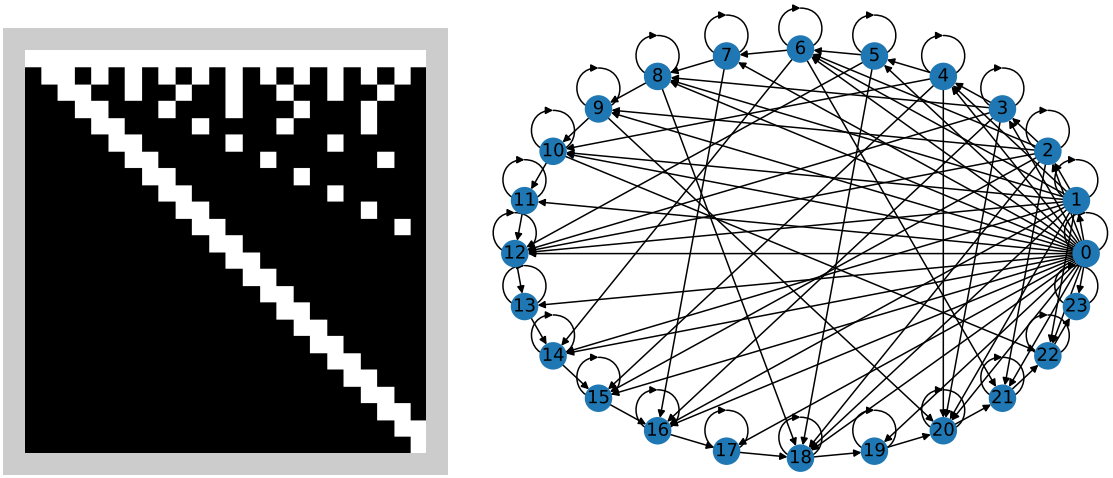

Fig. 1 Static connection scheme for a subsequence length of 24. The left diagram depicts the adjacency matrix with white elements indicating a connection and the right diagram displays the same connection scheme as a graph, starting with an index at 0. Apart from self-connections and connections to subsequent time steps, connections to future time steps are included with a decreasing frequency for later time steps.

3.2.1 Static Connection Scheme

Figure 1 presents a simple strategy for connecting time steps in the adjacency matrix. Besides self-connections and connections to subsequent time steps which originate from the temporal grid, it adds connections to future time steps with a decreasing frequency for later time steps. Let $e_{ij} \in \mathcal{E}$ be an edge connecting nodes i and j, then the edge set complementing the temporal grid is constructed by

$$\mathcal{E} = \bigcup_{i \in [1,N]} \{e_{ij} | j \in \{k * i | k \in \mathbb{N}_1 \wedge k * i \leq N\}\}. \tag{1}$$

3.2.2 Autocorrelation

Similar to the process of identifying autoregressive and moving-average components in an ARIMA model [33], the autocorrelation and the partial autocorrelation function can be calculated for many different lags and edge weights can be set according to local extrema. The sample autocorrelation function is defined as

$$\text{ACF}(s_t, s_{t-l}) = \frac{\text{Cov}(s_t, s_{t-l})}{\sigma(s_t) * \sigma(s_{t-l})}$$

with $\text{Cov}(s_t, s_{t-l})$ being the sample covariance and $\sigma(s_t)$ the sample standard deviation. The sample partial autocorrelation function is then defined as

$$\text{PACF}(s_t, s_{t-l}) = \text{ACF}(s_t - \tilde{s}_t, s_{t-l} - \tilde{s}_{t-l}),$$

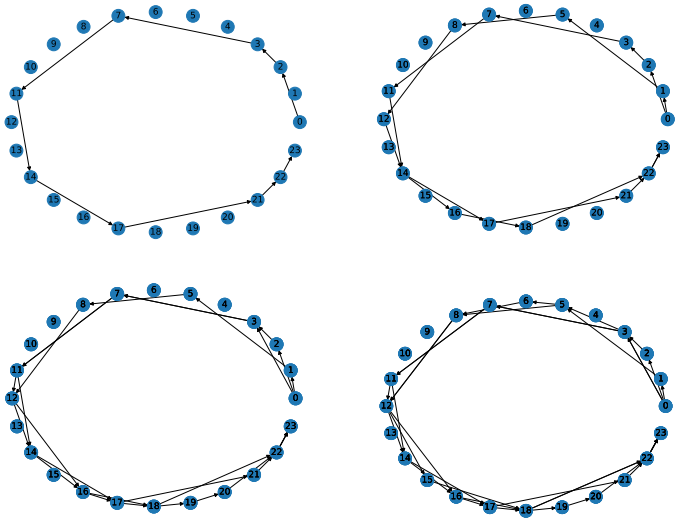

Fig. 2 Visualization of the process for drawing random routes and inserting them into the graph. The temporal grid is omitted for better visibility. The difference to the random connection strategy is that multiple connections (connecting the start and end node together as a route) are drawn at the same time.

where $\tilde{s}_t$ and $\tilde{s}_{t-l}$ are linear combinations of $\{s_{t-l+1}, ..., s_{t-1}\}$ that minimize $||s_t - \tilde{s}_t||_2$ and $||s_{t-l} - \tilde{s}_{t-l}||_2$. Let $\{l_1, ..., l_L\}$ be a list of lags for which either the ACF or the absolute PACF show peaks when compared to neighboring lags. The edge set complementing the temporal grid is then

$$\mathcal{E} = \bigcup_{i \in [1,N]} \{e_{ij} | j \in \{i + l | l \in \{l_1, ..., l_L\} \wedge i + l \leq N\}\}. \tag{2}$$

The process of finding peaks in the ACF and PACF is repeated for every channel $c \in [1, M]$ in the case of multivariate time series.

