PROCEEDINGS OF SCIENCE

Modeling Multivariate Sensor Dynamics with STGNNs: An Anomaly Detection Study at the INFN CNAF Tier-1 Facility

G. Zurlo,[a,*] **E. Ronchieri,**[a] **L.B. Scarponi,**[a] **D. Cesini**[a] **and A. Costantini**[a]

[a] *INFN-CNAF,*
Viale Berti Pichat 6/2, 40127 Bologna, IT

E-mail: giovanni.zurlo@cnaf.infn.it, elisabetta.ronchieri@cnaf.infn.it

The reliable operation of modern cyber-physical infrastructures, such as data centers, requires scalable and robust anomaly detection pipelines to monitor complex multivariate time-series data. However, traditional deep learning methods often struggle to capture the complex spatial relationships inherent in these environments. To bridge this gap, we evaluate an unsupervised anomaly-detection framework using Spatio-Temporal Graph Neural Networks (STGNNs), specifically comparing a Temporal Graph Convolutional Network (T-GCN) and a Graph Convolutional Gated Recurrent Unit (GCNConvGRU) integrated into a hybrid VAE-GAN architecture. The models represent sensor measurements as a sequence of discrete graph snapshots to jointly encode spatial and temporal structures. We validate these architectures through a real-world case study using electrical and cooling infrastructure data from the INFN CNAF Tier-1 facility, alongside comprehensive benchmarks on two public datasets: the Server Machine Dataset (SMD) and Pooled Server Metrics (PSM). Experimental results show that the STGNN models successfully identify operational irregularities in the Tier-1 facility and achieve better detection performance than previously proposed Graph Neural Network baselines. Furthermore, an efficiency benchmark clearly highlights the impact of introducing graph-native recurrence within a Spatio-Temporal Graph Neural Network: embedding topology-aware operators directly inside GRU gate computations increases per-step latency and memory pressure proportionally to the depth of graph propagation in the recurrent update loop.

International Symposium on Grids and Clouds (ISGC2026)
15-20, March, 2026
Academia Sinica, Taipei, Taiwan

*Speaker

© Copyright owned by the author(s) under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (CC BY-NC-ND 4.0).
All rights for text and data mining, AI training, and similar technologies for commercial purposes, are reserved.
ISSN 1824-8039. Published by SISSA Medialab.

PoS(ISGC2026)006

1. Introduction

Modern industrial systems generate massive, complex multivariate time-series data, making anomaly detection a critical yet highly challenging requirement. While traditional deep learning methods fail to capture the non-Euclidean topological inter-dependencies of these environments, Graph Neural Networks (GNNs) have emerged as a powerful alternative. However, a critical research gap persists: current GNN deployments rely either on static, rigidly predefined topologies designed by domain experts, or on naive data-driven learning strategies that may inject noise and obscure nuanced sensor relationships. This creates a pressing need for adaptive, scalable, fully unsupervised frameworks capable of learning graph structures without restrictive prior constraints.

To bridge this representational gap, this study evaluates and compares two Spatio-Temporal Graph Neural Network (STGNN) architectures—specifically, the Temporal Graph Convolutional Network (T-GCN) [1] and a Graph Convolutional Gated Recurrent Unit (GCNConvGRU) [2]—against previously proposed GNN baselines [3]. We assess these models through a real-world case study using data from the electrical and cooling infrastructure of the INFN CNAF Tier-1 data center [4] (Section 5.1). To ensure rigorous validation, the architectures are further compared on two public datasets (the Server Machine Dataset [5] and Pooled Server Metrics [6]), where they demonstrate improved anomaly detection results (Section 5.2). A final benchmark further suggests a clear efficiency trade-off: introducing graph-native recurrence, embedding topology-aware operators directly inside GRU gate computations, increases per-step propagation cost and memory footprint, even when the model as a whole is smaller in terms of parameter count (Section 5.3).

This work investigates data-driven spatio-temporal graph learning as a promising approach to anomaly detection in complex industrial systems.

2. Related Works

Recent advancements in multivariate time-series anomaly detection have increasingly relied on graph neural networks (GNNs) to explicitly model the complex spatial and temporal interdependencies among sensors. Existing literature can be broadly differentiated along two critical dimensions: the strategies employed to construct the underlying graph, and the structural evolution of the graph over time.

A fundamental challenge in graph-based anomaly detection is determining the dependency structure between variables. Methods typically fall into two paradigms: physical (knowledge-driven) construction and learned (data-driven) construction. When target systems are well understood, researchers frequently embed domain expertise directly into the graph topology. This explicitly grounds the model in the real-world mechanics of the system. For instance, [7] constructs a cyber-physical topology for grid-tied photovoltaic systems derived directly from Kirchhoff's circuit laws and control loops. Similarly, [8] formulates a hybrid mechanism model that hardcodes the spatial layout and physical distances of industrial controllers into the graph, effectively bridging the gap between physical process constraints and data-driven anomaly compensation. In blast furnace applications, [9] initializes its graph using existing process topologies before refining the connections using statistical distance metrics. While highly interpretable, these physical construction methods are often limited by the availability of precise domain knowledge. When explicit

PoS(ISGC2026)006

domain knowledge is unavailable or incomplete, models must infer the graph structure from the time-series data itself. This is achieved either through predefined heuristics or adaptive end-to-end learning. Certain approaches compute a fixed adjacency matrix as a preprocessing step prior to model training. For example, [10] utilizes a Synergistic Similarity Graph Construction (SSGC) strategy, generating a robust structural prior by calculating and fusing four distinct mathematical metrics: K-nearest neighbors, cosine similarity, Mahalanobis distance, and minimum spanning trees. Recent trends heavily favor optimizing the graph jointly with the downstream anomaly detection task. Approaches like [11] jointly learn a directed adjacency matrix of sensor relationships via trainable node embeddings, while [12] captures complex intra- and inter-modal correlations using a multimodal embedding-based attention mechanism. Similarly, [13] applies multi-scale spatial-temporal attention to fully connected initial graphs, relying entirely on the model to adaptively highlight relevant connections. To suppress noise and spurious correlations during this learning process, [14] incorporates a differentiable threshold function and an l_0 norm, effectively pruning redundant edges and yielding a highly optimized, sparse dependency graph.

Beyond construction, these methods diverge in how they handle the temporal evolution of the graph architecture, balancing computational efficiency against the capacity to model non-stationary environments. The majority of existing anomaly detection architectures operate on static graphs with temporal signals. In this paradigm, the underlying dependency topology is fixed (whether physically defined or learned), and only node features (i.e., sensor readings) vary over time. Methods such as [10–13] leverage this structure. By maintaining a stationary adjacency matrix, these models benefit from improved training stability and computational efficiency. However, they implicitly assume that the fundamental relationships between sensors do not undergo regime shifts during operation. Real-world systems, however, frequently experience operational shifts that fundamentally alter sensor interdependencies. To address this, several frameworks propose dynamic graphs with temporal signal.

For continuous or discrete time-series systems where both the topology and the signal evolve concurrently, fully dynamic approaches are required. In [9], edge weights are dynamically re-calculated at each time instant using normalized Euclidean distances between changing process states. Similarly, [14] continuously adjusts graph connectivity layer-by-layer based on the evolving feature representations of the sensors. Even in physically grounded systems, [7] utilizes a dynamic weighted adjacency matrix that shifts in response to changing non-stationary system dynamics, enabling the model to track evolving operational states.

3. Preliminaries

3.1 Neural Reconstruction-Based Anomaly Detection

Neural networks can model complex data distributions through generative frameworks such as variational autoencoders (VAEs) [15] and generative adversarial networks (GANs) [16]. These methods learn compact, low-dimensional latent representations that capture the essential structure of the training data and enable the generation of new samples. Typically, they combine a generative component, which aims to reproduce the data distribution, with an auxiliary network—such as an encoder or discriminator—that evaluates how likely a sample is to belong to that distribution.

PoS(ISGC2026)006

The use of low-dimensional representations also enables reconstruction-based anomaly detection approaches, where models are trained to accurately reconstruct normal data from their latent embeddings; anomalies, which deviate from the learned patterns, tend to yield higher reconstruction errors and can thus be identified.

Extensions of these generative approaches to graph data include variational graph autoencoders (VGAE) [17] and Graph-GAN for static graphs [18], while in dynamic graph settings, similar models are employed to capture how data distributions evolve over time [19].

3.2 Dynamic Graph Modeling

A central design choice when modeling a dynamic network is how to define its temporal span T. Two main options exist for the time domain: a discrete-time setting, where T is a discrete set and the evolving network is represented as a sequence of timestamped static graphs, and a continuous-time setting, where T is treated as continuous and the evolution is captured at fine temporal resolution through timestamped events. From a computational standpoint, discrete-time dynamic graph models are generally easier to handle, and most existing dynamic graph embedding methods, including ours, adopt this snapshot-based formulation. Nonetheless, several works instead use continuous-time formulations to model richer stochastic phenomena and support tasks such as event-time prediction, where accurately capturing the timing of interactions is crucial [20].

In the Graph Snapshots model, a dynamic graph is represented as a sequence of static graphs $G = \{G(t_0), \ldots, G(t_{N_S-1})\}$, where each snapshot $G(t_k) = (V(t_k), E(t_k))$ is a static graph associated with timestamp t_k for $k \in \{0, \ldots, N_S - 1\}$, N_S denotes the number of snapshots, $V(t_k)$ is the node set at time t_k, and $E(t_k)$ is the edge set, typically containing all edges whose timestamps fall within the interval $[t_k, t_{k+1})$. Most dynamic graph embedding methods take such a sequence of snapshots as input, either by directly using successive state graphs that discretely track the network over time or by partitioning the time axis into non-overlapping windows of fixed length and constructing one static graph per window [21].

Different architectures based on neural networks are used both to extract the topological properties of a network and to capture temporal dependencies therein. Recurrent graph-convolutions layers based on recurrent neural networks (RNNs), such as long short-term memory units (LSTMs) and gated recurrent units (GRUs), leverage a sequence of graphs, or their representations, in order to learn embeddings taking into consideration temporal correlation and store information over time to handle more complex correlations beyond consecutive timestamps. Depending on how the temporal and spatial modules are combined, the main architectures are classified into sequentially integrated (stacking) models like [1, 22, 23], where the two operations are performed in a cascade[1], or embedded convolutions, where the usual matrix multiplications within the gate equations of recurrent networks are replaced with graph convolution operations [24–26]. In this way, the recurrent unit simultaneously processes the topological and temporal dependencies of the data.