3.2.3 Random Connections

With a total number of $|\mathcal{E}| = \frac{1}{2}(N-2)(N-1)$ potential edges, the probability for choosing an individual edge when sampling connections within the upper triangle of the adjacency matrix not yet included in the temporal grid is

$$P_U(e_{ij}) = \frac{2}{(N-2)(N-1)}, \qquad \forall j > i + 1. \tag{3}$$

With an exponential decay for increasing time differences, it is

$$P_E(e_{ij}) = u^{-\lambda(j-i)}\, P_U(e_{ij}), \qquad \forall j > i + 1, \quad u \in [0, 1]. \tag{4}$$

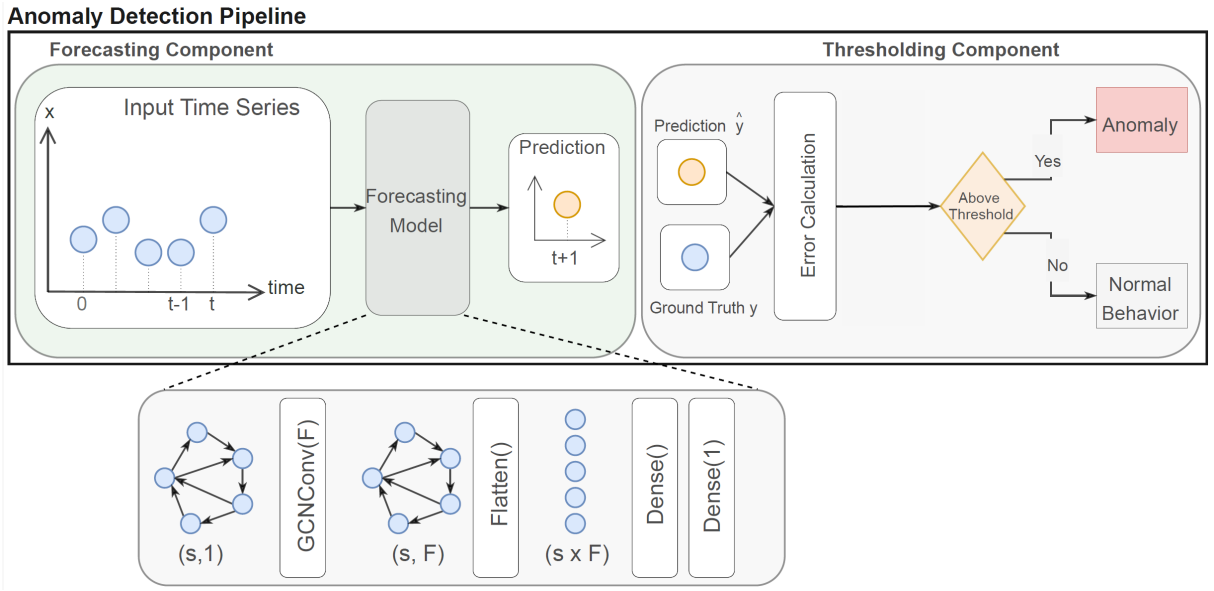

Fig. 3 GCN model with arbitrary connection strategy used for forecasting the subsequent time step of a time interval. The model depicted has one GCN layer and two feed-forward ("Dense") layers. "Flatten" in this context denotes a reshaping operation. The prediction is compared to the ground truth value by a distance metric and a threshold is applied to decide whether an anomaly is discovered.

† Figure adapted from Kiprit *et al.* [34], where preliminary results have been published.

3.2.4 Random Routes

Let a route r_{ij} be defined as multiple consecutive directed edges connecting nodes i and j,

$$r_{ij} \coloneqq \{e_{ij'}, ..., e_{j''j}\} \in R_{ij} \coloneqq \{\{e_{ij}\}, \{e_{ij'}, e_{j'j}\}, ...\}, \quad j', j'' \in \{i, ..., j\} \tag{5}$$

with arbitrary many nodes in between nodes i and j. The total number of routes between nodes i and j can be calculated as

$$|R_{ij}| = 1 + \sum_{j'=(j-i-1)}^{1} 2^{(j-i-1)-j'}. \tag{6}$$

For a more intuitive understanding, the summation is carried out from back to front. With every decreasing index j', the previous node at $j' + 1$ can either be skipped or added to all previous paths, thereby adding double as many paths as for the previous step. For the complete window of size W, this results in a total number of routes

$$|R| = 1 + \sum_{j'=W-1}^{1} 2^{(W-1)-j'}. \tag{7}$$

Consequently, the probability for choosing any route connecting the first and last time step is

$$P_U(r_{1W}) = \frac{1}{|R|}. \tag{8}$$

Again, an exponential decay can be introduced to punish any route segments skipping longer distances,

$$P_E(r_{1W}) = \prod_{e_{ij} \in r_{1W}} u^{-\lambda(j-i)} P_U(r_{1W}), \qquad \forall j > i+1, \quad u \in [0,1]. \tag{9}$$

The overall process of sampling routes connecting the start and end node of the graph is displayed in Figure 2.

3.3 Detect Anomalies via Forecasting Methods

As all GNN layers with graphs constructed according to the strategies introduced in the previous chapter are used as forecasting methods in this work, the task of time series forecasting for anomaly detection is formalized. Returning to the context window $x_t := [s_{t-W+1}, ..., s_t]$, a model-based forecasting method with model parameters θ_t produces a prediction for time step $t+1$,

$$\hat{s}_{t+1} = f_{\text{forecasting}}(x_t, \theta_t).$$

The nonconformity score is calculated by applying an arbitrary distance between the prediction and the original stream vector,

$$n_{t+1} = D(s_{t+1}, \hat{s}_{t+1}).$$

In this work, the distance metric is chosen to be the cosine distance $D_C(a,b) = 1 - S_C(a,b)$ for multivariate time series, with $S_C(a,b)$ as the cosine similarity, and the absolute distance $D_A(a,b) = |a-b|$ for univariate time series. The overall structure of a GNN-based forecasting model is displayed in Figure 3, which also shows the application of a static threshold to the nonconformity score. According to the SAFARI framework [30], anomaly scores can further be applied as a post-processing step. As an example, the anomaly likelihood [35] emphasizes short-term differences to long term characteristics of the nonconformity scores.