[1] The input snapshot graph first passes through the GCN to extract spatial features; then, the "spatialized" output is fed into the recurrent network.

PoS(ISGC2026)006

4. Materials and Methods

4.1 Graph Construction

To construct the spatio-temporal graph sequence, we pre-process the sensor time series with first-order differencing to remove common trends and potential spurious correlations. For each timestamp t_k, a graph snapshot $G(t_k) = (V(t_k), E(t_k))$ is then created by computing the pairwise Pearson correlation coefficients over a sliding window of length $w = 100$ past samples; edges are kept when the absolute correlation is above a chosen threshold τ, while adding self-loops and stability checks to avoid isolated or numerically problematic nodes. In this way, the dynamic graph $G = \{G(t_0), \ldots, G(t_{N_S-1})\}$ captures how relationships between nodes evolve over time while node attributes and prediction targets are kept on the original normalized signal levels.

For scalable training, snapshots are organized into mini-batches by concatenating multiple consecutive graphs as disconnected subgraphs of one larger sparse graph (block-diagonal connectivity), while node and target features are concatenated along the node dimension. This standard graph mini-batching strategy is crucial for large datasets because it enables parallel processing without padding and cross-graph information leakage.

4.2 Dataset

We evaluate the proposed model on monitoring data from the electrical and cooling infrastructure of the INFN CNAF Tier-1 data center in Bologna, Italy. The facility covers about 2000 m^2, including 500 m^2 of IT space, and is designed to sustain an IT load of 1.2 MW. The dataset contains 71 time series sampled every 15 minutes over 547 days, yielding 52,512 observations [4], each converted into a correlation-based graph snapshot following the procedure described in Section 4.1 with $w = 100$ and $\tau = 0.4$. The recorded electrical variables include power consumption, phase and neutral currents, power factor, active power, and DRUPS power losses. The monitored cooling variables include chiller and pump power consumption, water inlet and outlet temperatures, ambient temperature, and, for each chiller, circuit pressures and operating temperatures. All measurements were acquired through the facility Building Management System (BMS), which supervises plant operation.

4.3 Models

We adapted the hybrid VAE-GAN architecture already introduced in [3], redesigning it for batched spatio-temporal graph snapshots and graph-level latent learning. Temporal modeling is performed through stacked Temporal Graph Convolutional Network (T-GCN) blocks arranged sequentially, so that spatial and recurrent operations are applied in cascade: at each step, historical node signals are first processed by graph convolution to encode topological dependencies, and the resulting spatialized representations are then propagated through gated recurrence to capture temporal evolution [1]. The encoder aggregates node embeddings by global mean pooling and parameterizes a variational graph-level latent distribution, while the decoder reconstructs each snapshot by jointly estimating node features and sparse edge scores in a batch-consistent way; adversarial regularization is preserved through graph- and latent-level critics.

Furthermore, we implemented a second recurrent variant with the same overall VAE-GAN structure, training protocol, and batched graph handling, differing only in the recurrent cell design.

PoS(ISGC2026)006

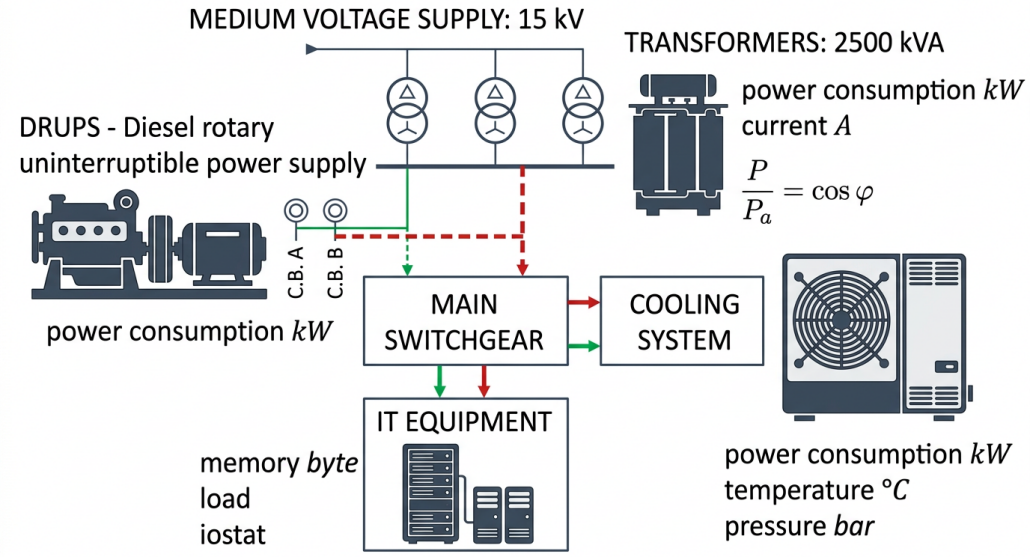

Figure 1: INFN CNAF Tier-1 Data Sources Overview: electrical infrastructure (DRUPS, medium-voltage supply, transformers) feeding the main switchgear, with cooling systems on the right and IT equipment loads below.