4 Results

Measuring the effectiveness of future time step connections within the graph of a Graph Neural Network temporal module can be achieved by evaluating many GNN configurations on a multitude of anomaly detection benchmark datasets. As the focus of this paper lies in the onboard analysis of spacecraft telemetry, both the open-source multivariate ESA Anomaly Dataset (ESA-ADB) [36] mission 1 and the univariate NASA-SMAP and NASA-MSL [8] are selected. Both contain telemetry data with anonymized

Table 1 Hyperparameters chosen for the raster search of the GCN models for the different graph connection strategies. Configurations are found by applying a cartesian product on all hyperparameters, i.e. forming all possible combinations. The parameter "Num. Adj." denotes the number of adjacency matrices randomly sampled per configuration.

Hyperparameters						
	Window Size	GCN Hidden	Dense Hidden	Num. GCN	Num. Dense	Num. Adj.
Autocorrelation	$25, 50$	$64, 256$	$64, 256$	$1, 3$	$1, 2, 3$	–
Random Connections	25	32	32	1	$2, 3$	750
Random Routes	$25, 50$	$64, 256$	$64, 256$	$1, 3$	$1, 2, 3$	10

channels. The spacecraft for the ESA-ADB are unknown while the NASA-SMAP dataset originates from the Soil Moisture Active Passive satellite and the NASA-MSL dataset from the Mars Science Laboratory. Apart from the spacecraft telemetry datasets, three public anomaly detection benchmarks are selected: Daphnet [37], which includes measurements of acceleration sensors on Parkinson's patients with the goal to detect the freezing of gait condition as anomalies, Exathlon [38], consisting of traces from stream processing jobs in Apache Spark clusters, and SMD [39], which includes traces from server machines.

4.1 Evaluation Process

Anomaly detection approaches can be evaluated by a variety of evaluation metrics. Each metric provides certain advantages and emphasizes a different aspect of the detection capability of anomaly detection algorithms. The first major distinction is whether predictions are evaluated as individual time steps that coincide with a true anomalous region in "point-wise" metrics or whether the predictions are combined to form predicted anomaly sequences which are supposed to overlap with true anomalous regions in "range-based" metrics. Point-wise metrics emphasize the exact detection of all points in true anomaly regions while penalizing all falsely predicted time steps equally. The condition of an exact detection is relaxed to count all time steps within a true anomaly region as true positives if a single time step has been correctly identified by Xu *et al.* [43]. The precision and recall are defined to be

$$Prec_{PW} = \frac{\mathrm{TP}_{PW}}{\mathrm{TP}_{PW} + \mathrm{FP}_{PW}}, \qquad Rec_{PW} = \frac{\mathrm{TP}_{PW}}{\mathrm{TP}_{PW} + \mathrm{FN}_{PW}}, \tag{10}$$

where the subscript PW indicates point-wise counting. Range-based metrics combine consecutive predicted time steps to form anomalous intervals. A true positive is counted if a predicted and true anomaly region overlap. False positive intervals are counted as 1 FP for range-based methods, whereas they are

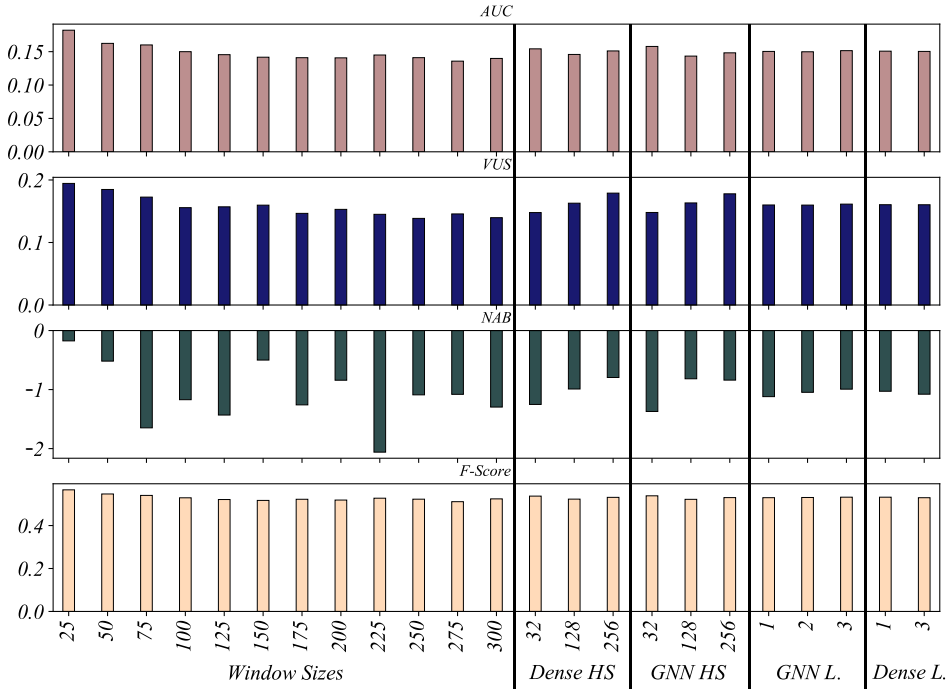

Fig. 4 Sensitivity of the GCN-AC model w.r.t. individual hyperparameters, measured on the SMD dataset. The results show the measured AUC, VUS, NAB and $F_{0.5}$-Score, summed over all commbinations containing the respective hyperparameter. As this summation leads to worse scores than the best performance of individual configurations reported in table 4, it can be concluded that no individual hyperparameter is largely reponsible for the model performance. Small differences can be observed for the AUC, VUS and NAB scores, suggesting shorter window sizes (25, 50) and larger hidden sizes (Dense HS, GNN HS) lead to improvements. The difference in layers (Dense L., GNN L.) does not seem to affect the overall performance, at least not for the difference of $1-3$ layers.

counted as L_p FPs for point-wise metrics with L_p as the length of the falsely predicted interval. The range-based versions of precision and recall are defined accordingly, however true positives, false positives and false negatives indicate time step intervals,

$$Prec_{RB} = \frac{\mathrm{TP}_{RB}}{\mathrm{TP}_{RB} + \mathrm{FP}_{RB}}, \qquad Rec_{RB} = \frac{\mathrm{TP}_{RB}}{\mathrm{TP}_{RB} + \mathrm{FN}_{RB}}. \tag{11}$$

A combination of precision and recall is realized by the F-Score,

$$F_\beta = \frac{(1+\beta^2) \cdot Prec_m \cdot Rec_m}{(\beta^2 \cdot Prec_m) + Rec_m}, \quad m \in \{PW, RB\} \tag{12}$$

where both precision and recall are calculated either point-wise or range-based.

Another important aspect for the evaluation of anomaly detection algorithms is the timing of a detection, i.e. where the detection is located within an anomaly region. The Numenta anomaly benchmark scoring function (NAB) [44] rewards early detection of a ground truth anomaly region by a sigmoidal weighting

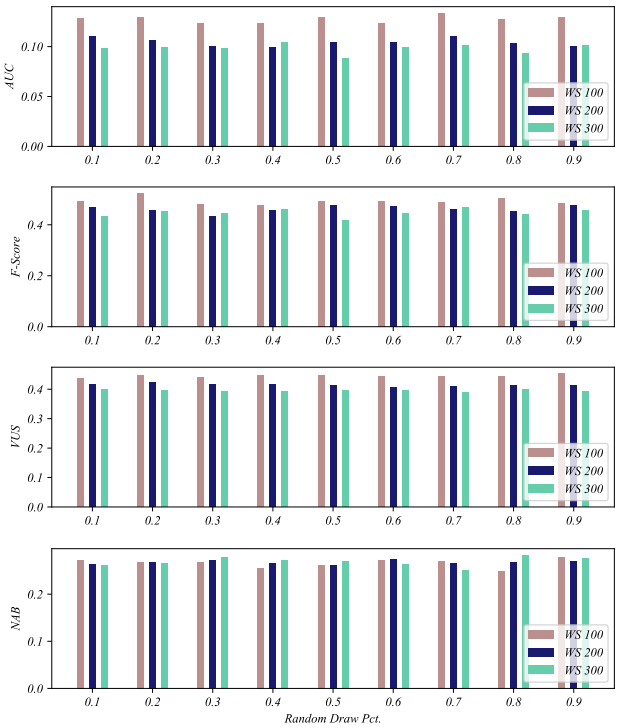

Fig. 5 Sensitivity of the GCN-RC model to the percentage of randomly selected edges (range: 10% - 90%) in the upper triangular matrix of the graph. The performance of the models is measured on the SMD benchmark for three different window sizes (100, 200, 300). While the previous correlation of decreasing AUC, VUS and F-scores for longer window sizes can be observed, the percentage of randomly selected edges shows no significant contribution. This suggests the overall density of edges in the graph is not substantial and the specific combination of selected edges might be more relevant.

function $\sigma_{\text{NAB}} \in [0, 1]$ and counts the score of the earliest detection within a true anomaly region as the score assigned for detecting the complete region. Every false positive point on the other hand contributes as -1, leading to negative scores in case of a high number of false positives. In order to estimate the anomaly detection capability for a wide choice of static thresholds, the area under the precision-recall curve (AUC) is calculated by applying the trapezoidal rule to the 2D plot of recalls on the x-axis and precisions on the y-axis that result from applying many different static thresholds to the nonconformity scores. It is possible to calculate this measure for either point-wise or range-based precisions and recalls, although in this evaluation it is restricted to be range-based. The volume under the surface (VUS) [45] combines point-wise and range-based metrics by including a reward for an overlapping predicted and true anomaly region and multiplying it with the point-wise recall. It further introduces a buffer region around the ground truth anomaly regions to take into account near misses and stacks the respective AUC curves to calculate the volume under the resulting surface.

Table 2 GCN models with the random connection (GCN-RC), random routes (GCN-RR), and auto-correlation (GCN-AR) strategies being evaluated on the univariate spacecraft telemetry dataset NASA SMAP & MSL. The GCN approaches perform comparably with $F_{0.5}$ scores at around 0.9, surpassing the baseline fully-connected model (FC w/o GCN). Both GCN-RC models are further quantized ($F_{0.5}$ A.Q.) and smoothing is applied to eliminate peaks that occurred due to quantization ($F_{0.5}$ A.Q.&S.). It is noteworthy that the GCN-AC shows competitive results without the need for extensive random sampling.

† Preliminary results that were published in Kiprit *et al.* [34]

	SMAP & MSL							
	$Prec_{RB}$	Rec_{RB}	$F_{0.5}$	$F_{0.5}$ A.Q.	$F_{0.5}$ A.Q.&S.	*AUC*	*VUS*	*NAB*
GCN-RC-2Dense†	0.94	0.70	0.88	0.71	0.83	0.42	0.43	0.68
GCN-RC-3Dense†	**0.96**	0.70	0.89	0.78	0.86	0.42	0.47	0.70
GCN-RR	0.91	0.76	0.87	–	–	0.48	0.61	0.81
GCN-AC	0.93	0.83	**0.91**	–	–	0.50	0.61	0.79
FC w/o GCN†	0.72	0.74	0.72	–	–	–	–	–
LSTM Network [8]	0.87	0.80	0.85	–	–	–	–	–
Channel-Specific LSTM [40]	0.79	0.83	0.79	–	–	–	–	–
TadGAN [41]	0.51	0.78	0.54	–	–	–	–	–
StackedPredictor [42]	0.87	**0.89**	0.87	–	–	–	–	–

Table 3 Pointwise evaluation comparing GCN models with a graph constructed via the autocorrelation (GCN-AR) strategy to ATCN and GTA. Both models outperform the GCN-AC model, suggesting several longer anomaly regions (with more time steps) were missed by GCN-AC.

	SMAP			MSL		
	$Prec_{PW}$	Rec_{PW}	$F_{1,PW}$	$Prec_{PW}$	Rec_{PW}	$F_{1,PW}$
GCN-AC	0.94	0.72	0.81	0.81	0.65	0.72
ATCN [28]	0.95	0.90	0.93	0.94	0.98	0.96
GTA [29]	0.89	0.92	0.90	0.91	0.91	0.91

As the datasets used in this evaluation each contain a set of time series $\mathcal{D} \coloneqq \{T_1, ..., T_M\}$, the measurements for each evaluation metric need to be accumulated to result in one overall score. For this purpose, the true positives, false positives and false negatives are summed across the complete dataset,

$$\text{TP} = \sum_{T_i \in \mathcal{D}} \text{TP}_{T_i}, \quad \text{FP} = \sum_{T_i \in \mathcal{D}} \text{FP}_{T_i}, \quad \text{FN} = \sum_{T_i \in \mathcal{D}} \text{FN}_{T_i}, \tag{13}$$

Table 4 Benchmarking results of a multivariate GCN with graph connections set via the autocorrelation strategy (GCN-AC). It is compared to the best result achieved in a previous publication (N-BEATS model). The improvements in NAB score with similar precisions for the Daphnet and SMD benchmarks suggest a better locating of the beginning of ground truth anomaly regions. The higher VUS score for the Exathlon and SMD benchmarks further indicates false positive detections just outside of the ground truth anomaly regions.