In this model, the standard affine transformations used in the GRU gate equations are replaced by embedded graph convolution operations based on GCNConv [27], so that both input-to-state and state-to-state updates are topology-aware. This substitution yields a Graph Convolutional GRU (GCNConvGRU) formulation that preserves the original gating logic while enforcing message passing directly inside the update, reset, and candidate-state computations. The cell itself was adapted, for comparability with T-GCN, from the Chebyshev Graph Convolutional GRU cell (GConvGRU) [2].

Temporal memory is stored at the node level during the recurrent stage, so each block carries a hidden state with one entry per node in the fused batch, from one batched input to the next. This means that the model does not keep a single memory for the whole graph; instead, it keeps a separate temporal trace for each node in the fused batch, allowing it to follow how local signals change over time (with a step s) while still taking the graph structure into account. Only after this step, node representations are combined through pooling to obtain a graph-level summary, which is then used for latent encoding and adversarial evaluation.

A formal and extensive description of the training procedure is provided in Section 4.3 of [3]. Here, we mention additional mechanisms introduced to stabilize the autoencoder component training, which we found to be significantly more prone to instability than in our previous work. In particular, we employ gradient clipping and an adaptive balancing strategy for the three loss terms that define the autoencoder as a multi-task learning network. We adopt Dynamic Weight Averaging (DWA) [28], a simple yet effective method that adjusts task weights over time based on the relative rate of change of their respective losses.

PoS(ISGC2026)006

5. Results

5.1 INFN CNAF Tier-1 Case Study

For this experiment, we employ an adversarial autoencoding setup with moderate-to-high representational capacity and conservative regularization. The encoder, decoder, graph discriminator, and latent discriminator all use a hidden size of 32 and three recurrent or dense blocks. The latent dimension is 48 (per graph), with single-channel node inputs.

We trained the models for 60 epochs and two graph-discriminator updates per iteration, a batch size of 64, combined with a step of 64 to have non-overlapping windows. The objective combines Wasserstein adversarial losses with gradient penalty ($\lambda = 10$) and edge-aware reconstruction regularization. Optimization uses Adam with $\beta_1 = 0.5$ and $\beta_2 = 0.999$; autoencoder dropout is set at 0.15; learning rates are 10^{-4} for the graph discriminator and 10^{-3} for both latent discriminator and autoencoder.

To obtain a high-level overview of the anomaly patterns in the data, we compute the mean reconstruction error of node and edge attributes across all sensor signals at each timestamp (Figure 2). A threshold corresponding to the 99th percentile ($q = 0.99$) is set after manual inspection of scores. Previous analysis on the same dataset revealed several distinct anomalous events [3]. Many of these are short in duration and correspond to sudden spikes or drops in electrical parameters associated with power transformers (January 26th, March 23rd, April 1st and 6th of 2022, as well as various events in May 2023). Both models fail to detect most of these in-train anomalies; this suggests that operational transitions (such as transformer handovers) were easily learned as normal behavior given the models' design and configuration. Nevertheless, the model successfully confirmed other anomalies and irregularities. These include the intense power consumption of the pumping system in June 2023 (Figure 3), along with an unconfirmed "anomalous" regime during January and February 2023 that was flagged by the edge reconstruction errors (Figure 4). Further investigation into this period revealed novel irregular patterns in the temperature and pressure readings from the refrigeration units [3].

5.2 Performance Benchmarking

To assess the anomaly detection capabilities of the T-GCN and GCNConvGRU models, we evaluate their performance using two publicly available IT-infrastructure datasets: the Server Machine Dataset (SMD) [5] and Pooled Server Metrics (PSM) [6]. These datasets provide a solid foundation for evaluating anomaly detection models on real-world IT environment time-series data, as they include predefined train-test splits, as well as ground-truth anomaly labels against which to evaluate the models. SMD encompasses 5 weeks of high-dimensional sensor data across 28 machines, while PSM collects anonymized application server metrics from eBay, and additionally tests a model's resilience to unlabelled anomalies present during the training phase.

We benchmark the new architectures against our previously proposed Graph Neural Network (GNN) framework, based on a comparable VAE-GAN setup. To ensure a direct and fair comparison, we follow some of the graph-structuring conditions used in our prior contribution [3]: 100-timestep ($w = 100$) signal windows and a cross-correlation threshold $\tau = 0.4$ for defining nodes connectivity. Model hyperparameters already introduced in Section 5.1 are also employed.

PoS(ISGC2026)006

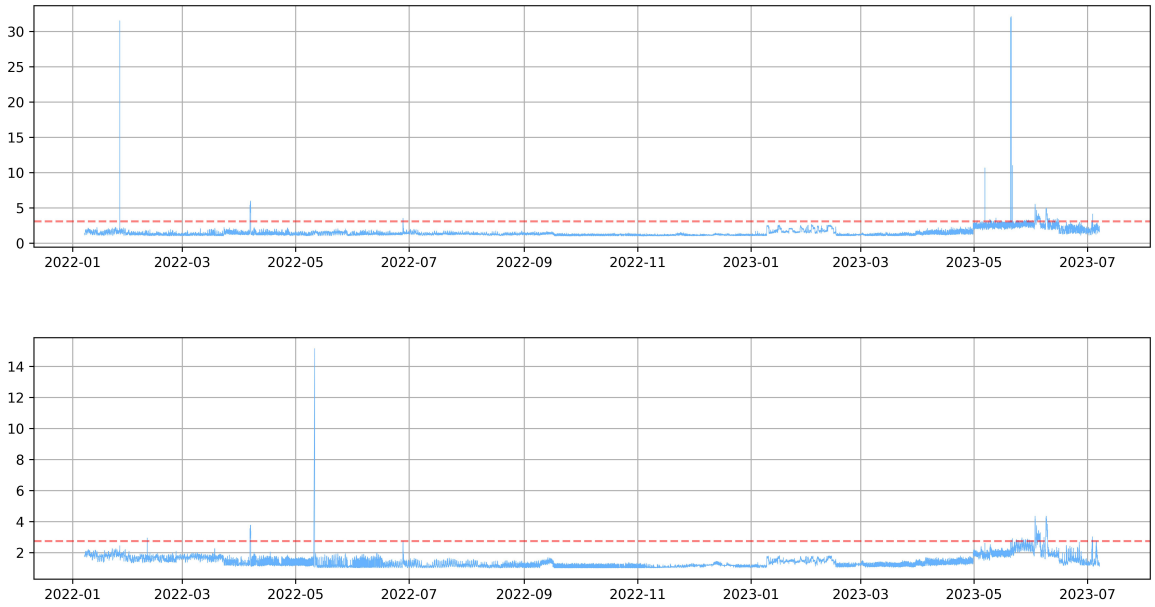

Figure 2: Mean node reconstruction errors as obtained from the two proposed models. **(Top)** T-GCN-based VAE. **(Bottom)** GCNConvGRU VAE. In the absence of proper confirmed anomalies, a threshold (red line) is set at $q = 0.99$.

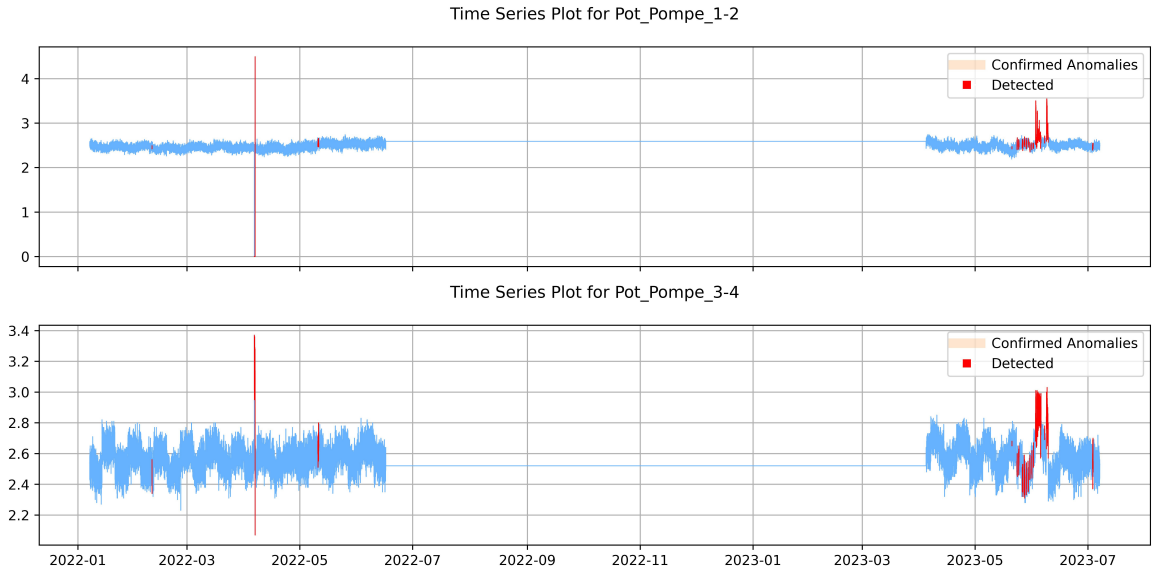

Figure 3: Time series plots showing anomalies detected by the GCNConvGRU model in the power consumption measurements of the pumping system [kW] running on a 2N redundancy.

PoS(ISGC2026)006

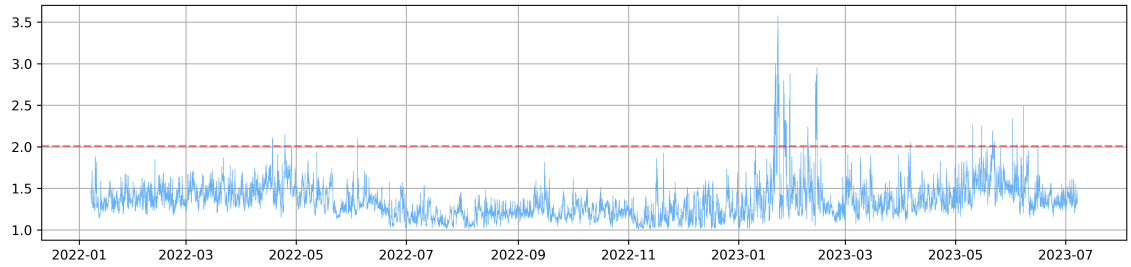

Figure 4: Mean edge reconstruction errors as obtained from the GCNConvGRU model.