	Daphnet						**Exathlon**					
	$Prec_{RB}$	Rec_{RB}	$F_{0.5}$	*AUC*	*VUS*	*NAB*	$Prec_{RB}$	Rec_{RB}	$F_{0.5}$	*AUC*	*VUS*	*NAB*
GCN-AC	0.82	0.52	0.74	0.38	0.24	0.31	0.92	0.82	0.90	0.57	0.78	0.74
N-BEATS [46]	0.81	0.54	0.74	0.40	0.26	0.09	0.96	0.93	0.95	0.68	0.39	0.80
	SMD						**ESA-ADB1**					
	$Prec_{RB}$	Rec_{RB}	$F_{0.5}$	*AUC*	*VUS*	*NAB*	$Prec_{RB}$	Rec_{RB}	$F_{0.5}$	*AUC*	*VUS*	*NAB*
GCN-AC	0.89	0.22	0.55	0.24	0.38	0.50	0.89	0.33	0.66	0.40	0.10	0.04
N-BEATS [46]	0.91	0.43	0.74	0.40	0.20	0.19	–	–	–	–	–	–

whereas the NAB (q_{NAB}), AUC (q_{AUC}) and VUS (q_{VUS}) scores are averaged,

$$q_{\text{m}} = \frac{1}{M} \sum_{T_i \in \mathcal{D}} q_{\text{m},T_i}, \quad m \in \{\text{AUC}, \text{VUS}, \text{NAB}\}. \tag{14}$$

4.2 Benchmarking Results

The previously presented strategies are incorporated into a multitude of GCN models, denoted by **GCN-AC** for the autocorrelation strategy, **GCN-RC** for the random connection strategy and **GCN-RR** for the random routes. A raster search is applied for each time series in a dataset for the autocorrelation strategy and in the case of both random strategies, the number of randomly sampled adjacency matrices is supplied. The hyperparameters are depicted in table 1. An analysis regarding the sensitivity of the GCN-AC to the choice of hyperparameters is presented in Figure 4. The contribution of the number of randomly selected edges to the anomaly detection capability of the GCN-RC model is presented in Figure 5. Note that due to high compute costs, the raster search for the random connection and random routes strategies is only applied to a single dataset, the NASA SMAP & MSL dataset. It was performed for the random connection strategy with 750 sampled random adjacency matrices. Table 2 exhibits corresponding results. It is apparent that all GCN models perform well with $F_{0.5}$ scores at around 0.9 when measuring range-based metrics. However, they yield worse results for point-wise measurements (table 3) compared to other graph neural network based anomaly detection models, suggesting longer false positive ranges. Since the

autocorrelation strategy results in a model scoring higher AUC, VUS and NAB scores, compared to the random connection models, we can draw the conclusion that the autocorrelation strategy already provides a competitive GCN model, without the need for extensive random sampling. In order for deployment on the AMD-Xilinx DPU, two GCN models with random connections (GCN-RC-2Dense, GCN-RC-3Dense) are quantized ($F_{0.5}$ A.Q.) to 8-Bit integers. As they show significant performance degradation, smoothing is applied to eliminate the peaks that occurred due to quantization ($F_{0.5}$ A.Q.&S.). Table 4 depicts the benchmarking results of GCN models incorporating the autocorrelation strategy. It is evaluated on the multivariate datasets previously mentioned and the results are compared to the best results achieved by the N-BEATS model in a previous publication. While the AUC for the GCN-AC is lower for all three datasets, the NAB scores are significantly higher for both the Daphnet and the SMD datasets, indicating a superior locating of the anomaly starting points, since the precision is almost the same. The higher VUS values for the Exathlon and SMD datasets further suggest false positive detections just outside the ground truth anomaly regions. The GCN-AC configuration $(\text{WS: } 25, \text{DHS: } 256, \text{GHS: } 64, \text{DL: } 1, \text{GL: } 1)$ achieved a range-based $F_{0.5}$-Score of 0.66 on the ESA-ADB Mission 1 time series. Note that in this setting, overlapping ground truth anomaly regions of different channels are joined into one, yielding a total of 56 ground truth anomaly regions. As the ESA-ADB publication [36] reports point-wise metrics for the case of affected channels (channel-aware) or affected subsystems (subsystem-aware) being predicted, the reported scores cannot be compared to our results.

4.3 Onboard Resource Utilization

In order to measure the resources required for different sizes of GCN models, univariate GCN-RC models with an increasing number of parameters are deployed to the AMD-Xilinx Versal AI Core via the AMD-Xilinx Deep Learning Processing Unit (DPU). Out of a total of 400 specialized vector processors called AI engines, the configuration DPU-B1 uses 32 AI engines and the DPU-B5 utilizes 160 AI engines. Naturally, this difference is reflected in the idle power of 14.6W for the DPU-B1 and 24.5W for the DPU-B5. Both of these values are calculated by summation of measurements of all power system rails available on the Versal AI Core SoC[1]. The measurements of the INA226 power sensors are read out via the HWMON linux driver in the Petalinux OS, which is used for deployment. The difference between idle and active power (during inference) is used to derive the spent energy per Bit, which is depicted in Figure 6 on the left. While the energy per Bit is slightly lower for the larger DPU-B5 configuration compared to

[1] https://docs.amd.com/r/en-US/ug863-versal-pcb-design/Versal-Adaptive-SoC-Power-Rails