We report scores under two distinct aggregation approaches. The first is the standard point-adjusted scoring, a widely used but highly biased method that records a true positive if any point within a true anomalous segment is detected. Because this method easily inflates performance metrics, we also evaluate all models using a time-gap aggregation scheme. This stricter, more operationally realistic approach groups predicted anomalies within a 12-hour maximum gap before scoring against the ground truth. Table 1 contextualizes the performance of the T-GCN and GCNConvGRU models against our previous GNN baselines [3]. We report the maximum $F_{0.2}$ score achieved through temporal mean aggregation and an optimal combination of critic scores and node or edge reconstruction errors.

While the point-adjusted metrics provide a necessary baseline for comparison with legacy literature, we place more emphasis on the time-gap $F_{0.2}$ scores. Under the point-adjusted scheme, the Sequential Multivariate Nodes approach [3] still dominates, achieving the highest scores across both datasets (an average of 0.820 on SMD and 0.968 on PSM). However, this performance proves highly fragile [29]. When subjected to the time-gap aggregation scheme, this baseline sees a performance degradation, plummeting to 0.466 on SMD and 0.397 on PSM. In a production environment, this behavior would translate to noisy, false-positive detections and inevitable operator fatigue. Conversely, the newly evaluated space-temporal graph models (T-GCN and GCNConvGRU) demonstrate greater robustness under the stricter evaluation criteria. On the SMD dataset, the GCNConvGRU architecture achieves the highest averaged time-gap score of 0.739. The outcome changes on the PSM dataset, which introduces the unique challenge of undetected anomalies within the training data. Here, the T-GCN model emerges as the most robust architecture, recording the highest time-gap score of 0.667 and significantly outperforming prior baselines.

It is important to mention a limitation regarding these benchmark results. Due to constraints in the available computational resources, we were unable to perform extensive hyperparameter tuning or extend the number of training epochs for the evaluated models across each dataset. Consequently, the reported metrics for the proposed spatio-temporal graph models (T-GCN and GCNConvGRU) likely represent a conservative estimate of their true capabilities. One would reasonably expect these architectures to achieve improved detection performance under better training conditions. Given these limitations, although the findings demonstrate clear advantages in the integration of spatial and temporal graphical structures, we avoid drawing definitive conclusions regarding the comparative superiority of either of these two models.

PoS(ISGC2026)006

Table 1: Performance benchmarking of spatio-temporal graph models on SMD and PSM datasets ($F_{0.2}$ scores). Results are presented as (Point-Adjusted / Time-Gap).

Model Architecture	Graph Strategy	SMD	PSM
[3]	Single-Layer Correlation Network	0.763 / 0.630	0.943 / 0.520
[3]	Sequential Multivariate Nodes	**0.820** / 0.466	**0.968** / 0.397
T-GCN	Space-Temporal Graphs	0.741 / 0.702	0.757 / **0.667**
GCNConvGRU	Space-Temporal Graphs	0.770 / **0.739**	0.881 / 0.497

5.3 Efficiency and Scalability Benchmarking

A scalability benchmark was also designed as a controlled, reproducible test of the compared recurrent graph blocks under the same training loop and optimization schedule. To isolate architecture effects from dataset-specific structure, we used a synthetic graph generator based on the *Erdős–Rényi* topology [30] with fixed node cardinality (71 as for the previous use-case), controlled edge probability (0.4), explicit self-loops, and univariate random node features and edge weights. Batches of 128 synthetic graphs were used as the experiment input unit.

Robustness was addressed through 10 repeated seed-based executions, where each run re-initialized model and synthetic stream deterministically to perform a fixed budget of warmup (20) and measured training iterations (1000). Metrics were therefore collected over multiple independent realizations, reducing sensitivity to random fluctuations.

Performance measurements include runtime and throughput (step latency distribution and processed samples per second) along with normalized efficiency indicators that scale latency/throughput by graph size (1k nodes/edges-normalized terms), enabling fairer comparison across structurally equivalent but computationally different model variants. Results were aggregated for each candidate model using cross-seed summary statistics, reporting both central tendency and variability in Table 2. For further information on measuring efficiency and scalability in GNNs training, please refer to [31] and [32].

Under this synthetic benchmark protocol and the hyperparameter configuration already introduced in Section 5.1, T-GCN exhibited the best computational profile when training on a data-center grade NVIDIA A100 GPU (CUDA 12.1, PyTorch 2.2.0, PyG 2.5.0), with lower step latency (549 ms), higher throughput (233 samples/s), and lower peak allocated memory (1.90 GB) than GCNConvGRU (676 ms, 189 samples/s, 3.04 GB). This result is consistent with the architectural distinction: in a GCNConvGRU cell, the standard affine transforms in GRU gate equations are replaced by topology-aware GCNConv operators, preserving GRU gating semantics while embedding message passing directly into update, reset, and candidate-state computations [2]. Empirically, this tighter graph-coupled recurrence increases per-step cost, which in these experiments translated into an approximately 19% runtime penalty and 37% higher memory footprint, despite the whole GCNConvGRU-based model using fewer parameters.