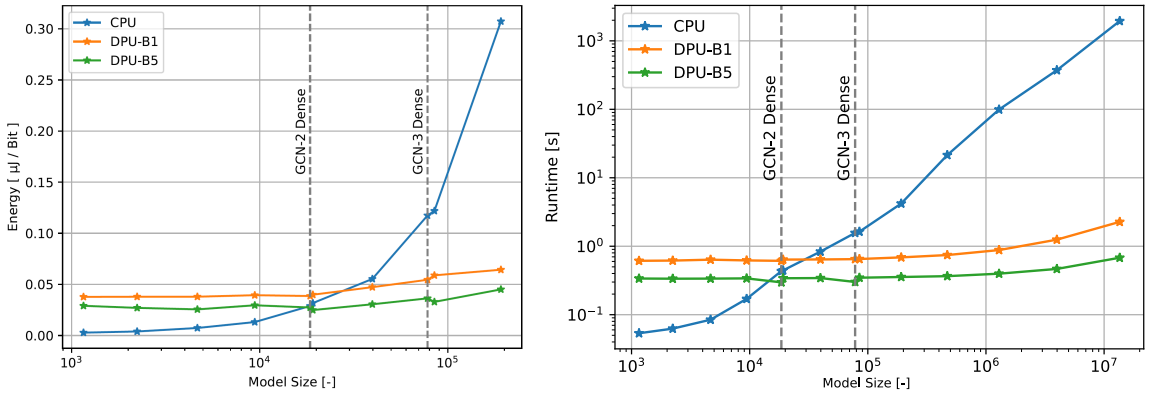

Fig. 6 Runtime comparison of the three processors (right) and energy consumption per Bit (left) with respect to the model size. The energy per Bit is calculated via the difference between idle and active power of the power system rails on the VCK190 evaluation board. The configurations ”DPU-B1” and ”DPU-B5” include the use of the AMD-Xilinx DPU and 32 or 160 AI engines respectively. The runtime is measured for processing 3000 samples at once for the GCN-RC models of sizes ranging from 1.2k to 13.5M parameters. A better runtime is achieved on either DPU configuration compared to a CPU configuration for model sizes above 50k parameters.

† Figure reproduced from Kiprit *et al.* [34], where preliminary results have been published.

the DPU-B1 configuration, the energy per Bit required when executing the inference exclusively on CPU rises for larger models to be above either DPU configuration. Naturally, the total power necessary for either DPU configuration still eclipses a CPU only configuration. The right diagram of Figure 6 displays the runtime required to process 3000 samples for GCN models of different sizes ranging from 1.2k to 13.5M parameters. In line with expectations, it shows a steep increase for larger models being executed on a CPU only configuration, while both DPU configurations display a modest incline. In terms of runtime, a DPU configuration makes sense for model sizes above 50k parameters. However, due to the high idle power requirements, a DPU configuration should only be employed for the inference of larger models when used in consistently high throughput scenarios.

5 Conclusion

By extending the adjacency matrix, used in the temporal module of Graph Neural Networks, to incorporate connections to future time steps in addition to the temporal grid, we are able to empirically show a competitive anomaly detection capability. We found that setting graph connections via peaks in the

autocorrelation function results in a GCN model which is comparable to the best graph connections obtained from extensive random sampling. While all three connection strategies exhibit promising results, the role of the random routes strategy still needs further examination on multiple benchmark datasets. As the extracted features of a Graph Neural Network layer with connections to future time steps are crucial for the anomaly detection capability, analyzing the effect of the presented strategies and other connection strategies on the extracted features and the resulting anomaly detection capability is a good topic for future research. As a next step towards applying the presented GCN approach to analyze spacecraft telemetry in real-time, it can be demonstrated with a dynamic thresholding policy on space-grade hardware.

Acknowledgments

This activity has been funded by Deutsches Zentrum für Luft- und Raumfahrt (DLR) within the project Machine Learning for Telecom Satellites (MaLeTeSa) - 50YB2103.

Declaration of competing interest

The authors have no competing interests to declare that are relevant to the content of this article.

References

[1] NASA. Fault-Detection, Fault-Isolation and Recovery (FDIR) Techniques. https://llis.nasa.gov/lesson/839, 1994.

[2] Koch A, Krstova A, Hegwein F, De Lera MC, Ales F, Petry M, Ali R, Mallah M, Hili L, Ghiglione M, Werner M. On-Board Anomaly Detection on a Flight-Ready System. In *2023 European Data Handling Data Processing Conference (EDHPC)*, 2023, 1–4, doi:10.23919/EDHPC59100.2023.10395967.

[3] Schmidl S, Wenig P, Papenbrock T. Anomaly Detection in Time Series: A Comprehensive Evaluation. 15(9): 1779–1797, doi:10.14778/3538598.3538602.

[4] Bianco AM, García Ben M, Martínez EJ, Yohai VJ. Outlier Detection in Regression Models with ARIMA Errors using Robust Estimates. *Journal of Forecasting*, 2001, 20(8): 565–579, doi:https://doi.org/10.1002/for.768.

[5] Chandola V, Banerjee A, Kumar V. Anomaly detection: A survey. *ACM Computing Surveys*, 2009, 41(3): 15:1–15:58, doi:10.1145/1541880.1541882.

[6] Munir M, Siddiqui SA, Chattha MA, Dengel A, Ahmed S. FuseAD: Unsupervised Anomaly Detection in Streaming Sensors Data by Fusing Statistical and Deep Learning Models. *Sensors*, 2019, 19(11): 2451, doi:10.3390/s19112451.

[7] Oreshkin BN, Carpov D, Chapados N, Bengio Y. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. 2022.

[8] Hundman K, Constantinou V, Laporte C, Colwell I, Soderstrom T. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, 387–395.

[9] Breiman L. Random forests. *Machine learning*, 2001, 45: 5–32.

[10] Chen T, Guestrin C. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, New York, NY, USA: Association for Computing Machinery2016, 785–794, doi:10.1145/2939672.2939785.

[11] Koch A, Krstova A, Hegwein F, De Lera MC, Ales F, Petry M, Ali R, Mallah M, Hili L, Ghiglione M, Werner M. On-Board Anomaly Detection on a Flight-Ready System. In *2023 European Data Handling Data Processing Conference (EDHPC)*, 2023, 1–4, doi:10.23919/EDHPC59100.2023.10395967.

[12] Audibert J, Michiardi P, Guyard F, Marti S, Zuluaga MA. USAD: UnSupervised Anomaly Detection on Multivariate Time Series. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, New York, NY, USA: Association for Computing Machinery2020, 3395–3404, doi:10.1145/3394486.3403392.

[13] Su Y, Zhao Y, Niu C, Liu R, Sun W, Pei D. Robust Anomaly Detection for Multivariate Time Series through Stochastic Recurrent Neural Network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, New York, NY, USA: Association for Computing Machinery2019, 2828–2837, doi:10.1145/3292500.3330672.

[14] Paffenroth R, Kay K, Servi L. Robust PCA for Anomaly Detection in Cyber Networks, 2018.

[15] Senin P, Lin J, Wang X, Oates T, Gandhi S, Boedihardjo AP, Chen C, Frankenstein S. Time series anomaly discovery with grammar-based compression. In *Proceedings of the 18th International Conference on Extending Database Technology, EDBT 2015, Brussels, Belgium, March 23-27, 2015*, OpenProceedings.org2015, 481–492, doi:10.5441/002/EDBT.2015.42.

[16] Ramaswamy S, Rastogi R, Shim K. Efficient Algorithms for Mining Outliers from Large Data Sets. *SIGMOD Rec.*, 2000, 29(2): 427–438, doi:10.1145/335191.335437.

[17] Yeh CCM, Zhu Y, Ulanova L, Begum N, Ding Y, Dau HA, Silva DF, Mueen A, Keogh E. Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, 2016, 1317–1322, doi:10.1109/ICDM.2016.0179.

[18] Breunig MM, Kriegel HP, Ng RT, Sander J. LOF: Identifying Density-Based Local Outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, New York, NY, USA: Association for Computing Machinery2000, 93–104, doi: 10.1145/342009.335388.

[19] Yairi T, Kato Y, Hori K. Fault Detection by Mining Association Rules from House-keeping Data. In *Proceedings of the 6th International Symposium on Artificial Intelligence and Robotics & Automation in Space: i-SAIRAS 2001*, Quebec, Canada2001.

[20] Liu FT, Ting KM, Zhou ZH. Isolation Forest. In *2008 Eighth IEEE International Conference on Data Mining*, 2008, 413–422, doi:10.1109/ICDM.2008.17.

[21] Hariri S, Kind MC, Brunner RJ. Extended Isolation Forest. *IEEE Transactions on Knowledge and Data Engineering*, 2021, 33(4): 1479–1489, doi:10.1109/tkde.2019.2947676.

[22] Jin M, Koh HY, Wen Q, Zambon D, Alippi C, Webb GI, King I, Pan S. A survey on graph neural networks for time series: Forecasting, classification, imputation, and anomaly detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[23] Li Y, Yu R, Shahabi C, Liu Y. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*, 2017.

[24] Yu B, Yin H, Zhu Z. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875*, 2017.

[25] Liang Y, Zhao Z, Sun L. Dynamic spatiotemporal graph convolutional neural networks for traffic data imputation with complex missing patterns. *arXiv preprint arXiv:2109.08357*, 2021.

[26] Guo K, Hu Y, Sun Y, Qian S, Gao J, Yin B. Hierarchical graph convolution network for traffic forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 2021, 151–159.

[27] Zheng C, Fan X, Wang C, Qi J. Gman: A graph multi-attention network for traffic prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 2020, 1234–1241.

[28] Liu L, Tian L, Kang Z, Wan T. Spacecraft anomaly detection with attention temporal convolution networks. *Neural Computing and Applications*, 2023, 35(13): 9753–9761.

[29] Chen Z, Chen D, Zhang X, Yuan Z, Cheng X. Learning graph structures with transformer for multivariate time-series anomaly detection in IoT. *IEEE Internet of Things Journal*, 2021, 9(12): 9179–9189.

[30] Calikus E, Nowaczyk S, Sant'Anna A, Dikmen O. No free lunch but a cheaper supper: A general framework for streaming anomaly detection. *Expert Systems with Applications*, 2020, 155: 113453, doi:https://doi.org/10.1016/j.eswa.2020.113453.

[31] Wu L, Cui P, Pei J, Zhao L, Guo X. Graph neural networks: foundation, frontiers and applications. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, 2022, 4840–4841.

[32] Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[33] Box GE, Jenkins GM. Time series. *Analysis, Forecasting and Control//San Francisco*, 1970.

[34] Kiprit GN, Koch A, Petry M, Werner M. Graph Neural Networks for Anomaly Detection in Spacecraft. In *Proceedings of SPAICE2024: The First Joint European Space Agency / IAA Conference on AI in and for Space*, Zenodo2024, 105–109, doi:10.5281/zenodo.13885527.

[35] Ahmad S, Purdy S. Real-time anomaly detection for streaming analytics. *arXiv preprint arXiv:1607.02480*, 2016.

[36] De Canio G, Kotowski K, Haskamp C. ESA Anomaly Dataset, 2024, doi:10.5281/zenodo.12528696.

[37] Bachlin M, Plotnik M, Roggen D, Maidan I, Hausdorff JM, Giladi N, Troster G. Wearable Assistant for Parkinson's Disease Patients With the Freezing of Gait Symptom. *IEEE Transactions on Information Technology in Biomedicine*, 2010, 14(2): 436–446, doi:10.1109/TITB.2009.2036165.

[38] Jacob V, Song F, Stiegler A, Rad B, Diao Y, Tatbul N. Exathlon: a benchmark for explainable anomaly detection over time series. *Proc. VLDB Endow.*, 2021, 14(11): 2613–2626, doi:10.14778/3476249.3476307.

[39] Su Y, Zhao Y, Niu C, Liu R, Sun W, Pei D. Robust Anomaly Detection for Multivariate Time Series through Stochastic Recurrent Neural Network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, New York, NY, USA: Association for Computing Machinery2019, 2828–2837, doi:10.1145/3292500.3330672.

[40] Baireddy S, Desai SR, Mathieson JL, Foster RH, Chan MW, Comer ML, Delp EJ. Spacecraft Time-Series Anomaly Detection Using Transfer Learning. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2021, 1951–1960, doi:

10.1109/CVPRW53098.2021.00223.