A further increase in computational load is observed when using the Chebyshev-based GConvGRU, where a spectral filter size of $K = 3$ is used. Under this configuration, the step time rises to 1056 ms, throughput drops to 121 samples/s, and peak allocated memory increases to 5.15 GB, indicating that the richer spectral operator clearly has a heavier efficiency cost in this setup.

PoS(ISGC2026)006

Table 2: Fixed synthetic benchmark summary (mean ± std across 10 runs per model). Arrows indicate preferred direction: ↓ lower is better, ↑ higher is better.

Metric	T-GCN	GCNConvGRU	GConvGRU
Step Time Mean [ms] ↓	**549.24 ± 5.92**	676.23 ± 7.63	1056.17 ± 4.50
Step Time p95 [ms] ↓	**574.71 ± 7.04**	702.32 ± 7.49	1085.48 ± 9.09
Throughput [samples/s] ↑	**233.08 ± 2.50**	189.31 ± 2.11	121.20 ± 0.51
Latency per 1k Edges [ms] ↓	**2.084 ± 0.023**	2.566 ± 0.029	4.007 ± 0.017
Latency per 1k Nodes [ms] ↓	**60.44 ± 0.65**	74.41 ± 0.84	116.22 ± 0.49
Samples/s per Million Edges ↑	**884.38 ± 9.45**	718.31 ± 8.02	459.86 ± 1.94
GPU Peak Memory [MB] ↓	**1903.12 ± 1.65**	3036.91 ± 1.84	5153.93 ± 3.05
Parameters	303841	259297	404449

6. Conclusions

This work investigated two spatio-temporal graph architectures, namely T-GCN and GCNConvGRU, for anomaly detection in multivariate sensor data. The proposed graph-based VAE-GAN formulation was able to model evolving dependencies between electrical and cooling signals from our INFN CNAF Tier-1 case study, while the benchmark study on SMD and PSM showed that both recurrent graph variants remain competitive under stricter event-oriented evaluation criteria. The efficiency benchmark showed that graph-integrated recurrent mechanisms such as GCNConvGRU can be introduced without prohibitive runtime penalties, although deeper recurrence results in progressively greater latency and memory requirements.

At the same time, current methodological limitations should be acknowledged. The graph-level decoder assumes a fixed node cardinality and reconstructs dense symmetric adjacency scores, which restricts the flexibility and application field of the approach. Likewise, the use of global mean pooling provides a compact graph representation, but may mask sparse and localized abnormal behaviors that arise in monitoring scenarios. On the temporal side, the fused inter-batch recurrence adopted here should be interpreted as a scalability-driven compromise: it enabled reasonable training runtimes thanks to batching, but it is not equivalent to a full temporal recurrence over long windows, which remains difficult to employ for our use cases.

A second major limitation concerns graph construction itself. In the current pipeline, graph snapshots are still derived from thresholded local correlations, which could still make the learned topology sensitive to noisy correlations, finite-window effects, and non-stationary regimes. Developing more robust graph construction criteria, for example by incorporating significance testing or temporal persistence constraints on edges, is therefore one of the main directions for future work. Finally, although the training procedure was substantially improved from a numerical perspective, stability cannot yet be considered fully solved: adversarial multi-objective balancing remains challenging and may still affect detection quality. Future research should therefore address these issues jointly, combining more robust graph construction, more locality-aware anomaly scoring, and more scalable temporal modeling in order to obtain STGNN-based detectors that are both accurate and deployable at scale in real-world data center operations.

PoS(ISGC2026)006

References

[1] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin et al., *T-GCN: A temporal graph convolutional network for traffic prediction*, *IEEE Transactions on Intelligent Transportation Systems* **21** (2018) 3848.

[2] Y. Seo, M. Defferrard, P. Vandergheynst and X. Bresson, *Structured sequence modeling with graph convolutional recurrent networks*, in *Neural Information Processing: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13-16, 2018, Proceedings, Part I*, (Berlin, Heidelberg), p. 362–373, Springer-Verlag, 2018, DOI.

[3] G. Zurlo, E. Ronchieri, R. Cornali, L. Scarponi, D. Cesini and L. Dell'Agnello, *Graph neural network solutions for interpretable anomaly detection in it infrastructure monitoring time series*, *Journal of Business Analytics* (2026) 1.

[4] E. Ronchieri, L. Giommi, L.B. Scarponi, L. Torzi, A. Costantini, D.C. Duma et al., *Anomaly Detection in Data Center IT & Physical Infrastructure*, *EPJ Web Conf.* **295** (2024) 07004.

[5] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun and D. Pei, *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), p. 2828–2837, Association for Computing Machinery, 2019, DOI.

[6] A. Abdulaal, Z. Liu and T. Lancewicki, *Practical approach to asynchronous multivariate time series anomaly detection and localization*, in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, (New York, NY, USA), p. 2485–2494, Association for Computing Machinery, 2021, DOI.

[7] S. Peng, M. Liu, L. Chai and R. Deng, *DST-GNN: A dynamic spatiotemporal graph neural network for cyberattack detection in grid-tied photovoltaic systems*, *IEEE Transactions on Smart Grid* **16** (2025) 330.

[8] X. Du, C. Zhou, Y.-C. Tian and B. Xu, *Anomaly detection based on graph neural networks incorporating with domain knowledge for industrial cyber-physical systems*, *Expert Syst. Appl.* **298** (2025) 129645.

[9] P. Wu, Y. Wang, J. Gao, X. Zhang, S. Lou and C. Yang, *Multichannel dynamic graph convolutional network-based fault diagnosis and its application in blast furnace ironmaking process*, *IEEE Sensors Journal* **23** (2023) 29293.

[10] X. Wang, Z. Xiao, J. Ge, J. Xiang and D. Zhou, *SSGC-GAT: Synergistic similarity graph construction strategy combined with GAT network for wind turbine anomaly identification using SCADA data*, *IEEE Transactions on Instrumentation and Measurement* **73** (2024) 1.

[11] A. Deng and B. Hooi, *Graph neural network-based anomaly detection in multivariate time series*, in *AAAI Conference on Artificial Intelligence*, 2021, https://api.semanticscholar.org/CorpusID:231646473.

PoS(ISGC2026)006

[12] C. Ding, S. Sun and J. Zhao, *Mst-gat: A multimodal spatial-temporal graph attention network for time series anomaly detection*, *ArXiv* **abs/2310.11169** (2022) .

[13] J. Zhan, S. Wang, X. Ma, C. Wu, C. Yang, D. Zeng et al., *STGAT-MAD: Spatial-temporal graph attention network for multivariate time series anomaly detection*, *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2022) 3568.

[14] R. Gao, Z. Chen, X. Wu, Y. Yu and L. Zhang, *Dynamic deep graph convolution with enhanced transformer networks for time series anomaly detection in iot*, *Cluster Computing* **28** (2024) .

[15] D.P. Kingma and M. Welling, *Auto-encoding variational bayes*, *CoRR* **abs/1312.6114** (2013) .

[16] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair et al., *Generative adversarial networks*, 2014. 10.48550/arxiv.1406.2661.

[17] T.N. Kipf and M. Welling, *Variational graph auto-encoders*, *CoRR* **abs/1611.07308** (2016) [1611.07308].

[18] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang et al., *GraphGAN: Graph representation learning with generative adversarial nets*, 2017.

[19] L. Zhang, L. Zhao, S. Qin, D. Pfoser and C. Ling, *TG-GAN: Continuous-time temporal graph deep generative models with time-validity constraints*, in *Proceedings of the Web Conference 2021*, WWW '21, (New York, NY, USA), p. 2104–2116, Association for Computing Machinery, 2021, DOI.

[20] H. Dai, Y. Wang, R. Trivedi and L. Song, *Recurrent coevolutionary latent feature processes for continuous-time recommendation*, in *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, DLRS 2016, (New York, NY, USA), p. 29–34, Association for Computing Machinery, 2016, DOI.

[21] C.D.T. Barros, M.R.F. Mendonça, A.B. Vieira and A. Ziviani, *A survey on embedding dynamic graphs*, 2021.

[22] G. Panagopoulos, G. Nikolentzos and M. Vazirgiannis, *United we stand: Transfer graph neural networks for pandemic forecasting*, *ArXiv* **abs/2009.08388** (2020) .

[23] A. Taheri, K. Gimpel and T. Berger-Wolf, *Learning to represent the evolution of dynamic graphs with recurrent models*, in *Companion Proceedings of The 2019 World Wide Web Conference*, WWW '19, (New York, NY, USA), p. 301–307, Association for Computing Machinery, 2019, DOI.

[24] J. Chen, X. Wang and X. Xu, *GC-LSTM: graph convolution embedded LSTM for dynamic network link prediction*, *Applied Intelligence* **52** (2022) 7513–7528.

PoS(ISGC2026)006

[25] Y. Li, R. Yu, C. Shahabi and Y. Liu, *Graph convolutional recurrent neural network: Data-driven traffic forecasting*, *CoRR* **abs/1707.01926** (2017) [1707.01926].

[26] J. Li, Z. Han, H. Cheng, J. Su, P. Wang, J. Zhang et al., *Predicting path failure in time-evolving graphs*, in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, (New York, NY, USA), p. 1279–1289, Association for Computing Machinery, 2019, DOI.

[27] T.N. Kipf and M. Welling, *Semi-supervised classification with graph convolutional networks*, *CoRR* **abs/1609.02907** (2016) [1609.02907].

[28] S. Liu, E. Johns and A.J. Davison, *End-to-end multi-task learning with attention*, *CoRR* **abs/1803.10704** (2018) [1803.10704].

[29] S. Kim, K. Choi, H.-S. Choi, B. Lee and S. Yoon, *Towards a rigorous evaluation of time-series anomaly detection*, *Proceedings of the AAAI Conference on Artificial Intelligence* **36** (2022) 7194.

[30] P. Erdös and A. Rényi, *On random graphs I*, *Publicationes Mathematicae Debrecen* **6** (1959) 290.

[31] K. Duan, Z. Liu, P. Wang, W. Zheng, K. Zhou, T. Chen et al., *A comprehensive study on large-scale graph training: Benchmarking and rethinking*, 2023.

[32] T. Baruah, K. Shivdikar, S. Dong, Y. Sun, S.A. Mojumder, K. Jung et al., *GNNMark: A benchmark suite to characterize graph neural network training on gpus*, in *2021 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 13–23, 2021, DOI.

PoS(ISGC2026)006