[41] Geiger A, Liu D, Alnegheimish S, Cuesta-Infante A, Veeramachaneni K. Tadgan: Time series anomaly detection using generative adversarial networks. In *2020 ieee international conference on big data (big data)*, IEEE2020, 33–43.

[42] Li T, Comer M, Delp E, Desai SR, Mathieson JL, Foster RH, Chan MW. A Stacked Predictor and Dynamic Thresholding Algorithm for Anomaly Detection in Spacecraft. In *MILCOM 2019 - 2019 IEEE Military Communications Conference (MILCOM)*, 2019, 165–170, doi: 10.1109/MILCOM47813.2019.9021055.

[43] Xu H, Chen W, Zhao N, Li Z, Bu J, Li Z, Liu Y, Zhao Y, Pei D, Feng Y, et al.. Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications. In *Proceedings of the 2018 world wide web conference*, 2018, 187–196.

[44] Lavin A, Ahmad S. Evaluating Real-Time Anomaly Detection Algorithms – The Numenta Anomaly Benchmark. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, Miami, FL, USA: IEEE2015, 38–44, doi:10.1109/ICMLA.2015.141.

[45] Paparrizos J, Boniol P, Palpanas T, Tsay RS, Elmore A, Franklin MJ. Volume under the Surface: A New Accuracy Evaluation Measure for Time-Series Anomaly Detection. *Proc. VLDB Endow.*, 2022, 15(11): 2774–2787, doi:10.14778/3551793.3551830.

[46] Koch A, Petry M, Werner M. Extended Framework and Evaluation for Multivariate Streaming Anomaly Detection with Machine Learning. In *2024 IEEE 40th International Conference on Data Engineering Workshops (ICDEW)*, 2024, 144–152, doi:10.1109/ICDEW61823.2024.00025.

Author Biographies

Gamze Naz Kiprit holds M.Sc. and B.Sc. degrees in Electrical and Computer Engineering from the Technical University of Munich (TUM). During her studies, she contributed to developing AI-driven FMCW radar applications with Infineon Technologies. For her master's thesis, she joined Airbus Defence and Space, where she subsequently worked for another year as Embedded AI Engineer. Her work there focused on anomaly detection, computer vision, and neuromorphic computing. She is currently part of Accenture, working as Software Engineer and specializing in IoT and Cloud applications.

Andreas Koch holds a B.Sc. degree in Physics at Ludwig-Maximilians-Universität and a M.Sc. degree in Robotics, Cognition and Intelligence at Technical University of Munich (TUM). He is pursuing his PhD with the topic of "Onboard Streaming Anomaly Detection in Spacecraft" at the Professorship of Big Geospatial Data Management in the School of Engineering and Design at TUM. This work encompasses the identification of suitable spacecraft sensor signals and corresponding anomalies, research on algorithms in the field of streaming anomaly detection and the deployment of algorithms on space-grade hardware, such as SoC-, FPGA- and ASIC-based compute platforms.

Graphical table of contents

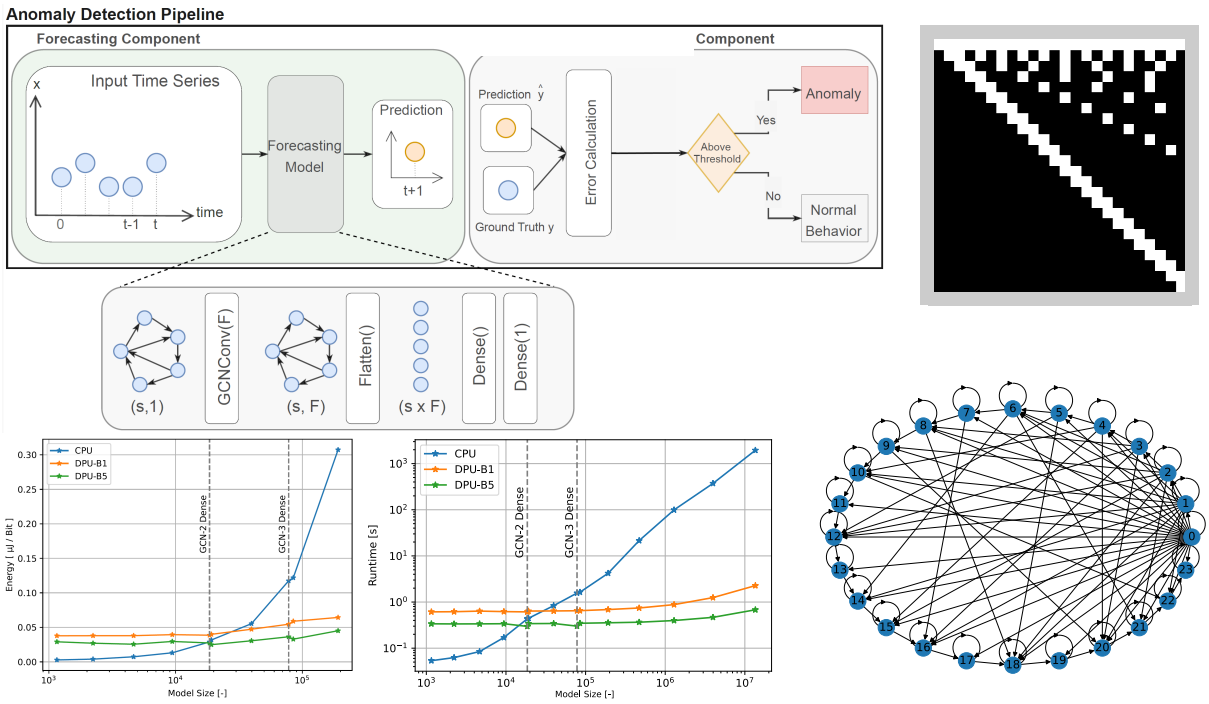

Fig. 7 This work explores connections to future time steps in the adjacency matrix of a Graph Neural Network layer for the purpose of anomaly detection in time series. The resulting Graph Neural Network is evaluated on spacecraft telemetry anomaly detection datasets and it is deployed on the AMD-Xilinx Versal AI Core SoC.